GEM3D: GEnerative Medial Abstractions for 3D Shape Synthesis

DMITRY PETROV, University of Massachusetts Amherst, USA
PRADYUMN GOYAL, University of Massachusetts Amherst, USA
VIKAS THAMIZHARASAN, University of Massachusetts Amherst, USA
VLADIMIR G. KIM, Adobe Research, USA
MATHEUS GADELHA, Adobe Research, USA
MELINOS AVERKIOU, CYENS Centre of Excellence and University of Cyprus, Cyprus
SIDDHARTHA CHAUDHURI, Adobe Research, USA
EVANGELOS KALOGERAKIS, University of Massachusetts Amherst and CYENS Centre of Excellence, USA

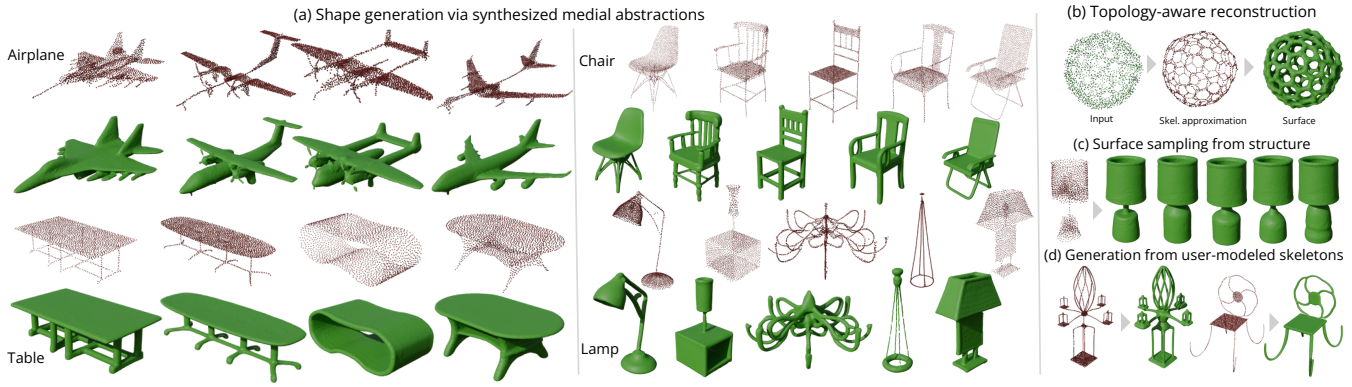

Fig. 1. We present GEM3D, a neural generative model able to (a) generate skeleton-based representations, or medial abstractions (in red), and 3D surfaces (in green) conforming to the structure and topology information encoded in the skeletons. Our method can be used in various applications, such as (b) reconstructing topologically complex surfaces, (c) synthesizing diverse shape surface samples conforming to a given skeleton, or even (d) synthesizing surfaces from user-specified skeletons far from the ones observed during training.

We introduce GEM3D – a new deep, topology-aware generative model of 3D shapes. The key ingredient of our method is a neural skeleton-based representation encoding information on both shape topology and geometry. Through a denoising diffusion probabilistic model, our method first generates skeleton-based representations following the Medial Axis Transform (MAT), then generates surfaces through a skeleton-driven neural implicit formulation. The neural implicit takes into account the topological and geometric information stored in the generated skeleton representations to yield surfaces that are more topologically and geometrically accurate compared to previous neural field formulations. We discuss applications of our method in shape synthesis and point cloud reconstruction tasks, and evaluate our method both qualitatively and quantitatively. We demonstrate significantly more faithful surface reconstruction and diverse shape generation results compared to the state-of-the-art, also involving challenging scenarios of reconstructing and synthesizing structurally complex, high-genus shape surfaces from Thingi10K and ShapeNet.

1 INTRODUCTION

Automated or semi-automated 3D shape synthesis is a significant and challenging problem in geometric modeling, with wide-ranging applications to computer-aided design, fabrication, architecture, art, and entertainment. While early work in this space primarily focused on handwritten models [Merrell et al. 2010; Musialski et al. 2013; Prusinkiewicz and Lindenmayer 1990], subsequent work employed statistical learning to infer generative design principles from data [Fisher et al. 2011; Kalogerakis et al. 2012]. In recent years, a variety of approaches have developed deep neural network-based architectures for 3D synthesis [Chaudhuri et al. 2020; Patil et al. 2020; Shi et al. 2023; Xu et al. 2023]. While these methods can capture diverse macro-level appearances, they rarely model shape structure or topology explicitly, relying instead on the representational power of the network to generate plausible-looking voxel grids [Liu et al. 2017], point clouds [Achlioptas et al. 2018a], meshes [Dai and Nießner 2019], or implicit fields [Chen and Zhang 2019]. Since 3D networks are hampered by the additional resource overheads incurred by the extra dimension compared to 2D image generation networks, they often struggle to model fine detail and connectivity. Some approaches do model part layouts [Li et al. 2017], but are limited in the complexity of the structures they can generate.

At the same time, these prior 3D synthesis methods rarely give artists flexible, precise control. They act more as black boxes for unconditional generation, or reconstruction from images or 3D scans. Recent methods introduce synthesis based on text prompts [Lin et al. 2023; Poole et al. 2023], with remarkable results but only high-level, global control via prompt engineering. 3D character artists have long been accustomed to posing skeleton rigs for accurate character configuration. However, such direct local control and interpretability through intuitive abstractions has had limited success in general 3D shape synthesis. Approaches without explicit structural modeling lack the ability to specify a particular desired topology, e.g.

a chair with a particular slat configuration in the back. On the other hand, approaches that do model part-level structure are restricted to simple topologies defined by assemblies of a few coarse primitives and cannot model e.g. complex fretwork or ornamentation.

We are interested in realistic 3D shape generation that accurately models complex topological and geometrical details, and supports more interpretable control of shape structure and geometry. To achieve this, we build upon three key insights: (1) topological detail can often be captured in a "skeletal abstraction", like that obtained by a medial axis transform [Tagliasacchi et al. 2016], which serves as a simplified structural proxy for the shape, even in the absence of meaningful part decompositions; (2) these abstractions can be synthesized with generative methods [Karras et al. 2022], predicted from sparse point clouds [Nie et al. 2020; Yin et al. 2018], or created manually by an artist, and need not be perfect since they are simply intermediate representations; and (3) each abstraction can be decoded into realistic surfaces by another trained model.

Our approach implements the surface generation step by inferring and assembling a collection of locally-supported neural implicit functions, conditioned on the skeletal abstraction. We draw inspiration from recent work in this area which associates a latent code with each 3D point in a sparse set, and generates a local implicit from a latent grid [Zhang et al. 2022]. The mixture of these implicits defines the overall synthesized shape, and allows for better generation of fine geometric details than a single large implicit. However, the sparse set of point supports in prior work tends to be arbitrary and not very interpretable. Follow-up work based on 3D neural fields and cross-attention [Zhang et al. 2023] drops explicit spatial grounding on the latent grids altogether. In contrast, our skeleton-based latent grids are more structure-aware, providing more interpretable supports for latent codes in 3D space, while remaining capable of representing complex, fine-grained topological structure.

We summarize our contributions as follows:

- We propose a generative model based on diffusion to automatically synthesize skeleton-based shape representations along with their supporting latents encoding shape information. The model's training procedure is performed without any form of user input or manual tuning.
- We also devise a neural implicit representation that can be used to accurately regress the shape surface from a skeletal representation and associated latent field.
- According to our experiments, our method produces significantly more faithful surface reconstruction and diverse shape generation results compared to the state-of-the-art. Our method handles challenging scenarios of reconstructing and synthesizing structurally complex, high-genus shape surfaces from Thingi10K and ShapeNet (Figure 1), including synthesizing surfaces from user-specified skeletons largely different from ones observed during training.

2 RELATED WORK

Skeletonization. 3D skeletonization, the computation of medial skeletons from a surface representation, is a well-studied problem in geometry processing. For a full discussion, we refer readers to the survey of Tagliasacchi et al. [2012]. Skeletons, 1D curves, 2D sheets or a combination of them, have been used as an intermediate representation for creating complete shapes. Traditional optimization strategies have been proposed via property-grouping and sinking consolidation to extract skeletons [Tagliasacchi et al. 2009; Wu et al. 2015]. More recently, Yin et al. [2018] proposed a neural network that can translate surface point clouds to skeletal point clouds, and Nie et al. [2020] followed up by regressing complete skeletons from partial surfaces. Clemot et al. [2023] fitted an implicit neural representation to surface points and extracted skeletons as the inner points with minimum SDF value that lie along the opposite direction to the gradient. On the other hand, our work is focused not only on computing those representations, but also on creating efficient *shape generation* models grounded on medial abstractions. While we also explore how to estimate such abstractions from point clouds, we primarily demonstrate how these intermediate representations can be used to guide the generation process, leading to more interpretable and topologically faithful shapes.

Skeleton-Guided Surface Representation. 3D shapes can be approximated as unions of spheres. For instance, a surface can be reconstructed by piecewise linear interpolation of medial spheres centered on skeleton points [Li et al. 2016]. Shape modeling tools like ZBrush [Pixologic 2022] allow artists to manually create such a skeleton of spheres via a feature called *ZSpheres*. *Sphere Meshes* [Thiery et al. 2013] used this representation for automatic shape approximation and extended it to non-tubular geometry. *Convolution Surfaces* [Bloomenthal and Shoemake 1991] are another class of skeleton-based implicit surfaces where the surface is defined as the level set of a function obtained by integrating a kernel function along the skeleton. To avoid blending artifacts, Zanni et al. [2013] proposed scale-invariant blending. Our work investigates using such abstractions to guide the generative process for the best of two worlds – efficient topological representation and interpretability from medial abstractions, and detailed surface generation from data-driven neural fields.

We are inspired by Yin et al. [Yin et al. 2018] and Nie et al. [2020], who developed neural networks that translate (complete and partial, respectively) surface point clouds to meso-skeletons, and back to surface points from which continuous surfaces can be extracted with Poisson reconstruction. In contrast, we focus on the generation rather than the regression task, with consequent major differences in the technical design and applications. We also develop a novel neural surface representation based on skeleton-supported local implicits that directly yields a continuous surface, and is designed to be efficient, detailed, and artifact-free.

Neural Fields. Scalar fields parameterized by neural networks have been frequently used as a shape representation in many deep generative models [Chen and Zhang 2019; Mescheder et al. 2019; Park et al. 2019]. When compared to other representations like voxels or meshes, those *neural fields* are capable of representing shapes with varying topology with reasonable amount of detail without requiring prohibitive memory footprint. More recent approaches encode spatially-varying features that locally modulate the neural fields to improve the accuracy of the generated shapes [Peng et al. 2020; Zhang et al. 2022, 2023]. Unfortunately, those approaches can

generate topologically incorrect and hard to control shapes. Our approach addresses both of these issues by representing shapes as neural fields conditioned on medial abstractions. More specifically, we learn scalar fields that encode a "thickening" operation on a skeletal representation, that naturally lead to more interpretable, controllable and topologically-faithful shapes.

Structure-Aware 3D Generation. A full review of 3D generative models is beyond our scope: we refer the reader to excellent surveys [Shi et al. 2023; Xu et al. 2023]. Methods that directly generate "raw" representations such as voxels, meshes, or point clouds frequently yield topological artifacts, omissions, and other inaccuracies in regions with thin/fine details. Further, they lack interpretable intermediates to help users control the generative process. Structure-aware methods [Chaudhuri et al. 2020], in particular part-based models (e.g., [Li et al. 2017]), have potentially higher topological fidelity and interpretability. However, they are currently suitable for shapes with small numbers of meaningful parts, and not for more topologically complex structures with ambiguous part decompositions. We use medial abstractions as a structure-aware intermediate representation to avoid these limitations.

3 METHOD

At the heart of our method lies a neural generative model (Figure 2) that first generates a skeletal representation of a shape dedicated to compactly capturing its topology. Then, conditioned on this skeletal representation, our generative model synthesizes the surface in the form of an implicit function locally modulated by the skeletal data. The skeletal representation approximates the medial axis of a 3D shape, which has widely been used to capture shape topology. In the following section, we briefly overview preliminaries, then we discuss our neural implicit function and generative approach.

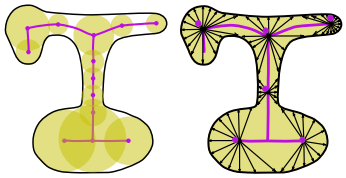

Fig. 3. A shape and its medial axis (purple) (a) medial balls (yellow) vs (b) enveloping primitives [Guo et al. 2023].

3.1 Preliminaries

Medial Axis Transformation. Consider a closed, oriented, and bounded shape S in $\mathbb{R}^3$, the medial axis is represented as a set of centers of maximal spheres inscribed within the shape (Figure 3, left). Each of these medial spheres is tangent to at least two points on the boundary ∂S of S and does not contain any other boundary points in its interior. The medial axis transformation (MAT) comprises both the medial axis and the radius associated with each sphere center [Tagliasacchi et al. 2016]. The MAT can be used to compactly capture both topological shape information and its geometry [Li et al. 2016], yet as also recently noted in [Guo et al. 2023], its ability to capture local surface details, such as shape protrusions and other high curvature regions, is limited, since a substantial number of medial spheres are often needed (Figure 3, left).

Enveloping Primitives. To circumvent the above limitation, Guo et al. [2023] introduced an implicit function, called "generalized enveloping primitive", which represents the shape's surface as a directional distance field around the medial axis. Specifically, given a query point $\mathbf{x}$ in $\mathbb{R}^3$ and a medial mesh S discretizing the medial axis of a given shape, their enveloping function outputs a signed distance value for query points depending on their closest medial elements (medial mesh vertex, edge, or face in their case):

$$E_S(\mathbf{x}) = ||\mathbf{x} - \mathbf{s}_\mathbf{x}|| - r(\mathbf{d}_{\mathbf{s},\mathbf{x}}) \tag{1}$$

where $\mathbf{s}_x$ is the closest medial mesh element to the query point $\mathbf{x}$ in Euclidean sense, $\mathbf{d}_{\mathbf{s},\mathbf{x}} = (\mathbf{x} - \mathbf{s}_\mathbf{x})/\|\mathbf{x} - \mathbf{s}_\mathbf{x}\|$ represents a unit direction vector from the closest medial mesh element towards the query point, and $r(\cdot)$ is a radius function. The radius function essentially defines an envelope, or displacement, around the medial axis, which is modulated differently depending on different directions around it (Figure 3, right), providing much better surface approximation compared to using medial spheres. Guo et al. [2023] estimates the above enveloping function for a given input 3D shape through a global optimization and iterative refinement procedure.

3.2 Surface generation

Neural enveloping. We extend the above primitives to "neural envelopes", such that can be used in our neural generative model. We employ a neural network, parameterized by learnable parameters $\boldsymbol{\theta}$, to approximate the radius function. In our case, the radius function depends not only on the direction from a medial element to the query point but also a medial latent code, which is specific to the medial element and aims to encode surface information around the medial axis. The latent code helps decoding towards a more accurate surface, since it is trained to encode shape information. In addition, instead of finding the medial element closest to the query point, as done in [Guo et al. 2023], we found that a significantly better surface approximation can be achieved by finding the medial element whose surface envelope is closest to the query point (Figure 4, see also our ablation). Specifically, our neural enveloping function defines the following implicit:

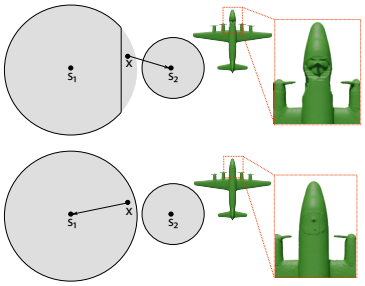

Fig. 4. (Top) Using the closest medial point for queries yields wrong ball reconstructions. (Bottom) Using the closest envelope yields the right result. Surface reconstruction is shown in green for both cases.

$$F(\mathbf{x};\boldsymbol{\theta}) = \min_i \left(||\mathbf{x} - \mathbf{s}_i|| - r(\mathbf{d}_{i,\mathbf{x}}, \mathbf{z}_i; \boldsymbol{\theta}) \right) \tag{2}$$

where $\mathbf{s} = \{\mathbf{s}_i\}_{i=1}^N$ is a set of N medial element positions in $\mathbb{R}^3$ produced by our generative model, $\mathbf{z} = \{\mathbf{z}_i\}_{i=1}^N$ are corresponding latent codes (256-dimensional in our implementation) also produced by our generative model, and $\mathbf{d}_{i,\mathbf{x}} = (\mathbf{x} - \mathbf{s}_i)/\|\mathbf{x} - \mathbf{s}_i\|$ are unit vectors from each medial element towards the query point.

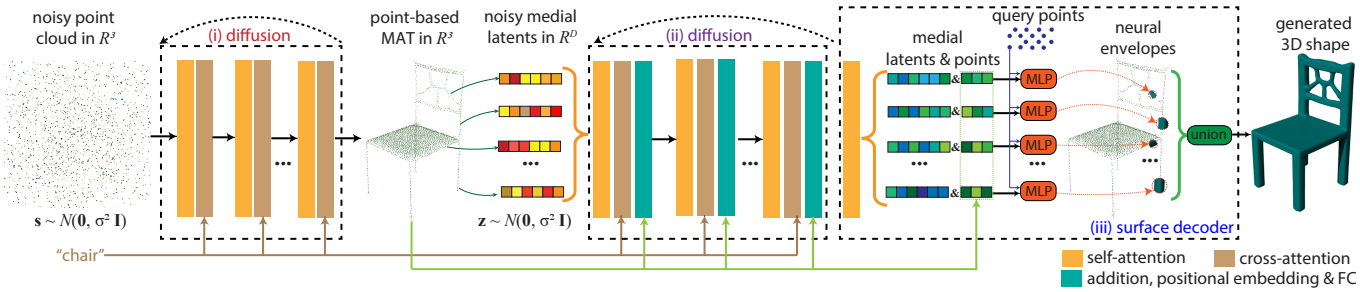

Fig. 2. GEM3D generative architecture: starting with Gaussian noise in $\mathbb{R}^3$, our first diffusion stage generates a point-based medial (skeletal) shape representation conditioned on a shape category embedding. Conditioned on this representation, our second diffusion stage generates latent codes capturing shape information around the medial points. In the last stage, our surface decoder decodes the medial latent codes and points to local neural implicit surface representations, which are then aggregated to create an output 3D shape.

The neural network function r has the form of a multi-layer MLP. Given the values of the implicit function extracted at a dense grid of query points (256^3 in our implementation), the zero-level iso-surface can be extracted using the marching cubes algorithm [Lorensen and Cline 1987]. Note that evaluating the Eq. 2 is computationally expensive given that it requires evaluating the min over the envelopes of all queries and medial elements. In practice, to accelerate computation, we sample a dense number of unit directions (1000 in our implementation) around each medial element, estimate the largest value for the radius across all directions for each medial element i.e., form a sphere bounding the envelope of each medial element, then compute the min over only the medial elements whose bounding spheres include the query point. In this manner, we find for each query point, the medial elements that approximately lie in the vicinity of the query point.

Training. Given the surfaces of training shapes in a dataset along with their medial elements, the parameters of the MLP can be trained to minimize the L_1 loss measuring error between predicted locations of sample points and the location of training surface sample points sampled at different directions around each medial element:

$$L_{surf}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{D}(i)|}\sum_{d\in\mathcal{D}(i)}\|\hat{\mathbf{x}}_{i,d} - \mathbf{x}_{i,d}\|_1, where \quad (3)$$

$$\mathbf{x}_{i,d} = \mathbf{s}_i + \mathbf{d}_i \cdot r(\mathbf{d}_i, \mathbf{z}_i; \boldsymbol{\theta}) \quad (4)$$

where $\mathcal{D}(i)$ represents a set of training surface points found by casting rays from each medial element i along several sample directions (1000 in our implementation), $\hat{\mathbf{x}}_{i,d}$ is the 3D location of each training surface point, and $\mathbf{d}_i$ is a sample unit direction vector. The loss is averaged over all shapes of the training datasets. Note that in contrast to other neural implicit surface formulations that often use point samples all over in $\mathbb{R}^3$ to avoid trivial solutions (i.e., zero implicit values everywhere), our MLP only needs surface sample points to be trained on.

3.3 Generative model

Central to our approach is the generation of medial elements along with their latent codes. Our generative model proceeds in two stages: first, we generate the positions of medial elements through a denoising diffusion process [Ho et al. 2020; Karras et al. 2022], then we generate their latent codes conditioned on their position through another subsequent diffusion process. Our two-stage process allows the generation of a multitude of different shapes conforming to the same skeleton structure, including user-specified skeletons, as discussed in our results and applications.

Generation of medial elements. Our first stage synthesizes a point-based representation of the medial axis i.e., a simplified skeleton form including only points as medial elements (as opposed to a mesh). The diffusion process starts by sampling N 3D point positions from Gaussian noise $\mathbf{s} \sim \mathcal{N}(0, \sigma_{\max}^2\mathbf{I})$, where $\sigma_{\max}$ is an initially prescribed standard deviation. Then an iterative denoising procedure is initiated to synthesize a point-based skeleton. The point denoising is executed by iteratively solving a probability flow ODE [Karras et al. 2022] in a series of time steps: $d\mathbf{s} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{s}}\log p(\mathbf{s};\sigma(t))dt$ where $\sigma(t)$ is a schedule defining the desired noise level at time t, $\dot{\sigma}(t)$ is its derivative, and $\nabla_{\mathbf{s}}\log p(\mathbf{s};\sigma(t))dt$ is the score function for diffusion models [Song et al. 2021]. Following this vector field nudges the sample towards areas of higher density of the data distribution i.e., the distribution over plausible point-based skeletons in our case. The vector field is approximated with the help of a denoiser neural network. The network takes as input: (i) a noisy skeleton sample $\mathbf{s}$ consisting of 2048 noisy points in $\mathbb{R}^3$ in our implementation, (ii) the noise level σ_t at a time step t, (iii) and a learnable embedding vector $\mathbf{c}$ representing a desired category of shapes (e.g., 55 different embedding vectors for 55 categories in ShapeNet). The embedding is used for category-conditioned generation i.e., generate skeletons conditioned on a desired category, such as "airplanes". With the help of the denoiser network, the diffusion process outputs a skeletal approximation consisting of 2048 sample points. The denoiser network consists of a set of blocks each producing a feature representation for each medial point based on self-attention and cross-attention operations [Vaswani et al. 2017; Zhang et al. 2023]. Specifically, each block performs the following operations:

$$\{\mathbf{f}_i^{(l)}\}_{i=1}^N = \text{SelfAttn}\big(\{\mathbf{f}_i^{(l-1)}\}_{i=1}^N\big) \quad (5)$$

$$\{\mathbf{f}_i'^{(l)}\}_{i=1}^N = \text{CrossAttn}\big(\{\mathbf{f}_i^{(l)}\}_{i=1}^N, \mathbf{c}\big) \quad (6)$$

where $\mathbf{f}_i^{(l)}$, $\mathbf{f}_i'^{(l)}$ are feature vectors (256-dimensional in our implementation) for each medial point computed after self attention and cross attention respectively from the block l. The first block uses as features the current sample positions per medial point. The noise level is taken into account the network through σ-dependent skip

connections [Karras et al. 2022]. We refer to the same paper for more details on noise variance scheduling and hyperparameters.

Training of the medial point denoiser network. The parameters $\boldsymbol{\phi}_1$ of denoising network $D_1(\mathbf{s}, \sigma_t, \mathbf{c}; \boldsymbol{\phi}_1)$ are trained by minimizing the L_2 denoising loss for samples drawn from training skeletons:

$$L_{\text{diff},1}(\boldsymbol{\phi}_1) = \mathbb{E}_{\hat{\mathbf{s}}\sim p_{\text{data}}} \mathbb{E}_{\mathbf{n}\sim\mathcal{N}(\mathbf{0},\sigma_t^2\mathbf{I})} \|D_1(\hat{\mathbf{s}} + \mathbf{n}, \sigma_t, \mathbf{c}; \boldsymbol{\phi}_1) - \mathbf{s}\|_2^2 \quad (7)$$

where $\hat{\mathbf{s}}$ are training point-based skeletons and $\mathbf{n}$ is sampled Gaussian noise. The training requires obtaining reference point-based skeletons for shapes. We developed a fast, distance field-based skeletonization algorithm whose basic idea is to move surface sample points set towards the negative gradient of the signed distance function through an iterative gradient descent procedure such that surface shrinks and moves towards the medial axis. We provide more details about our skeletonization in the supplementary material. We stress that the skeletonization was performed fully automatically for all our datasets without any manual parameter tuning per shape category or testing scenario.

Generation of medial latents. Given a generated point-based skeleton $\mathbf{s} = \{\mathbf{s}_i\}_{i=1}^N$ from the first diffusion model, or a point-sampled skeleton provided as input for skeleton-based shape synthesis, our second diffusion stage aims at generating a set of latent vectors $\mathbf{z} = \{\mathbf{z}_i\}_{i=1}^N$. Each latent vector $\mathbf{z}_i$ is generated such that it corresponds to the medial point $\mathbf{s}_i$. This diffusion stage proceeds in a similar manner to the first one. We start by sampling N 3D latent vectors from Gaussian noise, then these are iteratively denoised by iteratively solving the same probability flow ODE [Karras et al. 2022] as in our first stage (i.e., substituting medial points with medial latents). An important difference is that the denoiser network used to compute the score function is additionally conditioned on the medial point positions so that the network outputs latents tailored to each medial point. Specifically, the denoiser network $D_2(\mathbf{z}, \mathbf{s}, \sigma_t, \mathbf{c}; \boldsymbol{\phi}_2)$ consists of a set of blocks implementing the self-attention and cross-attention operations with the input category embedding, similarly to Eq. 5 and 6 respectively. As an additional operation, each block adds a positional embedding to the input feature vector $\mathbf{h}_i$ of each medial point based on its position: $\mathbf{h}_i'^{(l)} = \mathbf{h}_i^l + g(\mathbf{s}_i)$, where g represents a frequency-based positional embedding of the medial point positions [Sitzmann et al. 2020], followed by a fully connected layer. We observed that adding this positional embedding offered significantly better reconstruction results, since it made each latent aware of its corresponding medial position represented at different frequencies. The feature representations of the last block is finally processed through a block of self-attention layers to exchange the information within the latent set – this last block produces the final latents used in our surface decoder (Eq. 2).

Training of the medial latent denoiser network. The parameters $\boldsymbol{\phi}_2$ of denoising network $D_2(\mathbf{z}, \mathbf{s}, \sigma_t, \mathbf{c}; \boldsymbol{\phi}_2)$ are trained by minimizing the expected L_2 denoising error loss for training latents:

$$L_{\text{diff},2}(\boldsymbol{\phi}_2) = \mathbb{E}_{\hat{z}\sim p_{\text{data}}} \mathbb{E}_{\mathbf{n}\sim\mathcal{N}(\mathbf{0},\sigma_t^2\mathbf{I})} \|D_2(\hat{\mathbf{z}} + \mathbf{n}, \sigma_t, \mathbf{c}; \boldsymbol{\phi}_2) - \mathbf{z}\|_2^2 \quad (8)$$

where $\hat{\mathbf{z}}$ are training skeletal latent codes. To provide these latent codes, we devised an autoencoder-based, unsupervised learning strategy where the training latents are estimated such that they yield an optimal surface reconstruction error for the training shapes. More specifically, in a pre-training step that aims to estimate latents for training shapes, we train an encoder with learnable parameters $\boldsymbol{\omega}$ that takes as input a set of dense surface sample points $\{\hat{\mathbf{x}}_k\}_{k=1}^K$ ($200K$ in our implementation) along with the training medial points $\{\hat{\mathbf{s}}_i\}_{i=1}^N$, encodes them into latent codes, then decodes them back to predict surface points:

$$\{\hat{\mathbf{z}}_i\}_{i=1}^N = \text{Encoder}(\{\hat{\mathbf{x}}_k\}_{k=1}^K, \{\hat{\mathbf{s}}_i\}_{i=1}^N; \boldsymbol{\omega}) \quad (9)$$

$$\{\mathbf{x}_k\}_{k=1}^K = \text{Decoder}(\{\hat{\mathbf{s}}_i\}_{i=1}^N, \{\hat{\mathbf{z}}_i\}_{i=1}^N; \boldsymbol{\theta}) \quad (10)$$

The encoder consists of cross-attention blocks that estimate features by taking into account both the surface samples and medial points so that the medial latents encode surface information. The decoder implements Eq. 2, i.e., it computes surface points based on the radius function, given medial points and latents. The encoder and decoder parameters are trained to minimize the surface loss of Eq. 3 along with a KL regularization loss, as commonly used in variational autoencoders [Kingma and Welling 2014].

4 RESULTS & APPLICATIONS

We now discuss the experiments and validation of our method for three different applications: category-conditioned shape generation, surface reconstruction from point clouds, and skeleton-driven shape generation. We note that all our code, data, and evaluation will be publicly released.

4.1 Category-driven shape generation

In this application, the input is a given category (e.g., "lamp", "chair", and so on) and the output is a sample set of 3D generated shapes from this category. Our method here makes use of the diffusion stages discussed in Section 3.3: the first stage generates a point-based MAT given the input category and the second stage generates the medial latents. Then our surface decoder generates the surface under the guidance of the medial points and latents (Section 3.2).

Baselines. We compare our method with the following recent neural 3D generative methods: the autoregressive transformer-based *3DILG* model [Zhang et al. 2022] and latent diffusion-based *3DS2VS* [Zhang et al. 2023]. For both competing methods, we use the source code provided by the authors.

Metrics. As generative evaluation metrics, we employ the metrics also used in 3DS2VS. First, we report the Maximum Mean Discrepancy based on Chamfer Distance (**MMD-CD**) and Earth Mover's Distance (**MMD-EMD**) – the MMD metrics quantify the *fidelity* of generated examples i.e., how well the generated shapes match the reference ShapeNet test splits. In addition, we report *coverage* measured as the fraction of the generated shapes matching the reference test splits in terms of Chamfer distance (**COV-CD**) and Earth Mover's Distance (**COV-EMD**). For implementation details of these metrics, we refer readers to [Achlioptas et al. 2018b] who introduced them in the context of 3D generation.

Metric	3DILG	3DS2VS	GEM3D (Ours)
MMD-CD ($\times 10^2$) (↓)	9.64	9.34	**8.64**
MMD-EMD ($\times 10^2$) (↓)	11.0	10.9	**10.1**
COV-CD ($\times 10^2$) (↑)	54.8	50.6	**58.5**
COV-EMD ($\times 10^2$) (↑)	53.1	49.9	**57.2**
precision (%) (↑)	73.1	72.4	**80.6**
recall (%) (↑)	79.4	80.3	**89.2**
PointBert-FID ($\times 10^{-2}$) (↓)	1.46	1.62	**1.25**
PointBert-KID ($\times 1$) (↓)	4.06	4.77	**2.54**

Table 1. Evaluation measures on category-conditioned shape generation. The measures are averaged over the 10 largest categories of ShapeNet. For evaluation per-category, please see the supplement.

In addition, as proposed in the literature of image generative models, we use the Fréchet Inception Distance and Kernel Inception Distance. While prior work [Zhang et al. 2022, 2023] relies on PointNet++ embeddings to evaluate these metrics, we use the 768-D embeddings taken by the more recent and more discriminative model of PointBert [Yu et al. 2022] trained on ShapeNetCore. We refer to these two metrics as **PointBert-FID** and **PointBert-KID**. Finally, we report generative **precision** and **recall** [Sajjadi et al. 2018] as alternative measures to assess how well generated data covers the reference test splits and vice versa. For precision and recall, the similarities are assessed via PointBert embeddings.

Experimental setup. We follow the experimental setup of 3DS2VS [Zhang et al. 2023], which was also demonstrated in the same application. We use the same dataset of ShapeNet-v2 [Chang et al. 2015] with the same splits. We use the same watertight ShapeNet meshes provided by [Zhang et al. 2022]. In constrast to 3DS2VS that evaluated category-conditioned generation in 2 categories, in our case we evaluate on the largest 10 categories from ShapeNet, namely: tables, cars, chairs, airplanes, sofas, rifles, lamps, watercrafts, benches, and speakers. To further improve our evaluation, we generate a much larger set of generated samples – for each method and category, we generate a number of samples equal to 3x the size of the ground truth test split per category, which is also consistent with what Achlioptas et al. [2018b] originally proposed.

Quantitative results. Table 1 shows quantitative evaluation of all competing methods for all above measures. The measures are averaged over the 10 largest categories of ShapeNet. For all metrics, our method outperforms the competing work demonstrating results of better fidelity, coverage, precision and recall. In addition, we observe that according to FID/KID, our method outputs samples whose feature distributions match better the reference splits. Our supplement includes per-category evaluation for all measures – again, for the majority of categories, GEM3D outperforms the competing methods for all metrics. In particular, we observe that the gap widens for categories containing shapes with thin or tubular-like parts, such as benches, rifles, airplanes, chairs, and tables.

Qualitative results. Figure 5 demonstrates generated samples from our method for various categories from ShapeNet. We observe that our method is able to generate structurally challenging patterns (e.g., grid-like patterns for chair backs), thin parts (e.g, wings for airplanes), tubular or generalized cylinder-like parts (e.g., lamp wires or other connecting pieces). Our teaser (Figure 1) shows additional results. For more sample visualizations, we refer to the supplement.

Timings. We note that our method takes approximately 30 seconds to generate one shape measured on a NVidia RTX A6000.

4.2 Surface reconstruction from point clouds

In this application, the input to our method is a sparse point cloud of a shape and the output is a reconstructed surface. Here, we also follow the "auto-encoding" setting also used in 3DILG [Zhang et al. 2022] and 3DS2VS [Zhang et al. 2023], where the point cloud is obtained after sampling the original input surface. In this autoencoding setting, we only use the second diffusion stage of our method i.e., given a point-based skeleton, the second stage generates the medial latents, then our surface decoder uses them to reconstruct the surface. To apply our method on this task, we need as input a point-sampled skeleton. To obtain such a skeleton, we use a variant of P2PNet [Yin et al. 2018], an encoder-decoder network that takes as input a point cloud of a shape's surface and converts it to a point-based skeleton representation. Compared to the original P2PNet, we replaced the original PointNet++ encoder [Qi et al. 2017] with the cross-attention-based shape encoder from 3DS2VS [Zhang et al. 2023], including its positional-based embedding functions. We train this P2PNet variant in the training split of ShapeNet. Then at test time, we use its output skeletons as input to our second diffusion stage. We emphasize that no manual input or supervision were needed to obtain the skeletons – P2PNet is trained in a unsupervised manner.

Baselines. We again compare our method with *3DILG* [Zhang et al. 2022] and 3DS2VS [Zhang et al. 2023], which were also applied in the same auto-encoding setting. All methods use the same number of latent codes (2048 codes).

Metrics. For evaluation purposes, we compare reconstructed shapes to their corresponding reference ("ground-truth") shapes, which the input point clouds were sampled from. Following 3DS2VS [Zhang et al. 2023], we use the (squared) Chamfer Distance (**CD**), Intersection Over Union (**IoU**) and F-Score (**F1**). Compared to 3DS2VS' protocol, we made the following changes: (i) for computing IoU, we use a higher-resolution grid of 256^3 instead of 128^3 to better characterize reconstruction of small-scale surface details, thin parts, and holes, (ii) for CD and F1, we use farthest point sampling instead or random point sampling to better cover small and thing parts – random sampling tends to miss such parts. In the supplement, we provide comparisons using the original 3DS2VS' protocol.

Experimental setup. Following [Zhang et al. 2022] and [Zhang et al. 2023], we use the same training and test split from ShapeNet-v2 to train and test our method respectively. Apart from ShapeNet, we also test all competing methods in a more challenging out-of-distribution testing scenario: after training on ShapeNet, we test all methods on Thingi-10K [Zhou and Jacobson 2016]. We note that the whole dataset is $10K$ test shapes, 4 times larger than the ShapeNet's test split. The Thingi10K shapes contain man-made objects that

Metric	Test dataset	3DILG	3D2VS	GEM3D (Ours)
CD (↓)	Shapenet (10 cat.)	1.69	1.32	**1.17**
CD (↓)	Shapenet (all cat.)	2.05	1.70	**1.38**
IOU (↑)	Shapenet (10 cat.)	90.1	**96.2**	95.9
IOU (↑)	Shapenet (all cat.)	90.5	95.4	**95.5**
F1 (↑)	Shapenet (10 cat.)	97.7	98.5	**99.3**
F1 (↑)	Shapenet (all cat.)	96.3	97.4	**98.7**

Table 2. Evaluation measures on point cloud surface reconstruction ("auto-encoding") in ShapeNet. The rows "ShapeNet (10 cat.)" report the measures averaged over the 10 largest categories of ShapeNet, while the rows "ShapeNet (all)" report the measures averaged over all 55 ShapeNet categories. For evaluation per-category, please see the supplement.

	Thingi10K	*#shapes*	3DILG	3D2VS	GEM3D (Ours)
CD ↓	all	9997	2.12	1.74	**1.68**
	$genus \geq 0$	5608	1.93	1.62	**1.61**
	$genus \geq 5$	1037	2.41	1.96	**1.78**
	$genus \geq 10$	489	2.88	2.30	**1.98**
	$genus \geq 20$	229	3.67	2.91	**2.32**
F1 ↑	all	9997	93.6	96.4	**96.7**
	$genus \geq 0$	5608	95.3	**97.4**	**97.4**
	$genus \geq 5$	1037	91.1	94.8	**95.9**
	$genus \geq 10$	489	87.5	92.3	**93.7**
	$genus \geq 20$	229	82.6	88.0	**90.4**

Table 3. Evaluation measures on point cloud surface reconstruction in the Thingi10K dataset. We note that none of the methods are trained on Thingi10K, or any subset of it. We report performance in terms of CD and F1 scores for shapes of increasing genus.

often possess highly complex topology (e.g., a large genus number). None of the Thingi10K shapes exists in ShapeNet. The vast majority of them do not even relate to any of the ShapeNet categories. Thingi10K also provides topological complexity information (genus) for a large subset of the dataset. We use this information as a proxy of shapes' topological complexity for additional evaluation.

Quantitative results. Table 2 shows quantitative evaluation on ShapeNet-v2. The odd rows report the measures averaged over the 10 largest categories of ShapeNet, and the even rows report the measures averaged over all 55 categories. In terms of the surface-based metrics of CD and F1 scores, our model is consistently better than the baselines. In terms of IoU, our model has comparable performance. The results reflect a more accurate surface reconstruction, without compromising volumetric (IoU) accuracy. The supplement reports the performance for each of the largest 10 categories – again, we outperform prior methods in most categories based on the surface-based metrics. The gap is particularly larger for topologically challenging categories, such as lamps and watercrafts.

The out-of-distribution evaluation of reconstruction on Thingi10K is shown in Table 3. Our model clearly outperforms the baselines – the performance gap increases for topologically challenging shapes with higher genus. We note that this evaluation does not include IoU since some Thingi10K ground-truth shapes are not watertight, thus, volume-based metrics cannot be accurately assessed.

Qualitative results. Figures 6 and 7 shows qualitative evaluation on ShapeNet-v2 and Thingi10K respectively. In particular, the Thingi10K results show that our method can better reconstruct surface topology e.g., the connectivity of different shape parts and their surface holes. We suspect that this is due to the ability of our method to capture structure and topology information through the neural medial representation, which in turn guides the reconstruction. In contrast, 3DILG and 3D2VS tend to oversmooth surface details, miss connections, and alter the topology in an unpredicted manner.

4.3 Skeleton-driven shape synthesis

An alternative application of our method is to generate surfaces driven by an input skeleton in the form of a point-based MAT. From a practical point of view, one possibility for users of our method is to execute our first diffusion stage, obtain various shape structures represented in their MATs, then select the one matching their target structure or topology goals more. Then the user can execute the second stage to obtain various shape alternatives conforming to the given structure. Figure 8 highlights several examples of this application scenario. We show a generated MAT, then diverse surfaces conforming to this MAT. For example, our method can generate diverse lamp stands from a table lamp structure or chairs of diverse part thicknesses conforming to a particular chair structure.

Moreover, instead of obtaining a MAT from our method, an alternative scenario is that the user provides such as input. We asked an artist to draw the skeletons of completely fictional objects using 3D B-spline curves and patches. Our method was still able to generate diverse surfaces conforming to these out-of-distribution skeletons, as shown in Figure 9.

4.4 Ablation studies

In our supplementary material, we show the impact of various choices in our method, including testing with different number of medial points and using closest medial points versus closest envolopes in our implicit function.

5 CONCLUSION

We presented a skeleton-centered generative model of 3D shapes. Our method first captures structure and topology in the form of a generated skeleton, then synthesizes the surface guided from it. Users can also provide their own modeled skeletons in the process to control shape synthesis.

Our method still has a number of limitations and offers avenues for further research. First, it relies on a simplified form of point-based skeletons with fixed resolution. Generalizing our method to create more expressive representations, such as skeletal diagrams [Guo et al. 2023], may further enhance the topology information encoded in our model. Despite the improvements in the topology of reconstructed shapes, our method does not guarantee certain topology characteristics; the surfaces might still mismatch a target topology (e.g., a certain genus) or contain undesired, noisy shape components. Finally, an interesting future avenue would be to enable interactive skeleton editing, exploration, and shape synthesis from the edits.

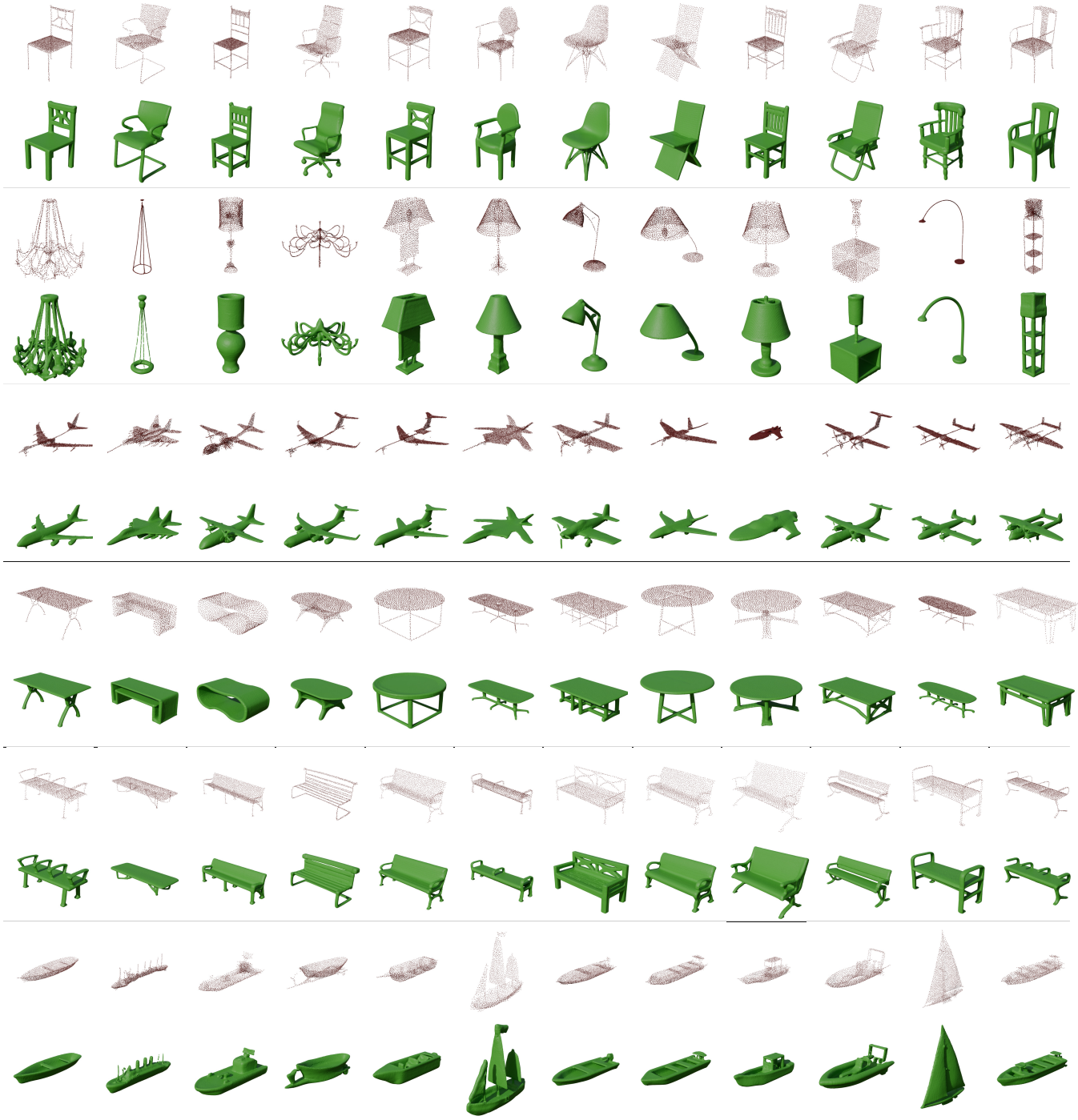

Fig. 5. **Category-conditioned shape generation on ShapeNet.** We show generated shapes from GEM3D for five categories: chair, lamp, airplane, table, bench and watercraft. Odd rows are skeletons generated by our model; even rows are surfaces sampled from them.

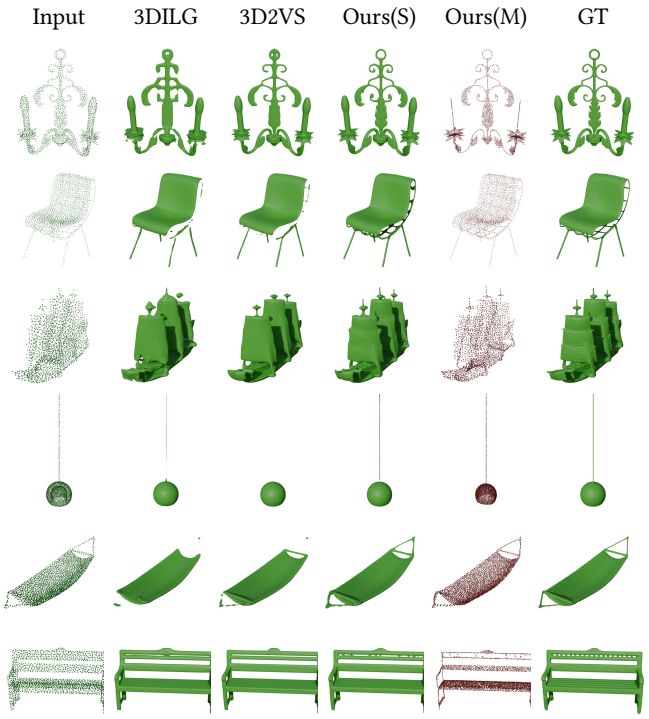

Fig. 6. **Point cloud reconstruction on ShapeNet.** GEM3D yields better reconstruction esp for thin and tubular parts, and with better connectivity.

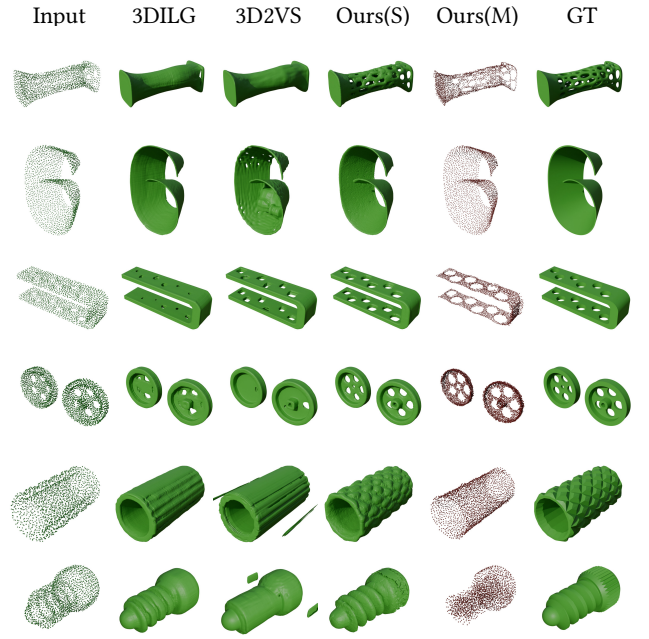

Fig. 7. **Point cloud reconstruction on Thingi10K.** In this out-of-distribution test setting our model is able to reconstruct topology (e.g., holes, connectivity) and surface details with less artifacts than prior work.

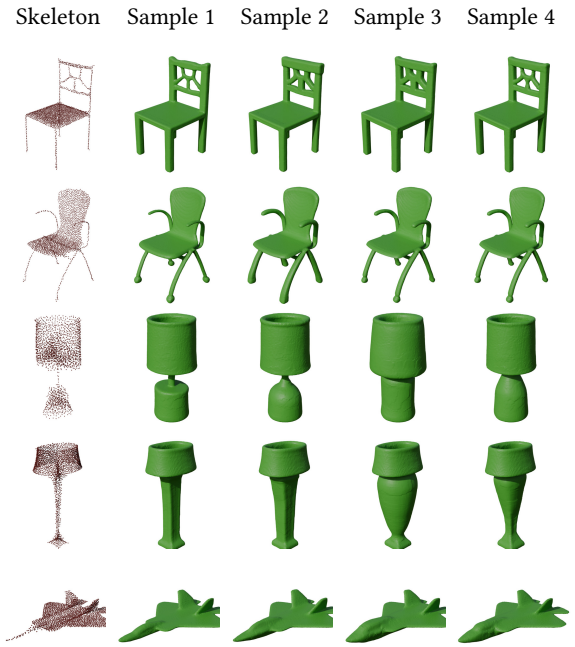

Fig. 8. **Surface sampling conditioned on skeleton.** Our model can sample diverse and structurally consistent surfaces from generated skeletons.

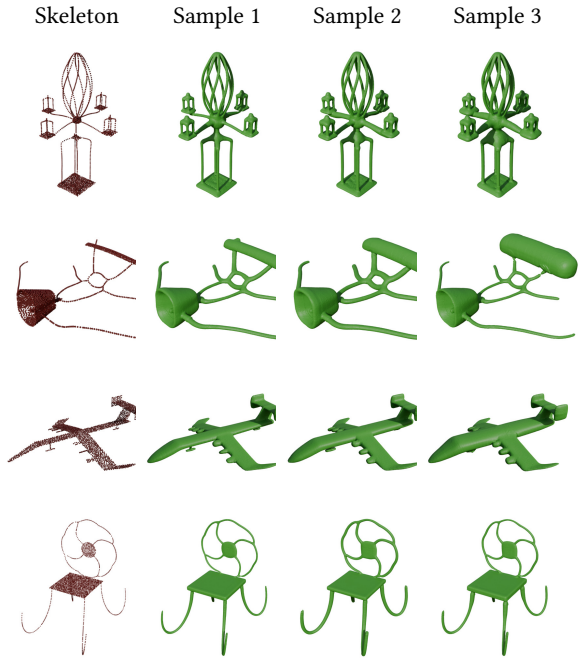

Fig. 9. **Surface generation based on user-modeled skeletons.** Our model is able to generalize to unseen structures encoded in user-provided skeletons, and produces plausible surface samples from them.

REFERENCES

P. Achlioptas, O. Diamanti, I. Mitliagkas, and L.J. Guibas. 2018a. Learning representations and generative models for 3D point clouds. In *Proc. ICML*.

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018b. Learning Representations and Generative Models for 3D Point Clouds.

Jules Bloomenthal and Ken Shoemake. 1991. Convolution surfaces. *Proc. ACM SIGGRAPH*, 251–256.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012* (2015).

Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao (Richard) Zhang. 2020. Learning Generative Models of 3D Structures. In *Computer Graphics Forum*.

Zhiqin Chen and Hao Zhang. 2019. Learning Implicit Fields for Generative Shape Modeling. *CVPR* (2019).

Mattéo Clémot and Julie Digne. 2023. Neural skeleton: Implicit neural representation away from the surface. *Computers & Graphics* 114 (2023), 368–378.

Angela Dai and Matthias Nießner. 2019. Scan2Mesh: From Unstructured Range Scans to 3D Meshes. In *Proc. CVPR*.

Matthew Fisher, Manolis Savva, and Pat Hanrahan. 2011. Characterizing structural relationships in scenes using graph kernels. In *ACM Transactions on Graphics*, Vol. 30. ACM, 34.

Minghao Guo, Bohan Wang, and Wojciech Matusik. 2023. Medial Skeletal Diagram: A Generalized Medial Axis Approach for Compact 3D Shape Representation. arXiv:2310.09395

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Proc. NIPS*.

Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. 2012. A Probabilistic Model for Component-Based Shape Synthesis. In *Proc. ACM SIGGRAPH*.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the Design Space of Diffusion-Based Generative Models. In *Proc. NIPS*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proc. ICLR*.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *Trans. Graphics* 36, 4 (2017).

Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2016. Q-MAT: Computing Medial Axis Transform By Quadratic Error Minimization. *ACM Transactions on Graphics* 35, 1 (2016).

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2023. Magic3D: High-Resolution Text-to-3D Content Creation. In *Proc. CVPR*.

J. Liu, F. Yu, and T. Funkhouser. 2017. Interactive 3D modeling with a generative adversarial network. In *Proc. 3DV*.

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Transactions on Graphics* 21, 4 (1987).

Paul Merrell, Eric Schkufza, , and Vladlen Koltun. 2010. Computer-Generated Residential Building Layouts. In *ACM Transactions on Graphics*.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3D reconstruction in function space. In *Proc. CVPR*.

Przemyslaw Musialski, Peter Wonka, Daniel G. Aliaga, Michael Wimmer, Luc van Gool, and Werner Purgathofer. 2013. A Survey of Urban Reconstruction. In *Computer Graphics Forum*.

Yinyu Nie, Yiqun Lin, Xiaoguang Han, Shihui Guo, Jian Chang, Shuguang Cui, and Jian Jun Zhang. 2020. Skeleton-bridged point completion: from global inference to local adjustment. In *Proc. NIPS*.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*.

Akshay Gadi Patil, Supriya Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, , and Hao Zhang. 2020. Advances in Data-Driven Analysis and Synthesis of 3D Indoor Scenes. In *Computer Graphics Forum*.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In *Proc. ECCV*.

Pixologic. 2022. *ZBrush*. https://www.maxon.net/en/zbrush

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. DreamFusion: Text-to-3D using 2D Diffusion. *ICLR* (2023).

Przemyslaw Prusinkiewicz and Aristid Lindenmayer. 1990. The Algorithmic Beauty of Plants.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. NIPS*.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lučić, Olivier Bousquet, and Sylvain Gelly. 2018. Assessing Generative Models via Precision and Recall. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zifan Shi, Sida Peng, Yinghao Xu, Andreas Geiger, Yiyi Liao, and Yujun Shen. 2023. Deep Generative Models on 3D Representations: A Survey. arXiv:2210.15663 [cs.CV]

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NIPS*.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *Proc. ICLR*.

A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang. 2012. Mean Curvature Skeletons. 31, 5 (2012).

Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum* 35, 2 (2016).

Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. 2009. Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics* 28 (2009).

Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-Meshes: shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics* 32, 6 (2013).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NIPS*.

Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. 2015. Deep points consolidation. *ACM Transactions on Graphics* 34 (2015).

Qun-Ce Xu, Tai-Jiang Mu, and Yong-Liang Yang. 2023. A survey of deep learning-based 3D shape generation. *Computational Visual Media* 9 (2023), 407–442. Issue 3.

Kangxue Yin, Hui Huang, Daniel Cohen-Or, and Hao Zhang. 2018. P2P-NET: Bidirectional Point Displacement Net for Shape Transform. *ACM Transactions on Graphics* 37, 4 (2018).

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. 2022. Point-bert: Pre-training 3D point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19313–19322.

C. Zanni, A. Bernhardt, M. Quiblier, and M.-P. Cani. 2013. SCALe-invariant Integral Surfaces. *Computer Graphics Forum* (2013).

Biao Zhang, Matthias Nießner, and Peter Wonka. 2022. 3DILG: Irregular Latent Grids for 3D Generative Modeling. In *Proc. NIPS*.

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 2023. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *ACM Transactions on Graphics* 42, 4 (2023).

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

SUPPLEMENTARY MATERIAL

A PER-CATEGORY EVALUATION

In this appendix, we discuss more detailed evaluation, including per-category evaluation for generation and reconstruction.

Shape generation. In terms of generative measures, we report precision and recall per category in Table 4, and PointBert-FID and PointBert-KID per category in Table 5. We also report the per-category MMD-CD, MMD-COV, COV-CD, COV-MD in Table 6. Overall, we observe that for the majority of categories, GEM3D outperforms the competing methods in terms of all metrics, especially in terms of generative recall. We hypothesize that is due to fact that our two step generative procedure allows model to learn better structural priors leading to generation of more structurally diverse shapes.

	Precision (↑)			Recall (↑)		
	3DILG	3DS2VS	Ours	3DILG	3DS2VS	Ours
table	76.0	74.0	**91.3**	74.0	77.4	**95.6**
car	48.5	**74.8**	73.5	61.6	73.5	**89.2**
chair	**91.7**	82.4	88.9	90.9	84.7	**93.2**
plane	82.2	74.4	**88.8**	78.8	86.4	**95.3**
sofa	64.8	**79.4**	78.6	82.5	88.9	**93.9**
rifle	69.3	69.6	**75.2**	71.0	74.9	**85.8**
lamp	**82.8**	50.7	69.1	**86.1**	72.2	74.4
w-craft	72.3	77.6	**84.4**	86.1	83.8	**87.1**
bench	71.6	70.4	**81.3**	84.5	79.2	**88.4**
speaker	71.7	71.0	**74.4**	78.9	81.5	**89.5**

Table 4. Per-category evaluation of generated samples based on precision and recall using PointBert as a backbone to assess shape similarity. We note that for each method we generate a number of samples equal to 3x the size of the ground truth test split per category.

	PointBert-FID ($\times 10^{-2}$) (↓)			PointBert-KID (↓)		
	3DILG	3DS2VS	Ours	3DILG	3DS2VS	Ours
table	1.57	1.53	**0.71**	6.87	7.49	**1.31**
car	1.39	0.98	**0.82**	7.68	3.56	**3.13**
chair	0.84	1.24	**0.79**	1.76	3.74	**1.20**
plane	0.90	1.06	**0.79**	2.10	4.08	**1.46**
sofa	1.29	1.07	**1.05**	3.37	**1.46**	1.96
rifle	1.52	1.51	**1.07**	6.53	5.87	**2.51**
lamp	**1.84**	3.81	2.88	**2.28**	15.2	9.32
w-craft	1.70	1.49	**1.34**	4.18	1.70	**0.86**
bench	1.70	1.75	**1.47**	2.84	2.71	**1.16**
speaker	1.85	1.79	**1.62**	2.95	**1.85**	2.50

Table 5. Per-category evaluation of generated samples based on FID and KID scores using PointBert as a backbone for feature extraction. We note that for each method we generate a number of samples equal to 3x the size of the ground truth test split per category.

Surface reconstruction from point clouds. For reconstruction, we report CD, IoU and F1 scores for each of the largest 10 categories of ShapeNet in Table 7. We also report performance using 512 latent codes vs 2048 latent codes for all methods. For most categories, GEM3D outperforms the competing methods especially in terms of the surface-based metrics (CD and F1 scores), while it offers comparable performance in terms of IoU. As expected, the performance is improved with more latent codes for all methods. As explained in our main text, this evaluation is done on a higher-resolution grid and with more uniform point-based surface sampling (furthest point sampling). In contrast, 3DShape2VecSet follows a different evaluation protocol based on a lower-resolution grid and random point sampling. We report the evaluation measures for reconstruction based on the original 3DShape2VecSet protocol on the categories reported in their paper in Table 8. We observe the same trends, yet, our gap with other methods slightly decreases given that this evaluation protocol is less sensitive to topological and surface details.

B ABLATION

We perform the following ablation studies:
(a) We tested using 2048 vs 512 different number of medial points and correspondingly, different number of latent codes. Results are evaluated in the task of surface reconstruction and are shown in Table 7. With more latents, our performance is improved, as expected.
(b) We tested using closest medial points vs closest medial envolope in our implicit function. Using closest medial envelopes yielded the best performance, as shown in Table 9.

ALGORITHM 1: Medial extraction algorithm

Input: Surface point samples $P = \{\mathbf{p}_i\}_{i=1}^{N}$
Parameters: Learning rate $\lambda = 0.1$, local shape diameter threshold $\tau_{\max} = 0.6$, number of neighbors $K = 20$ for kernel SDF estimation, kernel bandwidth $\sigma^2 = 0.002$, number of iterations $M = 50$
Result: Medial points $S = \{\mathbf{s}_i\}_{i=1}^{N}$
Initialize: For each surface point $\mathbf{p}_i \in P$ estimate local shape diameter function β_i through ray casting and its normal $\mathbf{n}_i$ from the original mesh; initialize medial points $\mathbf{q}_i = \mathbf{p}_i - \frac{1}{2}\beta_i \mathbf{n}_i$
for *iterations* $< M$ **do**
  For each $\mathbf{q}_i$ find K nearest neighbors $\mathbf{p}_{ij}$;
  Estimate local SDF $f(q_i) = \frac{\sum_j \alpha_{ij} f(\mathbf{p}_{ij})}{\sum_j \alpha_j}$, where:
  $\alpha_{ij} = \exp - \frac{||\mathbf{q}_i - \mathbf{p}_{ij}||^2}{\sigma^2}$ and
  $f(\mathbf{p}_{ij})$ is the signed distance of $\mathbf{q}_i$ to $\mathbf{p}_{ij}$'s tangent plane;
  Compute updated skeleton points $\hat{\mathbf{q}}_i = \mathbf{q}_i - \lambda \nabla_{\mathbf{q}_i} f(\mathbf{q}_i)$;
  if $||\hat{\mathbf{q}}_i - \mathbf{p}_i|| \leq \tau_{max}\beta_i$ **then**
    $\mathbf{s}_i \leftarrow \hat{\mathbf{q}}_i$
  end
end

C SKELETONIZATION

Skeletonization is a well-studied topic in the geometry processing literature [Tagliasacchi et al. 2016]. All skeletonization methods rely on different assumptions, approximation heuristics, and convergence criteria that might lead to different MAT approximations. We

	MMD-CD ($\times 10^2$) (↓)			MMD-EMD ($\times 10^2$) (↓)			COV-CD ($\times 10^2$) (↑)			COV-EMD ($\times 10^2$) (↑)		
	3DILG	3DS2VS	Ours	3DILG	3DS2VS	Ours	3DILG	3DS2VS	Ours	3DILG	3DS2VS	Ours
table	11.00	9.87	**9.01**	12.63	11.74	**10.67**	44.6	59.4	**65.0**	46.7	53.0	**62.9**
car	5.42	5.38	**5.07**	6.37	6.16	**5.92**	41.9	**44.0**	42.8	34.2	**46.6**	46.4
chair	11.89	12.44	**10.99**	12.86	13.66	**12.37**	**61.4**	51.9	61.3	**61.1**	48.8	56.0
airplane	5.13	5.60	**4.70**	6.67	7.23	**6.21**	56.2	46.5	**64.3**	61.3	47.0	**63.3**
sofa	12.2	9.7	**9.4**	12.51	10.63	**9.82**	43.2	**54.8**	53.1	40.3	**54.0**	53.8
rifle	5.27	5.35	**5.05**	6.83	6.86	**6.79**	**59.3**	40.4	55.9	**60.4**	44.3	53.9
lamp	**15.06**	15.93	15.32	**19.24**	20.88	19.69	**64.6**	41.7	51.0	**61.7**	43.4	49.2
w-craft	8.53	7.70	**7.16**	9.55	8.93	**8.33**	65.2	60.0	**71.8**	59.3	56.6	**65.9**
bench	9.68	9.31	**9.05**	10.72	10.39	**10.03**	**54.8**	51.8	54.0	53.3	52.5	**55.1**
speaker	12.27	12.05	**10.56**	12.57	12.66	**11.45**	57.3	55.7	**65.4**	52.3	53.1	**64.9**

Table 6. Per-category evaluation of generated samples based on MMD-CD, MMD-COV, COV-CD, and COV-MD. We note that for each method we generate a number of samples equal to 3x the size of the ground truth test split per category.

		3DILG	3D2VS	Ours	3DILG	3D2VS	Ours
	# latents	512	512	512	2048	2048	2048
CD ↓	table	1.38	1.05	**1.01**	1.36	1.03	**0.99**
	car	2.29	1.86	**1.24**	2.27	1.76	**1.18**
	chair	1.42	1.09	**1.05**	1.39	1.06	**1.03**
	airplane	0.98	**0.59**	**0.59**	0.97	**0.58**	**0.58**
	sofa	1.37	1.06	**1.02**	1.35	1.04	**1.00**
	rifle	0.94	0.47	**0.45**	0.92	0.46	**0.45**
	lamp	1.82	1.02	**0.72**	1.8	1.02	**0.72**
	w-craft	1.25	0.84	**0.71**	1.23	0.79	**0.69**
	bench	1.28	0.94	**0.84**	1.27	0.93	**0.82**
	speaker	1.8	1.46	**1.28**	1.77	1.42	**1.23**
IOU ↑	table	88.9	**95.9**	93.5	88.9	**96.2**	94.3
	car	92.9	**95.8**	94.8	93.1	**96.2**	95.7
	chair	90.5	**95.9**	94.8	90.6	**96.4**	95.3
	airplane	89.3	**96.5**	**96.5**	89.6	**96.9**	96.7
	sofa	95.1	**98.1**	97.5	95.2	**98.3**	97.8
	rifle	87.0	96.0	**96.5**	87.2	96.2	**96.6**
	lamp	85.6	94.0	**94.5**	86.5	**94.6**	93.9
	w-craft	90.8	96.4	**96.5**	91.0	96.7	**96.8**
	bench	85.7	**94.4**	94.0	86.0	94.7	**94.8**
	speaker	92.4	**95.9**	94.7	92.6	96.3	**97.7**
F1 ↑	table	99.0	99.4	**99.6**	99.1	99.5	**99.7**
	car	91.6	93.2	**96.5**	91.9	93.9	**97.4**
	chair	98.5	99.2	**99.4**	98.7	99.3	**99.4**
	airplane	99.6	**99.9**	**99.9**	99.6	99.9	**99.9**
	sofa	98.7	99.2	**99.5**	98.9	99.4	**99.6**
	rifle	99.7	99.9	**100.0**	99.7	**100.0**	**100.0**
	lamp	97.4	98.6	**99.5**	97.4	98.7	**99.4**
	w-craft	98.1	98.6	**99.7**	98.2	98.9	**99.8**
	bench	98.7	99.3	**99.7**	98.9	99.3	**99.7**
	speaker	95.9	96.9	**98.3**	96.2	97.4	**98.7**

Table 7. Evaluation of reconstructed surfaces on ShapeNet based on Chamfer Distance (CD), Intersection over Union (IoU), and F1 scores for the auto-encoding task. All numbers are scaled by 100.

tried various skeletonizations methods, including mean curvature flow skeletons [Tagliasacchi et al. 2012], medial skeletal diagrams [Guo et al. 2023], and the "neural skeletons" by Clemot et al. [2023]. However, we found that all methods either were too slow to process large datasets or needed manual parameter runing. For our purposes, we needed a skeletonization method that is scalable i.e., is able to handle large shape collections, such as ShapeNet (in other words, it is able to extract a skeleton from a mesh efficiently e.g., a few seconds for the largest mesh). The algorithm should also be robust to varying mesh tesselations, and most importantly should not not require manual parameter adjustment for different shape categories.

During our early experiments, we found out that none of existing methods satisfy all the above criteria. These issues led us to develop our own skeletonization algorithm that balances computational efficiency with the need for accurate skeletons. A key component of this algorithm is a gradient descent procedure on the signed distance function (SDF) of the shape approximated through a local RBF kernel to shrink the surface iteratively towards its interior. We note that our skeletonization is inspired by [Clémot and Digne 2023], yet with several important differences, including in the surface sampling, initialization, gradient descent procedure, and stopping criteria. The algorithm is summarized in Algorithm 1. It initiates medial points from surface sample points from a given mesh by shifting the surface points according to their negated surface normal and a multiplier of the local shape diameter estimated per point through ray casting. This initialization bootstraps the procedure, since the subsequent iterative phase of the method is slower. In the iterative phase, for each current position of medial point, the algorithm computes a Signed Distance Function approximation (SDF) by averaging its signed distances to the tangent planes of the nearest surface points. The averaging is performed with the help of RBF kernel. Following the gradient of the SDF gradually shrinks the shape. The update is constrained: it is accepted if the updated point remains within a distance less than a given threshold expressed as a multiplier (approximately half) of the local shape diameter. This ensures that the points will meet close to the middle of the shape and will not drift too further away. The method continues until the skeleton point

		OccNet	ConvOccNet	IF-Net	3DILG	3D2VS-LQ	3D2VS-PQ	GEM3D
CD ↓	table	0.041	0.036	0.029	0.026	0.026	0.026	**0.014**
	car	0.082	0.083	0.067	0.066	0.062	0.062	**0.016**
	chair	0.058	0.044	0.031	0.029	0.028	0.027	**0.015**
	airplane	0.037	0.028	0.020	0.019	0.018	0.017	**0.008**
	sofa	0.051	0.042	0.032	0.030	0.030	0.029	**0.015**
	rifle	0.046	0.025	0.018	0.017	0.016	0.014	**0.006**
	lamp	0.090	0.050	0.038	0.036	0.035	0.032	**0.010**
IoU ↑	table	0.823	0.847	0.901	0.963	0.965	**0.971**	0.960
	car	0.911	0.921	0.952	0.961	0.966	**0.969**	0.936
	chair	0.803	0.856	0.927	0.950	0.957	**0.964**	0.958
	airplane	0.835	0.881	0.937	0.952	0.962	**0.969**	0.961
	sofa	0.894	0.930	0.960	0.975	0.975	**0.982**	0.974
	rifle	0.755	0.871	0.914	0.938	0.947	**0.960**	0.956
	lamp	0.735	0.859	0.914	0.926	0.931	**0.956**	0.934
F1 ↑	table	0.961	0.982	0.998	**0.999**	**0.999**	**0.999**	0.991
	car	0.830	0.852	0.888	0.892	0.898	0.899	**0.965**
	chair	0.890	0.943	0.990	0.992	0.994	**0.997**	0.984
	airplane	0.948	0.982	0.994	0.993	0.994	0.995	**0.998**
	sofa	0.918	0.967	0.988	0.986	0.986	**0.990**	0.987
	rifle	0.922	0.987	0.998	0.997	0.998	**0.999**	**0.999**
	lamp	0.820	0.945	0.970	0.971	0.970	0.975	**0.990**

Table 8. Evaluation of reconstructed surfaces on ShapeNet based on Chamfer Distance (CD), Intersection over Union (IoU), and F1 scores for the auto-encoding task. Here we use the original 3DShape2VecSet protocol. We report the same seven categories and numbers are not scaled as in their paper.

Choice in the decoder	CD (↓)	IOU (↑)	F1(↑)
Closest medial point	1.41	89.9	98.6
Closest medial envelope	**1.38**	**95.5**	**98.7**

Table 9. Ablation study based on the ShapeNet point cloud reconstruction task (we report averages over all 55 categories).

positions converge up to a tolerance threshold. The parameters used in our method are listed in Algorithm 1. For all our shapes involved in our experiments, these parameters were fixed.