RPG: Learning Recursive Point Cloud Generation

Wei-Jan Ko[1] ts771164@gmail.com
Chen-Yi Chiu[1] charles.en07@nycu.edu.tw
Yu-Liang Kuo[1] hank.cs08g@nctu.edu.tw
Wei-Chen Chiu[1] walon@cs.nctu.edu.tw

***Abstract*— In this paper we propose a novel point cloud generator that is able to reconstruct and generate 3D point clouds composed of semantic parts. Given a latent representation of the target 3D model, the generation starts from a single point and gets expanded recursively to produce the high-resolution point cloud via a sequence of point expansion stages. During the recursive procedure of generation, we not only obtain the coarse-to-fine point clouds for the target 3D model from every expansion stage, but also unsupervisedly discover the semantic segmentation of the target model according to the hierarchical/parent-child relation between the points across expansion stages. Moreover, the expansion modules and other elements used in our recursive generator are mostly sharing weights thus making the overall framework light and efficient. Extensive experiments are conducted to show that our point cloud generator has comparable or even superior performance on both generation and reconstruction tasks in comparison to various baselines, and provides the consistent co-segmentation among instances of the same object class.**

I. Introduction

As the increasing usage of 3D sensors nowadays, understanding the 3D data and the rich geometric information of it becomes a crucial problem in many applications such as robotics and autonomous vehicles. Among various formats of 3D data (e.g. meshes, voxels, and point clouds), the point cloud is one of the most intuitive and popular representations, not only because of being the default output of several 3D sensors (e.g. LiDARs) but also its ability of well capturing the geometric details. In this paper, we focus on better understanding the underlying geometry of 3D data via learning the generative procedure of point clouds.

Basically, understanding the geometry of 3D point clouds can be categorized into three levels of granularity from fine to coarse, as sequentially described in the following: **(1)** *Modelling structure details*, where many works [1], [2], [3], [4], [5], [6] proposed for generating or reconstructing point clouds belongs to such level, in which various modelling techniques are adopted to describe the geometric details. For instance, [2] utilizes the transformation of multiple 2D grids/patches to reconstruct the 3D object surface, while [3] builds the probabilistic model based on point densities for generating sophisticated surfaces through sampling; **(2)** *Extracting semantic parts*, where a point cloud is represented as the (hierarchical) combination of semantic parts. For instance, both [7] and [8] propose tree-structured generative models of point clouds, where the former explicitly constructs hierarchical part-based representations while the latter produces part segmentation of the generated point cloud via tracing the tree built upon graph convolutions; and **(3)** *Discovering shape correspondences*, where the co-analysis of multiple point clouds of the same class is applied to discover the correspondences across shapes. For example, [9] learns the co-segmentation while [10] finds the semantic-aware structural points shared across object instances.

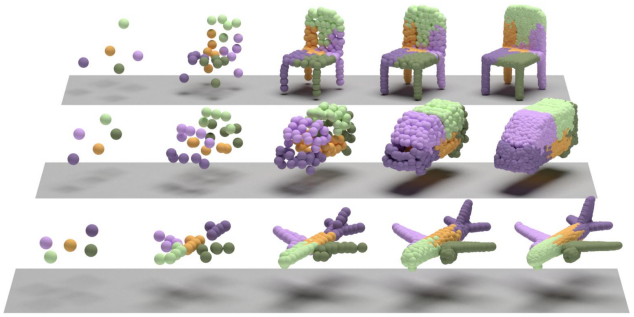


Fig. 1. Our model based on novel design of having recursive point expansion stages is able to not only produce coarse-to-fine generation of point clouds, but also unsupervisedly discover their semantic parts, which provides more fine-grained understanding on the geometry of 3D shapes.

While most of the research works of understanding the geometry of point clouds only focus on one of the aforementioned levels (with few of them taking care of two levels), in this paper we aim to model the generative procedure of point cloud data with including all the granularities in a unified framework. We propose a **recursive point cloud generator (RPG)** which takes a latent vector (standing for the representation of the whole target 3D shape) as input, and generates the point clouds in a coarse-to-fine manner via a sequence of point expansion stages, as shown in Figure 1. Basically, the generation starts from initializing the simplest/coarsest point cloud composed of only a single 3D point at origin, the expansion module at each expansion stage recursively duplicates every point obtained from the previous stage certain times to form a finer point cloud (i.e. having more points to detailedly represent the target 3D shape). The point clouds produced from all the stages naturally form a tree-structure, where we can partition the points at an arbitrary stage according to their ancestor assignments on a previous stage thus resulting the hierarchical part segmentation in an unsupervised way. Moreover, as our RPG is learnt on multiple 3D instances at the same time, the shape correspondences across point clouds can be automatically discovered. With having the expansion modules sharing weights across expansion stages by the recursive nature of our RPG, our proposed framework results to have small model size thus leading to more efficient model training. We experimentally

[1]All authors are with the Department of Computer Science, National Chiao Tung University

∗ Project page: https://odie2630463.github.io/rpg-page/

show that our RPG framework is able to achieve comparable generation and reconstruction performance with respect to the state-of-the-art baselines, and produce compact and semantically-meaningful part segmentation/co-segmentation on point clouds.

II. Related Work

In the following we briefly review the related works for understanding geometry of 3D point cloud data from three granularities, as aforementioned in the introduction.

Reconstruction and Generation. Though the popularity of using point cloud to represent 3D data, the tasks of reconstructing or generating the point cloud data are still considered to be challenging due to its irregularity (e.g. permutation-invariance). Various research works have been proposed these years, for instance: [1] and [2] represent a 3D shape as the deformation of 2D grid points or the 2D parametric surface elements, which however cannot well reconstruct the local details of point cloud data; [4] models the point clouds as a two-level hierarchy of distributions via adopting the frameworks based on continuous normalizing flow; [5] develops a progressive deconvolution network to generate the coarse-to-fine point clouds, which is trained via adversarial learning; and [3] proposes an energy-based probabilistic framework to model the distribution of 3D points via learning the gradient field of the point densities. While these prior works improve the fidelity of generated or reconstructed point clouds over the recent years, the information or the segmentation of semantic parts are not taken into consideration for their model designs.

Hierarchical and Part-based Representation. Treating the 3D shape as a hierarchical decomposition of parts has facilitated humans' understanding on the shape structures, and thus have attracted plenty research attentions. For instance, [11] and [12] represent 3D objects as hierarchical n-ary trees and train the graph neural network [13] and the conditional variational autoencoder respectively to learn the hierarchical structures, where they require the dataset with groundtruth part annotations (e.g. PartNet [14]) for the model learning. In contrast, [7] learns the hierarchical part-based representation of point clouds in an unsupervised manner, where they adopt the tree-structured Gaussian mixture distribution to probabilistically approximate the point cloud surface. However, their model struggles to produce small sharp details in the generated point cloud, as indicated by themselves in [7]. Similarly, TreeGAN [8] learns a point cloud generator built upon tree-structured graph ConvNet, but only outputs the point cloud with part segmentation at the final layer and lacks of the intermediate point clouds during the generation, i.e. having no hierarchical representation.

Cross-Object Shape Correspondence. Discovering the semantic correspondences across 3D object instances helps to investigate and study the variances among 3D shapes thus leading to more high-level understanding. [15], [16], [17] learn the decomposition of shapes with adopting the primitive-based representation, where the parts are however limited to the primitives such as cuboids, superquadrics, or convex hulls. [10] instead proposes an unsupervised method to discover the 3D structural points which are shared among the shape instances with similar structures and show the semantic consistency. And [9] leverages a branched autoencoder to achieve co-segmentation between similar shapes, where each branch represents a recurring part in the implicit field. Although producing semantic correspondences across shapes, [10], [9] do not support the point cloud generation.

In contrast to all the aforementioned works, our proposed recursive point cloud generator (RPG) is able to perform reconstruction and generation of point cloud data with high fidelity, produce hierarchical part-based segmentation without requiring any supervision on part annotations, and discover the semantic correspondence across instances of the same category, with all achieved by a unified lightweight model.

III. Recursive Point Cloud Generator (RPG)

The goal of our proposed method is to unsupervisedly learn a point cloud generator, named as **Recursive Point Cloud Generator** (**RPG**), which is able to generate 3D point clouds in a recursive coarse-to-fine manner. The learning of RPG is based on the autoencoder framework: Given a point cloud data $\mathcal{P}$, an encoder E (which is built upon the PointNet [18] followed by a fully-connected layer in our implementation) first maps it into a latent representation $z \in \mathbb{R}^U$, our proposed RPG then takes z as input and attempts to reconstruct $\hat{\mathcal{P}}$ which ideally should be identical to $\mathcal{P}$. The reconstruction error between $\hat{\mathcal{P}}$ and $\mathcal{P}$ is thus the objective to drive the training of encoder E and our RPG.

Now we detail the generative process of our proposed RPG for generating the point cloud data from the latent representation z. Basically, we decompose the generative process into D sequential stages of point expansion, as formulated in Eq. 1. At each stage d, the 3D points $\mathbb{S}^{(d)}$ together with their corresponding structural representations $\mathbb{H}^{(d)}$ are expanded by $k(d)$ times into $\mathbb{S}^{(d+1)}$ and $\mathbb{H}^{(d+1)}$ respectively via the expansion module $\mathcal{E}$, where $d = 0, 1, \cdots, D-1$. In other words, the number of 3D points in $\mathbb{S}^{(d+1)}$, denoted as $\left|\mathbb{S}^{(d+1)}\right|$, is $k(d)$ times more than the one in $\mathbb{S}^{(d)}$. The 3D points obtained after the last stage of expansion, i.e. $\mathbb{S}^{(D)}$, become the output $\hat{\mathcal{P}}$ of our RPG, in which $\left|\mathbb{S}^{(D)}\right|$ can be also calculated easily by $\prod_{d=0}^{D-1} k(d)$.

$$\begin{aligned} &\text{Stage 0:} && \mathbb{S}^{(1)}, \mathbb{H}^{(1)} = \mathcal{E}(\mathbb{S}^{(0)}, \mathbb{H}^{(0)}) \\ &\text{Stage 1:} && \mathbb{S}^{(2)}, \mathbb{H}^{(2)} = \mathcal{E}(\mathbb{S}^{(1)}, \mathbb{H}^{(1)}) \\ & && \vdots \\ &\text{Stage } D-1: && \mathbb{S}^{(D)}, \mathbb{H}^{(D)} = \mathcal{E}(\mathbb{S}^{(D-1)}, \mathbb{H}^{(D-1)}). \end{aligned} \tag{1}$$

The basic concept behind these sequential stages can be described in an intuitive way, as illustrated in Figure 2: Assuming that we are now asked to build the 3D point cloud of a specific chair, but at the beginning we can only use 5 points to represent the structure of that chair. It then become a reasonable choice to distribute these 5 points to the chair back and 4 chair legs respectively. Such scenario is analogous

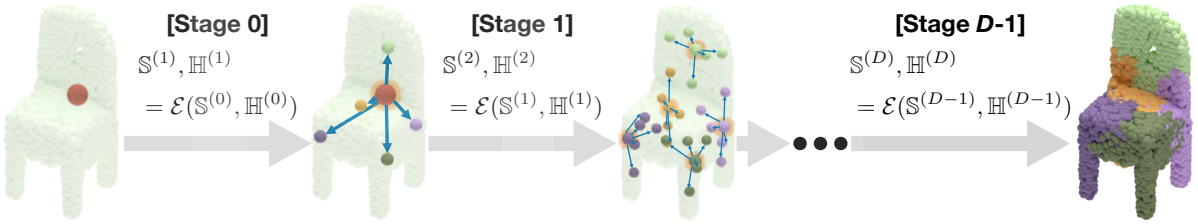


Fig. 2. Illustration for the basic concept behind our proposed Recursive Point Cloud Generator (RPG). Assuming now we are aiming to reconstruct the point cloud of a 3D chair model, with being given its latent/structural representation z. Our generation starts from having $\mathbb{S}^{(0)} = \{\mathbf{0}\}$ and $\mathbb{H}^{(0)} = \{z\}$. The expansion module $\mathcal{E}$ in the Stage 0 firstly expands from the origin point $\mathbf{0}$ into $k(0) = 5$ points $\mathbb{S}^{(1)} = \{s_i^{(1)}\}_{i=1}^{k(0)}$, which are roughly distributed to the locations of the chair back and chair legs, as well as evolves z into $k(0)$ structural representations $\mathbb{H}^{(1)} = \{h_i^{(1)}\}_{i=1}^{k(0)}$ indicating that now the chair model is assembled by $k(0)$ parts. The expansion module $\mathcal{E}$ in the Stage 1 can further expand every point in $\mathbb{S}^{(1)}$ and representation in $\mathbb{H}^{(1)}$ by $k(1)$ times in order to derive $\mathbb{S}^{(2)}$ and $\mathbb{H}^{(2)}$, where now the point cloud for the chair contains more points thus becoming more fine-grained. With having the expansion module recursively applied D times, we obtain $\mathbb{S}^{(D)}$ as the resultant point cloud for the 3D chair model, in which there are $\prod_{d=0}^{D-1} k(d)$ points in the point cloud. The overall procedure of our RPG shows the coarse-to-fine generation of the 3D point cloud, and we can easily obtain the part segmentation of the point cloud according to the corresponding ancestors (e.g. in $\mathbb{S}^{(1)}$) for each point in $\mathbb{S}^{(D)}$, as shown in the right-most figure.

to our stage 0 of point expansion with taking $\mathbb{H}^{(0)} = \{z\}$ and $\mathbb{S}^{(0)} = \{\mathbf{0}\}$, where the $k(0) = 5$ points (denoted as $\mathbb{S}^{(1)} = \{s_i^{(1)}\}_{i=1}^{k(0)}$) are expanded from the origin $\mathbf{0}$ according to the latent structural representation z of the whole chair, and z is evolved into $\mathbb{H}^{(1)} = \{h_i^{(1)}\}_{i=1}^{k(0)}$ representations indicating that now the chair is assembled by $k(0) = 5$ parts. When stepping further with being allowed to use more points to represent each part of this chair, these points would ideally be distributed to better cover the volume of their corresponding parts. Similarly, our stage 1 is analogous to this scenario, in which for every point $s_i^{(1)} \in \mathbb{S}^{(1)}$ we can expand it into $k(1)$ more points as well as evolve its structural representation $h_i^{(1)} \in \mathbb{H}^{(1)}$ into $k(1)$ corresponding structural representations, indicating that now the part related to $s_i^{(1)}$ is represented by $k(1)$ more sub-parts. All the points and structural representations expanded from $\mathbb{S}^{(1)}$ and $\mathbb{H}^{(1)}$ are denoted as $\mathbb{S}^{(2)}$ and $\mathbb{H}^{(2)}$ respectively, in which $\mathbb{S}^{(2)}$ shows more finer-grained 3D point cloud of the chair in comparison to $\mathbb{S}^{(1)}$. Following the similar principle as previous stages, more stages of point expansion can be recursively applied to build the 3D point cloud in a coarse-to-fine manner.

A. Expansion Module $\mathcal{E}$

The architecture of the expansion module $\mathcal{E}$ used in RPG is illustrated in Figure 3. Given a 3D point $s_i^{(d)} \in \mathbb{S}^{(d)}$ with its corresponding structural representation $h_i^{(d)}$ and scaling factor $\alpha_i^{(d)}$, the expansion module $\mathcal{E}$ expands $s_i^{(d)}$, $h_i^{(d)}$, and $\alpha_i^{(d)}$ respectively by $k(d)$ times:

$$\left\{s_m^{(d,i)\to(d+1)}, h_m^{(d,i)\to(d+1)}, \alpha_m^{(d,i)\to(d+1)}\right\}_{m=1}^{k(d)} = \mathcal{E}(s_i^{(d)}, h_i^{(d)}, \alpha_i^{(d)}) \quad (2)$$

where $h_1^{(d,i)\to(d+1)} = h_2^{(d,i)\to(d+1)} = \cdots = h_{k(d)}^{(d,i)\to(d+1)}$. Noting that in Eq. 1 we skip the scaling factors α for simplicity. With denoting all the 3D points and structural representations expanded from $s_i^{(d)}$ and $h_i^{(d)}$ respectively as $\mathbb{S}^{(d,i)\to(d+1)}$ and $\mathbb{H}^{(d,i)\to(d+1)}$, the overall points and their corresponding structural representations obtained at the stage d are then accumulated respectively by:

$$\mathbb{S}^{(d+1)} = \bigcup_{i=1}^{|\mathbb{S}^{(d)}|} \mathbb{S}^{(d,i)\to(d+1)}, \quad \mathbb{H}^{(d+1)} = \bigcup_{i=1}^{|\mathbb{H}^{(d)}|} \mathbb{H}^{(d,i)\to(d+1)} \quad (3)$$

We now provide more detailed explanation for how such expansion module works as well as the physical meanings behind each variable. We say, the parent point of $s_i^{(d)}$ (denoted as $P(s_i^{(d)}) \in \mathbb{S}^{(d-1)}$, which $s_i^{(d)}$ is expanded from) is in charge of representing a certain portion $\mathcal{V}$ of the target 3D model (e.g. a 3D chair). According to Eq. 2, all the sibling points of $s_i^{(d)}$, which are also expanded from $P(s_i^{(d)})$, do share the same structural representation $h_i^{(d)}$ as $s_i^{(d)}$. Therefore, $h_i^{(d)}$ actually carries the structural information of $\mathcal{V}$. Now, when we further expand $s_i^{(d)}$ into $k(d)$ points, a function $\mathcal{F}$ is used to extract from $h_i^{(d)}$ the structural information h' for the sub-portion $\mathcal{V}'$ of $\mathcal{V}$ that $s_i^{(d)}$ is taking care of.

$$h' = \mathcal{F}(s_i^{(d)}, h_i^{(d)}) = \tanh(\mathcal{M}_s * s_i^{(d)} + \mathcal{M}_h * h_i^{(d)}) \quad (4)$$

where $s_i^{(d)} \in \mathbb{R}^{3\times 1}$, $\mathcal{M}_s \in \mathbb{R}^{U\times 3}$, $h_i^{(d)} \in \mathbb{R}^{U\times 1}$, and $\mathcal{M}_h \in \mathbb{R}^{U\times U}$ (U is set to 512 in all our experiments). Then, as all the child points expanded from $s_i^{(d)}$ would share the same h', we duplicate it $k(d)$ times, and denote them as $\mathbb{H}^{(d,i)\to(d+1)} = \{h_m^{(d,i)\to(d+1)}\}_{m=1}^{k(d)}$. Regarding the scaling factor $\alpha_i^{(d)}$ of the point $s_i^{(d)}$, it is to constraint the range of movement for $s_i^{(d)}$ when $s_i^{(d)}$ is expanded from its parent $P(s_i^{(d)})$ as $\mathcal{V}$ is just a portion of the whole target 3D model, i.e. $s_i^{(d)}$ should not exceed the spatial range of $\mathcal{V}$.

Regarding the way of distributing $k(d)$ points in the sub-portion $\mathcal{V}'$ that $s_i^{(d)}$ takes care of, we particularly introduce the embeddings $\{e_m^{(d)}\}_{m=1}^{k(d)}$ which serve as the high-level representations for an initial spatial-arrangement, the concatenation of each $e_m^{(d)}$ with h' is then taken as the input for a multiple-layer perceptron (MLP) to produce the offset $o_m^{(d,i)\to(d+1)} \in \mathbb{R}^{3\times 1}$ and the scaling factor $\alpha_m^{(d,i)\to(d+1)}$ of the m-th point expanded from $s_i^{(d)}$. Please note that now the scaling factor $\alpha_m^{(d,i)\to(d+1)}$ has been multiplied with $\alpha_i^{(d)}$ after MLP (i.e. with denoting the m-th α-related

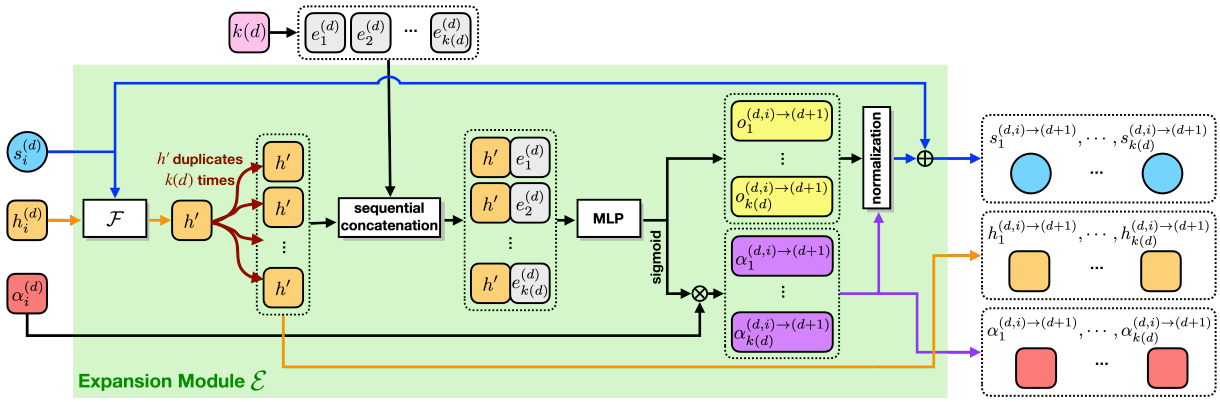


Fig. 3. Architecture of our expansion module $\mathcal{E}$. Please refer to its corresponding paragraphs for more detailed description.

element outputted by MLP$(h', e_1^{(d)}, \cdots, e_{k(d)}^{(d)})$ as $\widetilde{\alpha_m}$, we have $\alpha_m^{(d,i)\to(d+1)} = \alpha_i^{(d)} \cdot \widetilde{\alpha_m}$). The resultant 3D point coordinate $s_m^{(d,i)\to(d+1)}$ of m-th point is computed by:

$$s_m^{(d,i)\to(d+1)} = s_i^{(d)} + \frac{o_m^{(d,i)\to(d+1)}}{o_{\max}} * \alpha_m^{(d,i)\to(d+1)} \tag{5}$$

where $o_{\max} = \max(\{\left\| o_m^{(d,i)\to(d+1)} \right\|\}_{m=1}^{k(d)})$.

Insights. As the points in $\mathbb{S}^{(d)}$ are expanded the ones in $\mathbb{S}^{(d-1)}$, with leveraging the hierarchical/parent-child relation between them, we are able to straightforwardly construct a tree-structure upon all the point clouds (i.e. $\{\mathbb{S}^{(d)}\}_{d=0}^{D-1}$) obtained during the process of our recursive point generator. Note that, such tree-structure will form a perfect k-ary tree if $k(d)$ are equivalent $\forall d$. Moreover, when we consider the path of tracing each point in $\mathbb{S}^{(D)}$ (i.e. leave node) back to the origin point in $\mathbb{S}^{(0)}$ (i.e. root node), the function $\mathcal{F}$ with respect to all the points and their structural representations in such path actually form a simplest recurrent neural network (RNN), in which the points s and structural representations h in the path are the inputs and hidden states respectively, while $\mathcal{M}_s$ and $\mathcal{M}_t$ are the parameters of such RNN.

Lastly, when we consider the cluster assignments for all the points in $\mathbb{S}^{(d)}$ according to their corresponding ancestor points in $\mathbb{S}^{(d')}$, where $d' < d$, we can obtain the segmentation result of $\mathbb{S}^{(d)}$ with $|\mathbb{S}^{(d')}|$ clusters. Examples of using $\mathbb{S}^{(1)}$ as the reference of the ancestor points to perform segmentation on various $\mathbb{S}^{(d)}$ with $d > 1$ are shown in Figure 1.

B. Training Objective

The main objective for RPG training simply adopts the Chamfer Distance (CD) [19] for the reconstruction error between $\mathcal{P}$ and our RPG output $\hat{\mathcal{P}} = \mathbb{S}^{(D)}$.

$$\mathcal{L}_{\text{CD}} = \frac{1}{|\mathcal{P}|} \sum_{s \in \mathcal{P}} \min_{s' \in \hat{\mathcal{P}}} ||s - s'||_2^2 + \frac{1}{|\hat{\mathcal{P}}|} \sum_{s' \in \hat{\mathcal{P}}} \min_{s \in \mathcal{P}} ||s - s'||_2^2 \tag{6}$$

Moreover, we introduce a regularization term to generally encourage smaller scaling factors, in order to reduce the potential overlapping between different portions (i.e. $\mathbb{S}^{(d,i)\to(d+1)}$ and $\mathbb{S}^{(d,j)\to(d+1)}$, $i \neq j$) on each stage.

$$\mathcal{L}_{\text{reg}} = \frac{1}{D} \sum_{d=0}^{D-1} \left(\frac{1}{|\mathbb{S}^{(d)}|} \sum_{i=1}^{|\mathbb{S}^{(d)}|} \mathbb{A}^{(d)} \right) \tag{7}$$

where $\mathbb{A}^{(d)}$ denotes all the scaling factors obtained at stage d. The overall objective then becomes $\mathcal{L}_{\text{CD}} + \lambda \mathcal{L}_{\text{reg}}$, where λ is set to 5e$-$5 in our experiments.

Please especially note that, the parameters of $\mathcal{F}$ (i.e. $\mathcal{M}_s$ and $\mathcal{M}_t$) and the ones of MLP are shared globally across all the stages of point expansion, while the embeddings $\{e_m^{(d)}\}_{m=1}^{k(d)}$ in stage d are shared among all expansion operations of the points in $\mathbb{S}^{(d)}$ (where $e_m^{(d)} \in \mathbb{R}^{64\times 1}$ in our experiments). The initial value for the scaling factor α^0 at the stage 0 is set to 1.

IV. Experiments

Datasets. The experiments and evaluations are conducted on ShapeNet [20], which is a large scale dataset composed of 51,127 pre-aligned 3D shapes from 55 categories. We follow the experimental setting as [3], [4] to have 35,708/5,158 shapes for the train/test split of ShapeNet. Since the shapes in ShapeNet are originally in the format of meshes, we follow the common practice as [18] to sample 2048 points randomly on the triangles from the meshes in order to build up the point cloud data of each shape for our experiments.

Evaluation metrics. We follow the prior works [3], [4] to adopt the symmetric Chamfer Distance (CD) for evaluating the performance of point cloud reconstruction. Regarding the point cloud generation, we also follow [3], [4] to use the Minimum Matching Distance (MMD), Coverage (COV), and 1-Nearest Neighbor Accuracy (1-NNA) as metrics to access the quality of generated point clouds. While taking the test split of ShapeNet as the reference set, MMD and 1-NNA are the smaller the better, while COV is the larger the better.

Implementation Details Two variants are adopted for our RPG training. The first one, denoted as RPG^{3125}, is trained on the point cloud data with having 3125 points per point cloud, where there are 5 expansion stages with setting all $k(d)$ equally to 5. Another one, denoted as RPG^{2048}, is instead based on the points clouds of 2048 points, where there are also 5 expansion stages but setting $k(1)$ to 8 and

$k(2)$=$k(3)$=$k(4)$=$k(5)$ to 4. We mainly adopt RPG^{2048} for evaluating the tasks of reconstruction and generation in order to have fair comparison to the baselines (however please note that the performances of our RPG^{2048} and RPG^{3125} are almost the same), while using RPG^{3125} to illustrate the part segmentation for better visualization (where the segmentation into $k(1) = 5$ parts is easier for clear colorization of points).

TABLE I

ARCHITECTURE OF THE ENCODER E.

Layer	channel	kernel
Conv1d	128	1x1
BatchNorm1d	128	
ReLU		
Conv1d	128	1x1
BatchNorm1d	128	
ReLU		
Conv1d	256	1x1
BatchNorm1d	256	
ReLU		
Conv1d	512	1x1
BatchNorm1d	512	
MaxPooling1d		
Conv1d	256	1x1
BatchNorm1d	256	
ReLU		
Conv1d	128	1x1
BatchNorm1d	128	
ReLU		
Conv1d	128	1x1
Linear	512	

The architecture of the encoder E used in our method is shown in Table I. Regarding the expansion module $\mathcal{E}$, as described and illustrated in Section III-A and Figure 3, there are two subnetworks to be learned, i.e. function $\mathcal{F}$ and a multiple-layer perceptron MLP. The function $\mathcal{F}$ has parameters $\mathcal{M}_s \in \mathbb{R}^{512\times3}$ and $\mathcal{M}_h \in \mathbb{R}^{512\times512}$, while the multiple-layer perceptron MLP has three hidden layers, where each layer has 512 neurons and followed by a ReLU activation function. The input and output size of MLP are 576 and 4 respectively. For all the experiments, we use the AdamW optimizer with learning rate $1\mathrm{e}{-3}$ and batch size 64. The model is trained for 2000 epochs.

A. Reconstruction

We compare the reconstruction ability of auto-encoding the point cloud among our proposed RPG and various baselines, i.e. AtlasNet [2] with patches and with sphere, PointFlow [4] (abbreviated as PF), and the current state-of-the-art ShapeGF [3], based on the whole test set of ShapeNet as well as the three categories of it (i.e. Airplane, Car, and Chair). Please note that, we follow the same experimental setting as [4], [3].

The reconstruction errors in terms of Chamfer distance as well as the number of parameters for each method are provided in Table II. It is shown that our RPG has the comparable performance of point cloud reconstruction with respect to all the baselines, while having only 1.8M model parameters (with 0.7M from the PointNet encoder E), being almost half of the current state-of-the-art ShapeGF [3]. In particular, although PF [4] baseline has smaller number of model parameters than ours, but our reconstruction performance is superior to it on all the datasets. Qualitative examples of reconstruction produced by our RPG are provided in Figure 4 (more can be found in supplementary videos).

B. Generation

We follow the experimental setting as [3] to conduct the evaluation in terms of generation ability on two categories

TABLE II

EXPERIMENTAL RESULTS OF THE RECONSTRUCTION TASK CONDUCTED ON SHAPENET DATASET AND THREE CATEGORIES OF IT. THE RECONSTRUCTION ERRORS BETWEEN THE INPUT AND RECONSTRUCTED OUTPUT ARE GIVEN IN CHAMFER DISTANCE MULTIPLIED BY 10^4.

Dataset	Metric	AtlasNet Sphere	AtlasNet Patches	PF	ShapeGF	Ours
Airplane	CD	1.002	0.969	1.208	0.96	0.924
	EMD	2.672	2.612	2.757	2.562	3.336
Chair	CD	6.564	6.693	10.120	5.599	7.493
	EMD	5.790	5.509	6.434	4.917	5.502
Car	CD	5.392	5.441	6.531	5.328	5.711
	EMD	4.587	4.570	5.138	4.409	4.573
ShapeNet	CD	5.301	5.121	7.551	5.154	5.607
	EMD	5.553	5.493	5.176	4.603	4.536
Params		12M	12M	1.5M	3.5M	1.8M

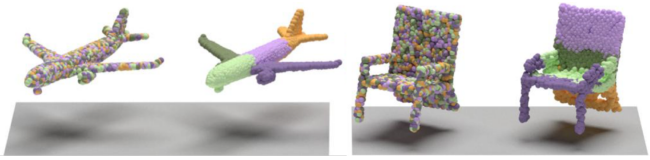
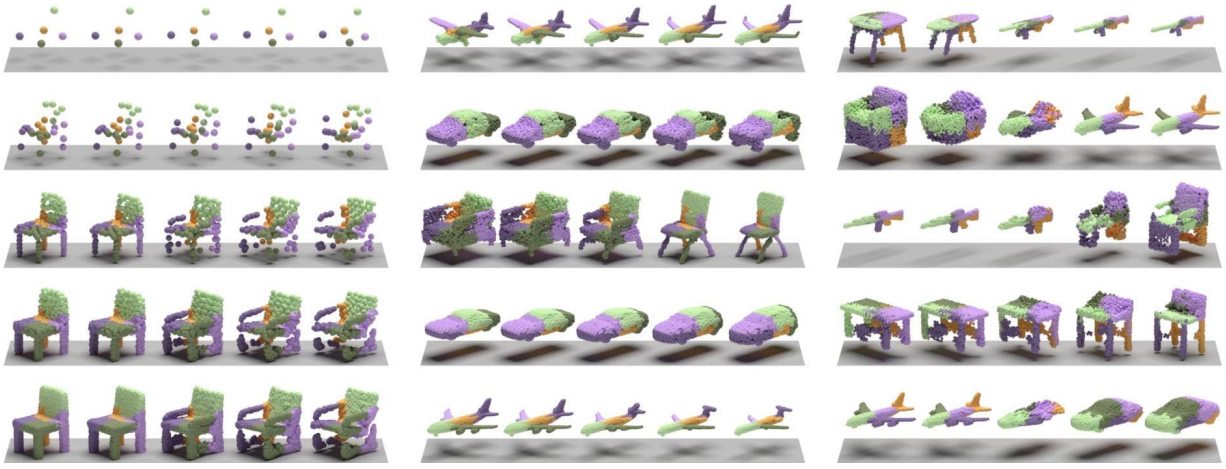


Fig. 4. Examples of reconstruction by our RPG. For each example pair, the shape on the left is the input while the output is on the right.

(i.e. Airplane and Chair) in ShapeNet. Several baselines are adopted for making comparison, including r-GAN [6], GCN [21], TreeGAN [8], PF [4], and ShapeGF [3]. In order to equip our RPG framework with the ability of generation, our model training is extended from the autoencoder to the variational autoencoder [22], where the distribution of latent representations of 3D shapes are modelled as a normal distribution, thus our RPG can take noise vectors sampled from such normal distribution as $\mathbb{H}^{(0)}$ and output the generated point clouds. The quantitative results of generation quality for various approaches as well as their model sizes are provided in Table III. We can see that, our proposed RPG framework achieves superior performance with respect to r-GAN, GAN, TreeGAN, and PF on almost all the metrics for both categories. While our RPG has comparable performance with ShapeGF, the amount of parameters of our RPG model is smaller than all the baselines thus contributing to less training time and lower hardware requirements. We also provide the qualitative examples of point clouds generated by our proposed RPG in Figure 6 to visualize their quality as well as the diversity. Moreover, in Figure 5(b) we provide the 5 example sets of interpolation between two 3D shapes of the same category (i.e. shapes generated from the interpolation between two noise vectors which can lead to the point clouds of the same category), where we can see the smooth and semantically-consistent transition between generated point clouds during the interpolation. In Figure 5(a) we visualize the point clouds generated on all the expansion stages for interpolated shapes. While in Figure 5(c) we provide even the examples of interpolation between different object categories. More examples for generation and interpolation are provided in the supplementary video.



(a) (b) (c)

Fig. 5. Examples for interpolation between shapes: (a) Rows sequentially show the point clouds generated on all the expansion stages while interpolating between the chairs on the bottom-left and bottom-right corners; (b) Each row shows interpolation between two 3D shapes of the same object class; (c) Each row shows interpolation between two shapes from different categories.

TABLE III

EXPERIMENTAL RESULTS OF THE GENERATION TASK CONDUCTED ON THE AIRPLANE AND CHAIR CATEGORIES OF SHAPENET DATASET. THE MMD NUMBERS ARE MULTIPLIED WITH 10^4 HERE.

	Model	Params	MMD(↓)	COV (↑)	l-NNA(↓)
Airplane	r-GAN	6.9M	1.657	38.52	95.80
	GCN	7.0M	2.623	9.38	95.16
	TreeGAN	40M	1.466	44.69	95.06
	PF	1.5M	1.408	39.51	83.21
	ShapeGF	3.5M	1.285	47.65	85.06
	Ours	1.1M	1.288	42.47	84.57
Chair	r-GAN	6.9M	18.187	19.49	84.82
	GCN	7.0M	23.098	6.95	86.52
	TreeGAN	40M	16.147	40.33	74.55
	PF	1.5M	15.027	40.94	67.60
	ShapeGF	3.5M	14.818	46.37	66.16
	Ours	1.1M	15.686	45.62	66.39

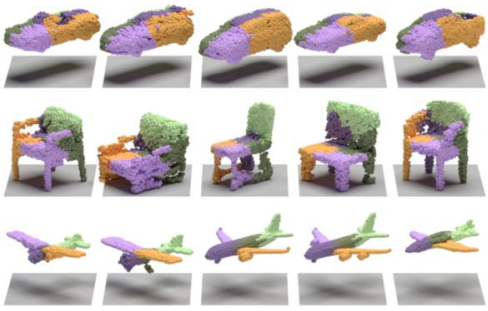


Fig. 6. Qualitative examples of the point clouds generated by our proposed recursive point cloud generator (RPG).

C. Part Segmentation and Co-Segmentation

As what can be seen from both Figure 7 and 5, the points in the generated point clouds are colorized according to their ancestor assignments in $\mathbb{S}^{(1)}$, which particularly highlight the most important feature, i.e. part segmentation, that distinguishing our proposed RPG from the baselines such as PointFlow [4] and ShapeGF [3] that can only generate point clouds without having any part segmentation.

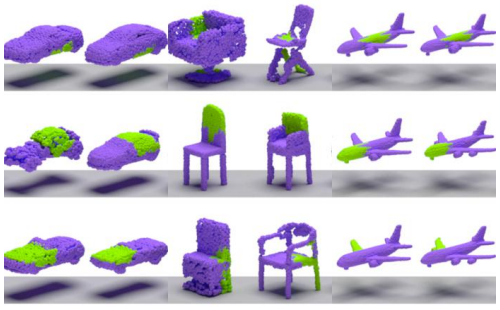


Fig. 7. Visualization of co-segmentation results among object instances from Car, Chair and Airplane categories in ShapeNet. For each object category, the rows sequentially highlight different common parts with green color shared across the instances.

In particular, the part segmentation is naturally obtained during the inference of our RPG, from the tree-structure built upon the point clouds produced by the sequential expansion stages without any supervision nor annotation on parts.

Moreover, as the generative procedures of different object instances are learnt by the same recursive process of expansions in our RPG framework, regardless of the tasks of generation or reconstruction, we are therefore able to discover the common patterns of part segmentation shared across object instances thus straightforwardly achieving the co-segmentation, where the common parts (e.g. the left and right wings, tail, and nose of airplanes) are segmented out among objects as shown in Figure 7. The co-segmentation helps us to understand the intra-class variance and similarity from a more fine-grained perspective according to the common parts (e.g. different forms of wings among airplanes).

As described in Section II, both TreeGAN [8] and PointGMM [7] are able to produce the part segmentation together with the generated point cloud, without requiring the groundtruth part annotations during their model training. We therefore provide more detailed analysis on part segmentation with respect to related works here.

Given the officially released model of TreeGAN [8], we first use it to randomly generate 64 samples of point clouds.

For each of the generated point cloud $\mathcal{P}$, we retrieve the most similar shape $\mathcal{P}'$ of $\mathcal{P}$ from the ShapeNetPart [23] dataset. As TreeGAN is able to provide the part segmentation for $\mathcal{P}$ and there are part annotations in ShapeNetPart dataset, we hence aim to investigate the consistency on the part segmentation of $\mathcal{P}$ with respect to $\mathcal{P}'$. Basically, for each point $s \in \mathcal{P}$, we assign it a label according to the annotation of its nearest neighbor $s' \in \mathcal{P}'$ provided by in ShapeNetPart. Assume there are $\mathcal{C}$ parts produced by TreeGAN for $\mathcal{P}$, where each part c has n_c points. The purity r_c of each part c is measured as a fraction of l_c to n_c, where l_c is the largest number of points being assigned to the same label (i.e. majority vote). We then measure the quality of part segmentation on $\mathcal{P}$ using the weighted purity (which is widely adopted for accessing the clustering results, the higher the better): $\mathcal{W} = \frac{1}{N}\sum_c n_c \cdot r_c$, where N is the total number of points in $\mathcal{P}$. In order to make fair comparison between TreeGAN and our RPG by using the same set of $\mathcal{P}'$ in ShapeNetPart, we input $\mathcal{P}'$ to our proposed framework (composed of encoder E and RPG decoder) to obtain $\hat{\mathcal{P}}'$. By adopting the same aforementioned procedure, we can also compute the weighted purity for $\hat{\mathcal{P}}'$. Please especially noting that, as TreeGAN has no encoder, we hence adopt such experimental setting in order to have both TreeGAN and our segmentation results being compared upon the same ground truth point cloud from ShapeNetPart. We hypothesize that the evaluation metric for semantic segmentation in terms of the weighted purity ideally should be unbiased towards the quality of point clouds, regardless the generation or reconstruction modes. Overall, **the average weighted purity over all the generated point clouds from TreeGAN is 74%, which is inferior than our RPG of 88%**, showing that our RPG is able to achieve more consistent part segmentation in comparison to TreeGAN, based on the ShapeNetPart groundtruth annotation. We also provide some qualitative examples of comparing part segmentation between TreeGAN and our RPG at the 25 second of supplementary video.

Furthermore, as TreeGAN shares several similarities in high-level concepts with our RPG, here we particularly highlight the differences. Basically, TreeGAN adopts the adversarial learning framework to train the generator built upon tree-structured graph convolution network, where multiple layers of branching operations and the graph convolutions are applied to generate the final point cloud. Due to its tree-structured design, the part segmentation of the generated point cloud is also available. Our work distinguishes from TreeGAN by the following aspects: **(1)** As the graph convolutions used in different layers of TreeGAN are not shared, its model size is relatively large; while our RPG framework introduces the novel recursive expansion module which is repeatedly applied on all the expansion stages (i.e. weights sharing), the model size of our RPG is much smaller than TreeGAN (cf. Table IV) thus leading to more efficient training; **(2)** Our RPG framework not only provides better performance in generation (cf. Table IV) but also shows more compact and semantically-meaningful part segmentation, as shown in Figure 8. We can see that TreeGAN's part segmentation would mix between the chair back and chair surface (or between chair legs and chair surfaces) or even produce some subtle parts, while ours better decomposes chair legs, surface, and back into different parts thus getting closer to the human interpretation. **(3)** TreeGAN can only output point cloud on the final layer of generator, while our RPG is able to obtain all the intermediate point clouds thus providing better understanding on the hierarchical segmentation and generative procedure of point cloud data.

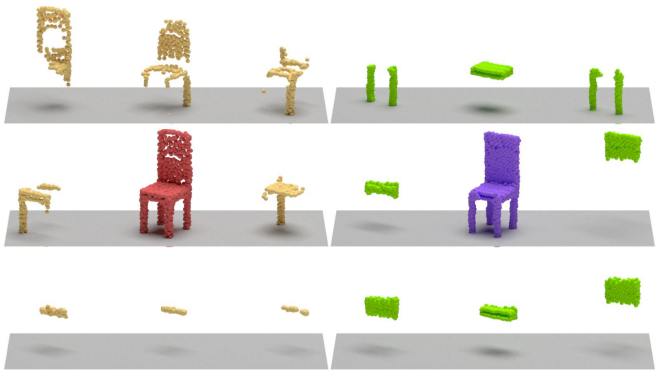


Fig. 8. Comparison between part segmentations obtained from TreeGAN (left) and our RPG (right), where the center image of the left or right part is the generated 3D shape, surrounded by its 8 segmented parts. It shows that our RPG provides more compact and semantically-meaningful parts.

Similarly, given the officially released model of PointGMM [7], we compare the average weighted purity between PointGMM and our RPG, and we obtain 80% for both PointGMM and RPG. Some qualitative examples of comparing part segmentation between PointGMM and our RPG are provided at the 28 second of supplementary video. Please note that, as PointGMM generally generates point clouds with more diverse shapes than the ones from TreeGAN (in which such situation is observable when comparing between the 25 and 28 second of supplementary video) , the set of $\mathcal{P}'$ retrieved from ShapeNetPart will be also more complicated thus being harder to obtain high value of the weighted purity. Please also note that, for TreeGAN, PointGMM, and our RPG, we all obtain the segmentation of point cloud into $\mathcal{C} = 8$ parts in order to have fair comparison. Although PointGMM has almost the same weighted purity as our RPG, indicating that PointGMM can also produce good part segmentation results, the quality of its generated point clouds is actually much worse than our RPG, as shown in Table IV and the 28 second of supplementary video. Noting that, in Table IV we provide the quantitative results of generation quality for various approaches (including TreeGAN [8], PointGMM [7], PF [4], and ShapeGF [3]) as well as their model sizes. In brief, our proposed RPG framework is able to produce better generation quality, superior or comparable part segmentation results, with much smaller model size in comparison to TreeGAN [8] and PointGMM [7].

V. Additional Qualitative Results In Supplement

In supplementary video we provide more qualitative examples for the reconstruction at the 31 second, the generation

TABLE IV

EXPERIMENTAL RESULTS OF THE GENERATION TASK CONDUCTED ON THE AIRPLANE AND CHAIR CATEGORIES OF THE SHAPENET DATASET. THE MMD NUMBERS ARE MULTIPLIED WITH 10^4 HERE.

	Model	Params	MMD(↓)	COV (↑)	l-NNA(↓)
Airplane	PF [4]	1.5M	1.408	39.51	83.21
	PointGMM [7]	10M	16.000	30.62	95.31
	TreeGAN [8]	40M	1.466	44.69	95.06
	ShapeGF [3]	3.5M	1.285	47.65	85.06
	Ours	1.1M	1.288	42.47	84.57
Chair	PF [4]	1.5M	15.027	40.94	67.60
	PointGMM [7]	10M	81.000	41.09	78.42
	TreeGAN [8]	40M	16.147	40.33	74.55
	ShapeGF [3]	3.5M	14.818	46.37	66.16
	Ours	1.1M	15.686	45.62	66.39

with comparing to several baselines on Airplane and Chair categories at the 34 and 37 second respectively, and the generation produced by our RPG^{3125} and RPG^{2048} at the 40 and 43 second respectively. Moreover, as our RPG has high flexibility and is efficient to learn and generate high-resolution point clouds, at the 46 second we particularly provide an example point cloud with 15625 points generated by a RPG which has the same architecture and the number of parameters as our RPG^{3125}. Additional qualitative examples of the intra-category and the inter-category interpolations are shown at the 48 and 50 second respectively. In particular, we provide the interpolation examples with more/denser intervals in at the 52 second, where we can find them changing quite smoothly thus verifying again the capacity of our RPG in terms of modelling the generative process and the distribution of point clouds. Also, more qualitative results of the co-segmentation are demonstrated at the 56 second. Moreover, we provide examples of point clouds generated by our RPG, with respective colorizations to visualize the part segmentation and co-segmentation at the 1 and 18 second.

VI. CONCLUSION

We propose the recursive point cloud generator (RPG) which is able to generate or reconstruct point clouds in a coarse-to-fine manner. Based on the generative procedure with multiple expansion stages, our RPG produces point clouds with semantic parts without requiring any supervision of part annotations. Our RPG achieves comparable performance against state-of-the-art baselines on point cloud generation and reconstruction, with having relatively small model size thanks to the design of sharing weights of the expansion module across all the stages. Moreover, the part segmentation and co-segmentation of point clouds naturally obtained from our RPG provide the potential of understanding point clouds from a more fine-grained perspective.

Acknowledgement. This project is supported by MOST 111-2636-E-A49-003 and MOST 111-2628-E-A49-018-MY4.

REFERENCES

[1] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[2] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan, "Learning gradient fields for shape generation," in *European Conference on Computer Vision (ECCV)*, 2020.

[4] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[5] L. Hui, R. Xu, J. Xie, J. Qian, and J. Yang, "Progressive point cloud deconvolution generation network," in *European Conference on Computer Vision (ECCV)*, 2020.

[6] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *International Conference on Machine Learning (ICML)*, 2018.

[7] A. Hertz, R. Hanocka, R. Giryes, and D. Cohen-Or, "Pointgmm: A neural gmm network for point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] D. W. Shu, S. W. Park, and J. Kwon, "3d point cloud generative adversarial network based on tree structured graph convolutions," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[9] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "Bae-net: Branched autoencoder for shape co-segmentation," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[10] N. Chen, L. Liu, Z. Cui, R. Chen, D. Ceylan, C. Tu, and W. Wang, "Unsupervised learning of intrinsic structural representation points," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. Mitra, and L. Guibas, "Structurenet: Hierarchical graph networks for 3d shape generation," *ACM Transactions on Graphics (TOG), SIGGRAPH Asia*, 2019.

[12] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. J. Mitra, and L. J. Guibas, "Structedit: Learning structural shape variations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[14] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "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)*, 2019.

[15] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, "Learning shape templates with structured implicit functions," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[16] D. Paschalidou, A. O. Ulusoy, and A. Geiger, "Superquadrics revisited: Learning 3d shape parsing beyond cuboids," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 1977.

[20] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," arXiv preprint, Tech. Rep. 1512.03012, Dec. 2015.

[21] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized generative models for 3d point clouds via graph convolution," in *International Conference on Learning Representations (ICLR)*, 2019.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *ArXiv:1312.6114*, 2013.

[23] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (ToG)*, 2016.