# SP-GAN: Sphere-Guided 3D Shape Generation and Manipulation

RUIHUI LI, XIANZHI LI, KA-HEI HUI, and CHI-WING FU, The Chinese University of Hong Kong, China

Fig. 1. SP-GAN not only enables the generation of diverse and realistic shapes as point clouds with fine details (see the two chairs on the left and right) but also embeds a dense correspondence across the generated shapes, thus facilitating part-wise interpolation between user-selected local parts in the generated shapes. Note how the left chair's back (blue part in top arc) and the right chair's legs (red part in bottom arc) morph in the two sequences.

We present SP-GAN, a new unsupervised sphere-guided generative model for direct synthesis of 3D shapes in the form of point clouds. Compared with existing models, SP-GAN is able to synthesize diverse and high-quality shapes with fine details and promote controllability for part-aware shape generation and manipulation, yet trainable without any parts annotations. In SP-GAN, we incorporate a global prior (uniform points on a sphere) to spatially guide the generative process and attach a local prior (a random latent code) to each sphere point to provide local details. The key insight in our design is to disentangle the complex 3D shape generation task into a global shape modeling and a local structure adjustment, to ease the learning process and enhance the shape generation quality. Also, our model forms an implicit dense correspondence between the sphere points and points in every generated shape, enabling various forms of structure-aware shape manipulations such as part editing, part-wise shape interpolation, and multi-shape part composition, etc., beyond the existing generative models. Experimental results, which include both visual and quantitative evaluations, demonstrate that our model is able to synthesize diverse point clouds with fine details and less noise, as compared with the state-of-the-art models.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Learning latent representations**; **Neural networks**; **Point-based models**.

Additional Key Words and Phrases: shape analysis and synthesis, generative model, 3D shape generation, 3D shape manipulation, point clouds

Authors' address: Ruihui Li; Xianzhi Li; Ka-Hei Hui; Chi-Wing Fu, The Chinese University of Hong Kong, Hong Kong, China.



**ACM Reference Format:**
Ruihui Li, Xianzhi Li, Ka-Hei Hui, and Chi-Wing Fu. 2021. SP-GAN: Sphere-Guided 3D Shape Generation and Manipulation. *ACM Trans. Graph.* 40, 4, Article 151 (August 2021), 12 pages. https://doi.org/10.1145/3450626.3459766

## 1 INTRODUCTION

A challenging problem in 3D shape creation is how to build generative models to synthesize new, diverse, and realistic-looking 3D shapes, while having structure-aware generation and manipulation. One solution is to decompose shapes into parts [Mo et al. 2019; Wu et al. 2020] and learn to compose new shapes according to part-level relations. With parts correspondence, these approaches can further enable various forms of structure-aware manipulation; however, the granularity of the shape generation is part-based and it depends on the availability and quality of the parts annotations.

Another feasible solution is to design deep generative models to characterize the shape distribution and learn to directly generate shapes in an unsupervised manner. Prior researches generate shapes represented as 3D point clouds, typically by learning to map random latent code to point clouds, via auto-regressive models [Sun et al. 2020], flow-based models [Kim et al. 2020; Klokov et al. 2020; Yang et al. 2019], and generative adversarial nets (GAN) [Achlioptas et al. 2018; Hui et al. 2020; Shu et al. 2019]. Though substantial progress has been made, the problem is still very challenging, due to the diverse shape variations and the high complexity in 3D space. Hence, existing generative models often struggle with the fine details and tend to produce noisy point samples in the generated shapes. Also, existing models lack structure controllability for part-aware generation and manipulation; it is because the learned mapping from a single latent code merely characterizes the overall shape variation,

so it is hard to obtain a plausible correspondence between the parts in generated shapes and the dimensions in latent code, as well as across different shapes generated by the learned mapping.

This paper presents a new GAN model called SP-GAN, in which we use a Sphere as Prior to guide the direct generation of 3D shapes (in the form of point clouds). Our new approach goes beyond generating diverse and realistic shapes, in which we are able to *generate point clouds with finer details and less noise*, as compared with the state-of-the-arts. Importantly, our design facilitates *controllability in the generative process*, since our model implicitly embeds a dense correspondence between the generated shapes, and training our model is *unsupervised*, without requiring any parts annotations. By this means, we can perform *part-aware generation and manipulation of shapes*, as demonstrated by the results shown in Figure 1.

Figure 2 illustrates the key design in SP-GAN. Instead of having a single latent code as input like the conventional generative models, we design our generator with two *decoupled* inputs: (i) *a global prior* $\mathcal{S}$, which is a fixed 3D point cloud in the form of a unit sphere, to provide an isotropic (unbiased) spatial guidance to the generative process; and (ii) *a local prior* $\mathbf{z}$, which is a random latent code to provide local details. We pack these two inputs into a *prior latent matrix* by attaching a latent code $\mathbf{z}$ to every point in $\mathcal{S}$, as illustrated in Figure 2. By this means, the generative process starts from a shared global initialization (*e.g.*, a common ground), yet accommodating the spatial variations, such that every point moves towards its desired location for forming the shape. A key insight behind our design is that we *formulate the generation task as a transformation* and *disentangle the complex 3D shape generation task* into (i) a global shape modeling (*e.g.*, chair) and (ii) a local structure adjustment (*e.g.*, chair's legs). Such a decoupling scheme eases the learning process and enhances the quality of the generated shapes.

Another important consequence is that our new model facilitates structure controllability in the generative process, *through an implicit dense correspondence between the input sphere and the generated shapes*. Metaphorically, the sphere provides a common working space (like the canvas in painting) for shape generation and manipulation; painting on a certain area on the sphere naturally manipulates the corresponding part in different generated shapes. So, if we modify the latent vectors associated with specific points on $\mathcal{S}$ while keeping others unchanged, we can manipulate local structures for the associated parts in the shape. Also, since parts are connected geometrically and semantically with one another, changing one part may cause slight changes in some other parts for structural compatibility; see Figure 1 (bottom). Furthermore, the dense correspondence extends *across all the generated shapes* produced from the generator, with $\mathcal{S}$ serving as a proxy. Hence, we can interpolate the latent code of different generated shapes to morph between shapes in a shape-wise or part-wise fashion. We shall show various structure-aware shape manipulations in Section 5. Further, the experimental evaluations confirm that SP-GAN is able to generate diverse and high-quality point clouds, compared with the state-of-the-art generative models.

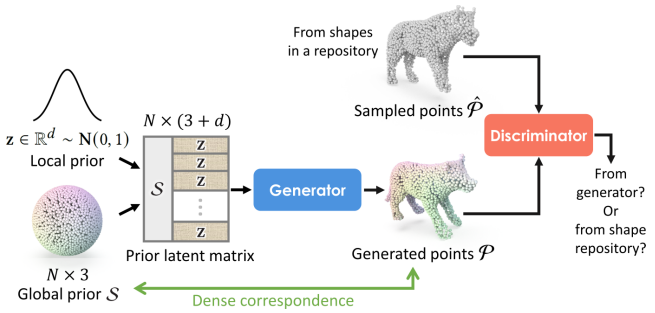



Fig. 2. An overview of SP-GAN. Its input is a *prior latent matrix* with (i) a random latent code $\mathbf{z}$, and (ii) a unit 3D sphere $\mathcal{S}$, which is represented as $N$ points evenly-distributed on the sphere. In SP-GAN, $\mathcal{S}$ provides an *unbiased spatial guidance* to the generator for it to learn to synthesize point cloud $\mathcal{P}$ from $\mathbf{z}$ with finer details and less noise. Also, SP-GAN implicitly embeds a *dense correspondence* between $\mathcal{S}$ and $\mathcal{P}$ (see their colors above), thus promoting structure-aware shape generation and manipulation.

## 2 RELATED WORK

3D shape generation based on deep neural networks has attracted immense research interest. Broadly speaking, it relates to many areas in graphics and vision, for example, 3D shape reconstruction from various forms of input (2D RGB images [Knyaz et al. 2018; Wu et al. 2017; Zheng et al. 2019], 2.5D depth images [Guo et al. 2017; Li et al. 2017; Waechter et al. 2017; Wu et al. 2015; Yang et al. 2018], and 3D scans [Aberman et al. 2017; Hanocka et al. 2020; Yuan et al. 2018]), shape transform [Yin et al. 2019, 2018],etc. In this work, we focus on 3D shape generation, specifically in the context of generating new and diverse shapes that are not necessarily in the given shape repository but yet look realistic when compared with the existing shapes in the repository.

In general, 3D shapes can be generated in a *part-conditioned* or *unconditioned* manner. Part-conditioned means that we employ an additional dataset with parts labels to train a part-wise variational autoencoder to encode each predefined (or learned) part into a latent distribution [Dubrovina et al. 2019; Mo et al. 2019, 2020; Wu et al. 2020], so that new shapes can be generated by sampling from the distributions of parts and composing them. On the other hand, unconditioned means that we directly synthesize 3D shapes from a random distribution, so the generation process has full freedom and the generated samples are not limited by the part-annotation data or any pre-defined structure relation.

Our SP-GAN model is unconditional, so we further review works on unconditional approaches from now on. With advances in direct 3D point cloud processing using deep neural networks, as inspired by pioneering works such as [Qi et al. 2017a,b; Wang et al. 2019], *etc.*, several new approaches were proposed recently to generate 3D shapes in the form of point clouds. These methods can be roughly classified into autoregressive-based, flow-based, and GAN-based.

*Autoregressive-based generative approach* models the joint 3D spatial distribution of points in a point cloud. Specifically, Sun *et al.* [2020] propose PointGrow to estimate the point coordinate distributions from the training shapes. During the generative phase,

points are sampled one-by-one based on the estimated probability distributions given the previously-generated points. However, due to the iterative property intrinsic to autoregressive models, the model cannot scale well with the size of the point cloud.

*Flow-based generative approach* learns to model the distribution of points in a shape mainly by an invertible parameterized transformation of the points. In the generative phase, the approach samples points from a given generic prior (*e.g.*, a Gaussian distribution) and moves them to the target shape using the learned parameterized transformation. For example, Yang *et al.* [2019] generate point clouds from a standard 3D Gaussian prior based on continuous normalizing flows. Klokov *et al.* [2020] relieve the computation by formulating a model using discrete normalizing flows with affine coupling layers [Dinh et al. 2016]. To better characterize the data distribution, Kim *et al.* [2020] propose to learn a conditional distribution of the perturbed training shapes. Cai *et al.* [2020] model the gradient of the log-density field of shapes and generate point clouds by moving the sampled points towards the high-likelihood regions.

While substantial progress has been made, the invertibility constraint in flow-based approach unavoidably limits the representation capability of the models. Also, the learned parameterized transformation can be regarded as a rough estimation of the averaged training data distribution, so the generated point samples tend to be blurry and noisy, as we shall show later in Section 6.

*GAN-based generative approach* explores adversarial learning to train the shape generation model with the help of a discriminator. Achlioptas *et al.* [2018] introduce the first set of deep generative models to produce point clouds from a Gaussian noise vector, including an r-GAN that operates on a raw point cloud input and an l-GAN that operates on the bottleneck latent variables of a pre-trained autoencoder. To overcome the redundancy and structural irregularity of point samples, Ramasinghe *et al.* [2019] propose Spectral-GANs to synthesize shapes using a spherical-harmonics-based representation. Shu *et al.* [2019] propose tree-GAN to perform graph convolutions in a tree and Gal *et al.* [2020] recently extend it into a multi-rooted version. Hui *et al.* [2020] design a progressive deconvolution network to generate 3D point clouds, while Arshad *et al.* [2020] create a conditional generative adversarial network to produce dense colored point clouds in a progressive manner.

Compared with the above GAN-based approaches, which attempt to synthesize point clouds from only a single latent code, we incorporate a fixed prior shape to guide the generative process in the form of a disentanglement model. Our model not only produces high-quality outputs with fine details but also promotes controllability for structure-aware shape generation and manipulation.

Other 3D representations such as voxel grid [Smith and Meger 2017; Wu et al. 2016], implicit function [Chen and Zhang 2019; Deng et al. 2021], and deformable mesh [Groueix et al. 2018; Sinha et al. 2017; Wang et al. 2018] are also explored for shape generation in recent years. Voxels are natural extensions of image pixels, allowing state-of-the-art techniques to migrate from image-space processing to 3D shape processing. However, 3D voxel grids suffer from large memory consumption and low-resolution representation, so the generated shapes contain only coarse structures but not fine details. Implicit-function-based methods [Mescheder et al. 2019; Park et al. 2019] are flexible but computationally expensive, since a large amount of points (e.g., $128^3$) must be sampled to get a surface from the representation, leading to a huge computational resource demand. Mesh-based methods, such as Pixel2Mesh [Wang et al. 2018], learn to deform a surface template to form the target shape given an input image. It is, however, hard to generate and interpolate shapes with arbitrary (and different) genus. Also, the reconstruction is conditional, meaning that it requires reference images and ground-truth 3D shapes, instead of being unconditional as in SP-GAN.

Using points for shape generation has several advantages. First, points are a generic representation for 3D shapes, without constraining the topologies and genus. Also, points are easy to use for shape manipulation and interpolation. Further, points are naturally acquired by scanning without tedious post-processing.

Contemporarily, Deng *et al.* [2021] propose a deformed implicit field representation for modeling dense correspondence among shapes. It learns an implicit-field-based template for each category and deforms the template towards each given shape to build the shape correspondence explicitly. In SP-GAN, we adopt a general "sphere prior" to enable high-quality shape synthesis, in which the dense correspondence is embedded implicitly and automatically via our novel design. Similarly, we noticed that P2P-Net [Yin et al. 2018] attaches an independent noise vector to each point feature for providing more freedom to the displacements of individual points. Differently, we attach noise latent code to each sphere point for providing local details and the point relations together define the global structure of the generated shapes.

## 3 OVERVIEW

The basic model of SP-GAN is illustrated in Figure 2, with shapes represented as point clouds like the previous works we discussed. In SP-GAN, the generator consumes two *decoupled inputs*: a global prior $\mathcal{S}$, which contains the 3D coordinates of $N$ points uniformly on a unit sphere, and a local prior $\mathbf{z}$, which is a $d$-dimensional random latent vector, with each element randomly sampled from a standard normal distribution. Very importantly, we pack one latent vector $\mathbf{z}$ with each point in $\mathcal{S}$ to form the *prior latent matrix* as the generator input; see again Figure 2. During the training, we use the generator to synthesize point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$ from $\mathcal{S}$ and $\mathbf{z}$; besides, we sample another point cloud $\hat{\mathcal{P}} \in \mathbb{R}^{N\times 3}$ from shape in a given 3D repository. So, when we train SP-GAN, the discriminator should learn to differentiate $\mathcal{P}$ and $\hat{\mathcal{P}}$, while the generator should learn to produce $\mathcal{P}$ that looks like $\{\hat{\mathcal{P}}\}$ from the 3D repository.

During the testing, we randomize a latent code and pack it with sphere $\mathcal{S}$ into a prior latent matrix, and feed it to the trained generator to produce a new shape; see $\mathcal{P}_a$ in Figure 3(a). Thanks to the sphere proxy, training SP-GAN creates an *implicit association* between points in $\mathcal{S}$ and points in the generated shape. This is a *dense point-wise correspondence*, as illustrated by the smoothly-varying point colors on the sphere and on the generated shape shown in Figure 3(a). Also, this dense correspondence extends across all the shapes produced from the generator; see the colors of the points in generated shapes $\mathcal{P}_b$ and $\mathcal{P}_c$ in Figure 3(b). With this important

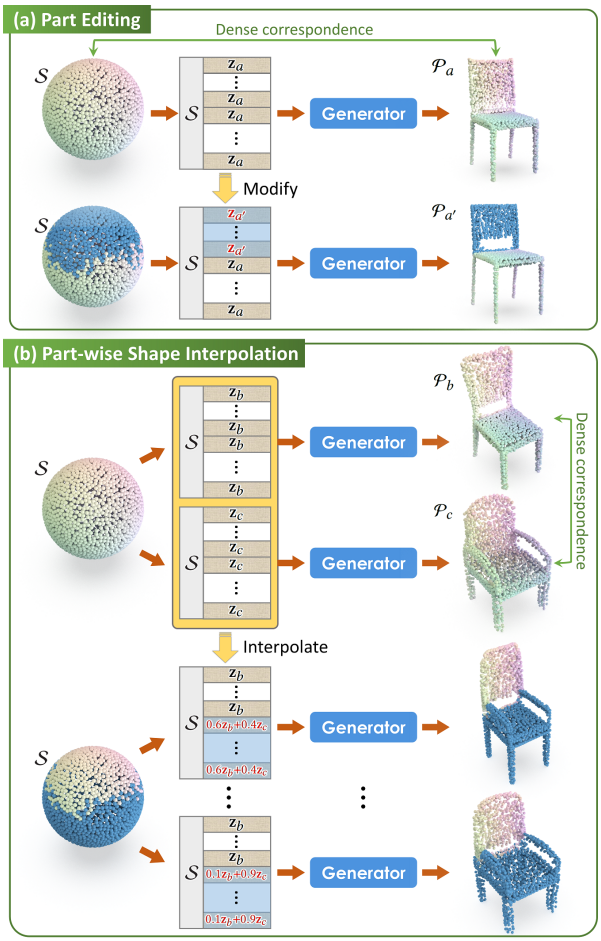



Fig. 3. SP-GAN establishes a dense correspondence implicitly between sphere $\mathcal{S}$ and all the generated shapes, with sphere $\mathcal{S}$ as a proxy. By this model, we can modify or interpolate specific latent code in the prior latent matrix, e.g., for part editing (a) and part-wise shape interpolation (b).

property, SP-GAN facilitates various forms of structure- and part-aware shape generation and manipulation. Below, we first describe two basic cases and more can be found later in Section 5:

- *Part editing.* The latent code associated with each point in sphere $\mathcal{S}$ is a local prior. If we change the latent code for some of the points, we can make local changes on the associated part in the generated shape. Figure 3(a) shows an example, in which we modify the latent code of the points associated with the chair's back (in blue) from $\mathbf{z}_a$ to $\mathbf{z}_{a'}$. By then, the generator produces a new chair $\mathcal{P}_{a'}$, whose back follows $\mathbf{z}_{a'}$ and other parts are slightly adjusted for compatibility with the newly-modified back.
- *Part-wise interpolation.* Further, the dense correspondence enables us to interpolate between the latent code of different generated shapes to morph shapes in part-wise fashion. This manipulation cannot be achieved by any existing unconditional generative models for 3D point clouds. Figure 3(b) shows an example, in which we interpolate the latent code associated with the chair's lower part (marked as blue) between $\mathbf{z}_b$ and $\mathbf{z}_c$ by setting different interpolation weights. With this change, the generator produces a set of new chairs, in which their lower parts morph from $\mathcal{P}_b$ to $\mathcal{P}_c$.

## 4 SP-GAN

In this section, we first present the architecture design of the generator and discriminator networks in SP-GAN, and then present the training and implementation details.

### 4.1 Generator

Figure 4 shows the architecture of the generator in SP-GAN, which produces point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$ from sphere $\mathcal{S}$ and a prior latent matrix. Inside the generator, $\mathbf{z}$ introduces diverse local styles and fine details into point features, whereas $\mathcal{S}$ serves as a prior shape for feature extraction and guides the generative process. Particularly, $\mathcal{S}$ also allows us to employ graph-related convolutions with spatial correlation for feature extraction. This is very different from existing works on 3D point cloud generation, which take only a single latent code as input; hence, existing models can only use fully-connected layers at early stages and require relatively large amount of learnable parameters, yet having limited expressiveness.

To elaborate on how our generator works, let's start with the bottom branch shown in Figure 4. First, the generator employs a graph attention module (to be detailed soon) to extract a point-wise feature map $F_1^G \in \mathbb{R}^{N\times 64}$ from $\mathcal{S}$. Now, to generate 3D shapes, an important question is how to insert the structure variations and local styles brought from latent code $\mathbf{z}$ into feature map $F_1^G$. Inspired by style transfer [Dumoulin et al. 2017; Karras et al. 2019], we propose to use an adaptive instance normalization layer (the left orange box in Figure 4) by modulating the mean and variance of the feature vectors in $F_1^G$. Specifically, as shown in the top branch, we employ a non-linear feature embedding, which is implemented using multi-layer perceptrons (MLPs), to transform the prior latent matrix into a high-level feature map $F^E \in \mathbb{R}^{N\times 128}$. Then, we use a style embedding (the left blue box in Figure 4), which is also implemented using MLPs, to specialize $F^E$ into styles $\mathbf{Y}_1 = (\mathbf{Y}_1^s, \mathbf{Y}_1^b)$, where $\mathbf{Y}_1^s \in \mathbb{R}^{N\times 64}$ and $\mathbf{Y}_1^b \in \mathbb{R}^{N\times 64}$ control the scale and bias, respectively, when normalizing each point feature vector in $F_1^G$. Thus, the adaptive instance normalization operation becomes

$$F_1^A(i) = \mathbf{Y}_1^s(i) \cdot \frac{F_1^G(i) - \mu(F_1^G(i))}{\sigma(F_1^G(i))} + \mathbf{Y}_1^b(i), i \in [1, ..., N], \quad (1)$$

where $F_1^G(i)$ is the $i$-th point feature vector in $F_1^G$, $F_1^A(i)$ is the corresponding feature vector after the normalization, and $\mu(\cdot)$ and $\sigma(\cdot)$ compute the mean and standard deviation across the spatial axes of its argument, which is a feature vector. In general, Eq. (1) allows us to encode the learned per-point style, *i.e.*, $(\mathbf{Y}_1^s(i), \mathbf{Y}_1^b(i))$, into the associated embedded feature vector $F_1^G(i)$, so that the final adjusted $F_1^A$ contains richer local details.

To further enrich the feature embedding, we pass $F_1^A$ to another set of graph attention module and adaptive instance normalization to obtain $F_2^G$ and $F_2^A$, respectively. As shown on the right side of Figure 4, we further regress the output 3D point cloud $\mathcal{P}$ by following

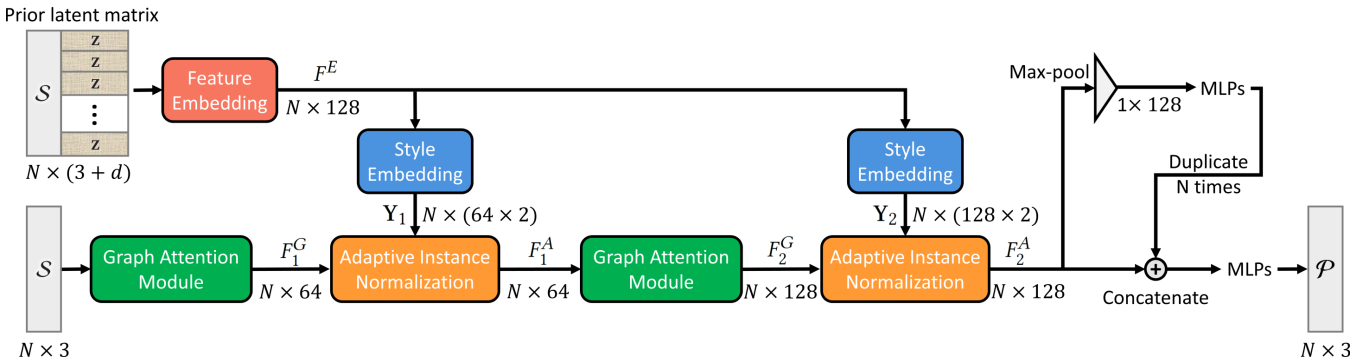



Fig. 4. Architecture of the generator in SP-GAN. With the prior latent matrix as input, the generator first feeds sphere points $\mathcal{S}$ to a graph attention module (green box; see Figure 5) to extract the point-wise feature map $F_1^G$. On the top branch, we use a nonlinear feature embedding (red box) to extract local style features $F^E$ from the prior latent matrix. Then, to fuse the local styles in $F^E$ with the spatial features in $F_1^G$, we propose to embed local styles $Y_1$ from $F^E$ (blue box) and then use adaptive instance normalization (orange box) to produce $F_1^A$ with richer local details. We repeat the process with another round of style embedding and normalization, and then follow PointNet [Qi et al. 2017a] to reconstruct point cloud $\mathcal{P}$ from the embedded feature map $F_2^A$.

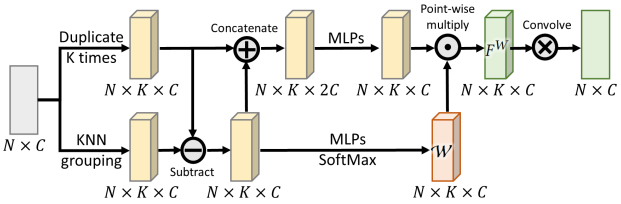



Fig. 5. Architecture of the graph attention module. Besides adopting the basic module (light yellow) in DGCNN [Wang et al. 2019], instead of equally treating the neighbor features, we regress weights $\mathcal{W}$ to obtain the weighted feature map $F^W$ by considering the relations among the $K$ point neighbors.

PointNet [Qi et al. 2017a] to reconstruct the 3D coordinates, *i.e.*, by concatenating $F_2^A$ with the duplicated global feature vectors. Please refer to [Qi et al. 2017a] for the details.

*Graph attention module.* Next, we elaborate on the design of the graph attention module; see Figure 5 for an illustration of its architecture. Here, we adopt the basic module (marked in light yellow) from DGCNN [Wang et al. 2019] and make our adjustments to further account for the relationships among the $K$ neighbors in the feature space. Instead of taking the neighboring features equally, we regress weights $\mathcal{W} \in \mathbb{R}^{N \times K \times C}$ (orange box) and employ point-wise multiplication to obtain the weighted neighbouring feature map $F^W$. Lastly, the output $N \times C$ feature map is obtained by applying a convolution of a kernel size of $1 \times K$ to $F^W$.

## 4.2 Discriminator

Figure 6 shows the architecture of the discriminator in SP-GAN. Its input is either a point cloud $\mathcal{P}$ produced by the generator or a point cloud $\hat{\mathcal{P}}$ sampled from shape in a given 3D repository. From the $N$ points in $\mathcal{P}$ or $\hat{\mathcal{P}}$, a conventional discriminator would learn a $1 \times C$ global feature vector for the whole shape and predict a single score that indicates the source of the point cloud. Considering that the "realism" of an input can be explored by looking into its local details, as inspired by [Schonfeld et al. 2020], we extend the conventional discriminator to additionally perform classifications on a per-point basis, *i.e.*, by predicting a score for each point in the input.

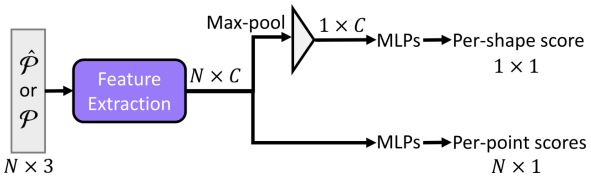



Fig. 6. Architecture of the discriminator in SP-GAN. To classify the input point cloud as being produced by the generator or sampled from a given 3D repository, we predict not only a per-shape score (top branch) but also per-point scores (bottom branch) to encourage fine-grained analysis.

As shown in Figure 6, after we extract point features from the input, we use the top branch to predict a per-shape score and the bottom branch to predict per-point scores. In this way, the discriminator can effectively regularize both global and local variations in the input point cloud. Correspondingly, we can then encourage the generator to focus on both global structures and local details, such that it can better synthesize point clouds to fool this more powerful discriminator. Note that in our implementation, we adopt PointNet [Qi et al. 2017a] as the backbone for the feature extraction, which is the violet box shown in Figure 6.

## 4.3 Training and Implementation details

*Loss functions.* SP-GAN can be trained end-to-end using one loss for generator G and another loss for discriminator D. In this work, we design the two losses based on the least squares loss [Mao et al. 2017], which is originally formulated by minimizing the following competing objectives in an alternating manner:

$$\mathcal{L}_G = \frac{1}{2}[D(\mathcal{P}) - 1]^2 \tag{2}$$

$$\text{and } \mathcal{L}_D = \frac{1}{2}[(D(\mathcal{P}) - 0)^2 + (D(\hat{\mathcal{P}}) - 1)^2], \tag{3}$$

where $D(\mathcal{P})$ and $D(\hat{\mathcal{P}})$ are the confidence values predicted by the discriminator on $\mathcal{P}$ and $\hat{\mathcal{P}}$, respectively, and $\mathcal{L}_G$ and $\mathcal{L}_D$ are the loss for training the generator and discriminator, respectively.

Since our discriminator outputs two types of predictions, *i.e.*, per-shape score and per-point scores. Hence, we model the discriminator

loss $\mathcal{L}_D$ as a summation of shape loss $\mathcal{L}_D^{\text{shape}}$ and point loss $\mathcal{L}_D^{\text{point}}$:

$$\mathcal{L}_D = \mathcal{L}_D^{\text{shape}} + \lambda \mathcal{L}_D^{\text{point}}, \tag{4}$$

$$\mathcal{L}_D^{\text{shape}} = \frac{1}{2}[(D(\mathcal{P}) - 0)^2 + (D(\hat{\mathcal{P}}) - 1)^2], \tag{5}$$

$$\mathcal{L}_D^{\text{point}} = \frac{1}{2N}\sum_{i=1}^{N}[(D(p_i) - 0)^2 + (D(\hat{p}_i) - 1)^2]. \tag{6}$$

where $\lambda$ is a balance parameter; $p_i$ and $\hat{p}_i$ are the $i$-th point in $\mathcal{P}$ and $\hat{\mathcal{P}}$, respectively; and $\mathcal{L}_D^{\text{point}}$ is computed by averaging the predictions of all the points.

Correspondingly, the objective for training the generator becomes

$$\mathcal{L}_G = \frac{1}{2}[D(\mathcal{P}) - 1]^2 + \beta \frac{1}{2N}\sum_{i=1}^{N}[D(p_i) - 1]^2, \tag{7}$$

where $\beta$ is a weight to balance the terms.

*Network training.* Overall, we train SP-GAN by alternatively optimizing the discriminator using Eq. (4) and the generator using Eq. (7). Note also that, we keep sphere $\mathcal{S}$ unchanged during the training and randomly sample a latent code $\mathbf{z}$ from a standard normal distribution in each of the training iterations.

*Network inference.* During the generation phase, we only need to use the trained generator to produce a point cloud, while keeping the same $\mathcal{S}$ as in the training. With different randomly-sampled $\mathbf{z}$ as inputs, the generator can produce diverse point cloud outputs.

*Implementation details.* We implement our framework using PyTorch and train it on a single NVidia Titan Xp GPU using the Adam optimizer. We use a learning rate of $10^{-4}$ to train both the generator and discriminator networks, and follow the alternative training strategy in [Goodfellow et al. 2014] to train each of them for 300 epochs. In both networks, we use LeakyReLU as a nonlinearity activation function. In the last layer of the generator, we use tanh as the activation function. Also, we set $K$=20 to extract point neighborhood. Our code is available for download on GitHub[1].

## 5 SHAPE GENERATION AND MANIPULATION

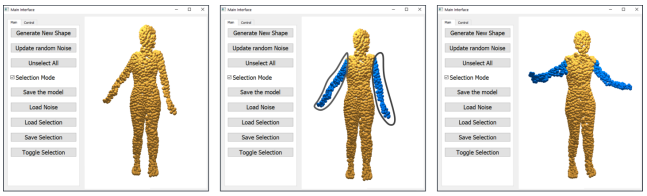
This section presents various forms of shape generation, manipulation, and analysis that are enabled by SP-GAN.

### 5.1 Shape Generation

*Galleries of generated shapes.* Figure 9 showcases varieties of shapes generated by SP-GAN in the form of point clouds, from which we can further reconstruct surfaces using [Liu et al. 2021] (see bottom row). Besides the commonly-used ShapeNet dataset [Chang et al. 2015], we adopted SMPL [Loper et al. 2015] (human body shapes) and SMAL [Zuffi et al. 2017] (animal shapes) as our training data. In detail, on the 3D mesh of each shape, we uniformly sampled $N$ points ($N$ = 2,048 by default) and normalized the points to fit a unit ball. Like the existing generative models [Achlioptas et al. 2018; Hui et al. 2020; Shu et al. 2019], we trained our network on each category separately. Note that we regard different kinds of animals as individual category. During the generation phase, we randomly sample a latent code from a standard Gaussian distribution and take it to SP-GAN to generate the associated shape (point cloud). As shown in Figure 9, our generated shapes exhibit fine details with less noise and also cover a rich variety of global and local structures.

[1]https://github.com/liruihui/SP-GAN

Fig. 7. Visualizing the dense correspondence between the sphere proxy and the generated shapes. Note that same color is assigned to associated points on the sphere proxy and on the generated shapes.




(a) Generate a new shape (b) Lasso to select parts (c) Show manipulation result

Fig. 8. Our interface for interactive shape generation and manipulation. Given a generated shape (a), one can use a lasso tool to select specific parts in the shape (b) and preview the manipulation result (c) in real-time.

*Visualization of the dense correspondence.* To visualize the learned implicit dense correspondence between the points on the sphere proxy $\mathcal{S}$ and the points in the generated shapes, we use same color to render associated points on the sphere and on the generated shapes; see Figure 7 for examples. Checking the point colors across sphere and generated shapes, we can see that different regions of the sphere correspond to different local parts in the shapes. For example, the lower left part of the sphere corresponds to the legs of the chairs. Also, in each category, the same semantic part of different objects, *e.g.*, the back of all chairs and the legs of all animals, receive similar colors. Particularly, as shown in the bottom row, these animals have different poses, yet our model can successfully establish a dense correspondence between the semantically-associated parts. This correspondence property is crucial to enable various structure-aware shape manipulations, as we shall show soon.

### 5.2 Single-shape Part Editing

With a dense correspondence established between the sphere proxy and each generated shape, we can easily locate "local" latent code associated with specific parts in a shape. As illustrated in Figure 3(a), we can perform part editing by re-sampling a new random latent code to replace the existing latent code associated with each specific part in the shape; by then, we can generate a new shape that is similar to the original one but with a different local part.

To facilitate part selection, we built the interactive interface shown in Figure 8, in which one can use a lasso tool to interactively select specific parts in a generated shape. The blue points in (b) indicate the lasso selection. Then, one can click a button to regenerate the shape with a real-time preview (c). Please watch our

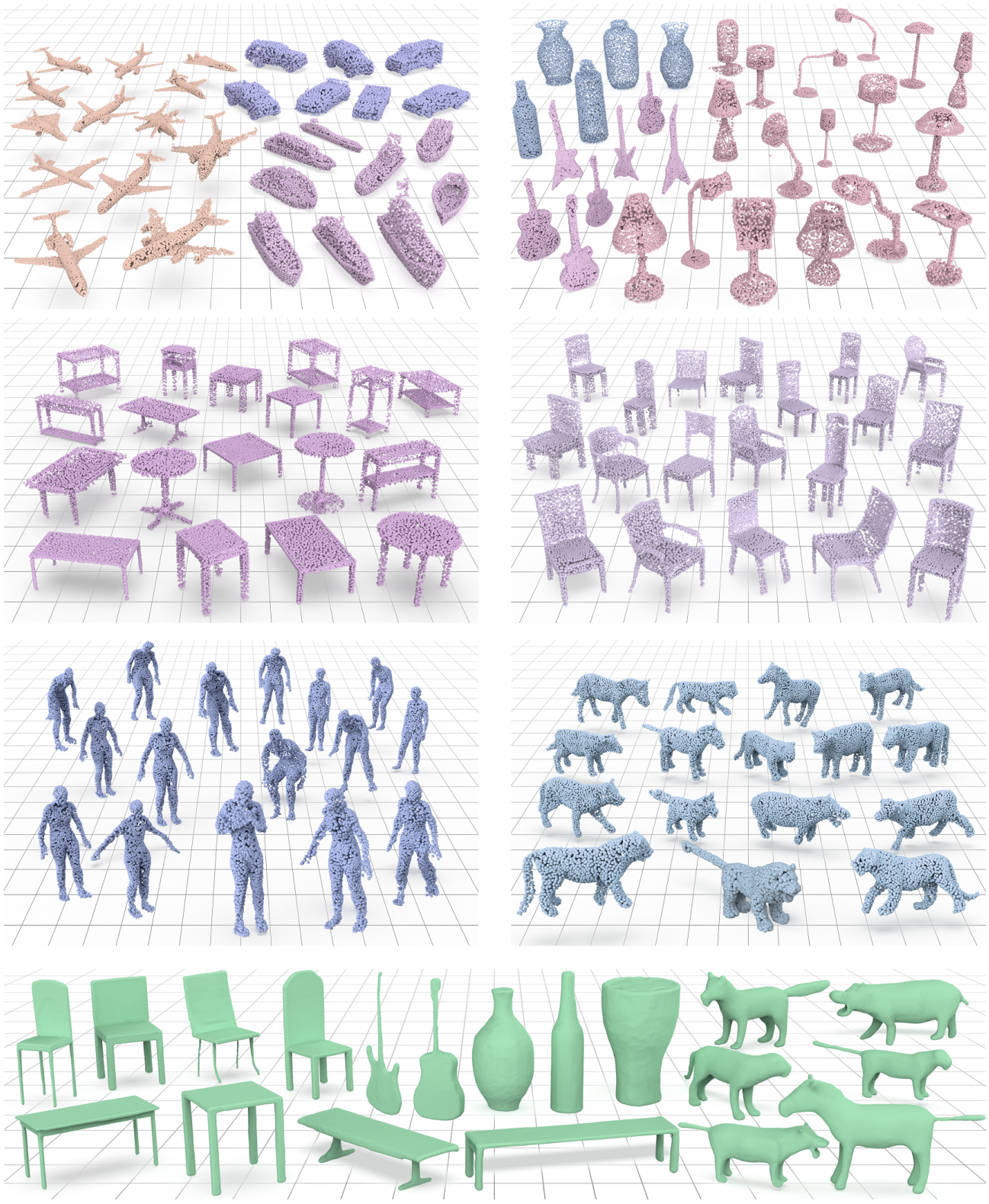


Fig. 9. Galleries showcasing shapes of various categories generated by SP-GAN, and the rich global structures and fine details exhibited in the shapes.

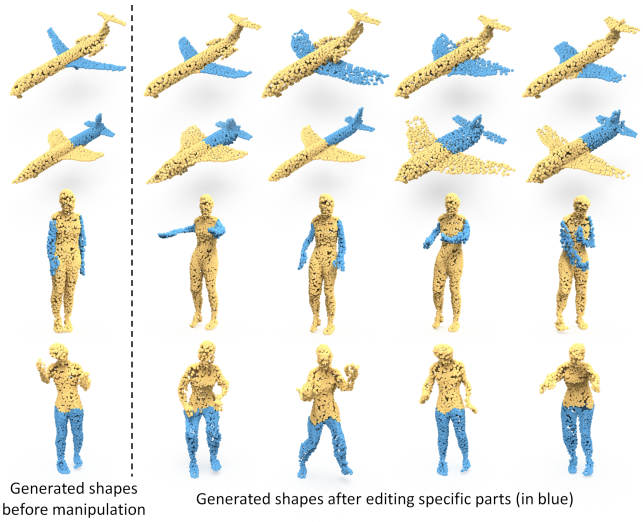



Fig. 10. Part editing examples. Given a generated shape (left), we can select specific parts (blue) and modify their associated latent codes to produce new shapes with different styles (right).

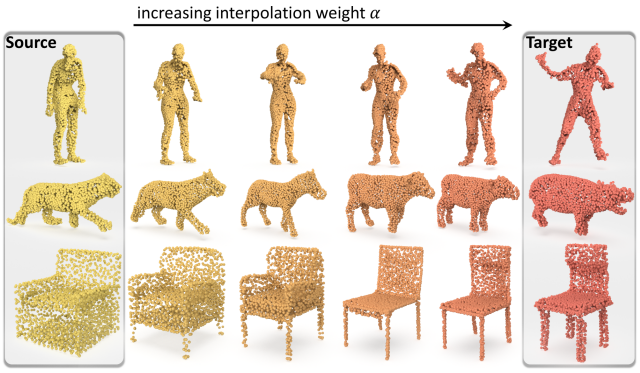



Fig. 11. Shape-wise interpolation between the source (left) and target (right-most) in each row. Note the smooth transition from left to right, including the two human bodies with different poses (top row). Also, the generated intermediate shapes are high-quality with fine details and little noise.

supplemental video. Figure 10 shows more part-zediting examples. On the left of each row is an initial generated shape without any manipulation. By selecting specific parts (blue) in the shape, we can randomly modify the associated latent codes and produce the new shapes shown on the right. We can observe that, after modifying a portion of all the latent codes, the generator will synthesize new shapes with different local styles on the user-marked blue parts, while other object parts (yellow) only exhibit small changes for compatibility with the newly-modified parts.

## 5.3 Shape Interpolation

*Shape-wise interpolation.* Like existing models [Achlioptas et al. 2018; Shu et al. 2019; Yang et al. 2019], we can also linearly interpolate between shapes of different appearance using SP-GAN. Specifically, given two different generated shapes $\mathcal{P}_a$ (source) and $\mathcal{P}_b$ (target) with their associated latent codes $\mathbf{z}_a$ and $\mathbf{z}_b$, respectively,

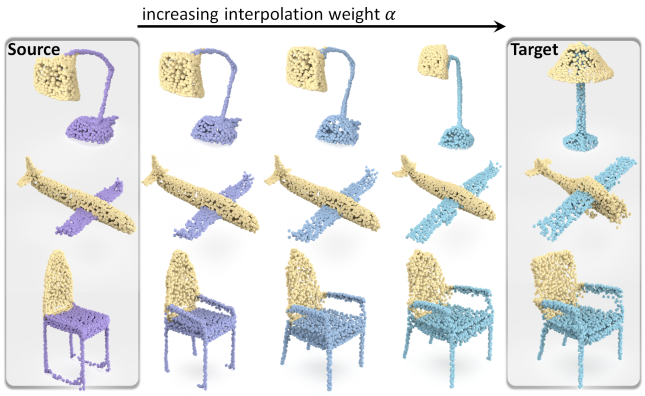



Fig. 12. Interpolating only the latent code associated with the user-selected parts (non-yellow) between the source (left) and target (right) in each row. Note the smooth and part-based transition, as well as the quality of the generated point samples in the interpolated shapes.

the interpolated shape $\mathcal{P}_c$ can be easily generated by feeding the interpolated latent code $\mathbf{z}_c = (1-\alpha)\cdot\mathbf{z}_a + \alpha\cdot\mathbf{z}_b$ into our generator, where $\alpha$ is the interpolation weight ranged from zero to one.

From the interpolation results shown in Figure 11, we can observe that as $\alpha$ increases, SP-GAN enables a smooth transition from the source to the target. See particularly the top row, the source and target reprensent the same human body of two different poses; the interpolation is able to produce rather smooth transitions between the two poses. Distinctively, just like the source and target, the intermediate (generated) shapes also contain fine details with little noise; this is very challenging to achieve, as many existing works easily introduce noise to the interpolated point clouds.

*Part-wise interpolation.* As illustrated in Figure 3(b), thanks to the dense correspondence established across the generated shapes, we can interpolate the latent code associated with specific part in a shape, instead of simply interpolating the whole shape.

Figure 12 shows three sets of part-wise interpolation results, where we interpolate only the latent codes associated with the user-selected points (non-yellow) between the source and target shapes. Here, we render the user-selected points using a color gradient from purple to blue to reveal the interpolation progress. From the results shown in the figure, we can observe that the interpolation happens mainly on the user-selected points, yet the remaining points (yellow) may exhibit small changes for the integrity of the generated shape. For example, in the first row, we select the bracket and base of the source and target lamps for interpolation, so the lamp's shape does not change significantly, while the style of the bracket and base gradually morphs from that of the source to the target.

Figure 13 further shows a two-dimensional part-wise interpolation example. Along each row, we fix the part being interpolated (non-yellow) and gradually increase the interpolation weight $\alpha$. Taking the first row as an example, we only selected the legs for interpolation. As $\alpha$ increases, the legs of the source (top-left) gradually becomes more and more similar to the legs of the target (bottom-right); see also the color gradient from purple to blue. On the other hand, along each column (top to bottom), we fix $\alpha$ but gradually enlarge the selected region. Taking the right-most column as an

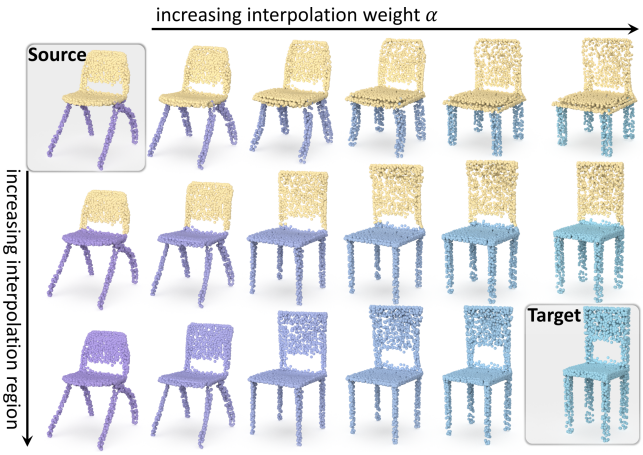



Fig. 13. A two-dimensional part-wise interpolation example. Along each row (left to right), we interpolate the user-selected part, whereas along each column (top to bottom), we gradually enlarge the user-selected region. From the results shown above, we can see the smooth transition and the quality of the interpolated shapes produced by SP-GAN.

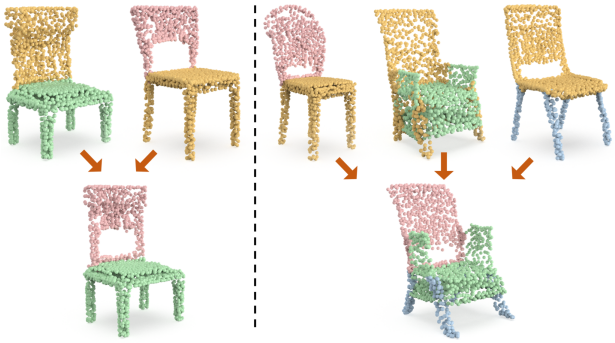


Fig. 14. Two multi-shape part composition examples (left and right), in which the non-yellow parts in the top row are selected in the composition.

example, in which $\alpha = 1$, we can observe that as the selected region (blue) increases, the source gradually deforms further to become the target. Again, we would like to highlight that shapes generated by SP-GAN have fine details and little noise, including also the interpolated ones; see again Figures 12 and 13.

### 5.4 Multi-shape Part Composition

The implicit dense correspondence across multiple generated shapes also facilitates the composition of parts from multiple shapes. Specifically, given different semantic parts (*e.g.*, chair's back, legs, *etc.*) and their associated latent codes from different shapes, we can pack a new prior latent matrix with $\mathcal{S}$ and feed the matrix to our generator to produce new shapes. In this way, the synthesized shape would integrate different local styles from the parent shapes.

Figure 14 presents two examples, where the non-yellow parts in the shapes (top row) are selected for composition. We can observe that, for the two re-generated shapes (bottom row), their local parts generally inherit the styles of the parent shapes.

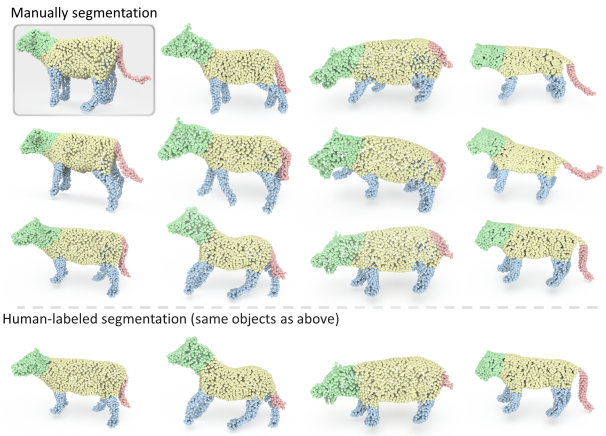



Fig. 15. Top: co-segmentation results relying on the dense correspondence across different generated shapes, where only the top-left object (marked over a gray background) is manually segmented and the others are automatically co-segmented. Note that each column represents a different kind of animal. Bottom row: the associated human-labeled results.

### 5.5 Part Co-Segmentation

Another benefit brought by the implicit dense correspondence is that SP-GAN enables part co-segmentation on a set of 3D shapes, *i.e.*, simultaneous segmentation of shapes into semantically-consistent parts. To do so, we only need to manually segment one of the shapes in the set, then the point-wise labels of the remaining shapes could be simultaneously obtained through the dense correspondence.

Figure 15 showcases an interesting co-segmentation example, in which only the top-left shape was manually segmented and all other shapes in the figure are *automatically* co-segmented. We can observe that, the learned implicit correspondence produced by SP-GAN can successfully facilitate part co-segmentation across shapes, even for shapes with different poses and local part orientations, *e.g.*, see the tails of the animals in the figure. Particularly, different columns actually feature different types of animals, yet our method can still achieve high robustness in co-segmenting these shapes.

Since generative tasks have no ground truths, we cannot directly evaluate the dense correspondence between the generated results. So, we resort to comparing our shape co-segmentation results with human-labeled results. Specifically, we manually label parts in 100 generated shapes of SMAL and calculate the mIoU between the co-segmentation results and human labels. The results are very positive (87.7%). Also, we conducted a user study with 10 participants (who are graduate students aged 23 to 28), to compare the 1-shot co-segmented sample and human-labeled sample; the preference statistics of our result is 48.5%, which is close to a random selection. Figure 15 (bottom) shows some of the human-labeled results.

## 6 EVALUATION AND DISCUSSION

### 6.1 Comparing with Other Methods

We evaluate the shape generation capability of our method against five state-of-the-art generative models, including r-GAN [Achlioptas et al. 2018], tree-GAN [Shu et al. 2019], PointFlow [Yang et al. 2019], PDGN [Hui et al. 2020], and ShapeGF [Cai et al. 2020]. Here,

Table 1. Quantitative comparison of the generated shapes produced by SP-GAN and five state-of-the-art methods, *i.e.*, r-GAN [Achlioptas et al. 2018], tree-GAN [Shu et al. 2019], PointFlow [Yang et al. 2019], PDGN [Hui et al. 2020], and ShapeGF [Cai et al. 2020]. We follow the same settings to conduct this experiment as in the state-of-the-art methods. From the table, we can see that the generated shapes produced by our method have the best quality (lowest MMD, largest COV, and lowest FPD) for both the Airplane and Chair datasets. Also, our method has the lowest complexity (fewest model parameters) and the highest training efficiency (shortest model training time). The unit of MMD is $10^{-3}$.

| Categories | Metrics | Methods | | | | | |
|---|---|---|---|---|---|---|---|
| | | r-GAN | tree-GAN | PointFlow | PDGN | ShapeGF | SP-GAN |
| Airplane | MMD(↓) | 3.81 | 4.32 | 3.68 | 3.27 | 3.72 | **1.95** |
| | COV(%,↑) | 42.17 | 39.37 | 44.98 | 45.68 | 46.79 | **50.50** |
| | FPD(↓) | 3.54 | 2.98 | 2.01 | 1.84 | 1.61 | **0.96** |
| Chair | MMD(↓) | 18.18 | 16.14 | 15.02 | 15.56 | 14.81 | **8.24** |
| | COV(%,↑) | 38.52 | 39.37 | 44.98 | 45.89 | 47.79 | **52.10** |
| | FPD(↓) | 5.28 | 4.44 | 3.83 | 3.77 | 3.52 | **2.13** |
| Parameter size (M) | | 7.22 | 40.69 | 1.61 | 12.01 | 5.23 | **1.01** |
| Training time (Days) | | 0.9 | 4.1 | 2.5 | 2.3 | 1.2 | **0.5** |

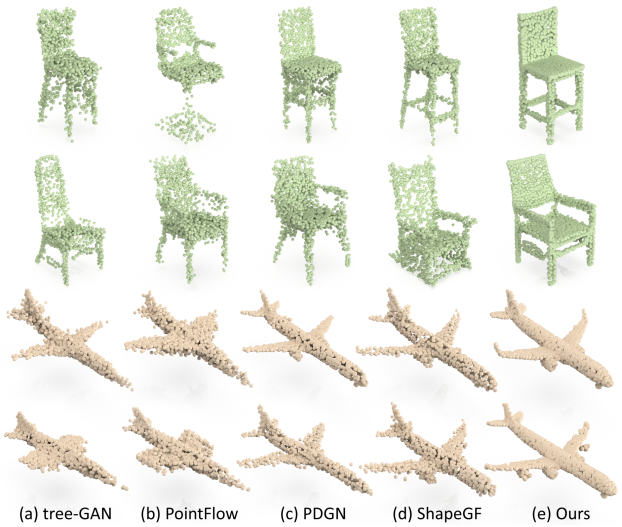



Fig. 16. Visual comparisons with state-of-the-art methods. Clearly, the point clouds generated by our method (e) exhibit more fine details and less noise, while other methods (a)-(d) tend to generate noisy point samples.

we follow the same experimental setup as the above works to conduct this experiment. That is, we trained our SP-GAN model also on the Chair (and Airplane) shapes in ShapeNet [Chang et al. 2015], randomly generated 1,000 shapes for each category, and then evaluated the generated shapes using the same set of metrics as in the previous works (details to be provided below). Also, for the five state-of-the-art models being compared, we directly employ their publicly-released trained network models to generate shapes.

*Evaluation metrics.* Different from supervised tasks, generation tasks have no explicit ground truths for evaluations. Hence, we directly follow the evaluation metrics in the existing works [Achlioptas et al. 2018; Hui et al. 2020; Shu et al. 2019]: (i) Minimum Matching Distance (MMD) measures how close the set of generated shapes is relative to the shapes in the given 3D repository $\{\hat{\mathcal{P}}\}$, so MMD indicates the fidelity of the generated shapes relative to $\{\hat{\mathcal{P}}\}$. (ii) Coverage (COV) measures the fraction of shapes in $\{\hat{\mathcal{P}}\}$ that can be matched (as a nearest shape) with at least one generated shape, so COV indicates how well the generated shapes cover the shapes in $\{\hat{\mathcal{P}}\}$; and (iii) Fréchet Point Cloud Distance (FPD) measures the 2-Wasserstein distance in the feature space between the generated shapes and the shapes in $\{\hat{\mathcal{P}}\}$, where we employ a pre-trained classification model, *i.e.*, DGCNN [Wang et al. 2019], for the feature extraction, so FPD indicates the distribution similarity between the generated shapes and $\{\hat{\mathcal{P}}\}$. For details of these metrics, readers may refer to [Achlioptas et al. 2018; Hui et al. 2020; Shu et al. 2019]. Although these metrics are not absolutely precise, they still reflect the quality of the generated shapes in various aspects. Overall, a good method should have a low MMD, high COV, and low FPD.

*Quantitative evaluation.* Table 1 reports the quantitative comparison results between different methods. For a fair comparison of training time, all methods (including ours) were run on the same desktop computer with the same GPU, and for the five comparison methods, we used code from their project websites. From the results shown in the table, we can see that SP-GAN outperforms all the other methods consistently on the three evaluation metrics for both datasets by a large margin. The low MMD and FPD suggest that our generated shapes have high fidelity compared with the shapes in the 3D repository in both the spatial and feature space, and the high COV suggests that our generated shapes have a good coverage of the shapes in the 3D repository. Also, our network has the fewest learnable parameters and requires much less time to train.

*Qualitative evaluation.* Figure 16 shows some visual comparison results. Here, we pick a random shape generated by our method and use the Chamfer distance to retrieve the nearest shapes generated by each of the other comparison methods. From these results, we can see that the point clouds generated by our method (e) clearly exhibit more fine details and less noise, while other methods (a)-(d) tend to generate noisy point samples.

## 6.2 Ablation Study

To evaluate the effectiveness of the major components in our framework, we conducted an ablation study by simplifying our full pipeline for four cases: (i) replace the graph attention module (GAM) with the original EdgeConv [Wang et al. 2019]; (ii) remove the adaptive instance normalization (AdaIN) and directly take the prior latent

Table 2. Comparing the generation performance of our full pipeline with various simplified cases in the ablation study. The unit of MMD is $10^{-3}$.

| Model | GAM | AdaIN | PPS | Sphere | MMD(↓) | COV(%,↑) | FPD(↓) |
|---|---|---|---|---|---|---|---|
| (i) | | ✓ | ✓ | ✓ | 10.38 | 48.24 | 2.88 |
| (ii) | ✓ | | ✓ | ✓ | 9.54 | 49.72 | 2.66 |
| (iii) | ✓ | ✓ | | ✓ | 9.81 | 50.63 | 2.52 |
| (iv) | ✓ | ✓ | ✓ | | 8.69 | 51.22 | 2.29 |
| Full | ✓ | ✓ | ✓ | ✓ | **8.24** | **52.10** | **2.13** |

matrix as the only input (see Figure 4); (iii) remove per-point score (PPS) (see Figure 6); and (iv) replace the sphere with a unit cube. In each case, we re-trained the network and tested the performance on the Chair category. Table 2 summarizes the results of (i) to (iv), demonstrating that our full pipeline (bottom row) performs the best and removing any component reduces the overall performance, thus revealing that each component contributes.

### 6.3 Shape Diversity Analysis

Further, to show that our generator is able to synthesize diverse shapes with fine details and it does not simply memorize the latent code of the shapes in the training set $\{\hat{\mathcal{P}}\}$, we conducted a shape retrieval experiment. Specifically, we take random shapes produced by our generator to query the shapes in the training set, *i.e.*, we find the shapes in training set $\{\hat{\mathcal{P}}\}$ that have the smallest Chamfer distance with each generated shape. Figure 17 shows some of the shape retrieval results. Compared with the top-five similar shapes retrieved from $\{\hat{\mathcal{P}}\}$, we can see that our generated shapes have similar overall structures yet different local fine details. Particularly, the points in our generated shapes have less noise (compared with those produced by other unsupervised generative models), just like the sampled ones from the training set.

### 6.4 Discussion on Limitations

Our approach still has several limitations. (i) SP-GAN is able to learn in an unsupervised manner but it still requires a large amount of shapes for training. Hence, for shapes with limited training samples or with complex or thin structures, the generated shapes may still be blurry (or noisy); see Figure 18 (left). (ii) Though the point cloud representation is flexible for shape generation and manipulation, we can not directly produce surface or topological information and require a post processing to reconstruct the surface. So, distorted edges and holes could be produced on the reconstructed surface; see Figure 18 (right). In the future, we plan to explore point normals in the generation process to enhance the mesh reconstruction. (iii) Lastly, parts relations are not explicitly or clearly represented in our model, even it embeds an implicit correspondence; looking at the human-labeled samples in Figure 14, we may incorporate certain parts priors into the network to enrich the generation.

## 7 CONCLUSION AND FUTURE WORK

This work presents SP-GAN, a new generative model for direct generation of 3D shapes represented as point clouds. By formulating the generator input using the prior latent matrix, we decouple the input into a global prior (sphere points) and a local prior (random latent code), and formulate the generator network with style embedding and adaptive instance normalization to bring local styles from the random latent code into the point features of the sphere points. This disentanglement scheme articulates the 3D shape generation task as a global shape modeling and a local structure regulation, and enables the generative process to start from a shared global initialization, yet accommodating the spatial variations.

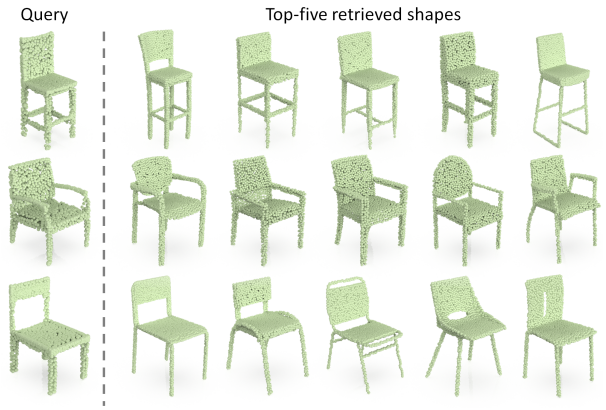



Fig. 17. Shape diversity analysis. We take shapes (left) randomly produced by our generator to query the shapes in the training set (the given 3D repository). The top-five most similar shapes for each case are presented on the right, showing that our generated shapes look similar to the retrieved shapes in overall structure, yet they have different local fine details.

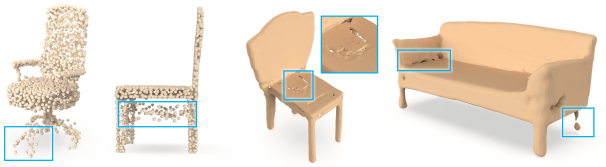


Fig. 18. Failure cases. Left: complex and thin structures may be blurry. Right: distorted edges and holes could be produced on the reconstructed surface.

Very importantly, distinctive from existing works on direct point cloud generation, our new design introduces structure controllability into the generative process through the implicit dense correspondence. So, we can modify or interpolate latent codes in a shape-wise or part-wise manner, and enable various forms of structure-aware shape manipulations that cannot be achieved by previous works on direct point cloud generation. Both quantitative and qualitative experimental results demonstrate that SP-GAN is able to generate diverse, new, and realistic shapes that exhibit finer details and less noise, beyond the generation capability of the previous works.

In the future, we plan to extend our current design by considering different forms of user feedbacks and to enable user-guided automatic shape generation. Also, we would like to explore the possibility of generating 3D shapes represented in other forms, such as polygonal meshes and implicit surfaces.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by Research Grants Council of the Hong Kong Special Administrative Region (Project No. CUHK 14206320 & 14201717 & 14201918).

## REFERENCES


Kfir Aberman, Oren Katzir, Qiang Zhou, Zegang Luo, Andrei Sharf, Chen Greif, Baoquan Chen, and Daniel Cohen-Or. 2017. Dip transform for 3D shape reconstruction. *ACM Transactions on Graphics (SIGGRAPH)* 36, 4 (2017), 79:1–79:11.

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. 2018. Learning representations and generative models for 3D point clouds. In *Proceedings of International Conference on Machine Learning (ICML)*. 40–49.

Mohammad Samiul Arshad and William J. Beksi. 2020. A progressive conditional generative adversarial network for generating dense and colored 3D point clouds. In *International Conference on 3D Vision (3DV)*.

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. 2020. Learning gradient fields for shape generation. In *European Conference on Computer Vision (ECCV)*.

Angel X. Chang, Thomas Funkhouser, Leonidas J. 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).

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.

Yu Deng, Jiaolong Yang, and Xin Tong. 2021. Deformed Implicit Field: Modeling 3D Shapes with Learned Dense Correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real NVP. In *International Conference on Learning Representations (ICLR)*.

Anastasia Dubrovina, Fei Xia, Panos Achlioptas, Mira Shalah, Raphaël Groscot, and Leonidas J. Guibas. 2019. Composite shape modeling via latent space factorization. In *IEEE International Conference on Computer Vision (ICCV)*. 8140–8149.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2017. A learned representation for artistic style. In *International Conference on Learning Representations (ICLR)*.

Rinon Gal, Amit Bermano, Hao Zhang, and Daniel Cohen-Or. 2020. MRGAN: Multi-Rooted 3D Shape Generation with Unsupervised Part Disentanglement. *arXiv preprint arXiv:2007.12944* (2020).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*. 2672–2680.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. A papier-mâché approach to learning 3D surface generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 216–224.

Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. 2017. Real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. *ACM Transactions on Graphics (SIGGRAPH)* 36, 3 (2017), 32:1–32:13.

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: A Self-Prior for Deformable Meshes. *ACM Transactions on Graphics (SIGGRAPH)* 39, 4 (2020), 126:1–126:12.

Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. 2020. Progressive point cloud deconvolution generation network. In *European Conference on Computer Vision (ECCV)*.

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4401–4410.

Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. 2020. SoftFlow: Probabilistic framework for normalizing flow on manifolds. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Roman Klokov, Edmond Boyer, and Jakob Verbeek. 2020. Discrete point flow networks for efficient point cloud generation. In *European Conference on Computer Vision (ECCV)*.

Vladimir A. Knyaz, Vladimir V Kniaz, and Fabio Remondino. 2018. Image-to-voxel model translation with conditional adversarial networks. In *European Conference on Computer Vision (ECCV)*.

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (SIGGRAPH)* 36, 4 (2017), 45:1–45:11.

Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Pengshuai Wang, Xin Tong, and Yang Liu. 2021. Deep Implicit Moving Least-Squares Functions for 3D Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (SIGGRAPH Asia)* 34, 6 (2015), 248:1–248:16.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*. 2794–2802.

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.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J. Guibas. 2019. StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Transactions on Graphics (SIGGRAPH Asia)* 38, 6 (2019), 242:1–242:19.

Kaichun Mo, He Wang, Xinchen Yan, and Leonidas J. Guibas. 2020. PT2PC: Learning to generate 3D point cloud shapes from part tree conditions. In *European Conference on Computer Vision (ECCV)*.

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.

Charles R. 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)*. 652–660.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017b. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Conference on Neural Information Processing Systems (NeurIPS)*. 5099–5108.

Sameera Ramasinghe, Salman Khan, Nick Barnes, and Stephen Gould. 2019. Spectral-GANs for high-resolution 3D point-cloud generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 8169–8176.

Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. 2020. A U-Net based discriminator for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 8207–8216.

Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 2019. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *IEEE International Conference on Computer Vision (ICCV)*. 3859–3868.

Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. 2017. SurfNet: Generating 3D shape surfaces using deep residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6040–6049.

Edward J. Smith and David Meger. 2017. Improved adversarial systems for 3D object generation and reconstruction. In *Conference on Robot Learning*. PMLR, 87–96.

Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. 2020. PointGrow: Autoregressively learned point cloud generation with self-attention. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. 61–70.

Michael Waechter, Mate Beljan, Simon Fuhrmann, Nils Moehrle, Johannes Kopf, and Michael Goesele. 2017. Virtual rephotography: Novel view prediction error for 3D reconstruction. *ACM Transactions on Graphics* 36, 1 (2017), 8:1–8:11.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *European Conference on Computer Vision (ECCV)*. 52–67.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics* 38, 5 (2019), 146:1–146:12.

Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. 2017. MarrNet: 3D shape reconstruction via 2.5D sketches. In *Conference on Neural Information Processing Systems (NeurIPS)*. 540–550.

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 *Conference on Neural Information Processing Systems (NeurIPS)*. 82–90.

Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. 2020. PQ-NET: A generative part Seq2Seq network for 3D shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 829–838.

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.

Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. 2018. Dense 3D object reconstruction from a single depth view. *IEEE Transactions Pattern Analysis & Machine Intelligence* 41, 12 (2018), 2820–2834.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. 2019. PointFlow: 3D point cloud generation with continuous normalizing flows. In *IEEE International Conference on Computer Vision (ICCV)*. 4541–4550.

Kangxue Yin, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Hao Zhang. 2019. LOGAN: Unpaired shape transform in latent overcomplete space. *ACM Transactions on Graphics (SIGGRAPH Asia)* 38, 6 (2019), 198:1–198:13.

Kangxue Yin, Hui Huang, Daniel Cohen-Or, and Hao Zhang. 2018. P2P-Net: Bidirectional point displacement net for shape transform. *ACM Transactions on Graphics (SIGGRAPH)* 37, 4 (2018), 152:1–152:13.

Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. 2018. PCN: Point completion network. In *International Conference on 3D Vision (3DV)*. 728–737.

Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. 2019. DeepHuman: 3D human reconstruction from a single image. In *IEEE International Conference on Computer Vision (ICCV)*. 7739–7749.

Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 2017. 3D Menagerie: modeling the 3D shape and pose of animals. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5524–5532.