BRepFacetGen: Reverse Engineering B-Reps By Generative Face Segmentation

Ka-Hei Hui[1], Vikas Thamizharasan[2], Pradeep Kumar Jayaraman[1], and Xiang Xu[1]

[1] Auotdesk AI Lab
{ka.hei.hui,xiang.xu,pradeep.kumar.jayaraman}@autodesk.com
[2] University of Massachusetts Amherst
vthamizharas@umass.edu

Abstract. Boundary representations (B-Reps) are fundamental to computer aided design (CAD), as they encode editable parametric surfaces together with the precise topological structure required by downstream engineering workflows. In contrast, modern 3D generative models produce meshes or implicit fields, leaving a gap between high-quality geometry synthesis and CAD-native representations. Existing reconstruction pipelines treat B-Rep recovery as deterministic segmentation and fitting on discrete samples, despite the inherent ambiguity of CAD-style face decompositions, often resulting in unstable topology recovery. We introduce BRepFacetGen, a generative framework that formulates CAD structure inference as a latent variable problem conditioned on geometry representing continuous surfaces. Our representation couples a pretrained geometry latent set with a geometry-conditioned label latent space that models multiple plausible dense surface segmentations, naturally handling permutation ambiguity and structural non-uniqueness. Joint decoding yields a faceted mesh where each segment maps to a B-Rep face that can be recovered by surface fitting and topology reconstruction. Experiments show improved segmentation and more reliable downstream B-Rep recovery, as well as strong performance in image-conditioned and point-conditioned B-Rep generation.

Keywords: 3D Foundation Models · CAD Reconstruction

1 Introduction

Computer-aided design (CAD) models underpin modern engineering because they encode not only geometry, but structured, editable parametric surfaces with exact topology using boundary representations [46] (B-Reps), that downstream tools for editing, simulation and manufacturing critically rely on. Meanwhile, learning-based 3D generative models have made remarkable progress in synthesizing high-quality geometry from diverse inputs such as images and text. However, their outputs are typically meshes or implicit fields [27, 49], which lack the parametric surfaces and exact topology required for engineering workflows. Consequently, today's powerful 3D generative models remain largely disconnected from CAD-native design pipelines.

Bridging this gap is often framed as a reverse-engineering problem: given discrete geometry, recover a valid B-Rep through segmentation, surface fitting, and topology reconstruction. Existing approaches [24, 40, 56] largely treat this as a deterministic prediction task, operating on resolution-limited point samples and committing to a single face decomposition. Yet CAD-style decompositions are inherently ambiguous since multiple valid segmentations can correspond to the same geometry, and discretized predictions on points frequently destabilize topology recovery. These limitations make current methods brittle and difficult to integrate with modern generative models.

In this work, we introduce BRepFacetGen and argue that CAD structure inference should be treated as a generative problem on continuous surfaces. Rather than deterministically predicting a single segmentation on discrete point clouds, we model CAD-style face decomposition as a latent variable conditioned on geometry. This separates geometry synthesis from structural reasoning and allows multiple plausible CAD interpretations of the same shape. Moreover, predicting segmentation densely over the surface yields well-supported, high-resolution patches, which stabilize subsequent surface fitting and topology recovery.

Concretely, we represent each CAD object with two coupled components: (i) a geometry latent set produced by a strong pretrained 3D geometry prior, and (ii) a label latent set space that encodes multiple plausible dense face segmentations, handling permutation ambiguity and structural non-uniqueness. Decoding both latent sets jointly produces a segmented mesh, which serves as an interface to classical surface fitting, intersection, and trimming to reconstruct a B-Rep. Finally, to support upstream generative pipelines that provide only geometry latent sets, *e.g.*, point- or image-conditioned generation, we train a geometry-conditioned generator to sample compatible label latent sets, enabling CAD-structure decoding across input modalities.

Our results demonstrate that treating CAD reconstruction as generative rather than purely predictive, leads to more robust segmentation and improved downstream B-Rep recovery, enabling CAD-ready synthesis from modern 3D priors. We make the following contributions in this paper: (i) **A generative CAD representation** that combines pretrained geometry latent sets with a geometry-conditioned distribution over label latent sets, leveraging strong geometry priors while modeling CAD structure. (ii) **Label-latent learning** that captures multiple plausible dense face decompositions, resolving permutation ambiguity and structural non-uniqueness. (iii) **A geometry-conditioned generator** that produces compatible label latents from geometry ones, enabling both point- and image-conditioned generation. Strong empirical results show improved point-cloud segmentation and downstream B-Rep reconstruction, as well as superior performance in the image-conditioned settings.

2 Related Work

Generative models for B-Rep. CAD generative modeling broadly follows two paradigms: construction sequence–based generation and direct B-Rep syn-

thesis. Program-based approaches represent CAD models as procedural command sequences rather than explicit topology. DeepCAD [47] first modeled sketch–extrude programs using Transformers, followed by SkexGen [55] and HNC-CAD [53], which improved controllability through disentangled or hierarchical representations. Recent works bridge language and CAD programs, including Text2CAD [18], CADmium [11], CAD-Coder [6], and Text-to-CadQuery [50], while methods such as CADOps-Net [7], eCAD-Net [58], and CADCL [31] recover operation sequences from final B-Reps to restore parametric editability. Although these approaches provide strong semantic structure and controllability, they are fundamentally constrained by the scarcity of large-scale construction sequence datasets for complex models. In contrast, direct B-Rep generation synthesizes native topology and geometry without modeling histories. SolidGen [17] first generated explicit vertices, edges, and faces, though limited to simple primitives. Subsequent methods improved representation and scalability: BrepGen [54] introduced hierarchical latent diffusion for structured free-form generation; NeuroNURBS [8] leveraged compact NURBS parameterizations for faster post-processing; DTGBrepGen [23] decoupled topology and geometry for improved validity; BrepDiff [22] proposed a single-stage diffusion model with implicit edge encoding; HoLa [34] learned a holistic latent space of geometry and topology; and autoregressive approaches such as AutoBrep [52] and BrepGPT [25] replaced continuous representations with discrete tokens and structured graph traversal for compact, consistent part-level synthesis. CAD-Dreamer [28] reconstructs a neural SDF from an image, extracts a mesh, segments it, and then fits B-Reps. Recently, BR-DF [59] proposed generating B-Reps using signed and unsigned distance fields that can be used to produce a segmented mesh. Different from these methods, we leverage pretrained 3D foundation models for geometry generation and focus our efforts on segmenting the geometry into B-Rep faces.

Segmentation-driven B-Rep reconstruction. Classical segmentation-driven reconstruction typically partitions a point cloud into smooth regions, fits analytic/spline surfaces per region, and recovers B-Rep topology via intersections and trimming of adjacent surfaces [1, 20, 41, 44]. Learning-based methods largely mirror this workflow by predicting point-cloud instance segmentations to drive downstream fitting [5, 9, 10, 15, 24, 29, 40, 56], while Point2CAD [35] fits a parametric surface per segment (including free-form patches) and assembles a B-Rep by deriving edges and corners from intersections of adjacent surfaces. Because boundaries and geometry are inferred from discrete samples, reconstruction fidelity is often limited by sampling resolution, especially near sharp transitions and segment boundaries; we instead define segmentation on the mesh surface to enable high-resolution sampling for subsequent fitting.

3D foundation models. Learned 3D generative modeling has rapidly expanded across shape representations, including voxels/octrees [45, 51, 61], point clouds [26, 37, 43], meshes [42], and implicit fields [4, 16, 57]. Building on these advances, 3D foundation models scale model capacity and training data to learn general-purpose 3D priors that transfer across tasks and input modalities, often

with multi-modal conditioning for text-to-3D [39, 49], image-to-3D [14, 27, 30, 39, 48, 49, 60], and representation conversion, *e.g.*, point cloud to mesh [2, 3, 13]. However, their outputs are typically discretized meshes or implicits and thus do not directly provide CAD-native representations with explicit parametric surfaces and trimming curves; we aim to bridge this gap by lifting strong 3D priors toward B-Rep generation for CAD-ready, multi-modal synthesis.

3 Method

In this work, our goal is to learn to generate/reconstruct B-Reps by leveraging 3D foundation models for the overall geometry, followed by a segmentation of the geometry into disjoint B-Rep faces that can be fitted in post-process. We first reuse a frozen pre-trained 3D foundation model to encode the shape into a geometry latent set $\mathbf{Z}_g$ and decode it into a mesh (Section 3.1). Second, we learn a label latent space: a probabilistic latent set $\mathbf{Z}_\ell$ based on the geometry that can be sampled and decoded into dense face-instance labels on the surface (Section 3.2). Decoding $(\mathbf{Z}_g, \mathbf{Z}_\ell)$ yields a segmented mesh, which serves as the interface to classical segment-fit-assemble post-processing for recovering a processed B-Rep (Section 3.3). Finally, because in the generative settings only $\mathbf{Z}_g$ is available (without paired ground-truth label latent sets), we train a geometry-conditioned generator to model $p_\theta(\mathbf{Z}_\ell \mid \mathbf{Z}_g)$, enabling consistent segmentation decoding for geometry latent sets produced by different sources (Section 3.4).

3.1 Preliminary

To obtain the geometry latent set $\mathbf{Z}_g$ of a B-Rep, we adopt the latent vector set representation [57] to encode a surface point cloud of it into a compact set of latent vectors, and decode it as a continuous signed-distance field (SDF) that can be meshed via marching cubes. Concretely, we first sample a dense surface point set with normals $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{n}_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^3$ and $\mathbf{n}_i \in \mathbb{R}^3$. Let $\phi(\cdot)$ be a positional embedding; we form per-point features $\mathbf{p}_i^g = \mathrm{MLP}\big([\phi(\mathbf{x}_i), \mathbf{n}_i]\big) \in \mathbb{R}^C$ and stack them as $\mathbf{P}_g = [\mathbf{p}_1^g, \ldots, \mathbf{p}_N^g]^\top \in \mathbb{R}^{N \times C}$.

Surface points encoding. To compress $\mathcal{X}$ into $M \ll N$ latent vectors, we subsample M per-point features by farthest-point sampling (FPS) on coordinates and gather their corresponding features as $\mathbf{P}'_g \in \mathbb{R}^{M \times C}$. We can then initialize the latent set via a cross-attention layer:

$$\mathbf{Z}_g^{\mathrm{enc}} = \mathrm{Attn}\big(\mathbf{P}'_g, \mathbf{P}_g, \mathbf{P}_g\big) \in \mathbb{R}^{M \times C}, \tag{1}$$

where $\mathrm{Attn}(\cdot, \cdot, \cdot)$ is the standard attention operator, and followed by L_{enc} self-attention blocks on the latent set to exchange information:

$$\tilde{\mathbf{Z}}_g = \mathrm{SA}^{L_{\mathrm{enc}}}\big(\mathbf{Z}_g^{\mathrm{enc}}\big) \in \mathbb{R}^{M \times C}. \tag{2}$$

Here $\mathrm{SA}(\cdot)$ denotes a standard Attention block operating on the latent set with query, key and value as arguments, and $\mathrm{SA}^L(\cdot)$ stacks L such blocks.

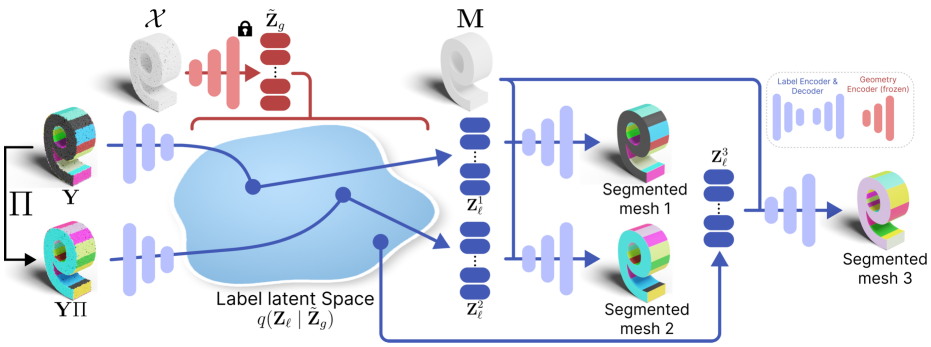

Fig. 1: Label latent space. Given dense surface samples $\mathcal{X}$, a frozen geometry encoder produces a pre-variational geometry latent set $\tilde{\mathbf{Z}}_g$ (red) and a mesh $\mathbf{M}$ via the geometry decoder. Conditioned on $\tilde{\mathbf{Z}}_g$, a label VAE (blue) encodes dense face-instance labels $\mathbf{Y}$ into a space over label latent sets, $q(\mathbf{Z}_\ell \mid \tilde{\mathbf{Z}}_g)$. During training, we randomly permute instance IDs, *e.g.*, $\mathbf{Y}\Pi$, to reflect the permutation ambiguity of segmentation labels, so that equivalent relabelings map to valid samples in the same geometry-conditioned latent space. At inference, sampling $\mathbf{Z}_\ell$ (blue) and decoding it on the mesh yields a segmented mesh; different samples can produce distinct, geometry-consistent segmentations (segmented meshes 1–3).

We refer to $\tilde{\mathbf{Z}}_g$ as a pre-variational latent set. To obtain the geometry latent set $\mathbf{Z}_g$, we map $\tilde{\mathbf{Z}}_g$ to per-latent Gaussian parameters and sample:

$$\begin{aligned} \boldsymbol{\mu}_g &= \mathrm{MLP}_\mu(\tilde{\mathbf{Z}}_g), \qquad \log \boldsymbol{\sigma}_g^2 = \mathrm{MLP}_\sigma(\tilde{\mathbf{Z}}_g), \\ \mathbf{Z}_g &= \boldsymbol{\mu}_g + \boldsymbol{\sigma}_g \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned} \tag{3}$$

Decoding SDF by querying the geometry latent set. Given query locations $\mathbf{x} \in \mathbb{R}^3$, the decoder first refines $\mathbf{Z}_g$ with L_{dec} self-attention blocks, and then uses cross-attention from query embeddings to the latent set to predict SDF values:

$$\begin{aligned} \mathbf{Z}_g^{\mathrm{dec}} &= \mathrm{Dec}(\mathbf{Z}_g) = \mathrm{SA}^{L_{\mathrm{dec}}}(\mathbf{Z}_g), \\ s(\mathbf{x}) &= \mathrm{MLP}\Big(\mathrm{Attn}(\phi(\mathbf{x}), \mathbf{Z}_g^{\mathrm{dec}}, \mathbf{Z}_g^{\mathrm{dec}})\Big), \end{aligned} \tag{4}$$

where $s(\mathbf{x})$ is the (truncated) signed distance at $\mathbf{x}$. Evaluating $s(\cdot)$ on a 3D grid and extracting the zero level-set, *e.g.*, via marching cubes, yields a mesh $\mathbf{M}$.

Geometry backbone. We consider the above as a geometry VAE in our formulation and implement it using a pre-trained TripoSG VAE model [27], which can decode the geometry component of the B-Rep. Note that we freeze the TripoSG encoder/decoder weights and only reuse the pre-variational latent set as the geometry context for our label VAE (detailed in the next section). Following the motivation in TripoSG, which posits that denser surface sampling improves the encoded geometric content, we feed N=16,384 surface points to the encoder.

3.2 Label Latent Space

Although the geometry latent set $\mathbf{Z}_g$ can be decoded into a high-quality mesh, it only captures shape geometry and does not provide the CAD-style structure needed for B-Rep reconstruction. To recover a B-Rep, we require an additional surface-level segmentation that partitions the mesh into face-like patches. However, predicting such a segmentation is challenging due to two reasons: (1) *Permutation ambiguity:* segmentation labels are discrete (one-hot), and the same partition admits many equivalent label permutations, making supervision and label alignment across shapes difficult (See segmented meshes 1 & 2 in Figure 1). (2) *Non-uniqueness:* a given mesh can admit multiple valid decompositions into CAD faces, *e.g.*, different plausible patch groupings can still yield reasonable B-Reps, making the learning target underdetermined (see segmented meshes 1 & 3 in Figure 1). Existing segmentation-driven pipelines [5,9,10,15,24,29,40,56] typically resolve permutation ambiguity using Hungarian matching [21] or feature clustering, but both introduce non-trivial complexity and can be brittle around boundaries. Moreover, when multiple decompositions are plausible, these methods often fail to model segmentation uncertainty.

Construction of Label Latent Space. To explicitly account for these ambiguities, we do not treat the segmentation as a deterministic prediction with a fixed label ordering. Instead, we introduce an label VAE that can encode multiple plausible segmentations into a probabilistic latent space $q(\mathbf{Z}_\ell \mid \tilde{\mathbf{Z}}_g)$ over label latent sets (as shown in Figure 1 (left)) given the geometry context $\tilde{\mathbf{Z}}_g$.

In practice, let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top \in \{0,1\}^{N \times K}$ be the one-hot face-instance labels on the N sampled surface points, where $\mathbf{Y}$ and $\mathbf{Y}\Pi$ (for any permutation matrix Π) represent the same partition. We encode labels by mirroring Section 3.1: we replace $\mathbf{n}_i$ with $\mathbf{y}_i$ to form $\mathbf{P}_\ell$ from $\mathbf{p}_i^\ell = \mathrm{MLP}\big([\phi(\mathbf{x}_i), \mathbf{y}_i]\big)$, and apply the same subsampling and self-attention refinement (Eq. (1)–(2)) to obtain the initial pre-variational latent set $\tilde{\mathbf{Z}}_\ell$. Next, we inject geometric context by cross-attending $\tilde{\mathbf{Z}}_\ell$ to the frozen geometry pre-variational latent set $\tilde{\mathbf{Z}}_g$ to produce a refined pre-variational latent set $\hat{\mathbf{Z}}_\ell$:

$$\hat{\mathbf{Z}}_\ell = \mathrm{Attn}\Big(\tilde{\mathbf{Z}}_\ell,\, \tilde{\mathbf{Z}}_g,\, \tilde{\mathbf{Z}}_g\Big)\,. \tag{5}$$

We can then sample the final label latent set $\mathbf{Z}_\ell$ as in Eq. (3).

Given a set of training query surface locations $\mathcal{Q} = \{\mathbf{q}_j\}_{j=1}^{N_q}$, we decode the label latents by following the same query-based decoding procedure as Eq. (4), but replacing the SDF regressor with a multiclass classifier. We supervise the decoder using a standard cross-entropy loss against the corresponding one-hot ground-truth labels (with the same label permutation applied when forming $\mathbf{Y}$) together with a KL regularization loss. Note that when the actual label count K_{active} in the shape is smaller than K, we will only employ a random permutation matrix that acts on the first K_{active} columns of Y.

3.3 B-Rep Decoding

Decoding of Segmented Meshes. Given a geometry latent set $\mathbf{Z}_g$ and a compatible label latent set $\mathbf{Z}_\ell$, we decode a segmented mesh as illustrated in Figure 1 (right). We first run the frozen geometry decoder to evaluate the SDF on a regular 3D grid, and extract the zero level-set with marching cubes [36] to obtain a triangle mesh $\mathbf{M}$. Next, we assign a face-instance label to each triangle by querying the label decoder (conditioned on $\mathbf{Z}_\ell$) at the triangle's face center. Repeating this procedure for all faces yields a fully segmented mesh, which is then passed to the subsequent B-Rep fitting and post-processing stage.

B-Rep Conversion. Given the segmented mesh $\mathbf{M}$ with per-triangle face-instance labels, we convert it to a processed B-Rep via a classical *segment–fit–assemble* pipeline. First, we resample $\mathbf{M}$ using Poisson-disk sampling to obtain a dense point set $\mathcal{X}_{\text{dense}} = \{\mathbf{x}_i\}_{i=1}^{N_d}$ on the surface (we use N_d=100,000 in all experiments). We then transfer instance labels from the segmented mesh to the sampled points by assigning each point the label of its triangle, producing a dense segmented point cloud. With this dense, labeled point set, we employ Point2CAD [35] for B-Rep reconstruction: for each instance label, Point2CAD fits a parametric surface patch to its corresponding point subset, and then recovers boundaries by intersecting adjacent fitted patches. Finally, it trims the fitted surfaces using these curves, derives vertices from curve intersections, and assembles the connectivity into a processed B-Rep.

3.4 Geometry-conditioned Label Latent Generation

In many use cases, we only have access to a geometry latent set $\mathbf{Z}_g$ produced by a shape-generation pipeline, but no paired label latent set $\mathbf{Z}_\ell$. To enable segmentation decoding in this setting, we learn a geometry-conditioned generator that models a conditional distribution over label latent sets. Specifically, we train $p_\theta(\mathbf{Z}_\ell \mid \mathbf{Z}_g)$ to match the label latent space induced by our label VAE for the same geometry (Section 3.2). At inference, we sample $\mathbf{Z}_\ell \sim p_\theta(\mathbf{Z}_\ell \mid \mathbf{Z}_g)$ and decode it with the label decoder to obtain a plausible face segmentation consistent with the generated geometry.

Flow-matching conditional generator. We model the conditional distribution $p_\theta(\mathbf{Z}_\ell \mid \mathbf{Z}_g)$ using flow matching [32]. During training, we construct paired latent sets $(\mathbf{Z}_g, \mathbf{Z}_\ell)$ for each CAD shape. Specifically, we obtain $\mathbf{Z}_g$ by encoding the input geometry with the frozen geometry encoder, and obtain $\mathbf{Z}_\ell$ by encoding the corresponding dense instance labels $\mathbf{Y}$ with the label encoder (Section 3.2) after applying a random permutation matrix Π to the instance IDs, *i.e.*, feeding $\mathbf{Y}\Pi$. Resampling Π yields multiple equivalent label latent sets for the same underlying partition, encouraging $p_\theta(\mathbf{Z}_\ell \mid \mathbf{Z}_g)$ to capture the distributional nature of the label latent space rather than a single canonical ordering.

Given such a sample $\mathbf{Z}_\ell$, we sample Gaussian noise $\mathbf{Z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (with the same shape as $\mathbf{Z}_\ell$) and draw $t \sim \mathcal{U}[0, 1]$. We then construct a linear interpolation

$$\mathbf{Z}_t = (1 - t)\mathbf{Z}_0 + t\mathbf{Z}_\ell, \tag{6}$$

whose target "denoising velocity" along this path is $\mathbf{v}^\star = \frac{d\mathbf{Z}_t}{dt} = \mathbf{Z}_\ell - \mathbf{Z}_0$. We train a geometry-conditioned velocity network $\mathbf{v}_\theta$ that takes the partially-noised label latent set $\mathbf{Z}_t$ (and a time embedding of t) while conditioning on $\mathbf{Z}_g$, and predicts the velocity field:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_\ell}\Big[\big\|\mathbf{v}_\theta(\mathbf{Z}_t, t, \mathbf{Z}_g) - (\mathbf{Z}_\ell - \mathbf{Z}_0)\big\|_2^2\Big]. \tag{7}$$

At inference, given $\mathbf{Z}_g$ we sample $\mathbf{Z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and solve the ODE $\frac{d\mathbf{Z}_t}{dt} = \mathbf{v}_\theta(\mathbf{Z}_t, t, \mathbf{Z}_g)$ from t=0 to t=1, producing a denoised sample $\mathbf{Z}_\ell \approx \mathbf{Z}_1$ that can be decoded by the label decoder.

Velocity network architecture. We parameterize the conditional velocity field $\mathbf{v}_\theta(\mathbf{Z}_t, t, \mathbf{Z}_g)$ using the TripoSG-style latent-set generator, but with a simpler conditioning mechanism. In TripoSG, conditioning is implemented via cross-attention; here we leverage the fact that both the geometry latent set $\mathbf{Z}_g \in \mathbb{R}^{M\times C}$ and our label latent set $\mathbf{Z}_t \in \mathbb{R}^{M\times C}$ are defined on the same M FPS anchors. The alignment enable performing latent-wise conditioning by channel concatenation:

$$\mathbf{H}_t = \mathrm{MLP}_{\mathrm{in}}\Big([\mathbf{Z}_t,\, \mathbf{Z}_g]\Big) \in \mathbb{R}^{M\times C}, \tag{8}$$

where $[\cdot,\cdot]$ denotes concatenation along the feature dimension and $\mathrm{MLP}_{\mathrm{in}}$ projects back to width C. We inject the timestep using a standard time embedding $\mathbf{e}(t)$ using the AdaIN-style conditioning adopted in [38], and process $\mathbf{H}_t$ with L_{gen} self-attention blocks:

$$\tilde{\mathbf{H}}_t = \mathrm{SA}^{L_{\mathrm{gen}}}\big(\mathbf{H}_t;\, \mathbf{e}(t)\big). \tag{9}$$

Finally, a lightweight output head predicts the denoising velocity with the same shape as $\mathbf{Z}_t$, $\mathbf{v}_\theta(\mathbf{Z}_t, t, \mathbf{Z}_g) = \mathrm{MLP}_{\mathrm{out}}(\tilde{\mathbf{H}}_t) \in \mathbb{R}^{M\times C}$. Overall, conditioning reduces to latent-wise concatenation and self-attention, avoiding cross-attention while still allowing global information exchange through the self-attention stack.

Sources of geometry latents. Our framework is agnostic to how the geometry latent $\mathbf{Z}_g$ is obtained, and supports two upstream generation/conditioning pipelines: (i) *Point-conditioned geometry:* given an input point cloud (with normals), we use the pre-trained TripoSG VAE to produce $\mathbf{Z}_g$. (ii) *Image-conditioned geometry:* given an input image, we obtain $\mathbf{Z}_g$ using the image-conditioned generator provided by TripoSG, which yields geometry latents compatible with the same frozen geometry decoder. In all cases, $\mathbf{Z}_g$ is passed to our geometry-conditioned generator to sample a compatible $\mathbf{Z}_\ell$ for segmentation decoding.

4 Experiments

Dataset. We train and evaluate on **ABC-1M** [52], a cleaned subset of ABC [19] built by decomposing multi-body STEP files into single-part solids and removing near-duplicates (∼1.3M unique CAD models). ABC-1M provides stratified `train/val/test` splits (70%/15%/15%) by face count. Ground-truth instance labels come directly from the B-Rep, treating each B-Rep face as one instance (without splitting periodic surfaces). We further restrict the dataset to shapes

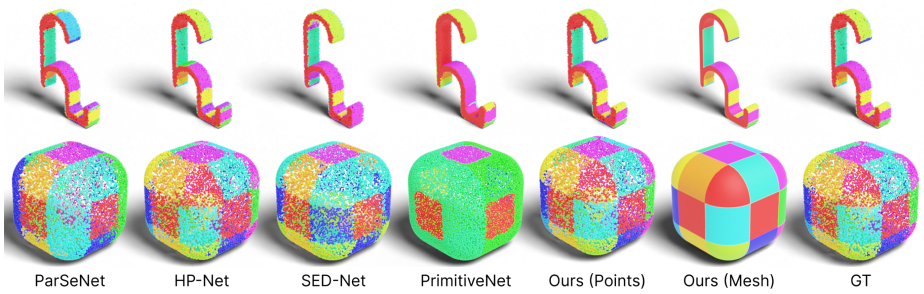

Fig. 2: Qualitative comparison of instance-level surface segmentation on representative shapes from the ParSeNet test set. Columns show ParSeNet, HP-Net, SED-Net, PrimitiveNet, our method queried on points (Ours (Points)), our method queried on a decoded mesh (Ours (Mesh)), and ground truth (GT). Colors indicate instance IDs; for visualization only, predicted IDs are aligned to GT via Hungarian matching.

with more than 10 and at most 100 faces to control complexity and match the operating range of our models. For evaluation, we use the intersection of our filtered ABC-1M test split with the released ParSeNet test set [40] (∼700 shapes), ensuring all methods are evaluated on the same CAD instances and ParSeNet-released surface samples. The resulting subset is challenging, as each shape contains at least one B-spline surface patch.

Training settings. We adopt the TripoSG architecture for both (i) our label VAE and (ii) the geometry-conditioned latent generator described in Section 3.4. For the label VAE, we train for 200k iterations with batch size 64 and learning rate 1e-4; we set the KL-loss weight to 1e-3. For the geometry-conditioned generator, we train for 700k iterations with batch size 128 and learning rate 5e-5, and maintain an exponential moving average (EMA) of the model parameters with decay 0.9999. All models are trained on 32 NVIDIA H100 GPUs; training the label VAE and the generator takes about 3 days and 7 days, respectively.

4.1 Point Segmentation

Baselines. We compare against four point-cloud instance segmentation methods: ParSeNet [40], HP-Net [56], SED-Net [29], and PrimitiveNet [15]. Following the ParSeNet protocol, we evaluate on the ParSeNet-released surface samples and read out the predicted instance label at each sampled point. While these baselines take a fixed-size input point set and output per-point predictions for that set, our method decouples conditioning from querying: we encode the input point cloud into geometry/label latent sets and query our label decoder at the evaluation samples. This enables evaluation on arbitrary query point sets (including ParSeNet samples) without retraining or changing input resolution. Note that for PrimitiveNet, the official release does not provide the full data split and preprocessing needed, so it is retrained on the ParSeNet dataset.

Metrics. We report standard instance-level segmentation metrics, including mean Intersection-over-Union (mIoU), mean accuracy, mean precision, mean re-

call, and mean F1 score. For each shape, we first compute per-instance statistics by treating each active instance label as a binary classification problem over the evaluated surface sample points. We then compute IoU, accuracy, precision, recall, and F1 for each instance label and average them across all active labels in the shape; finally, we report the mean across the test set.

Quantitative Comparisons. Table 1 reports results on the common test set, where our method achieves the best performance across all metrics. The largest gains are on point-wise correctness metrics (accuracy, precision, recall, and F1), indicating fewer false assignments and fewer missed regions, and thus cleaner separation of adjacent surface instances. Although trained with denser samples, our model remains robust when evaluated on the lower-resolution ParSeNet samples.

Table 1: Instance-level segmentation results. Higher is better for all metrics.

Method	mIoU	Acc.↑	Prec.↑	Rec.↑	F1↑
ParSeNet [40]	0.7671	0.5232	0.5472	0.6021	0.5582
HP-Net [56]	0.7486	0.5542	0.5928	0.6263	0.5896
SED-Net [29]	0.8010	0.5130	0.5262	0.5857	0.5420
PrimitiveNet [15]	0.7189	0.3069	0.3527	0.3479	0.3376
Ours	**0.8046**	**0.7040**	**0.7606**	**0.7575**	**0.7470**

Qualitative Comparisons. Figure 2 visualizes predictions on ParSeNet test shapes: we color surface samples by predicted instance ID and align IDs to ground truth via Hungarian matching for visualization. ParSeNet, HP-Net, SED-Net, PrimitiveNet, and Ours (Points) are rendered on the same ParSeNet samples, while Ours (Mesh) queries the decoder on the decoded mesh. Overall, our method produces less fragmentation and less label leakage across boundaries, especially on shapes with many small patches and curved regions; querying on the decoded mesh further reduces boundary aliasing and yields sharper transitions. Additional visualizations are provided in the supplementary materials.

4.2 Point to B-Rep

Baselines. We compare against segmentation-to-CAD pipelines built from the instance segmentation baselines in Section 4.1. For each baseline (ParSeNet/HP-Net/SED-Net/PrimitiveNet), we feed the input points and predicted instance labels into Point2CAD [35] to fit per-segment surface patches and assemble a B-Rep. For our method, we condition on the same input point cloud but run Point2CAD on a denser point set sampled from our reconstructed mesh (Section 3.3), with per-point labels obtained by querying our label decoder on the mesh. Beyond segmentation-driven pipelines, we include two direct point-to-B-Rep baselines: ComplexGen [12] predicts vertices, curves, surfaces, and their topology as a B-Rep chain complex, followed by constrained global optimization and geometric refinement, while Split-and-Fit [33] predicts a Voronoi-style cell decomposition and fits primitives per cell before recovering B-Rep structure through cell adjacencies and intersection reasoning (it focuses on simple analytic primitives and does not explicitly model spline patches).

Metrics. Following Split-and-Fit, we report three metric categories: (i) geometric error via Chamfer Distance (CD) for vertices/curves/surfaces, computed from point samples on each primitive; (ii) detection precision/recall/F1 by matching predicted primitives to ground truth using CD, averaged over thresholds (0.1,

Table 2: Geometric error, topology quality, and detection performance for B-Rep reconstruction. CD is measured for vertices (V), curves (C), and surfaces (S) (lower is better). FE/EV evaluate topological correctness (higher is better). Detection metrics are reported on V/C/S (higher is better). **Best results (excluded GT) are in bold**.

Method	CD ↓			Topology ↑		F1 ↑			Precision ↑			Recall ↑		
	V	C	S	FE	EV	V	C	S	V	C	S	V	C	S
ComplexGen [12]	**0.205**	0.173	0.106	0.186	0.298	0.236	0.205	0.273	0.250	0.275	0.259	0.238	0.175	0.305
Split-and-Fit [33]	0.416	0.154	**0.051**	0.167	0.182	0.216	0.211	0.347	0.258	0.344	0.460	0.264	0.212	0.335
ParSeNet [40] + Point2CAD [35]	0.399	0.144	0.086	0.236	0.213	0.203	0.208	0.293	0.270	0.348	0.385	0.183	0.158	0.249
SED-Net [29] + Point2CAD [35]	0.383	0.142	0.088	0.202	0.202	0.224	0.208	0.294	0.323	0.384	0.412	0.190	0.153	0.240
HP-Net [56] + Point2CAD [35]	0.357	0.108	0.076	0.245	0.231	0.232	0.240	0.328	0.311	0.400	0.423	0.210	0.185	0.281
PrimitiveNet [15] + Point2CAD [35]	0.563	0.177	0.091	0.119	0.116	0.144	0.137	0.250	0.280	0.343	0.420	0.111	0.090	0.188
Ours + Point2CAD [35]	0.233	**0.085**	0.057	**0.315**	**0.304**	**0.558**	**0.579**	**0.770**	**0.622**	**0.736**	**0.798**	**0.561**	**0.509**	**0.768**

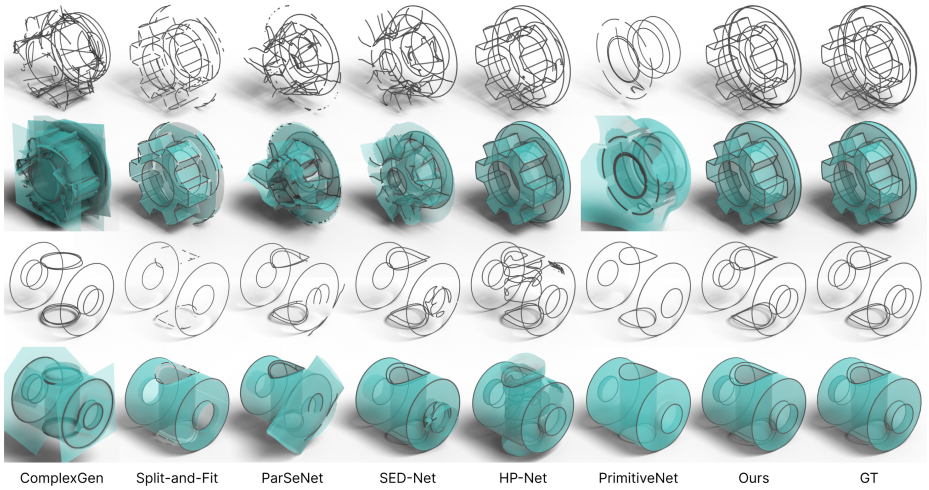

Fig. 3: Point-cloud to B-Rep qualitative comparison. Reconstructed B-Rep wireframes and surfaces from different methods obtained using Point2CAD [35]. Ours produces cleaner, complete curves and surfaces with consistent connectivity.

0.05, 0.02, 0.01, 0.005); and (iii) topological error via F1 for surface-curve connectivity (FE) and curve-vertex connectivity (EV).

Quantitative Comparisons. Table 2 summarizes point-cloud to B-Rep reconstruction results. Overall, our approach achieves the strongest performance on curve/surface geometry, detection, and topology. Compared to the strongest segmentation-to-fitting baseline (HP-Net+Point2CAD), we reduce curve and surface CD and substantially improve detection F1 for vertices, curves, and surfaces, indicating more complete and more recoverable surface regions for downstream fitting. Compared with direct point-to-B-Rep methods, although ComplexGen achieves strong vertex accuracy and Split-and-Fit can attain very low surface error on shapes well covered by its primitive library, our method remains consistently accurate across vertices/curves/surfaces and yields significantly higher topological correctness. This suggests that dense, face-consistent partitions coupled with fitting produce B-Reps with more reliable connectivity for intersection-and-trimming.

Table 3: Image-to-B-Rep results. Lower is better for Chamfer Distance (CD); higher is better for the other metrics.

	Geometry		Segmentation				
Method	CD↓	IoU↑	mIoU↑	Acc.↑	Prec.↑	Rec.↑	F1↑
CADDreamer [28]	0.2606	0.2835	0.0744	0.3413	0.1335	0.1488	0.1151
HP-Net [56]	-	-	0.2431	0.5583	0.3660	0.3626	0.3204
Ours	**0.1185**	**0.5502**	**0.3041**	**0.6054**	**0.4518**	**0.4225**	**0.3904**

Fig. 4: Image-to-B-Rep qualitative comparison. Mesh/Point colors indicate predicted/ground-truth face instances (aligned to the ground truth for visualization). Compared to CADDreamer [28] and HP-Net [56], our approach yields substantially cleaner surface reconstruction, and our geometry-compatible label latents produce more coherent, face-consistent segmentations thanks to the pre-trained geometry backbone.

Qualitative Comparisons. Figure 3 shows two challenging examples. ComplexGen can produce noisy outputs with many spurious curves and surfaces (top), and Split-and-Fit often struggles with curved geometry, yielding jagged piecewise approximations. For segmentation-driven pipelines (ParSeNet/SED-Net/HP-Net/PrimitiveNet + Point2CAD), segmentation errors commonly propagate to fitting as warped curves, broken loops, or inconsistent connectivity. In contrast, our method produces cleaner segmentations and denser geometric evidence for fitting, resulting in smoother curves and surfaces closer to the GT B-Rep. More visualizations are provided in the supplementary material.

4.3 Image to B-Rep

Baselines. We compare against CADDreamer [28], which reconstructs a mesh from a single image and partitions it into primitive patches using multi-view normal maps and semantic primitive maps, followed by patch extraction. Since both methods output a segmented mesh, we evaluate directly at the mesh level. Additionally, to isolate the contribution of the geometry backbone, we include a baseline that uses TripoSG [27] to reconstruct the geometry and applies HP-Net [56] to surface points sampled from the reconstructed mesh to obtain point-wise instance labels.

Metrics. For our method and CADDreamer, we first rigidly align the reconstructed mesh to the ground-truth mesh using Iterative Closest Point (ICP). We then report geometric fidelity using Chamfer Distance (CD) and volumetric IoU. To evaluate segmentation, we uniformly sample 10,000 points on the ground-truth surface and transfer instance labels from the predicted segmented mesh by assigning each sample the label of its nearest predicted face. For the HP-Net baseline, we follow the same label-transfer protocol, but assign each ground-truth surface sample the label of its nearest predicted point rather than

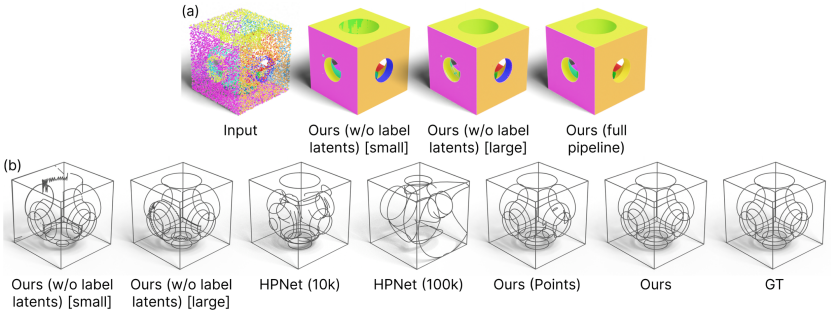

Fig. 5: Ablation study on label latents and fitting resolution. (a) Segmentation results. Deterministic variants without label latents produce less stable CAD face partitions, even with increased model capacity, while our full pipeline yields cleaner and more coherent regions. (b) Point2CAD reconstruction. Boundary leakage or unstable partitions from deterministic variants and HP-Net propagate to noisy or distorted fitted curves. Ours (Points) improves reconstruction from sparse points, and our default dense mesh-based setting recovers cleaner wireframes closer to the ground-truth B-Rep.

the label of the nearest predicted mesh face. We compute instance-level mIoU, accuracy, precision, recall, and F1 by averaging over active instances per shape and then averaging across the test set.

Quantitative Comparisons. Table 3 reports image-to-B-Rep results under this protocol. Our method consistently outperforms CADDreamer on both geometry and segmentation: the frozen pre-trained geometry backbone improves surface fidelity (lower CD, higher IoU), while our generative label latent formulation yields more accurate and more complete face partitions (higher mIoU and point-wise accuracy/precision/recall/F1). These results highlight the importance of combining a strong geometry prior with geometry-conditioned label generation for CAD-like segmented surfaces from a single image. We observe that applying HP-Net to TripoSG geometry still underperforms our approach, demonstrating that the gains do not come from the geometry backbone alone, but also from our geometry-conditioned label-latent segmentation formulation.

Qualitative Comparisons. Figure 4 shows representative examples. CADDreamer often exhibits geometric drift, *e.g.*, over-smoothed features, distorted silhouettes, or missing/merged structures, which leads to unstable patch extraction and fragmented segmentations with label leakage across boundaries. Consistent with the point-segmentation setting, HP-Net also exhibits label leakage across adjacent surface boundaries, leading to less coherent face partitions. In contrast, our reconstruction and segmentation better preserve global structure and sharper surface transitions, producing larger, more coherent face-like regions whose boundaries align more closely with geometric discontinuities.

4.4 Ablation Study

We ablate two key design choices: (i) modeling segmentation as a geometry-conditioned space over label latents (ours) versus a deterministic segmentation

Table 5: Ablation on Point2CAD input resolution and segmentation formulation. Ours (Points) fits from sparse labeled input points, while Ours fits from dense samples on the decoded mesh. Ours (w/o label latents) uses deterministic segmentation heads; *small* matches the label VAE size and *large* matches the VAE+generator size. HP-Net (100k) directly segments dense points, and HP-Net (10k) segments at its training resolution before transferring labels to dense samples. Lower is better for CD; higher is better for topology and detection.

	CD ↓			Topology ↑		F1 ↑			Precision ↑			Recall ↑		
Method	V	C	S	FE	EV	V	C	S	V	C	S	V	C	S
Ours (Points)	0.361	0.124	0.078	0.286	0.261	0.264	0.295	0.417	0.338	0.449	0.473	0.249	0.237	0.387
Ours (w/o label latents)[small]	0.268	0.099	0.059	0.308	0.280	0.292	0.344	0.470	0.309	0.423	0.463	0.324	0.318	0.493
Ours (w/o label latents)[large]	0.272	0.099	0.060	0.285	0.266	0.255	0.318	0.448	0.261	0.369	0.430	0.291	0.306	0.484
HP-Net (100k)	0.464	0.159	0.080	0.160	0.143	0.122	0.119	0.222	0.165	0.230	0.343	0.118	0.088	0.176
HP-Net (10k)	0.303	0.104	0.067	0.270	0.250	0.248	0.260	0.355	0.285	0.376	0.426	0.247	0.218	0.318
Ours	**0.233**	**0.085**	**0.057**	**0.315**	**0.304**	**0.558**	**0.579**	**0.770**	**0.622**	**0.736**	**0.798**	**0.561**	**0.509**	**0.768**

head, and (ii) running the downstream segment-fit-assemble stage on dense samples from the decoded mesh versus directly on the sparse input points.

Generative Label Latents vs. Deterministic Segmentation. To isolate the effect of our label-latent formulation, we remove the label VAE and train a deterministic segmentation head conditioned on the frozen geometry latent set. Since instance IDs are permutation-ambiguous, this baseline is trained with Hungarian matching to align predicted instances with ground-truth faces. We evaluate two deterministic variants: a *small* model matched to the parameter count of the label VAE, and a *large* model matched to the combined parameter count of the label VAE and the label-latent generator. As shown in Table 4, both deterministic variants underperform our generative formulation, even when the deterministic model is given comparable capacity to the full label-generation stack. This indicates that the improvement is not simply due to model size, but to modeling a geometry-conditioned distribution over plausible CAD face partitions.

Table 4: Ablation on segmentation formulation. The deterministic variants remove label latents and are trained with Hungarian matching; *small* and *large* denote capacity-matched settings. Best of 10 reports oracle selection over 10 sampled segmentations. Higher is better.

Method	mIoU↑	Acc.↑	Prec.↑	Rec.↑	F1↑
Ours (w/o label latents) [small]	0.7360	0.6136	0.6954	0.6687	0.6633
Ours (w/o label latents) [large]	0.7167	0.5862	0.6725	0.6435	0.6372
Ours	**0.8046**	**0.7040**	**0.7606**	**0.7575**	**0.7470**
Best of 10 Generations	0.8368	0.7952	0.8500	0.9223	0.8397

Although our default result reports one generated segmentation per input, the probabilistic formulation also enables sampling multiple plausible segmentations. Across 10 generations per input, the average pairwise adjusted Rand index (ARI) is 0.934, indicating that the generated segmentations are largely consistent while still allowing meaningful alternatives. Moreover, selecting the best of 10 generations improves mIoU from 0.8046 to 0.8368 and F1 from 0.7470 to 0.8397, suggesting that the label-latent space captures useful segmentation variations that can further improve downstream quality.

Dense Mesh Fitting vs. Sparse-point Fitting. To study the effect of geometric evidence, Ours (Points) evaluates segmentation on the sparse input point cloud and runs Point2CAD on those labeled points, while our default setting decodes a mesh from the frozen geometry backbone, densely samples points on the surface, and assigns labels by querying the label decoder on the mesh before fitting. As shown in Table 5, dense mesh sampling improves reconstruction quality across geometry, detection, and topology, indicating fewer missing primitives and more reliable connectivity. In Figure 5, sparse-point fitting can produce distorted curves, whereas mesh-based fitting recovers smoother, more complete curve wireframes that better match the ground-truth B-Rep.

To ensure that this gain is not merely due to using more points, we also evaluate HP-Net under denser reconstruction settings. HP-Net (100k) applies HP-Net directly to 100k points, while HP-Net (10k) runs HP-Net at its training resolution and transfers the predicted labels to the denser point set used by Point2CAD. Both variants remain below our method, showing that dense sampling alone is insufficient: accurate and resolution-consistent segmentation is necessary for reliable downstream fitting. Qualitatively, Figure 5 shows that HP-Net and deterministic variants introduce boundary leakage or unstable partitions, whereas our mesh-based pipeline recovers cleaner, more complete wireframes that better match the ground-truth B-Rep.

5 Limitations and Future Work

First, our image-to-geometry quality is bounded by the frozen TripoSG backbone, which is trained on general 3D shapes and can miss CAD-specific surfaces in occluded regions (examples in supplementary material); an interesting direction is CAD-aware fine-tuning of the pretrained model. Second, we currently focus on point- and image-conditioned inputs due to the availability of TripoSG, and extending to other conditions, *e.g.*, text or sketches, by training new geometry-latent generators would broaden applicability. Finally, our geometry-conditioned label-latent generation uses flow matching and requires multiple inference steps; exploring faster sampling, *e.g.*, fewer-step solvers or distillation, could reduce inference cost. Lastly, while we leveraged the Point2CAD [35] algorithm to convert our segmented mesh to B-Rep, advanced surface fitting techniques available in solid modeling kernels could yield higher quality B-Reps suitable for mainstream CAD applications.

6 Conclusion

We presented BRepFacetGen, a generative framework that bridges modern 3D priors and CAD-native B-Rep representations by modeling face decomposition as a geometry-conditioned latent variable on continuous surfaces. By coupling pretrained geometry latent sets with a space over label latent sets, our approach produces dense, face-consistent segmentations that better support downstream surface fitting and topology recovery. Experiments show improved instance segmentation from point clouds and more accurate B-Rep reconstruction, while also enabling CAD generation in both point- and image-conditioned settings.

References

1. Benkő, P., Martin, R.R., Várady, T.: Algorithms for reverse engineering boundary representation models. Computer-Aided Design **33**(11), 839–851 (2001)
2. Chen, S., Chen, X., Pang, A., Zeng, X., Cheng, W., Fu, Y., Yin, F., Wang, B., Yu, J., Yu, G., et al.: Meshxl: Neural coordinate field for generative 3d foundation models. Advances in Neural Information Processing Systems **37**, 97141–97166 (2024)
3. Chen, Y., He, T., Huang, D., Ye, W., Chen, S., Tang, J., Chen, X., Cai, Z., Yang, L., Yu, G., et al.: Meshanything: Artist-created mesh generation with autoregressive transformers. arXiv preprint arXiv:2406.10163 (2024)
4. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5939–5948 (2019)
5. Cherenkova, K., Dupont, E., Kacem, A., Gusev, G., Aouada, D.: Spelsnet: Surface primitive elements segmentation by b-rep graph structure supervision. Conference on Neural Information Processing Systems (NeurIPS) (2024)
6. Doris, A.C., Alam, M.F., Nobari, A.H., Ahmed, F.: Cad-coder: An open-source vision-language model for computer-aided design code generation. arXiv preprint arXiv:2505.14646 (2025)
7. Dupont, E., Cherenkova, K., Kacem, A., Ali, S.A., Arzhannikov, I., Gusev, G., Aouada, D.: Cadops-net: Jointly learning cad operation types and steps from boundary-representations. In: 2022 International Conference on 3D Vision (3DV). pp. 114–123 (2022). https://doi.org/10.1109/3DV57658.2022.00024
8. Fan, J., Gholami, B., Bäck, T., Wang, H.: Neuronurbs: Learning efficient surface representations for 3d solids (2024), https://arxiv.org/abs/2411.10848
9. Fang, Z., Zhuang, C., Lu, Z., Wang, Y., Liu, L., Xiao, J.: Bgpseg: Boundary-guided primitive instance segmentation of point clouds. IEEE Transactions Image Processing (2025)
10. Fu, R., Wen, C., Li, Q., Xiao, X., Alliez, P.: Bpnet: B\'ezier primitive segmentation on 3d point clouds. arXiv preprint arXiv:2307.04013 (2023)
11. Govindarajan, P., Baldelli, D., Pathak, J., Fournier, Q., Chandar, S.: CADmium: Fine-tuning code language models for text-driven sequential CAD design. arXiv preprint arXiv:2507.09792 (2025)
12. Guo, H., Liu, S., Pan, H., Liu, Y., Tong, X., Guo, B.: Complexgen: Cad reconstruction by b-rep chain complex generation. ACM Transactions on Graphics (2022)
13. Hao, Z., Romero, D.W., Lin, T.Y., Liu, M.Y.: Meshtron: High-fidelity, artist-like 3d mesh generation at scale. arXiv preprint arXiv:2412.09548 (2024)
14. He, X., Zou, Z.X., Chen, C.H., Guo, Y.C., Liang, D., Yuan, C., Ouyang, W., Cao, Y.P., Li, Y.: Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. arXiv preprint arXiv:2503.21732 (2025)
15. Huang, J., Zhang, Y., Sun, M.: Primitivenet: Primitive instance segmentation with local primitive embedding under adversarial metric. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15343–15353 (2021)
16. Hui, K.H., Li, R., Hu, J., Fu, C.W.: Neural wavelet-domain diffusion for 3d shape generation. In: SIGGRAPH Asia 2022 conference papers. pp. 1–9 (2022)
17. Jayaraman, P.K., Lambourne, J.G., Desai, N., Willis, K.D.D., Sanghi, A., Morris, N.J.W.: Solidgen: An autoregressive model for direct b-rep synthesis. "arXiv Preprint" (2022). https://doi.org/10.48550/ARXIV.2203.13944, https://arxiv.org/abs/2203.13944

18. Khan, M.S., Sinha, S., Uddin, S.T., Stricker, D., Ali, S.A., Afzal, M.Z.: Text2cad: Generating sequential CAD designs from beginner-to-expert level text prompts. In: Conference on Neural Information Processing Systems (NeurIPS) (2024)
19. Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., Panozzo, D.: Abc: A big cad model dataset for geometric deep learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9601–9611 (2019)
20. Kovács, I., Várady, T., Salvi, P.: Applying geometric constraints for perfecting cad models in reverse engineering. Graphical Models **82**, 44–57 (2015)
21. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)
22. Lee, M., Zhang, D., Jambon, C., Kim, Y.M.: Brepdiff: Single-stage b-rep diffusion model. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–11 (2025)
23. Li, J., Fu, Y., Chen, F.: Dtgbrepgen: A novel b-rep generative model through decoupling topology and geometry. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 21438–21447 (2025)
24. Li, L., Sung, M., Dubrovina, A., Yi, L., Guibas, L.: Supervised fitting of geometric primitives to 3d point clouds. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
25. Li, P., Zhang, W., Quan, W., Zhang, B., Wonka, P., Yan, D.: Brepgpt: Autoregressive b-rep generation with voronoi half-patch. ACM Transactions on Graphics (TOG) **44**(6), 1–18 (2025)
26. Li, R., Li, X., Hui, K.H., Fu, C.W.: SP-GAN:sphere-guided 3D shape generation and manipulation. ACM Transactions on Graphics (SIGGRAPH) **40**(4) (2021)
27. Li, Y., Zou, Z.X., Liu, Z., Wang, D., Liang, Y., Yu, Z., Liu, X., Guo, Y.C., Liang, D., Ouyang, W., et al.: Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. arXiv preprint arXiv:2502.06608 (2025)
28. Li, Y., Lin, C., Liu, Y., Long, X., Zhang, C., Wang, N., Li, X., Wang, W., Guo, X.: Caddreamer: Cad object generation from single-view images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2025)
29. Li, Y., Liu, S., Yang, X., Guo, J., Guo, J., Guo, Y.: Surface and edge detection for primitive fitting of point clouds. In: ACM SIGGRAPH 2023 conference proceedings. pp. 1–10 (2023)
30. Li, Z., Wang, Y., Zheng, H., Luo, Y., Wen, B.: Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. arXiv preprint arXiv:2505.14521 (2025)
31. Liang, J., He, F., Fan, R., Chu, Y., Yan, X.: Cadcl: Reconstruct parametric cad models from b-rep via contrastive learning. Journal of Computational Design and Engineering **12**(10), 176–184 (10 2025). https://doi.org/10.1093/jcde/qwaf102, https://doi.org/10.1093/jcde/qwaf102
32. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
33. Liu, Y., Chen, J., Pan, S., Cohen-Or, D., Zhang, H., Huang, H.: Split-and-fit: Learning b-reps via structure-aware voronoi partitioning. Proceedings of SIGGRAPH (2024)
34. Liu, Y., Xu, D., Yu, X., Xu, X., Cohen-Or, D., Zhang, H., Huang, H.: Hola: B-rep generation using a holistic latent representation. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **44**(4) (2025)

35. Liu, Y., Obukhov, A., Wegner, J.D., Schindler, K.: Point2cad: Reverse engineering cad models from 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3763–3772 (2024)
36. Lorensen, W.E., Cline, H.E.: Marching Cubes: A high resolution 3D surface construction algorithm. In: Proceedings of SIGGRAPH. vol. 21, pp. 163–169 (1987)
37. Luo, S., Hu, W.: Diffusion probabilistic models for 3D point cloud generation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2837–2845 (2021)
38. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4195–4205 (2023)
39. Sanghi, A., Khani, A., Reddy, P., Rampini, A., Cheung, D., Malekshan, K.R., Madan, K., Shayani, H.: Wavelet latent diffusion (wala): Billion-parameter 3d generative model with compact wavelet encodings. arXiv preprint arXiv:2411.08017 (2024)
40. Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., Měch, R.: Parsenet: A parametric surface fitting network for 3d point clouds. In: European Conference on Computer Vision (ECCV). pp. 261–276 (2020)
41. Shen, Z., Zhao, M., Yan, D.M., Wang, W.: Mesh2brep: B-rep reconstruction via robust primitive fitting and intersection-aware constraints. IEEE Transactions Visualization & Computer Graphics (2025)
42. Siddiqui, Y., Alliegro, A., Artemov, A., Tommasi, T., Sirigatti, D., Rosov, V., Dai, A., Nießner, M.: Meshgpt: Generating triangle meshes with decoder-only transformers. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 19615–19625 (2024)
43. Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K., et al.: Lion: Latent point diffusion models for 3d shape generation. Conference on Neural Information Processing Systems (NeurIPS) **35**, 10021–10039 (2022)
44. Varady, T., Benkő, P., Kós, G., Renner, G., Weiß, V.: Segmentation and surface fitting in reverse engineering: Keynote paper. In: Machining Impossible Shapes: IFIP TC5 WG5. 3 International Conference on Sculptured Surface Machining (SSM98) November 9–11, 1998 Chrysler Technology Center, Michigan, USA. pp. 167–172. Springer (1999)
45. Wei, S.T., Wang, R.H., Zhou, C.Z., Chen, B., Wang, P.S.: Octgpt: Octree-based multiscale autoregressive models for 3d shape generation. In: SIGGRAPH 2025 conference papers. pp. 1–11 (2025)
46. Weiler, K.: Topological structures for geometric modeling. Technical report RPI, Center for Interactive Computer Graphics, University Microfilms (1986)
47. Wu, R., Xiao, C., Zheng, C.: Deepcad: A deep generative network for computer-aided design models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2021)
48. Wu, S., Lin, Y., Zhang, F., Zeng, Y., Yang, Y., Bao, Y., Qian, J., Zhu, S., Cao, X., Torr, P., et al.: Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. Conference on Neural Information Processing Systems (NeurIPS) (2025)
49. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. arXiv preprint arXiv:2412.01506 (2024)
50. Xie, H., Ju, F.: Text-to-cadquery: A new paradigm for cad generation with scalable large model capabilities (2025), `https://arxiv.org/abs/2505.06507`

51. Xiong, B., Wei, S.T., Zheng, X.Y., Cao, Y.P., Lian, Z., Wang, P.S.: Octfusion: Octree-based diffusion models for 3d shape generation. In: Computer Graphics Forum. vol. 44, p. e70198 (2025)
52. Xu, X., Jayaraman, P., Lambourne, J., Liu, Y., Malpure, D., Meltzer, P.: Autobrep: Autoregressive b-rep generation with unified topology and geometry. In: Proceedings of the SIGGRAPH Asia 2025 Conference Papers. pp. 1–12 (2025)
53. Xu, X., Jayaraman, P.K., Lambourne, J.G., Willis, K.D., Furukawa, Y.: Hierarchical neural coding for controllable cad model generation. arXiv preprint arXiv:2307.00149 (2023)
54. Xu, X., Lambourne, J., Jayaraman, P., Wang, Z., Willis, K., Furukawa, Y.: Brepgen: A b-rep generative diffusion model with structured latent geometry. ACM Transactions on Graphics (TOG) **43**(4), 1–14 (2024)
55. Xu, X., Willis, K.D., Lambourne, J.G., Cheng, C.Y., Jayaraman, P.K., Furukawa, Y.: Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In: International Conference on Machine Learning. pp. 24698–24724. PMLR (2022)
56. Yan, S., Yang, Z., Ma, C., Huang, H., Vouga, E., Huang, Q.: Hpnet: Deep primitive segmentation using hybrid representations. In: IEEE International Conference on Computer Vision (ICCV). pp. 2753–2762 (2021)
57. Zhang, B., Tang, J., Niessner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. Proceedings of SIGGRAPH (2023)
58. Zhang, C., Polette, A., Pinquié, R., Carasi, G., De Charnace, H., Pernot, J.P.: ecadnet: Editable parametric cad models reconstruction from dumb b-rep models using deep neural networks. Comput. Aided Des. **178**(C) (Jan 2025). `https://doi.org/10.1016/j.cad.2024.103806`, `https://doi.org/10.1016/j.cad.2024.103806`
59. Zhang, F., Jayaraman, P.K., Xu, X., Furukawa, Y.: B-rep distance functions (br-df) how to represent a b-rep model by volumetric distance functions? arXiv (2025)
60. Zhao, Z., Lai, Z., Lin, Q., Zhao, Y., Liu, H., Yang, S., Feng, Y., Yang, M., Zhang, S., Yang, X., et al.: Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. arXiv preprint arXiv:2501.12202 (2025)
61. Zheng, X.Y., Pan, H., Wang, P.S., Tong, X., Liu, Y., Shum, H.Y.: Locally attentional sdf diffusion for controllable 3d shape generation. Proceedings of SIGGRAPH pp. 1–13 (2023)