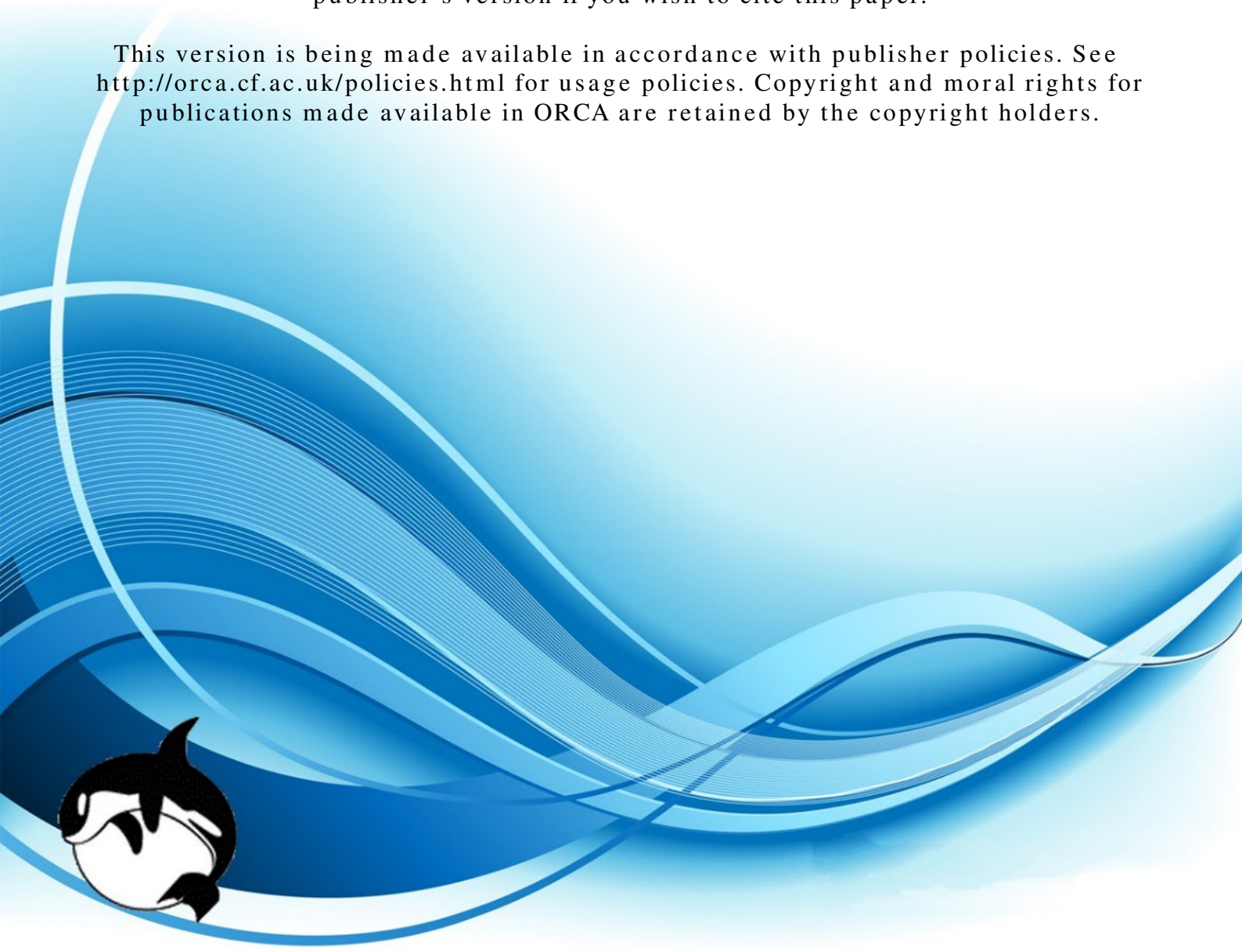


SDM-NET: Deep Generative Network for Structured Deformable Mesh

LIN GAO, Institute of Computing Technology, Chinese Academy of Sciences (CAS)
JIE YANG, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
TONG WU, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
YU-JIE YUAN, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
HONGBO FU, School of Creative Media, City University of Hong Kong
YU-KUN LAI, School of Computer Science and Informatics, Cardiff University
HAO ZHANG, School of Computing Science, Simon Fraser University

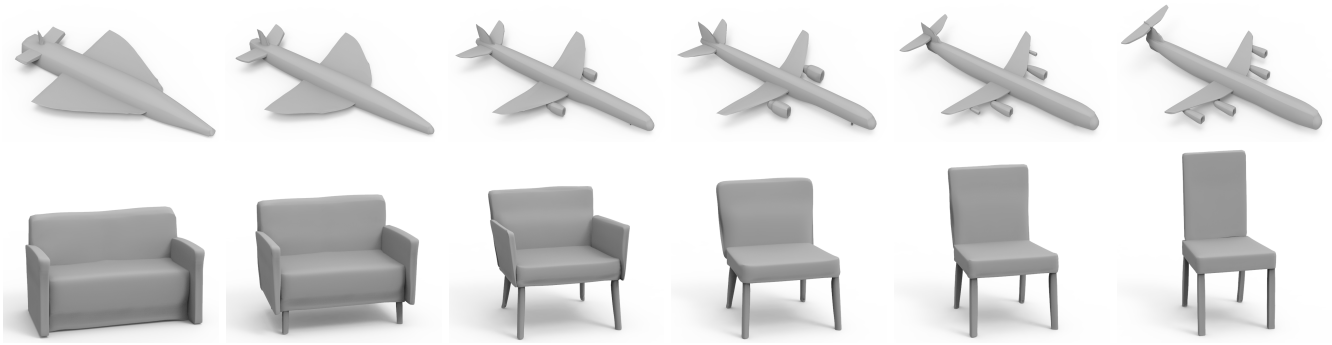


Fig. 1. Our deep generative neural network, SDM-NET, produces structured meshes composed of deformable parts. Part structures and geometries are jointly encoded into a latent space by an autoencoder, enabling quality 3D shape generation. We show shape interpolation results exhibiting flexible structure and fine geometric details. This is achieved by linearly interpolating airplane and chair latent codes and then reconstruction from the in-between codes.

We introduce SDM-NET, a deep generative neural network which produces *structured deformable meshes*. Specifically, the network is trained to generate a spatial arrangement of closed, deformable mesh parts, which respects the global part structure of a shape collection, e.g., chairs, airplanes, etc. Our key observation is that while the overall structure of a 3D shape can be complex, the shape can usually be decomposed into a set of parts, each homeomorphic to a box, and the finer-scale geometry of the part can be recovered by *deforming* the box. The architecture of SDM-NET is that of a *two-level variational autoencoder* (VAE). At the part level, a PartVAE learns a deformable model of part geometries. At the structural level, we train a Structured Parts VAE (SP-VAE), which *jointly* learns the part structure of a shape collection and the part geometries, ensuring the coherence between global shape structure and surface details. Through extensive experiments and comparisons with the state-of-the-art deep generative models of shapes, we demonstrate the superiority of SDM-NET in generating meshes with visual quality, flexible topology, and meaningful structures, benefiting shape interpolation and other subsequent modeling tasks.

CCS Concepts: • **Computing methodologies → Shape modeling**.

Authors' addresses: Lin Gao, Jie Yang, Tong Wu and Yu-Jie Yuan are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. Hongbo Fu is with the school of Creative Media, City University of Hong Kong. Yu-Kun Lai is with the School of Computer Science and Informatics, Cardiff University. Hao Zhang is with the school of Computing Science, Simon Fraser University. Authors' e-mails: {gaolin, yangjie01, wutong, yuanyujie}@ict.ac.cn, hongbofu@cityu.edu.hk, LaiY4@cardiff.ac.uk, haoz@sfu.ca.

Additional Key Words and Phrases: Shape representation, variational autoencoder, structure, geometric details, generation

ACM Reference Format:
Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. 2019. SDM-NET: Deep Generative Network for Structured Deformable Mesh. *ACM Trans. Graph.* 0, 0, Article 0 (2019), 15 pages. https://doi.org/10.1145/3355089.3356488

1 INTRODUCTION

Triangle meshes have been the dominant 3D shape representation in computer graphics, for modeling, rendering, manipulation, and animation. However, as deep learning becomes pervasive in visual computing, most deep convolutional neural networks (CNNs) developed for shape modeling and analysis have resorted to other representations including voxel grids [Girdhar et al. 2016; Qi et al. 2016; Wu et al. 2016, 2015], shape images [Sinha et al. 2016; Su et al. 2015], and point clouds [Qi et al. 2017a; Yin et al. 2018]. One of the main reasons is that the non-uniformity and irregularity of triangle tessellations do not naturally support conventional convolution and pooling operations [Hanocka et al. 2019]. Yet, advantages of meshes over other shape representations should not be overlooked.

Compared to voxels, meshes are more compact and better suited to representing finer surface details. Compared to points, meshes are more controllable and exhibit better visual quality. There have been recent attempts at developing mesh-specific convolutional operators designed for triangle tessellations [Hanocka et al. 2019; Poulenard and Ovsjanikov 2018]. Current deep generative models for meshes

are limited to either genus-zero meshes [Hamu et al. 2018; Maron et al. 2017] or meshes sharing the same connectivity [Gao et al. 2018; Tan et al. 2018]. Patch-based models which cover a shape with planar [Wang et al. 2018b] or curved [Groueix et al. 2018] patches, are more adaptive, but surface quality is often tampered by visible seams and the patches are otherwise unstructured and incoherent.

In this paper, we introduce a novel deep generative neural network for meshes which overcomes the above limitations. Our key observation is that while the overall structure of a 3D shape can be complex, the shape can usually be decomposed into a set of *parts*, each homeomorphic to a box, and the finer-scale geometry of the part can be recovered by *deforming* the box. Hence, the architecture of our network is that of a *two-level* variational autoencoder (VAE) [Kingma and Welling 2013] which produces *structured deformable meshes* (SDM). At the part level, a PartVAE learns a deformable model of shape parts, by means of autoencoding fixed-connectivity, genus-zero meshes. At the structural level, we train a Structured Parts VAE (SP-VAE), which *jointly* learns the part structure of a shape collection and the part geometries, ensuring the coherence between global shape structure and surface details.

We call our network SDM-NET, as it is trained to generate structured deformable meshes, that is, a spatial arrangement of closed, deformable mesh parts, which respects the global part structure (e.g., symmetry and support relations among shape parts) of a shape collection, e.g., chairs, airplanes, etc. However, our network can generate shapes with a varying number of parts, up to a maximum count. Besides the advantages afforded by meshes mentioned above, a structured representation allows shapes generated by SDM-NET to be immediately reusable, e.g., for assembly-based modeling [Mitra et al. 2013]. In addition, the deformability of the mesh parts further facilitates editing and interpolation of the generated shapes.

SDM-NET is trained with a shape collection equipped with a consistent part structure, e.g., semantic segmentation. However, the shapes in the collection can be with arbitrary topologies and mesh connectivities. Such data sets are now widely available, e.g., ShapeNet [Chang et al. 2015] and PartNet [Mo et al. 2019b], to name a few. While direct outputs from SDM-NET are not watertight meshes, each part is.

In summary, the main contributions of our work are:

- The first deep generative neural network which produces structured deformable meshes.
- A novel network architecture corresponding to a two-level variational autoencoder which jointly learns shape structure and geometry. This is in contrast to the recent work, GRASS [Li et al. 2017], which learns shape structure and part geometry using separate networks.
- A support-based part connection optimization to ensure the generation of plausible and physically valid shapes.

Figure 1 demonstrates the capability of our SDM-NET to reconstruct shapes with flexible structure and fine geometric details. By interpolating in the latent space, new plausible shapes with substantial structure change are generated.

Through extensive experiments and comparisons with the state-of-the-art deep generative models of shapes, we demonstrate the superiority of SDM-NET in generating quality meshes and shape interpolations. We also show the structured deformation meshes produced by SDM-NET enable other applications such as mesh editing, which are not directly supported by the output from other contemporary deep neural networks.

2 RELATED WORK

With the resurgence of deep neural networks, in particular CNNs, and an increasing availability of 3D shape collections [Chang et al. 2015], a steady stream of geometric deep learning methods have been developed for discriminative and generative processing of 3D shapes. In this section, we mainly discuss papers mostly related to our work, namely deep generative models of 3D shapes, and group them based on the underlying shape representations.

Voxel grids. The direct extension of pixels in 2D images to 3D is the voxel representation, which has a regular structure convenient for CNNs [Girdhar et al. 2016; Qi et al. 2016; Wu et al. 2016]. Variational autoencoders (VAEs) [Kingma and Welling 2013] and Generative Adversarial Networks (GANs) [Goodfellow et al. 2014] can be built with this representation to produce new shapes. Wu et al. [2019] utilize an autoencoder of two branches to encode geometry features and structure features separately, and fuse them into a single latent code to intertwine the two types of features for shape modeling. However, these voxel based representations have huge memory and calculation costs, when the volumetric resolution is high. To address this, sparse voxel-based methods use octrees to adaptively represent the geometry. However, although such adaptive representations can significantly reduce the memory cost, their expressiveness of geometric details is still limited by the resolution of leaf nodes of octrees [Tatarchenko et al. 2017; Wang et al. 2017]. As an improvement, recent work [Wang et al. 2018b] utilizes local planar patches to approximate local geometry in leaf nodes. However, planar patches still have limited capability of describing local geometry, especially for complex local shapes. The patches are in general not smooth or connected, and require further processing, which might degrade the quality of generated shapes.

Multi-view images. To exploit image-like structures while avoiding the high cost of voxels, projecting shapes to multiple 2D views is a feasible approach. Su et al. [2015] project 3D shapes to multi-view images, along with a novel pooling operation for 3D shape recognition. This representation is regular and efficient. However, it does not contain the full 3D shape information. So, although it can be directly used for recognition, additional efforts and processing are needed to reconstruct 3D shapes [Soltani et al. 2017]. It also may not fully recover geometric details due to the incomplete information in multi-view images.

Point clouds. Point clouds have been widely used to represent 3D shapes, since they are flexible and can easily represent the raw data obtained from 3D scanners. The major challenge for deep learning on point clouds is their irregular structure. Qi et al. [2017a; 2017b] propose PointNet and PointNet++ for 3D classification and segmentation, utilizing pooling operations that are order independent. Yang et al. [2017] exploit an interactive system for segmenting point clouds of indoor scenes. Fan et al. [2017] use point clouds to reconstruct 3D objects from a given image. Achlioptas et al. [2018]

introduce a deep autoencoder network for shape representation and generation. However, learning from irregular point clouds is still challenging and their method is only able to produce relatively coarse geometry.

Meshes and multi-chart representations. Deformable modeling of a shape collection, especially of human models [Anguelov et al. 2005; Pons-Moll et al. 2015], operates on meshes with the same connectivity while altering the mesh vertex positions; the shape collection can be viewed as deformations of a template model. For high quality shape generation, especially with large deformations, a manually crafted deformation representation [Gao et al. 2019] is employed by [Gao et al. 2018; Tan et al. 2018]. Although these methods can represent and generate shapes with fine details, they require meshes to have the same connectivity. Wang et al. [2018c] reconstruct a mesh-based 3D shape from an RGB image by deforming a sphere-like genus-zero mesh model. Dominic et al. [2018] use a CNN to infer the parameters of free-form deformation (FFD) to deform a template mesh, guided by a target RGB image. Both methods require an image as input to provide guidance, and thus cannot be used for general shape generation tasks without guidance. Moreover, deforming a single mesh limits the topological and geometric complexity of generated shapes.

Multi-chart representations attempt to overcome the above restriction by generating multiple patches that cover a 3D shape. This similar methodology has been applied in Zhou et al. [2004] previously to create texture atlases with less stretches for texture mapping. Hamu et al. [2018] generate a 3D shape as a collection of conformal toric charts [Maron et al. 2017], each of which provides a cover of the shape with low distortion. Since toric covers are restricted to genus-zero shapes, their multi-chart method still has the same limitation. AtlasNet [Groueix et al. 2018] generates a shape as a collection of patches, each of which is parameterized to a 2D domain as an atlas. While the patches together cover the shape well, visible seams can often be observed. In general, neither the atlases nor the toric charts correspond to meaningful shape parts; the collection is optimized to approximate a shape, but is otherwise unstructured. In contrast, SDM-NET produces structured deformable meshes.

Implicit representations. Several very recent works [Chen and Zhang 2019; Mescheder et al. 2019; Park et al. 2019] show great promise of generative shape modeling using implicit representations. These deep networks learn an implicit function which defines the inside/outside statuses of points with respect to a shape or a signed distance function. The generative models can be applied to various applications including shape autoencoding, generation, interpolation, completion, and single-view reconstruction, demonstrating superior visual quality over methods based on voxels, point clouds, as well as patch-based representations. However, none of these works generate structured or deformable shapes.

Shape structures. Man-made shapes are highly structured, which motivates structure-aware shape processing [Mitra et al. 2013]. Works on learning generative models of 3D shape structures can be roughly divided into two categories [Chaudhuri et al. 2019]: probabilistic graphical models and deep neural networks.

Huang et al. [2015] propose a probabilistic model which computes part templates, shape correspondences, and segmentations from clustered shape collections, and their points in each part are influenced by their correspondence in the template. Similar to [Huang et al. 2015], ShapeVAE [Nash and Williams 2017] generates point coordinates and normals based on different parts, but uses a deep neural network instead of a probabilistic model. Compared to the above two works, our method does not require point-wise correspondences, which can be difficult or expensive to obtain reliably. Moreover, our method encodes both global spatial structure like support relations, and local geometric deformation, producing shapes with reasonable structures and fine details.

Li et al. [2017] introduce GRASS, a generative recursive autoencoder for shape structures, based on Recursive Neural Networks (RvNNs). Like SDM-NET, GRASS also decouples structure and geometry representations. However, a key difference is that SDM-NET *jointly* encodes global shape structure and part geometry, while GRASS trains *independent* networks for structure and part geometry generations. In terms of outputs, GRASS generates a *hierarchical* organization of bounding boxes, and then fills them with voxel parts. SDM-NET produces a *set* of shape parts, each of which is a deformable mesh to better capture finer surface details. Lastly, the structural autoencoder of GRASS requires symmetry hierarchies for training while SDM-NET only requires consistent semantic segmentation and employs the support information to produce shapes with support stability.

Concurrent to SDM-NET, Mo et al. [2019a] develop *StructureNet*, which learns a generative autoencoder of shape structures based on graph neural networks. StructureNet shares much commonality with GRASS but extends it in two important ways. First, unlike GRASS, which is limited to encoding binary trees, StructureNet can directly encode shapes represented as n-ary graphs, aimed to facilitate a consistent hierarchical representation of shapes within the same category. Second, StructureNet also accounts for horizontal inter-part relationships between siblings. The outputs from StructureNet are either box structures or point cloud shapes. Our SDM-Net analyzes and encodes shape structures by not only using the consistent representation across the same shape families but also with support stability. In addition, it is expected our mesh-based representation with deformable parts is able to capture more geometry details than box and point cloud based representations adopted by StructureNet.

3 METHODOLOGY

Given a collection of shapes of the same category with part-level labels, our method represents them using a structured set of deformable boxes, each corresponding to a part. The pioneering works [Kim et al. 2013; Ovsjanikov et al. 2011] have shown the representation power of using a collection of boxes to analyze and explore shape collections. However, it is highly non-trivial to extend their techniques to shape generation, since boxes are generally of a coarse representation. We tackle this challenge by allowing individual boxes to be flexibly deformable and propose a two-level VAE architecture called SDM-NET, including PartVAE for encoding the geometry of deformable boxes, and SP-VAE for joint encoding of

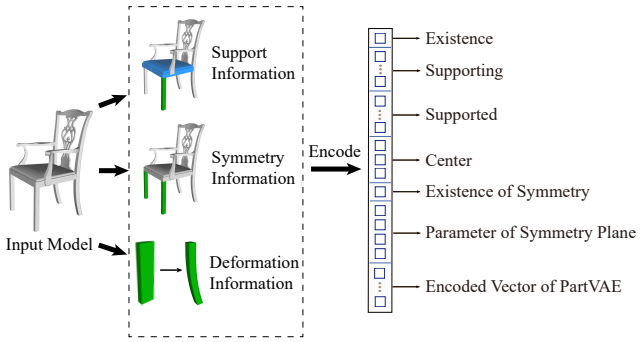


Fig. 2. Encoding of each shape part, including its structure and geometry information. The former includes both the support and symmetry information, and the latter describes the deformation of the bounding box, compactly represented as the latent vector of PartVAE. Detailed explanation of each entry of the code is presented in Section 3.1.

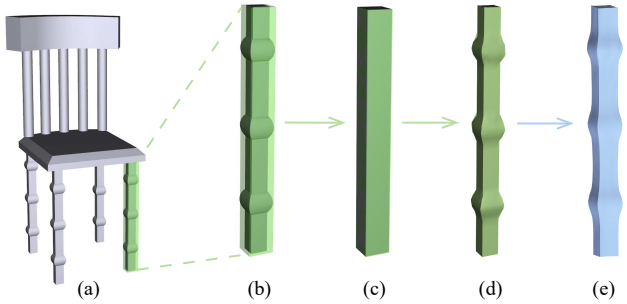


Fig. 3. An example of representing a chair leg with detailed geometry by a deformable bounding box. (a) a chair with one of its leg parts highlighted, (b) the targeted part in (a) with the bounding box overlaid, (c) the bounding box used as the template, (d) deformed bounding box after non-rigid registration, (e) recovered shape using PartVAE.

part geometry and global structure such as symmetry and support. Moreover, to ensure that decoded shapes are physically plausible and stable, we introduce an optimization based on multiple constraints including support stability, which can be compactly formulated and efficiently optimized. Our SDM-NET model allows easy generation of plausible meshes with flexible structures and fine details.

We first introduce the encoding of each part, including both the geometry and structure information. We then introduce our SDM-NET involving VAEs at both the local part geometry level (PartVAE), and global joint embedding of structure and geometry (SP-VAE). Then we briefly describe how the support relationships are extracted, and finally present our optimization for generating plausible and well supported shapes.

3.1 Encoding of a Shape Part

Based on semantic part-level labels, a shape is decomposed into a set of parts. Each part is represented using a deformable bounding box, as illustrated in Figure 3. Let n be the total number of part labels that appear across different shapes in the specified object category. For a given shape, it may contain a fewer number of parts as some parts may not be present. To make analysis and relationship representation easier, we assume the initial bounding box (before deformation) of each part is axis aligned. This is sufficient in practice, since each bounding box is allowed to have substantial deformation to fit the target geometry. The initial bounding primitive being a box does not prevent the internal part geometry from being complex, since geometric details can be captured and preserved through non-rigid registration (see Section 3.2 for details). Without loss of generality, the bounding boxes are used in our framework.

The geometry and associated relationships of each part are encoded by a representation vector $\mathbf{rv}$, as illustrated in Figure 2. The detailed definition of this vector is given as follows.

- $\mathbf{rv}_1 \in \{0, 1\}$ indicates the existence of this part.
- $\mathbf{rv}_2 \in \{0, 1\}^n$ is a vector with n dimensions to indicate which parts are supported by this part.
- $\mathbf{rv}_3 \in \{0, 1\}^n$ is a vector with n dimensions to indicate which parts support the current part.
- $\mathbf{rv}_4 \in \mathbb{R}^3$ is the 3D position of the bounding box center.
- $\mathbf{rv}_5 \in \{0, 1\}$ indicates the existence of a symmetric part.
- $\mathbf{rv}_6 \in \mathbb{R}^4$ records the parameters a-d of the symmetry plane represented in an implicit form, i.e., $ax + by + cz + d = 0$.
- $\mathbf{rv}_7$ is the encoded vector from the PartVAE described in Section 3.2, which encodes its geometry. By default, $\mathbf{rv}_7 \in \mathbb{R}^{64}$.

The ID of each part, used in $\mathbf{rv}_2$ and $\mathbf{rv}_3$, is pre-determined and stored in advance for the dataset. Each value in $\mathbf{rv}_1$, $\mathbf{rv}_2$, $\mathbf{rv}_3$ and $\mathbf{rv}_5$ is 1 if exists and 0 otherwise. For generated vectors, we treat a value above 0.5 as true and below as false. The length of this vector is $2n + 73$ and between 77 and 101 for all the examples in this paper. Note that other information such as the label of the part that is symmetric to the current one (if exists) is fixed for a collection (e.g. the right armrest of a chair is symmetric to the left armrest of the chair) and therefore not encoded in the vector. In our current implementation, we only consider reflection symmetry and keep one symmetric component (if any) for each part. Although this is somewhat restrictive, it is very common and sufficient to cope with most cases. In practice, we first perform global reflection symmetry detection [Podolak et al. 2006] to identify components that are symmetric to each other w.r.t. a symmetry plane. This is then supplemented by local reflection symmetry detection by checking if pairs of parts have reflective symmetry.

3.2 PartVAE for Encoding Part Geometry

For each part, the axis-aligned bounding box (AABB) is first calculated. The bounding box of the same part type provides a uniform domain across different shapes, and the geometry variations are viewed as different deformation functions applied to the same domain. We take a common template, namely a unit cube mesh box_0 with $19.2K$ triangles, to represent each part. We first translate and scale it to fit the bounding box of the part. Denote by $b_{i,j}$ the bounding box transformed from box_0 for the j^{th} part $c_{i,j}$ on the i^{th} shape s_i. We treat it as initialization, and apply non-rigid coarse-to-fine registration [Zollhöfer et al. 2014], which deforms $b_{i,j}$ to $b'_{i,j}$, as illustrated in Figure 3 (d). $b'_{i,j}$ shares the geometry details with the

part $c_{i,j}$ and has the same mesh connectivity as the unit cube box box_0.

The variational autoencoder has been used to encode the geometric priors for point cloud segmentation [Meng et al. 2019] and mesh generation [Tan et al. 2018]. Similar to [Tan et al. 2018], using meshes of the same connectivity makes it feasible to build a variational autoencoder to represent the deformation of $b'_{i,j}$. The convolutional VAE architecture in [Gao et al. 2018] is employed for compactly representing plausible deformation of each part, allowing new variations to be synthesized. The architecture is shown in Figure 4. The input is a $V \times 9$ dimensional matrix, where V is the number of vertices for the template bounding box mesh. Each row of the matrix is a 9-dimensional vector that characterizes the local deformation of 1-ring neighborhood of each vertex including the rotation axis, rotation angle and scaling factor. It passes through two convolutional layers followed by a fully connected layer to obtain the mean and variance. The decoder mirrors the structure of the encoder to recover the deformation representation, but with different trainable weights. Since each part type has its own characteristics, we train a PartVAE for all the parts with the same part type across different shapes.

3.3 Supporting Structure Analysis

Structure captures the relationships between parts, and proper encoding of shape structure is crucial to generate plausible shapes. Symmetry as one of the structural relationships has been well explored, for example effectively used in GRASS [Li et al. 2017]. Besides symmetry, support relationships have been demonstrated useful for structural analysis to synthesize physically plausible new structures with support and stability [Huang et al. 2016]. Compared to symmetry, support relationships provide a more direct mechanism to model the relations between adjacent parts. We thus use both symmetry and support to encode the structural relationships between parts (Section 3.1). Note that our work is the first attempt to encode support-based shape structures in deep neural networks.

Following [Huang et al. 2016], we detect the support relations between adjacent parts as one of three support sub-structures, namely, "support from below", "support from above", and "support from side". As illustrated in Figure 5, the detected support relations turn an undirected adjacency graph to a directed support graph. For each detected support relation of a part, we encode the labels of its supported and supporting parts in our part feature vector (Section 3.1). Our feature coding is flexible to represent the cases including one part being supported by multiple parts, as well as multiple parts being supported by one part. Since for all the cases in our shape dataset, the sub-structure type for each support relation between two adjacent parts is fixed, the support sub-structure types are kept in a look-up table but not encoded in our part feature vector. Given the supporting and supported part labels from a decoded part feature vector, we can efficiently obtain the corresponding sub-structures from this look-up table.

3.4 SP-VAE for Structured Deformable Mesh Encoding

We build SP-VAE to jointly encode the structure of a shape represented as the layout of boxes, and the geometry of its parts. By analyzing their joint distribution, it helps ensure that the geometry of the generated shape is coherent with the structure and the geometries of individual parts are consistent (i.e., of compatible styles). Our SP-VAE takes the concatenation of representation vectors for all the parts as input (see Section 3.1). It encodes parts in a consistent order during encoding and decoding. This concatenated vector covers both the geometric details of individual parts encoded using PartVAE, and the relationships between them. The SP-VAE uses multiple fully connected layers, and the architecture is illustrated in Figure 6.

Let $Enc_S(\cdot)$ and $Dec_S(\cdot)$ denote the encoder and decoder of our SP-VAE network, respectively. $\mathbf{x}$ represents the input concatenated feature vector of a shape, $\tilde{\mathbf{x}} = Enc_S(\mathbf{x})$ is the encoded latent vector, and $\mathbf{x}' = Dec_S(\tilde{\mathbf{x}})$ is the reconstructed feature vector. Our SP-VAE minimizes the following loss:

$$L_{SP-VAE} = \lambda_1 L_{recon} + \lambda_2 L_{KL} + L_{RegVAE}, \tag{1}$$

where λ_1 and λ_2 are the weights of different loss terms, and

$$L_{recon} = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{S}} ||\mathbf{x} - \mathbf{x}'||_2^2 \tag{2}$$

denotes the MSE (mean squared error) reconstruction loss. Here $\mathcal{S}$ is the training dataset and $N = |\mathcal{S}|$ is the number of shapes in the training set.

$$L_{KL} = D_{KL}(\hat{q}(\tilde{\mathbf{x}}|\mathbf{x})|\hat{p}(\tilde{\mathbf{x}})) \tag{3}$$

is the KL divergence to promote Gaussian distribution in the latent space, where $\hat{q}(\tilde{\mathbf{x}}|\mathbf{x})$ is the posterior distribution given feature vector $\mathbf{x}$, and $\hat{p}(\tilde{\mathbf{x}})$ is the Gaussian prior distribution. L_{RegVAE} is the squared ℓ_2 norm regularization term of the network parameters used to avoid overfitting. The Gaussian distribution makes it effective to generate new shapes by sampling in the latent space, which is used for random generation and interpolation.

3.5 Shape Generation and Refinement

The latent space of SP-VAE provides a meaningful space for shape generation and interpolation. Extensive experimental results are shown in Section 5. Random sampling in the latent space can generate novel shapes. Although the desired geometry and structure from the decoded feature vector are generally reasonable, they may not satisfy supporting and physical constraints exactly, resulting in shapes which may include parts not exactly in contact, or may be unstable. Inspired by [Averkiou et al. 2014], we propose to use an effective global optimization to refine the spatial relations between parts by mainly using the associated symmetry and support information.

Denote the center position and size (half of the length in each dimension) of the i^{th} part as $\mathbf{p}_i$ and $\mathbf{q}_i$, each being a 3-dimensional vector corresponding to x, y and z axis, where $\mathbf{p}_i$ is directly obtained from the representation vector, and $\mathbf{q}_i$ is determined by the bounding box after recovering the part geometry. Denote by $\mathbf{p}'_i$ and $\mathbf{q}'_i$ the position and size of the i^{th} part after global optimization. The objective of this optimization is to minimize the changes between the optimized position/scale and the original position/scale

$$\sum_i \|\mathbf{p}'_i - \mathbf{p}_i\|^2 + \alpha \|\mathbf{q}'_i - \mathbf{q}_i\|^2, \tag{4}$$

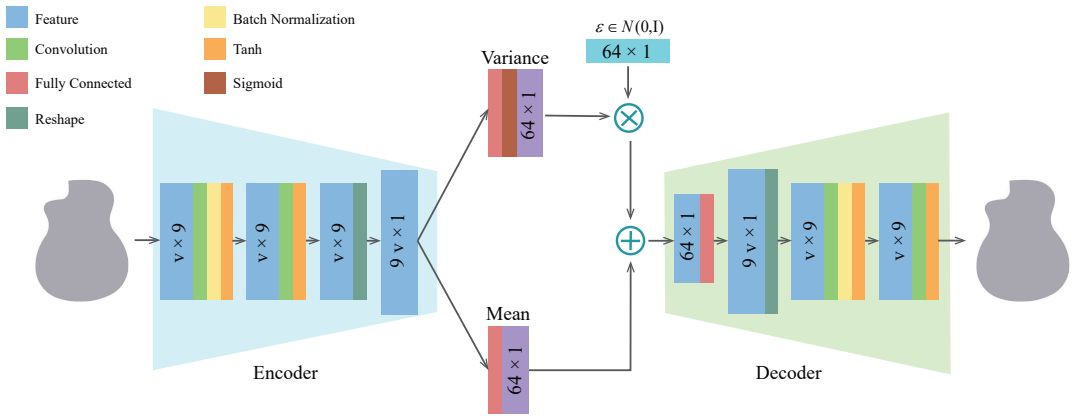


Fig. 4. Architecture of PartVAE for encoding the geometry details of a part represented as the deformation of the associated template box. V is the number of vertices in the template box. $N(\mathbf{0}, \mathbf{I})$ is the Gaussian distribution with $\mathbf{0}$ mean and identity covariance.

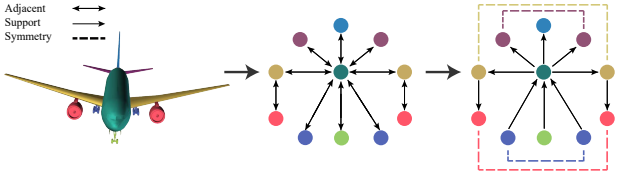


Fig. 5. Illustration of support and symmetry relations between parts of an airplane model. The support relations, detected by the approach in [Huang et al. 2016], turn an undirected adjacency graph (Middle) to a directed support graph (Right).

while ensuring the following constraints are satisfied. α is a weight to balance the two terms, and is fixed to 10 in our experiments. The symmetry and equal length constraints are from [Averkiou et al. 2014], though we use the support relationships to help identify equal length constraints more reliably. The remaining constraints are unique in our approach. Figure 7 illustrates typical problematic cases which are handled by our refinement optimization.

Symmetry Constraint. If the generated i^{th} part has the symmetry indicator flagged in its representation vector, its symmetry part (denoted using index j) also exists. Let $\mathbf{n}_i$ and d_i denote the normal and the intercept of the symmetry plane, respectively. Enforcing the symmetry constraint leads to the following constraints to be satisfied: $\frac{(\mathbf{p}'_i - \mathbf{p}'_j)}{2} \cdot \mathbf{n}_i + d_i = 0$, $(\mathbf{p}'_i - \mathbf{p}'_j) \times \mathbf{n}_i = 0$. The symmetry of two parts is viewed as an undirectional relationship. If the symmetry indicator ($\mathbf{rv}_5$ of the representation vector) of either part i or j is 1, we consider these two parts as symmetric.

Equal Length Constraint. A set of parts which are simultaneously supported by a common part, and simultaneously support another common part, are considered as a group to have the same length along the supporting direction. For this purpose, the ground is considered as a virtual part. For example, the four legs of a table supporting the same table top part and at the same time being supported by the ground should have the same height. These can be easily detected by traversing the support structure. An example that violates this constraint is illustrated in Figure 7 (a). The equal length constraints can be formulated as $\mathbf{q}'_i[t] = \mathbf{q}'_k[t], k \in g_i$, where g_i is an equal-length group containing the i^{th} part, and t is the supporting direction. $t = 0, 1, 2$ respectively represents x, y and z directions where y is the upright direction.

Support Relationship Constraint. In order for a supporting part to well support a part being supported, two requirements are needed: 1) in the supporting direction, the bounding box of the supporting part should have tangential relation (or a small amount of intersection in practice) with the bounding box of the part being supported (see Figure 7 (e) for a problematic case violating this constraint). If the i^{th} part supports the j^{th} part, the following inequality should be satisfied along the supporting direction. $\mathbf{p}'_j[t] - \mathbf{q}'_j[t] + \varepsilon\mathbf{q}'_j[t] \leq \mathbf{p}'_i[t] + \mathbf{q}'_i[t] \leq \mathbf{p}'_j[t] - \mathbf{q}'_j[t] + 2\varepsilon\mathbf{q}'_j[t]$, where t is the supporting direction, and ε controls the amount of overlap allowed and is set to $\varepsilon = 0.1$ in our experiments. 2) assuming $\tilde{b}_i$ and $\tilde{b}_j$ are the bounding boxes of parts i and j projected onto the plane orthogonal to the supporting direction t, it should satisfy that either $\tilde{b}_i \subseteq \tilde{b}_j$ or $\tilde{b}_j \subseteq \tilde{b}_i$ (see Figure 7 (c-d) for examples). This constraint can be formulated as an integer programming problem and solved efficiently during the optimization. The detailed formulation is provided in the Appendix.

Stable Support Constraint. For the "support above" relation ($t = 1$), the center of a supported part should be located in the supporting bounding box (the bounding box that covers all the bounding boxes of the supporting parts) for stable support (see Figure 7 (b) for an example violating this constraint). For a single supported part, the following constraints should be followed. $\mathbf{p}'_i[l] - \mathbf{q}'_i[l] \leq \mathbf{p}'_j[l] \leq \mathbf{p}'_i[l] + \mathbf{q}'_i[l], l \in \{0, 2\}$. For multiple supporting parts (e.g. four legs supporting the table top), the lower bound and upper bound of the x and z directions will be chosen from the corresponding parts.

This quadratic optimization with linear integer programming is solved by TOMLAB [Holmström and Edvall 2004] efficiently. We show an example of shape refinement in Figure 8.

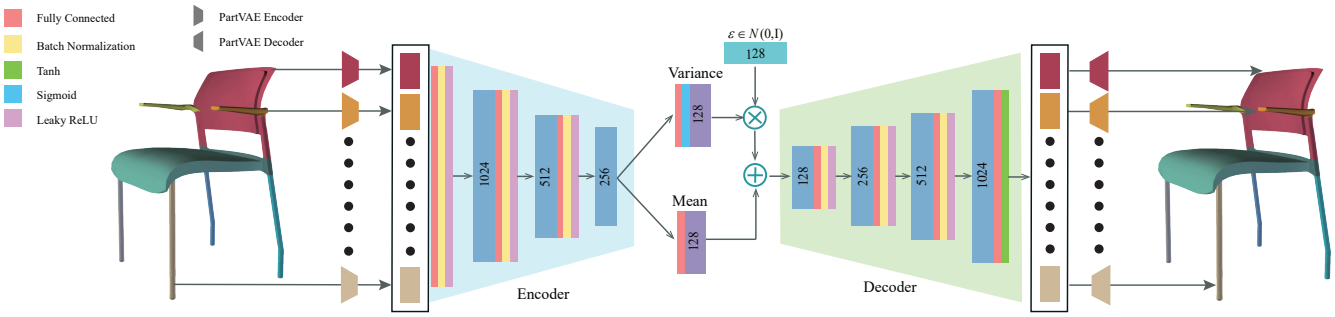


Fig. 6. Architecture of Structured Parts VAE (SP-VAE). The detailed geometry is encoded by PartVAE. The support- and symmetry-induced structure and the associated latent code of PartVAE are encoded by the Structured Parts VAE.

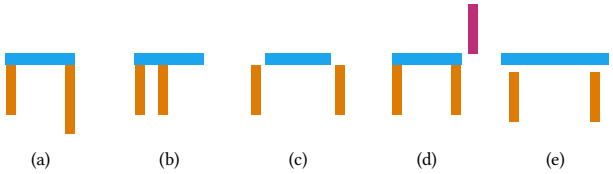


Fig. 7. Illustration of different cases addressed by refinement optimization: (a) equal length constraint, (b) stable support constraint, (c-e) support constraint.

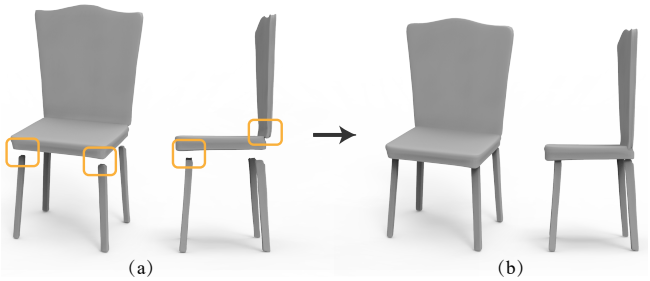


Fig. 8. An example showing the effect of shape refinement. (a) shape decoded by SDM-NET directly, (b) shape after optimization. The artifacts of gap and unstable support are fixed. The sub-figures show the same shape from two viewpoints, with problematic areas highlighted.

4 DATASET AND NETWORK IMPLEMENTATION

We now give the details of our network architecture and training process. The experiments were carried out on a computer with an I7 6850K CPU, 64GB RAM, and a GTX 1080 Ti GPU. The code and data are available at http://geometrylearning.com/sdm-net.

4.1 Dataset Preparation

The mesh models used in our paper are from [Yi et al. 2016], including a subset of ShapeNet Core V2 models [Chang et al. 2015], as well as ModelNet [Wu et al. 2015]. These datasets include pre-aligned models. However, ModelNet does not contain semantic segmentation, and models from [Yi et al. 2016] sometimes do not have sufficiently detailed segmentation to describe support structure (e.g. the car body and four wheels are treated as a single segment). To facilitate our processing, we use an active learning approach [Yi et al. 2016] to perform a refined semantic segmentation. We further use refined labels to represent individual parts of the same type, e.g., to have *left armrest* and *right armrest* labels for two armrest parts. The statistics of the resulting dataset are shown in Table 1.

Our network takes 3D shapes with consistent segmentation as input. The segmentation of test shapes can be obtained by some supervised methods such as [Qi et al. 2017a,b]. Each segmented part is registered from the bounding box by non-rigid deformation. Our method allows each part type to include substantial geometric variations, e.g., the swivel leg and bar stool in Figure 14. Within the confines of the consistent segmentation, the SP-VAE is capable of handling variations of part structure and topologies, as well as varying part counts (by marking certain parts as non-existing).

Table 1. The numbers of meshes and total part labels for each category in our dataset (after label refinement).

Category	Airplane	Car	Chair	Table	Mug	Monitor	Guitar
# Meshes	2690	1824	3746	5266	213	465	787
# Labels	14	7	10	9	2	3	3

4.2 Network Architecture

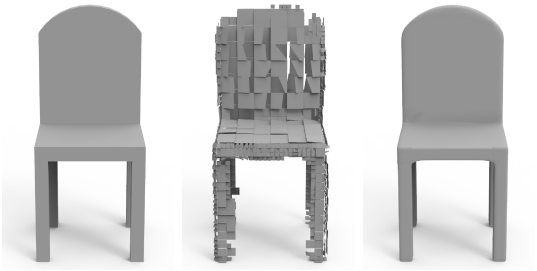
The whole network includes two components, namely *PartVAE* for encoding the deformation of each part of a shape, and *SP-VAE* for jointly encoding the global structure of the shape and the geometric details of each part.

As illustrated in Figure 4, the structure of the PartVAE has two convolutional layers and one fully connected layer. We use *tanh* as the activation function, and in the last convolution layer, we use the linear output. The output of the last convolution layer is reshaped to a vector and mapped into a 64-dimensional latent space by the fully connected layer. The decoder has a mirrored structure, but not sharing weights with the encoder. We train the PartVAE once for each part type.

The input of the SP-VAE is the concatenated representation vector of all parts as shown in Figure 6. The input is fully connected with

(a) Input shape (b) PSG (c) AtlasNet (d) Our Method

Fig. 9. Representative results of decoded shapes with different methods. Compared with PSG [Fan et al. 2017] and AtlasNet [Groueix et al. 2018], our method produces the decoded shapes of higher quality. PSG results are rather coarse point samples. The results by AtlasNet exhibit clearly noticeable patch artifacts.



(a) Input shape (b) Adaptive O-CNN (c) Our Method

Fig. 10. Visual comparison of the decoded shapes with Adaptive O-CNN [Wang et al. 2018b] and our method. Compared with Adaptive O-CNN, while the planar regions of the chair can be decoded by both methods, the curved regions such as the top of the chair back can be recovered only by our method.

dimensions 1024, 512 and 256, respectively, and the latent space dimension is 128. Leaky ReLU is set as the activation function.

4.3 Parameters

We use fixed hyper-parameters in our experiments for different shape categories. In the following, we perform experiments on the *table* data in the ShapeNet Core V2 to demonstrate how the method behaves with changing hyper-parameters. The dataset is randomly split into the training data (75%) and test data (25%). The generalization of SP-VAE is evaluated with different hyper-parameters in Table 2, where the bidirectional Chamfer distance is used to measure the reconstruction error on the test data (as unseen data). We perform such tests for 10 times and report the average errors in Table 2. As can be seen, SP-VAE has the lowest error with the hyper-parameters $\lambda_1 = 1.0$ and $\lambda_2 = 0.5$, where λ_1 and λ_2 are the weights of the reconstruction error term and KL divergence term, respectively. The hyper-parameters (weights of reconstruction, KL-divergence, and regularization) of PartVAE are set to the same numbers in [Gao et al. 2018]. We set the dimension of the latent space of PartVAE to 64, and the dimension of the latent space of SP-VAE to 128. These two parameters are evaluated in Tables 3 and 4 with the reconstruction error. When adjusting the dimension of one VAE, we leave the dimension of the other VAE unchanged.

Table 2. Comparison of average SP-VAE reconstruction errors (measured in bidirectional Chamfer distance) for unseen data on the *table* dataset w.r.t. changing hyper-parameters.

(λ_1, λ_2)	(0.5, 0.5)	(1.0, 0.5)	(0.5, 1.0)	(1.0, 1.0)
Recons. Error ($\times 10^{-3}$)	2.01	**1.85**	2.24	1.94

4.4 Training Details

Since a PartVAE encodes the geometry of a specific type of parts, it is trained separately. SP-VAE is then trained using PartVAE for encoding part geometry. Training of both VAEs is optimized using the Adam solver [Kingma and Ba 2015]. The PartVAE is trained with

Table 3. Comparison of average reconstruction errors (measured by bidirectional Chamfer distance) for PartVAE and SP-VAE w.r.t. changing dimension of the PartVAE latent space.

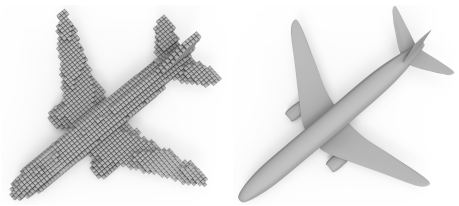
PartVAE Embedding Dimension	32	64	128	256
PartVAE Recons. Error ($\times 10^{-3}$)	1.92	1.76	**1.74**	1.82
SP-VAE Recons. Error ($\times 10^{-3}$)	2.16	**1.85**	1.91	2.03

Table 4. Comparison of average reconstruction errors (measured by bidirectional Chamfer distance) of SP-VAE w.r.t. changing embedding dimension.

SP-VAE Embedding Dimension	32	64	128	256
SP-VAE Recons. Error ($\times 10^{-3}$)	2.23	1.99	**1.85**	1.91

(a) (b) (c) (d)

Fig. 11. Visual comparison of shape reconstruction with GRASS [Li et al. 2017] and our technique. (a) input shape, (b)(c) GRASS results in voxels and extracted mesh, (d) Our result.



(a) (b)

Fig. 12. Visual comparison between the global-to-local method [Wang et al. 2018a] (a) and our technique (b) for shape generation.

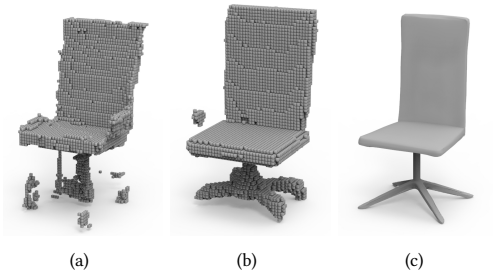
20,000 iterations and SP-VAE with 120,000 iterations by minimizing their loss functions. For both VAEs, we set the batch size as 512 and learning rate starting from 0.001 and decaying every 1000 steps with the decay rate set to 0.8. The training batch is randomly sampled from the training data set.

For a typical category, the training of both PartVAE and SP-VAE takes about 300 minutes. Once the networks are trained, shape generation is very efficient: generating one shape and structure optimization take only about 36 and 100 milliseconds, respectively.

5 RESULTS AND EVALUATION

We present the results of shape reconstruction, shape generation and shape interpolation to demonstrate the capability of our method, and compare them with those generated by the state-of-the-art methods. We also perform ablation studies to show the advantages of our design. Finally, we present examples to show generalizability (i.e., applying our learned model to new shapes of the same category), editability and limitations of our technique.

Shape Reconstruction. We compare our method with PSG [Fan et al. 2017], AtlasNet [Groueix et al. 2018] and Adaptive O-CNN [Wang et al. 2018b] on the ShapeNet Core V2 dataset. In this experiment, we choose four representative categories commonly used in the literature to perform both qualitative and quantitative comparisons. Each dataset is randomly split into the training set (75%) and test set (25%). For fair comparison, we train PSG, AtlasNet, and Adaptive O-CNN for individual shape categories, similar to ours. To prepare the input for PSG, we use rendered images under different viewpoints. Given the same input models in the test set, we compare the decoded shapes by different methods. Figures 9 and 10 show the visual comparison of representative results on several test shapes. It can be easily seen that the decoded shapes by PSG, Adaptive O-CNN and AtlasNet cannot capture the shapes faithfully. AtlasNet and Adaptive O-CNN are able to produce more details than PSG, but suffer from clearly noticeable patch artifacts. In contrast, SDM-NET recovers shapes with higher quality and finer-detailed geometry. Note that we compare the existing methods with SP-VAE followed by structure optimization instead of SP-VAE alone, since structure optimization, which is dependent on the output of SP-VAE, is a unique and essential component in our system, and cannot be directly used with the methods being compared due to their lack of structure information.



(a) (b) (c)

Fig. 13. Visual comparison of shape generation using different methods. (a) 3DGAN [Wu et al. 2016], (b) the global-to-local method [Wang et al. 2018a], (c) our technique.

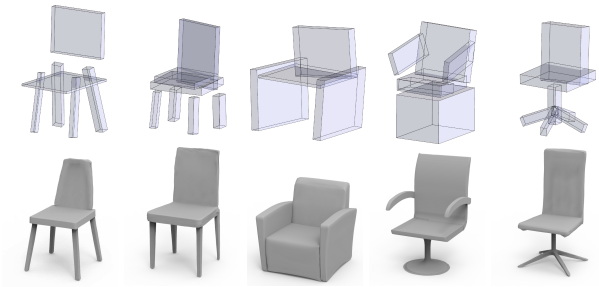


Fig. 14. Comparison between GRASS [Li et al. 2017] and our method for random generation. We visualize the structures generated by GRASS and shapes generated by our method.

Moreover, we quantitatively compare our method with the existing methods using common metrics for 3D shape sets, including Jensen-Shannon Divergence (JSD), Coverage (COV) and Minimum Matching Distance (MMD) [Achlioptas et al. 2018]. The latter two metrics are calculated using both the Chamfer Distance (CD) and Earth Mover's Distance (EMD) for measuring the distance between shapes. For JSD and MMD, the smaller the better, while for COV,

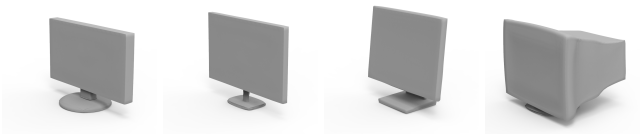


Fig. 15. Random generation of monitor shapes using our method, where the training data is from ModelNet [Wu et al. 2015].

Fig. 16. Visual comparison of shape interpolation by AtlasNet [Groueix et al. 2018] (top) and our technique (bottom). The first and last columns are the two shapes to be interpolated (which are decoded by respective methods).

Fig. 17. Shape interpolation of cups and chairs with different topologies using our method. The first and last columns are the input shapes for interpolation.

the larger the better. The average results for different methods on these datasets are shown in Table 5. It can be seen that our method achieves the best performance for nearly all the metrics.

In Table 6 we compare our method with three recent shape generation methods, namely, GRASS [Li et al. 2017], G2L [Wang et al. 2018a] and SAGNet [Wu et al. 2019]. For fair comparison, both of our reconstructed shapes and input shapes are voxelized. Particularly, we make comparisons with GRASS on their chair data since GRASS requires symmetry hierarchies as input for training. The results show that our method outperforms the compared methods in most cases with several metrics. We also show a visual comparison result between GRASS and our method in Figure 11. The GRASS result exhibits some artifacts due to the voxel representation. Even after surface extraction GRASS still fails to capture fine geometric details compared with our method.

Shape Generation. In Figure 12, we make a qualitative comparison between our technique and the global-to-local method [Wang et al. 2018a] by randomly generating shapes of airplanes. Their method uses an unconditional GAN architecture and thus cannot reconstruct a specific shape. So two randomly generated, visually similar planes are selected for comparisons. Their voxel based method fails to represent smooth, fine details of 3D shapes. We make further comparison with the global-to-local method [Wang et al. 2018a] as well as 3DGAN [Wu et al. 2016] in Figure 13. Again, we select visually similar shapes for comparison, and our method produces high quality shapes with plausible structure and fine details, whereas alternative methods have clear artifacts including fragmented output and rough surfaces. We also compare our technique with GRASS [Li et al. 2017] for random shape generation. As shown in Figure 14, the structures synthesized by GRASS might be problematic, containing parts which are disjoint and/or not well supported. In addition, since it is trained on symmetry hierarchies constructed on top of the shape segmentations, once trained, for new inputs GRASS utilizes automatically generated symmetry hierarchies, which, however, can be inconsistent. This is one of the main causes for GRASS to produce results with a greater level of structural noise including disconnections and asymmetries. In contrast, our results are physically stable and well connected. Note that our refinement step is an integral part of our pipeline and requires structure relations, so cannot be directly applied to GRASS.

As a generative model, our technique is able to generate new shapes. Because our architecture consists of two VAEs, i.e., PartVAE and SP-VAE, we can acquire different information from their latent spaces. Specifically, we extract various types of parts and structural information from the latent space of SP-VAE, and combine them with the deformation information from PartVAE, to produce novel shapes. Figure 15 gives an example, where our method is used to generate computer monitors with various shapes by sampling in the learned latent space. In this example, the training data is obtained from ModelNet [Wu et al. 2015].

Shape Interpolation. Shape interpolation is a useful technique to generate gradually changing shape sequences between a source shape and a target shape. With the help of SP-VAE, we first encode the source and target shapes into latent vectors and then perform linear interpolation in the latent space of VAE. A sequence of shapes between the input shape pairs are finally decoded from the linearly interpolated latent vectors. In Figure 16, we compare our technique with AtlasNet [Groueix et al. 2018] for their performance on shape interpolation. It can be easily seen that the results by AtlasNet suffer from patch artifacts and the surfaces of the interpolated shapes are often not very smooth. The interpolation in our latent space leads to much more realistic results. For example, the armrests gradually become thinner and then disappear in a more natural manner. This is because we combine the geometry and structure during the training of SDM-NET, which thus learns the implicit joint distribution of the geometry and structure.

The effectiveness of shape interpolation with SDM-NET is consistently observed with additional experiments on different datasets. Figure 17 shows two examples of natural interpolation between shapes with different topologies, thanks to our flexible structure representation. Figure 18 shows an additional interpolation result with substantial change of geometry.

Ablation Studies. We perform several ablation studies to demonstrate the necessity of key components of our architecture.

Support vs. adjacency relationships. We adopt support relationships in our method, rather than adjacency relationships to get well connected shapes, because support relationships ensure generating physically stable shapes, and provide a natural order which is useful

Table 5. Quantitative comparison of reconstruction capabilities of different methods on several metrics. For JSD and MMD, the smaller the better, while for COV, the larger the better.

Dataset	Methods	Metrics				
		JSD	MMD-CD	MMD-EMD	COV-CD	COV-EMD
Airplane	AOCNN	0.0665	0.0167	0.0157	84.3	**95.5**
	AtlasNet	0.0379	0.0147	0.0132	79.6	82.1
	PSG	0.0681	0.0244	0.0172	33.5	38.9
	Our	**0.0192**	**0.00462**	**0.00762**	**87.2**	90.6
Car	AOCNN	0.0649	0.0264	0.0223	60.6	60.8
	AtlasNet	0.0393	0.0228	0.0137	75.4	81.9
	PSG	0.0665	0.0365	0.0247	49.8	59.4
	Our	**0.0280**	**0.00247**	**0.00101**	**87.2**	**88.5**
Chair	AOCNN	0.0384	0.0159	0.0196	43.5	39.3
	AtlasNet	0.0369	0.0137	0.0124	51.1	52.6
	PSG	0.0391	0.0131	0.0152	42.9	49.1
	Our	**0.0364**	**0.00375**	**0.00764**	**47.3**	**55.3**
Table	AOCNN	0.0583	0.0393	0.0256	55.2	40.1
	AtlasNet	0.0324	0.0154	0.0146	59.1	63.7
	PSG	0.0354	0.0271	0.0276	41.2	42.5
	Our	**0.0123**	**0.00183**	**0.00127**	**63.3**	**76.8**

Table 6. Quantitative comparison of reconstruction capabilities of different methods (G2L [Wang et al. 2018a], GRASS [Li et al. 2017], SAGNet [Wu et al. 2019]) on several metrics. For JSD and MMD, the smaller the better, while for COV, the larger the better.

Dataset	Methods	Metrics				
		JSD	MMD-CD	MMD-EMD	COV-CD	COV-EMD
Chair	G2L	0.0357	0.0034	0.0682	83.7	83.4
	GRASS	0.0374	0.0030	0.0744	46.0	44.5
	SAGNet	0.0342	**0.0024**	0.0608	75.1	74.3
	Our	**0.0289**	0.00274	**0.00671**	**89.3**	**84.1**

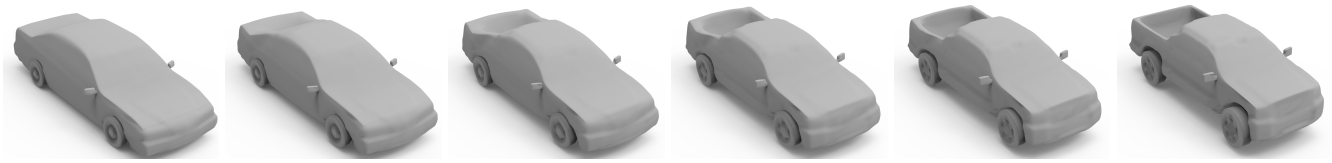


Fig. 18. Interpolating cars with different geometries using our method. The first and last columns are the shapes to be interpolated. The other columns are the in-between models by linear interpolation in the latent space.

(a) Generated shape (b) Adjacency constraint (c) Support constraint

Fig. 19. Comparison between adjacency relationship constraint and support relationship constraint. (a) is a randomly generated extreme case by SDM-NET before refinement. (b) and (c) are the results after refinement with only the adjacency constraint and the support constraint, respectively.

(a) Input shape (b) Separate training (c) Joint training

Fig. 20. Qualitative comparison between two different training strategies, i.e., separate training vs. end-to-end training. It can be seen that separate training keeps more geometric details.

(a) Decoupled structure and geometry (b) Our joint encoding

Fig. 21. Results of our method (b), compared with decoupled structure and geometry (a). The latter is produced by decoupling the geometry information from SP-VAE, and generating the geometry of individual parts independently.

to simplify the structure refinement optimization. In contrast, using a bidirectional adjacency, it would be much more complicated to formulate and optimize constraints between two adjacent parts. To evaluate the effectiveness of the support relationships, we replace the support constraints by simply minimizing the distance between every pair of adjacent parts to approximate the adjacency relationships. The effects of using support and adjacency constraints are shown in Figure 19. It can be seen that the support constraints lead to a physically more stable result.

Separate vs. end-to-end training. We adopt separate training for the two-level VAE, i.e. PartVAEs are trained for individual part types first, before training SP-VAE where the geometries of parts are encoded with the trained PartVAEs. The two-level VAE could also be trained end-to-end, i.e., optimizing both PartVAEs and SP-VAE simultaneously. We compare the average bidirectional Chamfer distance of the reconstruction of each part between end-to-end training and separate training adopted in our solution, as given in Table 7. The visual comparisons are shown in Figure 20. Since without the help of the well-trained distribution of the latent space of individual parts, end-to-end training would result in optimization stuck at a poor local minimum, leading to higher reconstruction errors and visually poor results.

Table 7. Comparison of reconstruction errors ($\times 10^{-3}$) under the metric of bidirectional Chamfer distance with two different training strategies, i.e., separate training vs. end-to-end training.

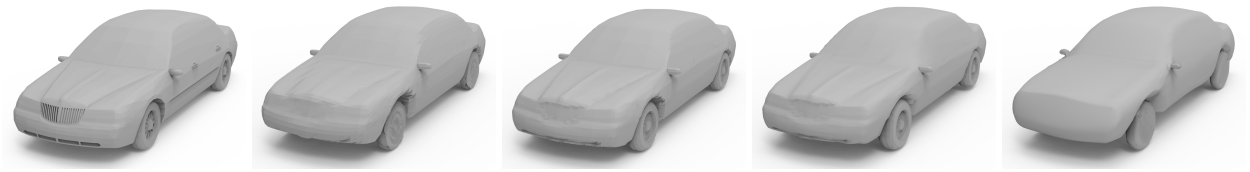
Dataset	Car	Chair	Guitar	Airplane	Table
Separate	**2.77**	**3.89**	**3.58**	**4.87**	**1.85**
End-to-End	5.07	6.73	7.44	11.38	4.86

Joint vs. decoupled structure and part geometry encoding. In this paper, the geometry details represented as part deformations are encoded into the SP-VAE embedding space jointly (see Section 3.4). This approach ensures that generated shapes have consistent structure and geometry, and the geometries of different parts are also coherent. We compare our solution with an alternative approach where the structure and geometry encodings are decoupled: SP-VAE only encodes the structure and the geometry of each part is separately encoded using a PartVAE. Figure 21 shows randomly generated shapes with both approaches. The structure of the first example implies the shape is a sofa, but the geometry of the seat part in (a) does not look like a sofa, whereas our method generates part geometry consistent with the structure. For the second example, our method produces parts with coherent geometry (b), whereas using decoupled structure and geometry leads to inconsistent part styles (a).

Resolution of bounding boxes. By default, our method uses bounding boxes each with $19.2K$ triangles. We also try using lower and higher resolution bounding boxes. As shown in Figure 22, using lower resolution (b) cannot capture the details of the shape, and using higher resolution (d) produces very similar result as our default setting (c), but takes longer time. Our default setting (c) provides a good balance between efficiency and quality.

PartVAE per part type vs. single PartVAE In our paper, we train a PartVAE for each part type. We compare this with an alternative approach where a single PartVAE is trained for all part categories. As shown in Figure 22 (e), this approach is not effective in capturing unique geometric features of different parts, leading to poor geometric reconstruction.

Generalizability. Figure 23 shows an example that demonstrates the generalizability of our method, to process new shapes of the same category without input semantic segmentation. We first train PointNet++ [Qi et al. 2017b] on our labeled dataset, which is then used for semantic segmentation of the new shape. Finally, we obtain the reconstruction result by our SDM-NET. An example is shown in Figure 23, which demonstrates that semantic segmentation obtained automatically can be effectively used as input to our method.



(a) (b) (c) (d) (e)

Fig. 22. Shape reconstruction using our SDM-NET with changing bounding box resolutions and using a single PartVAE for all part categories. (a) input shape, (b) our method with low-resolution bounding boxes ($4.8K$ triangles), (c) our method with default resolution bounding boxes ($19.2K$ triangles), (d) our method with high-resolution bounding boxes ($76.8K$ triangles), (e) result with a single PartVAE for all part categories.

(a) Input Shape (b) Segmentation (c) Reconstruction

Fig. 23. An example demonstrating the generalizability of our method. (a) input shape without semantic segmentation, (b) segmentation by PointNet++, (c) the reconstruction result by our method.

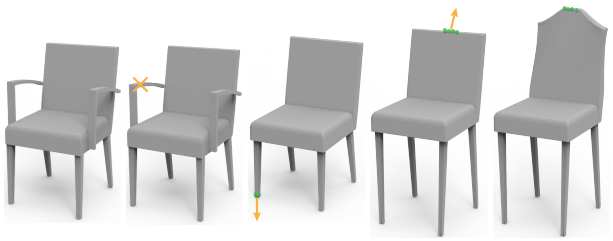
(a) Decoded models (b) Watertight models

Fig. 24. Watertight models derived from our decoded models using [Huang et al. 2018]

Watertight models. The direct output of our method includes watertight meshes for individual parts, but not the shape as a whole. As demonstrated in Figure 24, by applying a watertight reconstruction technique [Huang et al. 2018], watertight meshes can be obtained, which benefit certain downstream applications.

Editability. Our generative model produces Structure Deformable Meshes, which are immediately editable in a structure-aware manner. This is difficult for other generative methods (e.g. [Li et al. 2017; Wu et al. 2016]). An example is given in Figure 25, which shows an editing sequence, including removing parts (when a part is removed, its symmetric part is also removed), making a chair leg longer, which also affects other chair legs due to the equal length constraint, and further deforming the chair back using an off-the-shelf deformation method [Sorkine and Alexa 2007]. During shape deformation, the editing constraints are treated as hard constraints and the equal length constraints are used in the refinement step (see Section 3.5).

Limitations. Although our method can handle a large variety of shapes with flexible structures and fine details, it still suffers from several limitations. While our method can handle shapes with holes formed by multiple parts, if a part itself has holes in it, our deformable box is unable to represent it exactly as the topology of parts cannot be different from genus-zero boxes. In this case, our method will try to preserve the mesh geometry but cannot maintain the hole. For certain parts which are unusual (e.g. the legs and back of the chair and the headstock of the guitar in Figure 26), our VAE architecture considers such cases as outliers, and "projects" them back to deformations consistent with the training set. Another limitation is that currently SDM-NET is trained using a collection of shapes



(a) (b) (c) (d) (e)

Fig. 25. Our generated Structured Deformable Mesh is directly editable. We show an example editing sequence. (a) is the decoded shape of our method. After applying the deletion operation (b), we obtain a chair without armrests (c). Dragging a leg of the chair makes all four legs longer due to the equal length constraint (d). Finally, we deform the back of the chair to obtain (e).

with the same category. It thus cannot be used for interpolating shapes of different categories.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented SDM-NET, a novel deep generative model that generates 3D shapes as Structured Deformable Meshes. A shape is represented using a set of deformable boxes, and a two-level VAE is built to encode local geometry variations of individual parts, and global structure and geometries of all parts, respectively. Our representation achieves both flexible topology and fine geometric details, outperforming the state-of-the-art methods for both shape generation and shape interpolation.

(a) Input Shapes (b) Reconstructed Shapes

Fig. 26. Failure cases: the headstock of the guitar with a hole, the chair with complex legs and the chair with grid back could not be decoded by our SDM-NET since it requires each part to have the fixed topology, i.e., the same as a genus-zero box.

As future work, our method could be generalized to reconstruct shapes from images. Similar to [Xin et al. 2018], which uses a deep neural network to learn the segmentation masks of cylinder regions from a given image for reconstructing 3D models composed of cylindrical shapes, a possible approach to extend our method is to learn the segmentation of different parts in images and use such segmentation results as the conditions of the SP-VAE for 3D shape reconstruction. In this case, our SDM-NET makes it possible to generate rich shapes with details to better match given images. By exploiting the latent space of our network, our approach could also be generalized for data-driven deformation by incorporating user editing constraints in the optimization framework. It is also interesting to investigate how to extend our method to encode shapes of different categories using a single network. Our current approach can generate parts with the same resolution as the primitive bounding box mesh. We currently utilize a high-resolution mesh with fixed size, and therefore our generated shapes take up large storage space because of richer geometric details and higher resolution of meshes. However, since different kinds of parts have different geometric richness, it would be better to exploit (possibly different types of) primitives with adaptive resolutions for different parts so that we can preserve the same level of details but with significantly less storage space.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (No. 61828204 and No. 61872440), Beijing Municipal Natural Science Foundation (No. L182016), Youth Innovation Promotion Association CAS, CCF-Tencent Open Fund and SenseTime Research Fund. Hongbo Fu was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11237116, CityU 11300615), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning Representations and Generative Models for 3D Point Clouds. In *International Conference on Machine Learning (ICML)*, Vol. 80. 40–49.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. *ACM Trans. Graph.* 24, 3 (2005), 408–416.

Melinos Averkiou, Vladimir G Kim, Youyi Zheng, and Niloy J Mitra. 2014. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Comp. Graph. Forum* 33, 2 (2014), 125–134.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository.* Technical Report arXiv:1512.03012.

Siddhartha Chaudhuri, Daniel Ritchie, Kai Xu, and Hao Zhang. 2019. Learning Generative Models of 3D Structures. In *Eurographics Tutorial.*

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 5939–5948.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 605–613.

Lin Gao, Yu-Kun Lai, Jie Yang, Ling-Xiao Zhang, Shihong Xia, and Leif Kobbelt. 2019. Sparse Data Driven Mesh Deformation. *IEEE Trans. Vis. Comput. Graph.* (2019).

Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L. Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic unpaired shape deformation transfer. *ACM Trans. Graph.* 37, 6 (2018), 237:1–237:15.

Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. 2016. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV).* Springer, 484–499.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems (NIPS).* 2672–2680.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Heli Ben Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. 2018. Multi-chart generative surface modeling. *ACM Trans. Graph.* 37, 6 (2018), 215:1–215:15.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: A Network with an Edge. *ACM Trans. Graph.* 38, 4 (2019), 90:1–90:12.

Kenneth Holmström and Marcus M Edvall. 2004. The TOMLAB optimization environment. In *Modeling Languages in Mathematical Optimization.* Springer, 369–376.

Haibin Huang, Evangelos Kalogerakis, and Benjamin Marlin. 2015. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. *Comp. Graph. Forum* 34, 5 (2015), 25–38.

Jingwei Huang, Hao Su, and Leonidas Guibas. 2018. Robust watertight manifold surface generation method for ShapeNet models. *arXiv:1802.01698* (2018).

Shi-Sheng Huang, Hongbo Fu, Ling-Yu Wei, and Shi-Min Hu. 2016. Support Substructures: Support-Induced Part-Level Structural Representation. *IEEE Trans. Vis. Comput. Graph.* 22, 8 (2016), 2024–2036.

Dominic Jack, Jhony K Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frederic Maire, and Anders Eriksson. 2018. Learning free-form deformations for 3d object reconstruction. In *Asian Conference on Computer Vision.* Springer, 317–333.

Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. 2013. Learning Part-based Templates from Large Collections of 3D Shapes. *ACM Trans. Graph.* 32, 4 (2013), 70:1–70:12.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR).*

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv:1312.6114* (2013).

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas J. Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. Graph.* 36, 4 (2017), 52:1–52:14.

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.* 36, 4 (2017), 71:1–71:10.

Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. 2019. VV-Net: Voxel VAE Net with Group Convolutions for Point Cloud Segmentation. In *IEEE International Conference on Computer Vision (ICCV).*

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 4460–4470.

Niloy Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. 2013. Structure-aware Shape Processing. In *SIGGRAPH Asia 2013 Courses.*

1:1–1:20.
Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. 2019a. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Trans. Graph.* 38, 6 (2019), 242:1–242:29.
Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. 2019b. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 909–918.
Charlie Nash and Christopher KI Williams. 2017. The shape variational autoencoder: A deep generative model of part-segmented 3D objects. *Comp. Graph. Forum* 36, 5 (2017), 1–12.
Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. 2011. Exploration of Continuous Variability in Collections of 3D Shapes. *ACM Trans. Graph.* 30, 4 (2011), 33:1–33:10.
Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 165–174.
Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. 2006. A planar-reflective symmetry transform for 3D shapes. *ACM Trans. Graph.* 25, 3 (2006), 549–559.
Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. 2015. Dyna: A model of dynamic human shape in motion. *ACM Trans. Graph.* 34, 4 (2015), 120:1–120:14.
Adrien Poulenard and Maks Ovsjanikov. 2018. Multi-directional Geodesic Neural Networks via Equivariant Convolution. *ACM Trans. Graph.* 37, 6 (2018), 236:1–236:14.
Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 77–85.
Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. 2016. Volumetric and Multi-view CNNs for Object Classification on 3D Data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5648–5656.
Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.. In *Advances in Neural Information Processing Systems (NIPS)*. 5105–5114.
Ayan Sinha, Jing Bai, and Karthik Ramani. 2016. Deep Learning 3D Shape Surfaces Using Geometry Images. In *European Conference on Computer Vision (ECCV)*. Springer, 223–240.
Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. 2017. Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1511–1519.
Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*. 109–116.
Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3D shape recognition. In *IEEE International Conference on Computer Vision (ICCV)*. 945–953.
Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018. Variational Autoencoders for Deforming 3D Mesh Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5841–5850.
Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2017. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. In *IEEE International Conference on Computer Vision (ICCV)*. 2088–2096.
Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. 2018a. Global-to-local Generative Model for 3D Shapes. *ACM Trans. Graph.* 37, 6 (2018), 214:1–214:10.
Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018c. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conference on Computer Vision (ECCV)*. Springer, 52–67.
Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* 36, 4 (2017), 72:1–72:11.
Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018b. Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Trans. Graph.* 37, 6 (2018), 217:1–217:11.
Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling.. In *Advances in Neural Information Processing Systems (NIPS)*. 82–90.
Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1912–1920.
Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2019. SAGNet: Structure-aware Generative Network for 3D-Shape Modeling. *ACM Trans. Graph.* 38, 4 (2019), 91:1–91:14.
Chen Xin, Yuwei Li, Xi Luo, Tianjia Shao, Jingyi Yu, Kun Zhou, and Youyi Zheng. 2018. AutoSweep: Recovering 3D Editable Objects from a Single Photograph. *IEEE Trans. Vis. Comput. Graph.* (2018).
Sheng Yang, Jie Xu, Kang Chen, and Hongbo Fu. 2017. View suggestion for interactive segmentation of indoor scenes. *Computational Visual Media* 3, 2 (2017), 131–146.
Li Yi, Vladimir G Kim, Duygu Ceylan, I Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. 2016. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.* 35, 6 (2016), 210:1–210:12.
Kangxue Yin, Hui Huang, Daniel Cohen-Or, and Hao Zhang. 2018. P2P-NET: Bidirectional Point Displacement Net for Shape Transform. *ACM Trans. Graph.* 37, 4 (2018), 152:1–152:13.
Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 45–54.
Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. 2014. Real-time Non-rigid Reconstruction Using an RGB-D Camera. *ACM Trans. Graph.* 33, 4 (2014), 156:1–156:12.

APPENDIX: SUPPORT RELATIONSHIP FORMULATION.

Let $\tilde{b}_i$ and $\tilde{b}_j$ be the bounding boxes of parts i and j projected onto the plane orthogonal to the supporting direction. They should satisfy either $\tilde{b}_i \subseteq \tilde{b}_j$ or $\tilde{b}_j \subseteq \tilde{b}_i$. This constraint can be formulated as an integer programming problem and solved efficiently during the optimization as follows:

Let t_1 and t_2 be the two directions in the tangential plane. Denote by $\delta_1^{i,j}$ and $\delta_2^{i,j}$ two auxiliary binary variables, $\delta_1^{i,j}, \delta_2^{i,j} \in \{0, 1\}$, this is equivalent to

$$\begin{aligned}
\mathbf{p}'_j[t_1] - \mathbf{q}'_j[t_1] &\le \mathbf{p}'_i[t_1] - \mathbf{q}'_i[t_1] + M\delta_1^{i,j},\\
\mathbf{p}'_i[t_1] + \mathbf{q}'_i[t_1] &\le \mathbf{p}'_j[t_1] + \mathbf{q}'_j[t_1] + M\delta_1^{i,j},\\
\mathbf{p}'_j[t_2] - \mathbf{q}'_j[t_2] &\le \mathbf{p}'_i[t_2] - \mathbf{q}'_i[t_2] + M\delta_1^{i,j},\\
\mathbf{p}'_i[t_2] + \mathbf{q}'_i[t_2] &\le \mathbf{p}'_j[t_2] + \mathbf{q}'_j[t_2] + M\delta_1^{i,j}, \qquad (5)\\
\mathbf{p}'_i[t_1] - \mathbf{q}'_i[t_1] &\le \mathbf{p}'_j[t_1] - \mathbf{q}'_j[t_1] + M\delta_2^{i,j},\\
\mathbf{p}'_j[t_1] + \mathbf{q}'_j[t_1] &\le \mathbf{p}'_i[t_1] + \mathbf{q}'_i[t_1] + M\delta_2^{i,j},\\
\mathbf{p}'_i[t_2] - \mathbf{q}'_i[t_2] &\le \mathbf{p}'_j[t_2] - \mathbf{q}'_j[t_2] + M\delta_2^{i,j},\\
\mathbf{p}'_j[t_2] + \mathbf{q}'_j[t_2] &\le \mathbf{p}'_i[t_2] + \mathbf{q}'_i[t_2] + M\delta_2^{i,j}, \qquad (6)\\
\delta_1^{i,j} + \delta_2^{i,j} &\le 1, \qquad (7)
\end{aligned}$$

where M is a large positive number (larger than any possible coordinate in the shape), Eq. (7) is true if at most one of $\delta_1^{i,j}$ or $\delta_2^{i,j}$ can be 1, i.e., at least one of them is 0. Without loss of generality, assuming $\delta_1^{i,j} = 0$, then the set of equations in (5) without the term involving M is true, meaning $\tilde{b}_i \subseteq \tilde{b}_j$. Similarly, when $\delta_2^{i,j} = 0$, it satisfies that $\tilde{b}_j \subseteq \tilde{b}_i$.