Single Image 3D Reconstruction: Rethinking Point Cloud Deformation

Anh-Duc Nguyen, Seonghwa Choi, Woojae Kim, Jongyoo Kim, *Member, IEEE*, Heeseok Oh, Jiwoo Kang, and Sanghoon Lee, *Senior Member, IEEE*

***Abstract*—Single image 3D reconstruction has long been a challenging problem. Recent deep learning approaches have been introduced to this 3D area, but the ability to generate point clouds still remains limited due to either the inefficient and expensive 3D representations, the dependency between the output and number of model parameters or the lack of a suitable computing operation. In this paper, we present a novel deep-learning-based method to reconstruct a point cloud of an object from a single still image. The proposed method can be decomposed into two steps: feature fusion and deformation. The first step extracts both global and point-specific shape features from a 2D object image, then injects them into a randomly generated point cloud. In the second step, which is deformation, we introduce a new layer termed as GraphX that considers the inter-relationship between points like common graph convolutions but operates on unordered sets. The framework can be applicable to realistic image data with background as we optionally learn a mask branch to segment objects from input images. To complement the quality of point clouds, we further propose an objective function to control the point uniformity. Additionally, we introduce different variants of GraphX that cover from best performance to best memory budget. Moreover, the proposed model can generate an arbitrary-sized point cloud, which is the first deep method to do so. Extensive experiments demonstrate that we outperform existing models and set a new height for different performance metrics in single image 3D reconstruction.**

***Index Terms*—3D reconstruction, point cloud, deep learning, convolutional neural network.**

I. Introduction

The world is 3D, and so is human perception. Making machines see the world like human is the ultimate goal of computer vision. Humans are superior at understanding the underlying 3D space just by looking at a 2D image thanks to their ability to learn from experiences, and machines are still nowhere near humans' perception level. Thus, a crucial yet demanding question is whether we can help machines to achieve a similar 3D understanding and reasoning abilities. So far, we have made significant advancements in 2D machine vision tasks, and yet 3D reasoning from 2D still remains very challenging due to the loss of depth information. Even though 2D images do not contain absolute depth information, they still convey much relative one through various cues that machines can learn.

Early on, given multiple 2D images having different viewpoints, computers are able to estimate the shape of the interested object [1,2], but they suffer various visual problems related to holes and geometry, and some approaches may only be applied in a studio environment. Different from the aforementioned works, in this paper, we tackle the problem of single image 3D reconstruction. Although 2D visual signals discard occluded information of the 3D world, the shape of an object can be conveyed through different 2D factors, which constitutes a family of methods called "Shape from X". This family recovers 3D structures by assuming (or knowing) some knowledge about lighting condition and object surface, and hence, these methods usually work best in a controlled environment.

The availability of voluminous data sparks a new generation of learning systems that utilize deep convolutional neural networks (CNNs). However, there is not yet an easy and efficient way to apply them to 3D reconstruction. Most modern progress of deep learning is in areas where signals are ordered and regular; for *e.g.*, images, audios and languages [3]–[15], while common 3D representations such as meshes or point clouds are unordered and irregular. Therefore, there is no guarantee that all the bells and whistles from the 2D practice would work in the 3D counterpart. Some 3D structures may enable easier learning such as grid voxels but not only such structures are expensive but also quantization errors may diminish the natural invariance of the data [16]. Besides, some early works that applied deep learning into single image 3D reconstruction [17]–[20] synthesize an output point cloud directly, so the output size is constrained by the hardware memory. They also mostly consider synthetic data, so in order to apply to realistic scenes, they require segmentation masks of object, which are usually not available in practice.

A. Proposed Framework

To address all the aforementioned problems, we present a novel deep method that reconstructs a 3D representation of an object from a single 2D image. In this work, we consider point cloud as our output representation as it is simple and

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A2C3011697) and the Yonsei Signature Research Cluster Program of 2022 (2022-22-0002).

A.-D. Nguyen, S. Choi, W. Kim, and S. Lee are with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea ({adnguyen, csh0772, wooyoa, slee}@yonsei.ac.kr).

J. Kim is with Microsoft Research Asia in Beijing 100080, China (jongk@microsoft.com).

H. Oh is with Hansung University, Seoul 136792, South Korea (ohhs@hansung.ac.kr).

J. Kang is with the Division of Artificial Intelligence Engineering, Sookmyung Women's University, Seoul 04310, South Korea (e-mail: jwkang@sookmyung.ac.kr).

Corresponding authors: Sanghoon Lee (slee@yonsei.ac.kr) and Jongyoo Kim (jongk@microsoft.com)

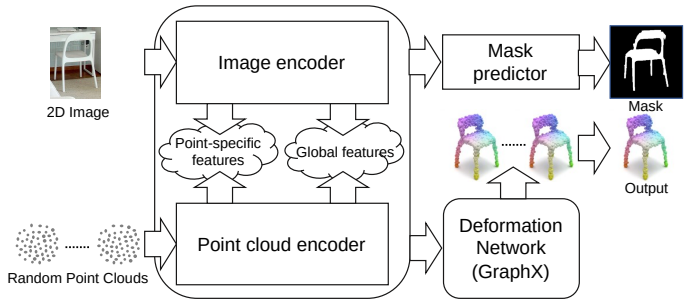


Fig. 1: A schematic diagram of our proposed method. Given a 2D image and a randomly initialized point cloud, an image encoder and point cloud encoder extract features and blend together to transfer the shape information to the point cloud. Then, the blended feature is fed to a deformation module to produce the final point cloud prediction.

sufficiently efficient when it comes to transformation and deformation, and can replicate fine details of an object surface. Meanwhile, volumetric representations are expensive to store in memory, and it is difficult to directly output meshes with diverse topologies.

Considering existing works, we identify several desired key properties of the prospective system as follows: (1) the model should make predictions based on not only local features but also high-level semantics, (2) the model should consider the spatial correlation between points, (3) the method should have a trade-off between performance and resources, (4) the output point clouds should be of high quality, (5) the method should be scalable, *i.e.*, the output point cloud can be of arbitrary size, and last but not least (6) the framework should have a built-in solution to deal with realistic scenes. To inherit all these properties, we propose to approach the problem in two steps: *feature fusion* and *deformation*. A schematic drawing of our approach can be seen in Fig. 1. Conceptually, the feature fusion step extracts features from the input image and a randomly initialized point cloud, and then fuse the two features together. The fused features are used in the deformation step to output a point cloud prediction.

In feature fusion, we first extract features from the given input image and a randomly initialized point cloud. Concretely, we develop two types of blended features, which are a *point-specific feature* and a *global feature*. The point-specific feature is obtained by projecting the 3D initial point cloud on to the 2D image feature maps. As points can be projected inside or outside of object's silhouette, this feature helps the network to distinguish the two possibilities so that the network can move points to the surface correctly. To achieve a global feature, inspired by a simple but powerful idea from image style transfer literature, we propose to transfer the shape information from the 2D feature maps to the point set by renormalizing point cloud features using statistics from the 2D feature maps. Since the global feature is a summarization of features over the whole spatial locations, the network is more aware of the overall shape. Additionally, we propose to learn a segmentation mask to deal with images containing background when ground truth segmentation masks are available, which is often overlooked in existing methods. This helps the network to be more shape-aware in the existence of background, and thus the performance can be much improved.

When generating points in a point cloud, it is necessary to take the inter-connection between them because any point can contain much information about other points. Thus, in the deformation step, we introduce a new layer termed GraphX that learns the non-local relationship among points like common graph convolutions [21] but can operate on unordered point sets. We build different variants of the deformation module and study their trade-offs, ranging from best performance to best memory usage.

As observed in previous works [17,22], generated point clouds typically do not exhibit a grid-like uniform distribution. To fix this problem, we propose a uniformity regularization that penalizes points that are too close to each other. We define the concept of local variance of distances and enforce every neighborhood shares the same variance. Thus, the result point clouds exhibit a more regular pattern as can be seen in Fig. 4.

Previous deep learning works output a point cloud prediction given an input image, so the size can only be as large as the hardware allows. By contrast, as our framework deforms a probabilistic initial point cloud, it is capable of producing a point cloud of arbitrary size by deforming multiple random point clouds. Indeed, we demonstrate that a variant of the deformation module allows arbitrary-sized point cloud generation via multiple forward passes, which is the first deep method to do so according to our knowledge.

We dub the proposed method Point Cloud Deformation NETwork v2 (PCDNetv2) for brevity. This work is an extensive and thorough update of our previously published work [23,24] with the following major contributions

- We refine feature extraction and adapt a better training regime. As a result, training time is much improved and memory consumption is lowered while maintaining the performance level.
- We predict a segmentation mask for input image in order to deal with realistic images having background compared with existing works that mainly consider synthetic data so far.
- We propose a uniformity regularization to enhance the quality of reconstructed point clouds.
- We propose several variants of GraphX ranging from best performance to best memory consumption to enable a wider usage coverage for the proposed layer.
- We present in-depth experiments and analyses regarding the proposed components in our framework.

II. Related Work

A. Multi-view 3D reconstruction

Traditionally, much research has been devoted to multi-view reconstruction following a matching step [25]. SfM [1] has been the most well-known method in this category. Some modern works rely on a photo-consistency principle [2,26]–[30] or graphs [31,32]. Still, these algorithms are critical of different problems such as dependency on matching accuracy, computational complexity, lack of details, and so on.

B. Single-view 3D reconstruction

A sibling to multi-view 3D reconstruction is single-view reconstruction. Shape-from-X [33,34] is perhaps the most popular approach to this problem. Shape-from-Shading, the most famous algorithm in this family, relies on the changes of the image brightness in the 2D domain to calculate the orientations of the surface points corresponding to the image pixels, which can be used to recover a depth map of the scene via integration. Despite its simple principle, its requirements including light sources and/or albedo maps may only be satisfied in a studio environment. In contrast, our method does not assume any condition regarding light sources or shading.

C. 3D reconstruction from depth

3D scenes can be recovered from depth maps, which can be obtained via either specialized hardwares [35] or estimation methods [25]. While the former incurs much infrastructure cost, the latter suffers many difficulties, and hence the estimates are quite noisy.

D. Learning-based 3D reconstruction

Rather than manually crafting priors, some early studies consider learning them from data. Notably, Saxena *et al.* [36] constructed a Markov random field to model the relationship between image depth and various visual cues to recreate a 3D "feeling" of the scene. In a similar study, the authors in [37] learned different semantic likelihoods to achieve the same goal. These models are quite simple and cannot learn sophisticated structures.

E. Deep learning-based 3D reconstruction

Deep learning-based methods can reconstruct an object from a single image by learning the geometries of the object from image(s). Below, we will categorize recent methods based on their output representations.

1) Volumetric: Wu *et al.* [38] employed a conditional deep belief network to model volumetric 3D shapes. Yan *et al.* [39] introduced an encoder-decoder network regularized by a perspective loss to predict 3D volumetric shapes from 2D images. In [40], the authors utilized a generative model to generate 3D voxel objects arbitrarily. Tulsianni *et al.* [41] introduced ray-tracing into the picture to predict multiple semantics from an image including a 3D voxel model. Tulsiani *et al.* [42] learned 3D shapes in an unsupervised way by a ray consistency loss with different views. Howbeit, voxel is known to be inefficient and computationally unfriendly [17,43], which makes it harder to scale beyond 32^3 or 64^3.

2) Mesh: Wang *et al.* [43] and Wen *et al.* [44] gradually deformed an elliptical mesh given an input image by using graph convolution. Mesh R-CNN [45] first generates a coarse voxel representation of the object, then meshing it and apply a similar processing pipeline as in [43] for refinement to obtain the final mesh. While mesh representation requires overhead construction in the former, the meshing operator introduced in [45] is not differentiable, and hence the final mesh quality still largely depends on the accuracy of the coarse voxel representation.

3) Implicit shapes: Implicit shape models have received much focus recently [46]–[49]. As the name implies, it takes more non-trivial effort to convert it into explicit 3D models but these implicit representations are very useful in rendering.

4) Point clouds: The earliest work by Fan *et al.* [17] directly regresses an image to a point cloud using a sophisticated encoder-decoder structure. Sun *et al.* [18] took an autoregressive approach for point cloud generation, with optionally an image to condition on. There has been a number of studies trying to reconstruct objects without 3D supervision [19,20] by leveraging multi-view images. However, in these methods, the number of trainable parameters is proportional to the number of output points, which creates an upper bound for the point cloud size. In contrast, we overcome this problem by deforming a point cloud instead of synthesizing one, which makes the system far more scalable.

F. Neural radiance field (NeRF)

Recently, there has been much noise created by neural radiance field, which started by the work of Mildenhall *et al.* [50] and followed by a considerable amount of research [51]–[54]. From multiple images of a scene, NeRF learns an implicit function that maps 3D locations to density and color, and a 3D mesh can be recovered using marching cubes [55]. The major difference between our methods and these works is generalization. While NeRF overfits an NN to a single scene and is not able to work with others, our framework can reconstruct unseen objects after training.

G. Deep learning on point clouds

There has been a surge of interest in applying deep NNs to point clouds started by [16] and followed by [56]–[62]. These methods mainly target classification, segmentation and detection problems.

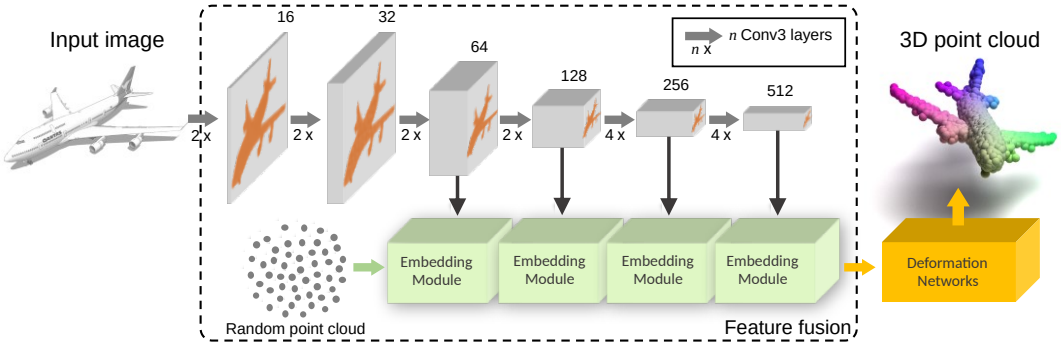
H. Generative adversarial networks

Our framework remotely resembles generative adversarial network (GAN) [63]–[66], especially StyleGAN [65], as it also utilizes a style transfer module, AdaIN [14], inside its architecture. Compared to StyleGAN, our work does not integrate the encoded noise into the main branch, but fuses the encoded image features into the noise (randomly initialized point cloud) branch.

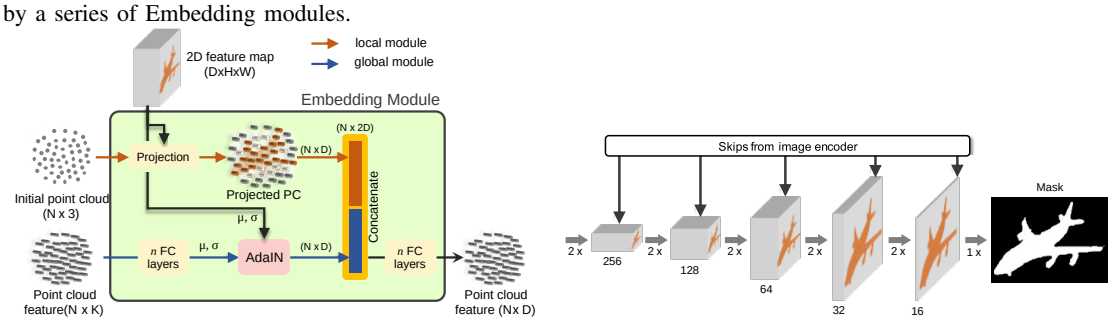
A few works taking a GAN approach for point cloud generation are [67]–[70]. In these works, input noise determines the output object shapes but in our method, the overall output shape is determined by the input image while the noise only affects the individual point position in the output point cloud.

III. Point Cloud Deformation Network

Our overall framework is shown in Fig. 2. Given a single image of an object, we first encode it into multi-scale feature maps using a CNN. From these features, we further distill global and point-specific shape information of the object. Then the two features are concatenated and sequentially processed by an MLP network. The output of this MLP is then fed to a deformation network. In the following sections, we will describe all the steps in detail.



(a) The network consists of two parallel branches. Image encoder (top) is a CNN that takes an input image and encodes it into multi-scale 2D feature maps. Point cloud encoder (bottom) takes a randomly initialized point cloud as input and processes the point cloud and 2D feature maps by a series of Embedding modules.



(b) **Left**: A schematic overview of the Embedding module. It takes three inputs: the initial point cloud, its features from the previous module and a 2D feature map extracted from the image encoder and outputs point cloud features for the next module. Inside this module, point-specific shape information and global shape information are extracted. More details are presented in the main text. **Right**: A U-Net decoder [71] that predicts a segmentation mask for the input image. The decoder mostly mirrors the encoder with the exception of the upsampling layers which use bilinear interpolation.

Fig. 2: Overview of PCDNetv2.

A. 2D Image Feature

In literature, 3D reconstruction from single view is widely studied where 2D information such as silhouette and shading plays an important role. In our method, 2D features act like an indicator of outside-ness and inside-ness of the silhouette. Moreover, sharp edges inside the object also signal that the corresponding 3D points are not locally planar. Disentanglement between the shading and texture can be resolved by the learned 3D prior in the network.

To extract 2D features, we use a VGG-like architecture [72] similar to [43] to encode the input image (Fig. 2 (a) top branch). Concretely, we divide the CNN into six blocks. The feature map dimension is doubled after each block, starting from 16. From the second block, the feature maps are four times decimated. The feature maps starting from the third block are used for the feature fusion as described next. All layers use 3×3 convolution kernels.

B. Feature Fusion Module

1) Point-specific shape information: Following [43], we extract a feature vector for each individual point by projecting the points onto the feature maps as illustrated in Fig. 2(b) (top branch). Given an initial point cloud, we compute the 2D pixel coordinate of each point using camera intrinsics and resample the feature vectors using bilinear interpolation. Let the initial point cloud and the extracted 2D feature maps at scale i be $\mathcal{Y} \subseteq \mathcal{R}^3$ and $X_i \in \mathcal{R}^{h_i \times w_i \times c_i}$ (c_i channels, height h_i, and width w_i), respectively. For simplicity, we drop the index i in the following discussion. In mathematical expression, the point-specific feature can be represented as

$$f_{\text{local}}(y_j) = u_{00} X_{\lfloor P_X[y_j] \rfloor} + u_{10} X_{\lfloor P_X[y_j] \rfloor + (1,0)} + u_{01} X_{\lfloor P_X[y_j] \rfloor + (0,1)} + u_{11} X_{\lfloor P_X[y_j] \rfloor + (1,1)}, \quad (1)$$

where $y_j \in \mathcal{Y}$ is a point in the cloud, $P_X[\cdot]$ is a projection operator that takes a 3D point and returns the 2D projection coordinates on X, $\lfloor \cdot \rfloor$ discards the floating point part, $X_{P_X[y_j]} \in \mathcal{R}^c$ is the feature vector of pixel $P_X[y_j]$, and u's are the corresponding bilinear coefficients. This scheme is simple and lightweight, but it is an efficient and creative way to embed features into the initial point cloud.

2) Global shape information: The point-specific features see only a small patch of the object, so the network is not aware of its overall shape. Therefore, in order for the network to be fully aware of the object shape, we propose a global shape feature that summarizes the object. The global shape information is obtained by the bottom branch in Fig. 2(b) **left**. We borrow a simple yet powerful concept from image style transfer literature. We find an analogy between style transfer and our problem as we want to transfer the object shape described in a 2D image to an intial point cloud. To this end, we propose to *stylize* the initial point cloud by a 2D-to-3D adaptive instance normalization (AdaIN$_{2D \to 3D}$) [14].

First, we process the incoming point cloud features by a stack of FC layers to project to the same spaces with the feature maps. For a point y_j in a point cloud $\mathcal{Y}$, we define the 2D-to-3D AdaIN$_{2D\to3D}$ as

$$f_{\text{global}}(y_j) = \text{AdaIN}_{2D\to3D}(X, y_j) = \sigma_X \frac{y - \mu_{\mathcal{Y}}}{\sigma_{\mathcal{Y}}} + \mu_X, \quad (2)$$

where $y_j \in \mathcal{Y}$ is the feature vector of point j in the cloud, μ_X and σ_X are the mean and standard deviation of X taken over all the spatial locations, and $\mu_{\mathcal{Y}}$ and $\sigma_{\mathcal{Y}}$ are the mean and standard deviation of the point cloud in feature space. The rationale of our definition is that from a global point of view, an object shape can be described by a mean shape and an associated variance. We can retrieve these mean shape and variance from the feature maps of the 2D input image, and then embed them into the normalized initial 3D point cloud. We apply AdaIN$_{2D\to3D}$ to the features in the coarse-to-fine order, starting with dimension 64 and gradually to 512. Note that we can reverse this order similar to [73], but it will require either a reverse point cloud encoder in which dimension becomes smaller as the network gets deeper or to add various 1×1 convolutional layers to reverse the feature dimensions. In our experiment, we found that doing so leads to unstable training and a lower performance. Therefore, we abandoned this design and stuck to our original order.

3) Mask prediction: To deal with the case in which background exists, we optionally predict a segmentation mask of the object in the input image in a supervised manner when ground truth masks are available, which is similar to Gkioxari *et al.* [45]. The mask prediction branch is shown in the right side of Fig. 2 (b) **right**. As can be seen from the figure, this branch consists of a U-Net [71] decoder that translates the last output feature maps of the image encoder to a segmentation mask. The decoder mostly mirrors the encoder architecture, with the exception of the upsampling layers that use bilinear interpolation. As in Ronneberger *et al.* [71], it concatenates feature maps from the image encoder blocks with the outputs of the corresponding decoder blocks. At the end of the network, the segmentation mask is produced by a pixel-wise sigmoid activation function.

After getting a predicted object mask, before extracting the point-specific and global feature vectors, we multiply the mask with the 2D feature maps to weight each pixel. For feature maps smaller than the mask, we resize the mask using bilinear downsampling. Multiplying the 2D feature maps with the masks will largely preserve the features in the foreground and discount those in the background. By providing the network an explicit hint about which pixels correspond to the object in the image, the network should be more aware about the object shape, and hence performance can be improved.

4) Point cloud feature extraction: The global and point-specific features are extracted from features in each of the four blocks as depicted in Fig. 2. At scale i, to obtain a single feature vector for each point, the two features are simply concatenated in feature space as

$$\hat{y}_j^{(i)} = [f_{\text{local}}^{(i)}; f_{\text{global}}^{(i)}]. \quad (3)$$

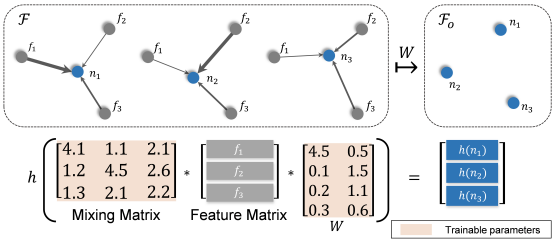


Fig. 3: A conceptual illustration of GraphX. First, the new points n_k are computed by combining all the given points f_i according to a mixing weight. Then the new points are mapped from $\mathcal{F}$ to a new space $\mathcal{F}_o$ by W and activated by a non-linear activation $h(\cdot)$. For brevity, biases are omitted.

Then this feature vector is fed to an FC layer to get $y_j^{(i+1)}$, which is the input feature for the next scale. Compared to our original work [24] that extracts and concatenates all features into a long vector at the end of the encoder, this sequential feature extraction scheme can reduce the memory footprint while retaining its expressivity.

C. Point Cloud Deformation Module

In order to generate a precise and representative point cloud, it is necessary to capture non-local patterns between points in the set. In this paper, inspired by the simplicity of graph convolution and the functionality of $\mathcal{X}$-conv [74], we propose *graphX-convolution* (GraphX) which possesses a similar functionality as the graph convolution but works on unordered point sets like $\mathcal{X}$-conv [21]. An intuitive illustration of GraphX is demonstrated in Fig. 3. The operation starts by mixing the features in the input via a *mixing matrix* and then applies a usual FC layer. Let $\mathcal{F}_j \subseteq \mathcal{R}^{d_j}$ be the set of d_j-dimensional features fed to j^{th} layer of the deformation network. For notation simplicity, we drop the layer index j and denote the output set as $\mathcal{F}_o \subseteq \mathcal{R}^{d_o}$. Mathematically, GraphX is defined as

$$f_k^{(o)} = h\left(n_k\right) = h\left(W^T\left(\sum_{f_i \in \mathcal{F}} w_{ik} f_i + b_k\right) + b\right), \quad (4)$$

where $f_k^{(o)}$ is the k^{th} output feature vector in $\mathcal{F}_o$, $w_{ik}, b_k \in \mathcal{R}$ are trainable mixing matrix and *mixing bias* corresponding to each pair $(f_i, f_k^{(o)})$, $W \in \mathcal{R}^{d \times d_o}$ and $b \in \mathcal{R}^{d_o}$ are the weight and bias of the FC layer, and h is an optional non-linear activation. Instead of learning locally like graph convolution, GraphX learns globally from the whole point set. The idea is that in a point cloud, every point can convey more or less information about others, thus we can let the learning decide where the network should concentrate. Our method is also similar to $\mathcal{X}$-conv in the way it takes the relationship of points into account and learn to be invariant to point order, but while the mixing matrix of $\mathcal{X}$-conv is computed by a neural network from a locality of points, ours is directly learned and works on the whole point set, and hence it is capable of learning a local-to-global prior.

Following the trend of employing residual connection [3] to boost gradient flow, we propose ResGraphX, which is a

residual version of GraphX. The main branch comprises an FC layer activated by ReLU and followed by a GraphX layer. As in [3], the residual branch is an identity when the output dimension of the layer does not change, and an FC layer otherwise.

D. GraphX Variants

1) Upsampling GraphX: If the size of the point cloud is large, learning a mixing operation is proportionally expensive. One workaround is to start with a small point cloud, and then gradually upsample it in such a way that $|\mathcal{F}_o| > |\mathcal{F}|$. Thus, the computation and memory can be reduced considerably. Alternatively, GraphX can also be utilized in the downsampling direction which is useful in point cloud encoding. By default, we use an upsampling version of ResGraphX, which is termed UpResGraphX. This variant provides a state-of-the-art performance while significantly reduce the memory footprint of the mixing matrix in the GraphX convolution.

2) Low-rank GraphX: As point cloud can be compressed [75], there must be some redundancy that we can leverage in order to further reduce the memory footprint. An intuitive way is to break the mixing matrix into a product of two low-rank matrices such that $W_{d\times d_o} = U_{d\times k}V_{k\times d_o}$. However, a naive choice of U and V would not bring any memory reduction benefit. Before introducing a constraint for k, we first define several terms that will enable us to analyze the low-rank property more conveniently.

Definition 1 (Rank ratio)**.** Let A be an $m \times n$ real matrix. Let $A = UV$ be a decomposition of A in which $U \in \mathcal{R}^{m\times k}$ and $V \in \mathcal{R}^{k\times n}$. Without loss of generality, assume that $m \leq n$. We define the *rank ratio* r to be the ratio between k and the lesser dimension, or

$$r = \frac{k}{m}. \tag{5}$$

For a reduction of memory, k should satisfy

$$k \times (d + d_o) < d \times d_o \iff k < \frac{d \times d_o}{d + d_o}, \tag{6}$$

or $r < \frac{d_o/d}{1+d_o/d}$. For a concrete example, if we have $d_o = 2d$, which means the output resolution doubles, we need to set k so that $r < \frac{2}{3}$ to enjoy lower memory consumption.

Next, we need the concept of *effective rank* (ER) to analyze the low-rank property of the mixing matrix.

Definition 2 (Effective rank)**.** Let A be an $m \times n$ matrix and ϵ a small real number. Without loss of generality, assume that $m \leq n$. ρ is called an effective rank of A at level ϵ if

$$\sum_{k=1}^{\rho} \sigma_k^2 > (1 - \epsilon) \sum_{k=1}^{m} \sigma_k^2 \tag{7}$$

where $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \geq 0$ are the singular values of A.

In other words, ER is the minimum number of retained principle or major components such that the change in the Frobenius norm (F-norm) of the matrix is negligible. We provide the analysis behind the low-rank mixing matrix in Section IV-D4.

(a) Training without VarMND. (b) Training with VarMND.

Fig. 4: Effect of VarMND on PCDNetv2.

E. Training PCDNetv2

1) Point cloud distance measure: We use Chamfer distance (CD) to measure the discrepancy between PCDNetv2's predictions and ground truths. For the sake of completeness, we present the CD measure between a 3D point cloud pair $(\mathcal{X}, \mathcal{Y})$ below

$$\begin{aligned}\mathcal{L}_{\text{CD}}(\mathcal{X}, \mathcal{Y}) &= d_{\mathcal{X}\to\mathcal{Y}}(\mathcal{X}, \mathcal{Y}) + d_{\mathcal{X}\leftarrow\mathcal{Y}}(\mathcal{X}, \mathcal{Y}) \\ &= \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} \|x - y\|_2^2 + \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}} \min_{x\in\mathcal{X}} \|y - x\|_2^2\end{aligned} \tag{8}$$

where $|\cdot|$ is the set cardinality.

2) Variance of mean neighborbood distances (VarMND): Given a point x in a point cloud $\mathcal{X}$, we define the mean neighborbood distance as the average of distances between x and its k nearest neighbors

$$d_{\mathcal{X}}(x) = \frac{1}{k}\sum_{y\in\mathcal{N}_k(x)} \|y - x\|_2^2 . \tag{9}$$

Then the expected variance of the mean neighborbood distances is calculated as

$$E_{\mathcal{P}}[Var_{x\in\mathcal{X}}(d_{\mathcal{X}})] = \int_{\mathcal{P}} Var_{x\in\mathcal{X}}(d_{\mathcal{X}}) \tag{10}$$

for a point cloud generating distribution $\mathcal{P}$. Intuitively, when points are regularly distributed, in the local neighborbood centering at any point, the mean distances from the center to their neigbors should be similar, or the variance should be small. We translate this intuition into the VarMND objective

$$\mathcal{L}_{var}(\mathcal{X}) = Var_{x\in\mathcal{X}}(d_{\mathcal{X}}), \tag{11}$$

and use a Monte Carlo method to estimate (10). A visual effect of this regularizer can be seen in Fig. 4. Clearly, when using VarMND, the output points distribute more regularly (Fig. 4 (b)) without any visible point cluster on the object surface compared with training without this loss (Fig. 4 (a)).

3) Segmentation mask learning: To learn a binary segmentation mask that can separate objects of interest and background, we use the common binary cross entropy loss between the predicted masks $\hat{M}$ and their corresponding ground truths M

$$\mathcal{L}_{\text{mask}}(\hat{M}, M) = -\frac{1}{h \times w}\sum_i M_i \log \hat{M}_i + (1 - M_i) \log (1 - \hat{M}_i), \tag{12}$$

and the sum is carried over all $h \times w$ pixels.

TABLE I: Comparison between PCDNetv2 and the original PCDNet [24]. The performance is comparable, but the training time (day) is significantly reduced.

	CD	IoU	Time
PCDNet	0.252	0.725	2
PCDNetv2	0.259	0.719	0.45

4) Total loss: We train the network using the following cost function

$$\mathcal{L}(\mathcal{X}, \mathcal{Y}) = \alpha \mathcal{L}_{\text{CD}}(\mathcal{X}, \mathcal{Y}) + \beta \mathcal{L}_{\text{VarMND}}(\mathcal{X}) (+\gamma \mathcal{L}_{\text{mask}}) \quad (13)$$

where α, β and γ are pre-defined weights to control the influence of each term on the model.

IV. Experimental results

A. Implementation Details

1) Training procedure: To limit the function space, we incorporated a small (1e-5) L2 regularization term into the loss. Compared with our original work [24], we used a batch size of 64 instead of 4. This helps significantly reduce the training time while maintaining a very similar performance, which can be seen from Table I. To accommodate for the increase of batch size, we increased the learning rate of Adam optimizer [76] to 3e-4. The exponential decay rates were initialized to the default values. We multiplied the learning rate by 0.2 at half and three fourth of the training. For the VarMND objective, by default we use sixteen neighbors to calculate the mean nearest distances. We set α and β in the total loss (13) to be 1e3 and 3e6, respectively, and γ to be 1e1 when applicable. At every iteration of the training, we initialized a random point cloud so that given fixed camera intrinsics, the projection of the point cloud covers the whole image plane. The point cloud can have 250 or 2k points depending on whether the upsampling variant of GraphX is used.

2) Data: Firstly, we trained and evaluated our model on the ShapeNet dataset [77]. We used a subset of the ShapeNet core consisting of around $50k$ models categorized into 13 major groups. As there is no ground truth segmentation masks accompanied with the database, and there is no background in images, we did not learn a mask prediction branch for this dataset. We utilized the default train/test split shipped with the database. The rendered images and ground truth point clouds were kindly provided by [78].

Secondly, to see how our proposed model can be applied to realistic images in the wild, we utilized the Pix3D dataset [79] and evaluated our framework in two scenario. The first scenario is that we simply used a pre-trained model on ShapeNet to test on the chair category following the evaluation protocol in Pix3D [79]. Next, we trained and tested the framework directly on Pix3D using the two splits in Gkioxari *et al.* [45]. Split $\mathcal{S}1$ is challenging because of the diversity in colors, orientations, lighting conditions and so on, while $\mathcal{S}2$ is even more difficult as the 3D models in training and testing are completely disjoint [79]. To prepare training images, we cropped the images so that each image contains exactly one object. We sampled 2000 points from the 3D object meshes using Poisson-disk sampling [80]. We used the ground truth segmentation masks to learn the mask prediction branch detailed in Section III-B.

3) Benchmarking methods: We pitted our PCDNetv2 against current state-of-the-art methods including 3D-R2N2 [78], point set generation network (PSG) [17], Pixel2Mesh [43], Pixel2Mesh++ [44], AtlasNet [81], Neural Mesh Renderer (NMR) [82], and GAL [83].

By default, we used a model powered by UpResGraphX (PCDNetv2-URG) if not otherwise mentioned. We additionally tested three more variants of PCDNetv2: (1) a naive model with an FC deformation network (PCDNetv2-FC), (2) a model using GraphX (PCDNetv2-GraphX), and (3) a model using a low-rank UpResGraphX (PCDNetv2-LRURG) with a rank ratio of 0.5. We also tried a variant with EdgeConv [58]. Because their method does not have an inherent solution to upsample a point cloud, we set the number of initial points to $2,000$. As a result, we have to reduce the batch size to 4.

In the deformation network of each variant, we use three corresponding GraphX-derived layers of sizes 512, 256 and 128, respectively. These layers are then followed by a linear FC layer to output a 3D point cloud. EdgeConv requires to specify the number of neighbors. We found that setting the numbers of neighbors for three EdgeConv layers to 5, 11 and 23 can balance best between performance and running time. Using more neighbors may enhance the performance, but the training was too slow, so we did not tune further.

4) Metrics: To make it easier for PCDNetv2 to serve as a baseline in subsequent research, we report three common metric scores: F-score, CD and intersection over union (IoU). All the scores are obtained by averaging the scores from three independent runs. Similar to [43,44], we use $F(\tau)$ and $F(2\tau)$ with $\tau = $ 1e-4. A higher score indicates a more favorable result for this metric. We note that we report F-score only for reference, as this metric and CD have a very strong correlation, meaning a low CD might well result in a high F-score.

IoU quantifies the overlapping region between two input sets. Regarding IoU, we first voxelized the point sets into a 32^3 grid using a simple method in [19] and calculated the scores. A method with higher IoU score is better.

B. Synthetic Data: Comparison to State-of-the-art Methods

1) Qualitative results: We start by comparing the results obtained by PCDNetv2-URG, PSG and Pixel2Mesh++ visually. PSG requires high computation, so it can generate only $1,024$ points. Pixel2Mesh and Pixel2Mesh++ synthesize meshes so we sampled $2,000$ points for visualization. We then render a point cloud from a random view point using Mitsuba [84]. We cherry-picked some difficult shapes that are not frequently seen in the database. The results are demonstrated in Fig. 5. As can be seen from the figure, we visually outperforms the competing method in all cases. While the estimated point clouds from PSG are very sparse and have high variance, and Pixel2Mesh++ fails to produce structures such as sharp lines and holes, those from PCDNetv2 have pretty sharp and solid shapes. As can be seen from the results, PCDNetv2 preserves both the appearances and fine details much better

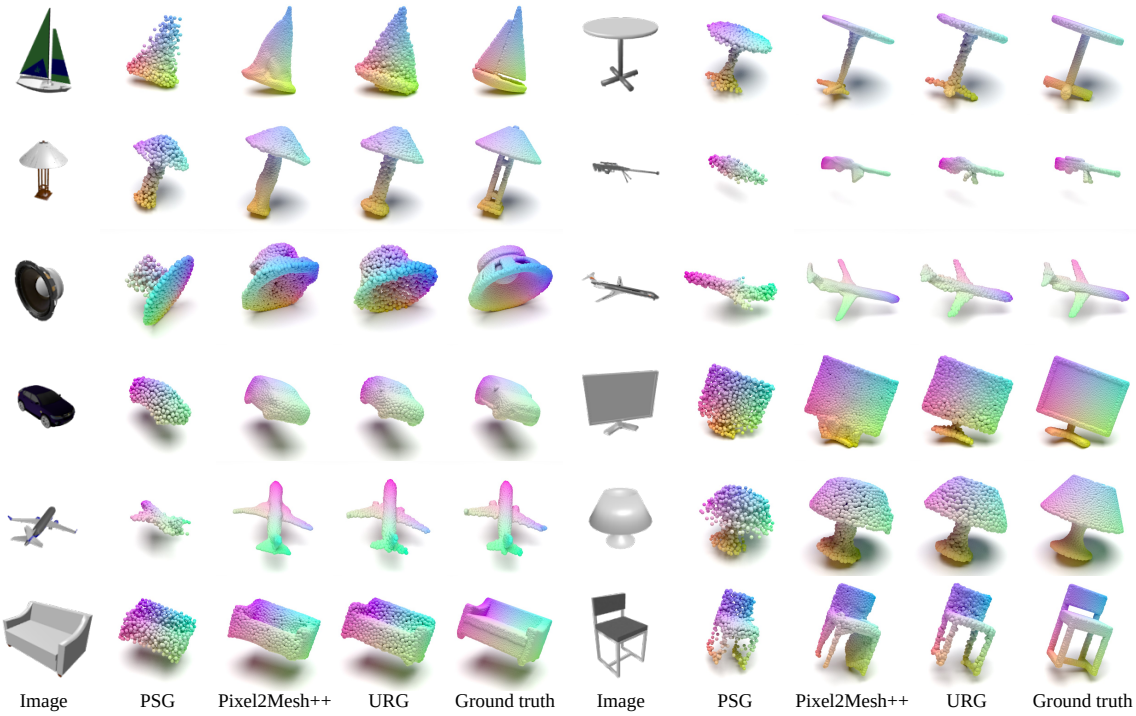


Fig. 5: Qualitative performance on ShapeNet.

TABLE II: CD results on ShapeNet. The best three scores are in **boldface**. "†" indicates training with half batch size and learning rate due to GPU memory limit.

Object	3D-R2N2	PSG	AtlasNet	Pixel2Mesh	Pixel2Mesh++	PCDNetv2-URG
Table	1.116	0.517	0.577	0.498	0.388	**0.335**
Car	0.845	0.333	0.340	0.268	0.249	**0.196**
Chair	1.432	0.645	0.724	0.610	0.461	**0.341**
Airplane	0.896	0.43	0.468	0.477	0.422	**0.126**
Sofa	1.135	0.549	0.621	0.490	0.439	**0.274**
Rifle	0.993	0.423	0.461	0.453	0.305	**0.132**
Lamp	4.009	1.193	1.575	1.295	1.135	**0.610**
Watercraft	1.215	0.633	0.839	0.670	0.508	**0.231**
Bench	1.891	0.629	0.703	0.624	0.549	**0.215**
Speaker	1.507	0.756	0.812	0.739	0.635	**0.456**
Cabinet	0.735	0.439	0.433	0.381	0.337	**0.281**
Display	1.707	0.722	0.848	0.755	0.566	**0.273**
Cellphone	1.137	0.438	0.443	0.421	0.325	**0.166**
Mean	1.445	0.593	0.68	0.591	0.486	**0.280**

thanks to the global and point-specific features embedded in our proposed method. Also, as the network is exposed to multiple views of various objects throughout training, it can generalize well to unseen objects and novel views.

2) Quantitative results: The metric scores of PCDNetv2-URG versus others are tabulated in Tables II, III and IV. As anticipated, PCDNetv2-URG outrun all the competing methods by a huge gap. For IoU, our method still tops the table and raises the performance bar previously set by our method [24]. Similarly, for F-score criterion, we also outperform Pixel2Mesh++.

C. Realistic Data: A Reality Check

First, we test the generalizability of a pretrained PCDNetv2-URG on ShapeNet by testing it on the realistic database Pix3D [79]. Because we did not train a mask branch on ShapeNet, we applied the available mask and cropped the images to make them similar to those from ShapeNet, which is the standard practice in [79]. Next, to check the joint segmentation and reconstruction ability, we trained PCDNetv2-URG with a mask branch from scratch directly on Pix3D as described in Section III-B. This will free the model from its dependency on ground truth mask, which is usually not available in practice. We detail the results of these two scenarios in the next two sections.

1) Evaluating pre-trained networks: Following [79], we tested PCDNetv2-URG on a subset of "chair" class that contains unoccluded chair images Then, we ran the provided evaluation protocol and tabulated the CD and Earth Mover's distance (EMD) scores in Table V. It can be seen that our model comfortably outperforms all existing methods by a large margin. Specifically when compared to PSG in Fig. 6,

TABLE III: IoU results on ShapeNet. The best scores are in **boldface**.

Object	3D-R2N2	PSG	GAL	PCDNetv2-URG
Table	0.580	0.606	0.714	**0.716**
Car	0.836	0.831	0.737	**0.834**
Chair	0.550	0.544	0.700	**0.713**
Airplane	0.561	0.601	0.685	**0.784**
Sofa	0.706	0.708	0.739	**0.797**
Rifle	0.600	0.604	0.715	**0.763**
Lamp	0.421	0.462	**0.670**	0.598
Watercraft	0.610	0.611	0.675	**0.776**
Bench	0.527	0.550	0.709	**0.763**
Speaker	0.717	**0.737**	0.698	**0.737**
Cabinet	0.772	0.771	0.772	**0.791**
Display	0.565	0.552	**0.804**	0.777
Cellphone	0.754	0.749	0.773	**0.860**
Mean	0.631	0.640	0.712	**0.762**

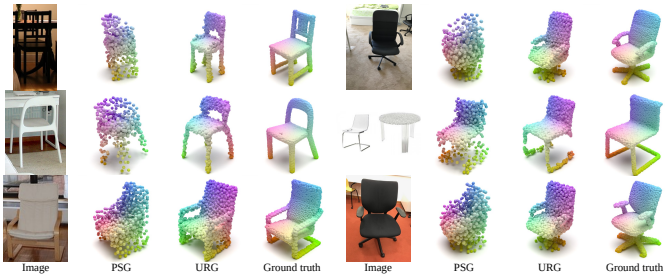


Fig. 6: Qualitative performance of PCDNetv2 and PSG on the Pix3D realistic dataset.

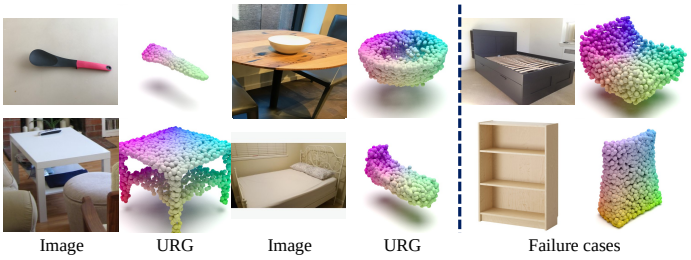


Fig. 7: Qualitative results of PCDNetv2-URG on different object categories in Pix3D. In most cases, the results are accurate (left), but sometimes the model mistake with its known objects (right).

our predicted point clouds look more faithfully to the ground truths. We went to extreme lengths to test the generalizability of our models by testing on the whole Pix3D dataset. We find that even though most of the shapes are out-of-distribution, our model can still make reasonable predictions in most cases. We cherry-pick some good examples and failure cases, and show in Fig. 7. Even thouth the model failed to predict the shape in these cases as they did not see anything similar in training, the shapes are still recognizable as the model relates the objects to known shapes thanks to the shape prior learned by GraphX. Additionally, even designed without occlusion handling in mind, our model can exhibit a certain level of robustness against occlusion as can be seen from the table example in Fig. 7.

2) Training from scratch: First, we compare our method with MeshRCNN [45] using their evaluation protocol, which rescales the point clouds to fit a fixed-length bounding box and measures CD and F1 scores at 0.1, 0.3 and 0.5. MeshRCNN [45] provided only their results on the $\mathcal{S}1$ split. This is a fair comparison because they used the ground truth 2D bounding boxes, so similar to ours, their model has to predict a segmentation mask and reconstruct the object. As can be seen from Table VI, we outperform MeshRCNN by a large margin. The low performance MeshRCNN can be the consequence of the non-differentiable voxel-to-mesh operator and the many regularizations needed to create a good mesh. Next, we conduct an ablation study with the masking branch and report CD (multiplied by 100) in Table VII. Note that we do not rescale the point clouds in this ablation study. As can be seen, we obtained better performance compared to the baseline that does not learn a mask branch. We also experimented a model similar to Gkioxari *et al.* [45] that we only predicted the mask without applying it (*w/ mask, w/o mul* in Table VII). Interestingly, in our experiments, we found that doing so did not benefit the model, and even hurt its performance.

D. Analysis

In this section, if not otherwise mentioned, all the experiments were perfomed on ShapeNet and all the models were trained without the mask branch.

1) Performance of different variants: In this section, we analyze the performance of different PCDNetv2 variants in order to validate our settings. The results can be seen in Table VIII. Among all the variants of PCDNetv2, in general, the GraphX family obtains better CD scores than the baseline whose deformation network is made of only FC layers. This is no surprise as GraphX is purposely architected to model both the global semantics and local relationship of points in the point cloud, which is necessary for characterizing point sets [16,74]. On the other hand, a deformation network with FC layers treats every point almost independently (points are processed independently in the forward pass but gradients are collectively computed in the backward pass), so the output coordinates are predicted without conditioning on the semantic shape information nor local coherence, which certainly degrades the performance. EdgeConv is more competitive to ours, but is too slow. We measured the wall clock time and while PCDNetv2-EdgeConv takes $72ms$ to infer one sample, PCDNetv2-URG needs only $13ms$. In conclusion, our operator is the best choice here as it can provide a decent performance while maintaining a good speed.

We note that the proposed loss function has a special effect on the GraphX layer. This loss operates on a locality of points, so it is difficult to optimize an FC layer that has only information about one point and is clueless about any other points in the point cloud. In that sense, GraphX benefits more from this loss, as the trained mixing matrix stores the interconnection between points.

2) Point-specific and global features: In this section, we quantitatively and visually verify the importance of the point-specific and global features produced by projection and

TABLE IV: F1(τ)/F1(2τ) results on ShapeNet. The best scores are in **boldface**.

Object	3D-R2N2	PSG	NMR	Pixel2Mesh	Pixel2Mesh++	PCDNetv2-URG
Table	43.79 / 59.49	53.44 / 73.10	28.40 / 41.73	66.30 / 79.20	**71.89** / **84.19**	69.10 / 83.69
Car	37.80 / 54.84	50.70 / 77.79	36.66 / 53.93	67.86 / 84.15	68.45 / 85.19	**74.28** / **90.05**
Chair	40.22 / 55.20	41.60 / 63.70	30.25 / 44.59	54.38 / 70.42	**62.05** / 77.68	61.59 / **79.84**
Airplane	41.46 / 63.23	68.20 / 81.22	62.10 / 77.15	71.12 / 81.38	76.79 / 86.62	**87.96** / **94.37**
Sofa	40.01 / 53.42	36.59 / 62.95	25.04 / 39.90	51.90 / 69.83	57.56 / 75.33	**62.19** / **83.29**
Rifle	28.34 / 46.87	69.96 / 82.65	52.22 / 63.28	73.20 / 83.47	80.74 / 89.29	**87.63** / **94.27**
Lamp	32.35 / 44.37	41.40 / 58.84	27.97 / 39.41	48.15 / 61.50	**62.56** / **74.00**	57.15 / 71.20
Watercraft	37.10 / 52.19	51.28 / 70.63	43.71 / 58.85	55.12 / 69.99	62.99 / 77.32	**73.87** / **87.12**
Bench	34.09 / 48.89	49.29 / 81.22	35.84 / 49.58	57.57 / 71.86	66.24 / 79.67	**76.46** / **88.88**
Speaker	45.30 / 57.86	32.61 / 56.79	19.46 / 32.20	48.84 / 65.61	**54.88** / 71.46	54.16 / **74.89**
Cabinet	49.88 / 64.83	39.93 / 67.03	21.04 / 35.16	60.39 / 77.19	65.72 / 81.57	**65.95** / **84.43**
Display	34.38 / 48.23	40.53 / 63.64	28.77 / 42.76	51.39 / 67.01	60.00 / 75.42	**69.72** / **85.22**
Cellphone	42.31 / 60.88	55.95 / 79.63	27.96 / 41.83	70.24 / 82.86	74.36 / 86.16	**84.19** / **92.96**
Mean	39.01 / 54.62	48.58 / 69.78	33.80 / 47.72	59.72 / 74.19	66.48 / 80.30	**71.10** / **85.40**

TABLE V: Quantitative performance on Pix3D. Results of competing methods are taken from [79].

	CD	EMD
3D-R2N2 [78]	0.239	0.211
PSG [17]	0.200	0.216
3D-VAE-GAN [85]	0.182	0.176
DRC [41]	0.160	0.144
MarrNet [86]	0.144	0.136
AtlasNet [81]	0.125	0.128
Pix3D (w/o pose) [79]	0.124	0.124
Pix3D (w/ pose) [79]	0.119	0.118
Ours	**0.091**	**0.105**

TABLE VI: Quantitative comparison with MeshRCNN [45] on the Pix3D $\mathcal{S}1$ split. We highlight the two best numbers.

Metrics	URG (*GT mask*)	URG (*w/ mask, w/ mul*)	MeshRCNN
CD	**0.207**	**0.694**	1.110
$F1^{0.1}$	**40.4**	10.8	**18.7**
$F1^{0.3}$	**86.4**	**56.9**	56.4
$F1^{0.5}$	**93.2**	**76.4**	73.5

TABLE VII: Ablation study of the mask branch on Pix3D when training PCDNetv2-URG from scratch.

Split	URG (*GT mask*)	URG (*w/o mask*)	URG (*w/ mask, w/o mul*)	URG (*w/ mask, w/ mul*)
$\mathcal{S}1$	0.42	1.59	1.55	**1.46**
$\mathcal{S}2$	1.58	2.13	2.29	**1.98**

TABLE VIII: Performance of different PCDNetv2 variants on ShapeNet. The best scores are in **boldface**.

Metric	FC	EdgeConv	GraphX	URG	LRURG
CD	0.295	0.285	0.289	**0.280**	**0.280**
IoU	0.759	**0.767**	0.755	0.762	0.761
F1(τ)	66.61	67.99	70.12	**71.10**	70.92
F1(2τ)	83.15	84.26	84.81	**85.40**	85.32

$\text{AdaIN}_{2D\to 3D}$, respectively. For time consideration, all the models in this comparison are trained without VarMND. All other hyperparameters and settings are kept the same.

Table IX demonstrates the quantitative and qualitative results of the ablation study in the mentioned order. As can be seen from the table, when either $\text{AdaIN}_{2D\to 3D}$ or projection is omitted from the model, it achieves roughly the same performance with some metrics higher and the others lower. The projection feature helps PCDNetv2 in CD and F-score while $\text{AdaIN}_{2D\to 3D}$ improves IoU. This can be easily explained by the fact that projection is a point-specific feature that embraces the details of the shape, which is favored by point-to-point metrics like CD and F-score. On the other hand, IoU measures the coverage percentage of two volumetric models, which can be high when two objects have roughly the same shape but not necessary all the subtleties. When the two are combined, both the two scores are significantly boosted, which validates our design of PCDNetv2.

3) Mixing matrix: We show a typical example of learned mixing matrices in Figure 8. As can be seen, there is minimal variation horizontally. This suggests that the output after multiplication with the mixing matrix is largely invariant to permutation in the point cloud as it gives each point roughly the same weight. In other words, the first layer of the deformation module builds permutation-invariant features from the input point features by summing over all points. We note that while other works [57] hard-code the summation over all points to learn permutation invariance, we let the network realize by itself through learning. The second layer continues to build more robust features from this previous output. As can be seen from the middle image in Figure 8, this layer interleaves summation and weighted summation of the points, and in the last layer, the network mainly performs weighted summation on the constructed feature set. This helps GraphX to extract a representative shape encoding from a point cloud, and as a result PCDNetv2 tends to produce an output with a clear and recognizable shape (albeit not very accurate) even in difficult cases like those in Section IV-C.

4) Rank ratio of mixing matrix: In (6), we introduce rank ratio to enable direct control over the memory consumption of GraphX by explicitly representing the mixing matrix as a low rank decomposition. We now discuss the rationale behind this approach by analyzing the mixing weights of learned PCDNetv2-URG networks. We observe the ERs at level 0.01 of the mixing matrices of three trained PCDNetv2-URG deformation networks together with their F-norms. In a PCDNetv2-URG deformation network, there are three ResGraphX layers

TABLE IX: Quantitative performance of PCDNetv2-URG when either point-specific or global feature is ablated. "-" denotes the feature that is ablated from the default PCDNetv2-URG. We style-code the best and second best numbers in **boldface**.

	-VarMND				-VarMND, -AdaIN$_{2D\to 3D}$				-VarMND, -Projection			
	CD	IoU	F1(τ)	F1(2τ)	CD	IoU	F1(τ)	F1(2τ)	CD	IoU	F1(τ)	F1(2τ)
Table	**0.298**	**0.591**	**67.54**	**82.92**	0.300	**0.518**	**67.09**	82.82	**0.298**	0.502	67.03	**82.87**
Car	**0.190**	**0.817**	**72.36**	**89.58**	0.192	**0.746**	71.49	89.39	**0.190**	0.726	**71.64**	**89.56**
Chair	**0.322**	**0.656**	**59.61**	**79.33**	0.324	**0.570**	**58.89**	79.11	**0.322**	0.547	58.82	**79.20**
Airplane	**0.122**	**0.753**	**87.53**	**94.52**	0.123	0.735	87.40	94.47	**0.121**	**0.736**	**87.74**	**94.62**
Sofa	**0.263**	**0.763**	**60.72**	**82.73**	0.267	**0.674**	59.15	82.17	**0.266**	0.654	**59.16**	**82.30**
Rifle	**0.124**	**0.735**	**87.64**	**94.68**	0.125	**0.722**	87.63	94.66	**0.124**	0.718	**87.64**	**94.70**
Lamp	**0.545**	**0.511**	**54.86**	**70.90**	0.548	**0.472**	54.71	70.76	**0.547**	0.468	**54.89**	**71.02**
Watercraft	**0.216**	**0.748**	**72.85**	**87.60**	0.219	**0.710**	72.33	87.43	**0.215**	0.707	**72.78**	**87.72**
Bench	**0.204**	**0.708**	**75.45**	**88.77**	0.206	**0.620**	74.91	88.57	**0.204**	0.606	**75.08**	**88.75**
Speaker	**0.412**	**0.704**	**54.09**	**74.69**	**0.417**	**0.637**	**52.92**	74.16	**0.417**	0.627	52.86	**74.20**
Cabinet	**0.261**	**0.768**	**65.89**	**84.20**	**0.264**	**0.707**	**64.88**	**83.87**	0.265	0.698	64.76	**83.87**
Display	**0.255**	**0.736**	**69.32**	**85.14**	0.257	**0.695**	68.52	84.93	**0.255**	0.688	**68.80**	**85.03**
Cellphone	0.161	**0.853**	**83.90**	**92.87**	**0.159**	0.833	**83.80**	**92.92**	**0.160**	**0.835**	83.64	92.82
Mean	**0.259**	**0.719**	**70.14**	**85.23**	0.262	**0.665**	69.52	85.02	**0.260**	0.655	**69.60**	**85.13**

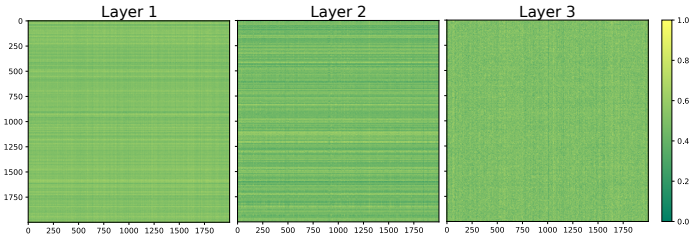


Fig. 8: Visualization of three mixing matrices in a learned GraphX network. The weights are in the first, second and third layers of the deformation module from left to right, respectively. We applied a sigmoid with high temperature for better visualization.

TABLE X: Quantitative performance of PCDNetv2-LRURG across different rank ratios.

	0.1	0.3	0.5	0.7	0.9
CD	0.284	0.281	**0.280**	0.281	0.282
IoU	0.760	**0.761**	**0.761**	0.762	0.761
F1	70.53	**71.05**	70.92	70.88	70.76
F1	85.12	**85.37**	85.32	85.29	85.23

each of which consists of one GraphX layer in the main branch and another in the residual branch. The mixing matrices in PCDNetv2-URG are also in charge of upsampling the initial point cloud from 250 to 2000 points.

We tabulate our findings in Table XI. As can be seen from this table, the normalized F-norm and ER tend to increase along the depth of the network. Also, the residual branch seems to have lower F-norm but higher ER than the main branch except for the last layer. However, the rank ratio exhibits a different pattern as it sinks to bottom at layer 2 and reaches its peak at layer 3. Our interpretation of these statistics is as follows. The deformation network first discards much of the input information by compressing from 250 points to roughly 50 points, and hallucinates these distilled points to 500 points. In the second layer, the distillation process continues and is even more intensive with a very high compression ratio of about 90%. After two consecutive compression and hallucination processes, most information retained in the point cloud is useful as evidenced by a high keep rate of 70%. As briefly mentioned following (6), a rank ratio of 0.7 is higher than the threshold of $2/3$ to actually save up some space, which suggests that the last layer is not meant for the low rank decomposition. Nonetheless, these statistics strongly advocate our explicit regularization on the rank of the mixing matrices.

Next, we trained PCDNetv2-LRURG of five different rank ratios from 0.1 to 0.9 with regular spacing and show the results in Table X. As can be seen, a ratio of 0.5 achieves the best performance. By contrast, IoU is not much affected by the ratio. This matter will be hashed over in the next section. In conclusion, a rank ratio of 0.5 is a suitable default choice for the PCDNetv2-LRURG layer.

5) Effect of VarMND: From Tables II, IV, III and IX, we can clearly observe the influence of the VarMND loss on the learned model. A positive change in performance is the gain in IoU, which increases from 0.719 to 0.762. Moreover, the effect of VarMND on IoU is very clear from Tables X and XII. When VarMND is used to regularize the networks, IoU scores across different rank ratios are similarly high. As the model is supervised by a point-to-point CD loss between the predictions and ground truths, it tries to match the fine details of the ground truth shapes and often the generated point clouds do not have a solid global appearance, which may result in a low IoU. To make the matter worse, the IoU metric is not differentiable. VarMND provides an alternative way to achieve higher IoU.

Tables XIII and XIV show the results of various hyperparameter settings for VarMND. When using only 8 neighbors, we achieved the best CD but the lowest IoU. Conversely, when using 24 neighbors, we obtained the highest IoU but other metrics are worse. Therefore, we used 16 neighbors, which settles for a satisfactory CD and IoU. Similarly, using larger β tends to increase IoU but increase CD as well. Thus, we chose a value that brings high IoU and low CD at the same time, which is 3e6.

6) Scalability: Last but not least, we demonstrate the scalability of our framework thanks to the stochasticity introduced by the randomly initialized input point cloud. In order to achieve a denser point cloud, we simply run the network

TABLE XI: Analyses of the mixing matrices in PCDNetv2-URG deformation networks.

	Main branch			Residual branch		
	Layer 1 (250×500)	Layer 2 (500×1000)	Layer 3 (1000×2000)	Layer 1 (250×500)	Layer 2 (500×1000)	Layer 3 (1000×2000)
F-norm	16.04	167.58	627.92	5.81	89.02	692.42
Normalized F-norm	1.00e-4	3.00e-4	3.00e-4	4.65e-5	2.00e-4	3.33e-4
Effective rank	54.7± 4.9	60.3± 6.0	703.7± 2.1	62.7± 10.5	74.0± 11.1	694.7± 2.5
Rank ratio	0.219	0.121	0.704	0.251	0.148	0.695

TABLE XII: Quantitative performance of PCDNetv2-LRURG without VarMND across different rank ratios.

Rank ratio	CD	IoU	F1(τ)	F1(2τ)
0.1	0.263	0.689	69.76	85.07
0.3	0.262	0.671	69.80	85.14
0.5	**0.261**	**0.706**	**70.00**	85.14
0.7	**0.261**	0.675	69.83	**85.16**
0.9	**0.261**	0.682	69.82	85.13

TABLE XIII: Quantitative performance of PCDNetv2-URG when using VarMND with different neighborhood sizes.

# neighbors	CD	IoU	F1(τ)	F1(2τ)
8	**0.274**	0.752	**71.15**	**85.50**
16	0.280	0.762	71.10	85.40
24	0.285	**0.765**	70.72	85.15
32	0.290	0.764	70.19	84.80

forward pass multiple times with different random initial input point clouds and then concatenate the generated point sets. We note that as shown in Section IV-D3, because GraphX learns to be invariant to set permutation by averaging the point set, it removes the randomness in the input and hence is not suitable for this task. Therefore, we use the variant with FC decoder. We produced point clouds of sizes 4000, 10000 and 20000 by running forward propagation two, five and ten times. A visualization of the outputs can be seen in Fig. 9. As can be seen, the point clouds can be arbitrarily dense, unlike previous works which always have an upper bound for the size.

V. Conclusion

We presented PCDNetv2, a point cloud deformation network that can deform an initial point cloud to the object shape given by a single object image. To deform the random point cloud, we first extracted global and point-specific features for every point. The point-specific features were obtained by projecting the random point cloud onto the 2D feature maps extracted from an image encoder, and the global features were distilled from the 2D feature maps by AdaIN$_{2D\rightarrow 3D}$, a concept borrowed from style transfer literature. To deal with realistic images containing background, we optionally learn a mask and apply it to the 2D feature maps before fusing point cloud and image features. Using the extracted features, we then deformed the point cloud by a network consisting of GraphX, a new layer that took into account the inter-correlation between points. To achieve better performance, we proposed a new loss function called VarMND, which aims to improve the regularity of the output point clouds. The framework is also able to scale the size of output point cloud thanks to the stochasticity in the random input. We conducted a series of experiments to validate the efficacy of the proposed method, and then thoroughly discussed the effectiveness of each individual component of our framework. Our future work should push further into the direction of single image 3D reconstruction directly from realistic data and possibly reconstructing the whole scene.

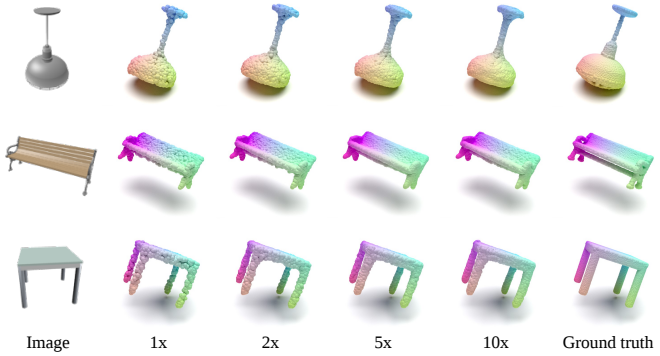


Fig. 9: A scalability test. Generated point clouds can have an arbitrary size, from 2000 (1×), 4000 (2×), 10000 (5×) to 20000 (10×) points.

TABLE XIV: Quantitative performance of PCDNetv2-URG when using VarMND with different β.

β	CD	IoU	F1(τ)	F1(2τ)
1e6	**0.272**	0.755	**71.10**	**85.55**
3e6	0.280	0.762	**71.10**	85.40
1e7	0.295	**0.764**	70.33	84.78
1e8	0.324	0.759	68.69	83.41

References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, mar 2004. [Online]. Available: https://www.cambridge.org/core/books/multiple-view-geometry-in-computer-vision/0B6F289C78B2B23F596CAA76D3D43F7A 1, 2

[2] Y. Furukawa and J. Ponce, "Carved visual hulls for image-based modeling," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS. Springer, Berlin, Heidelberg, 2006, pp. 564–577. [Online]. Available: https://link.springer.com/chapter/10.1007/11744023{_}44 1, 2

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, Conference Proceedings, pp. 770–778. 1, 5, 6

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," sep 2016. [Online]. Available: http://arxiv.org/abs/1609.03499 1

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," no. Mlm, 2018. 1

[6] W. Kim, A. D. Nguyen, S. Lee, and A. C. Bovik, "Dynamic Receptive Field Generation for Full-Reference Image Quality Assessment," *IEEE Transactions on Image Processing*, vol. 29, pp. 4219–4231, 2020. 1

[7] A. D. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee, "Distribution Padding in Convolutional Neural Networks," in *Proceedings - International Conference on Image Processing, ICIP*, vol. 2019-Septe. IEEE Computer Society, sep 2019, pp. 4275–4279. 1

[8] W. Kim, J. Kim, S. Ahn, J. Kim, and S. Lee, "Deep Video Quality Assessor: From Spatio-Temporal Visual Sensitivity to a Convolutional Neural Aggregation Network," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11205 LNCS. Munich, Germany: Springer, 2018, pp. 224–241. 1

[9] H. Oh, S. Ahn, J. Kim, and S. Lee, "Blind Deep S3D Image Quality Evaluation via Local to Global Feature Aggregation," *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4923–4936, 2017. 1

[10] J. Kim and S. Lee, "Fully deep blind image quality predictor," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, p. 206–220, 2017. 1

[11] J. Kim, A.-D. Nguyen, S. Ahn, C. Luo, and S. Lee, "Multiple level feature-based universal blind image quality assessment model," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, p. 291–295. 1

[12] J. Kim, A.-D. Nguyen, and S. Lee, "Deep cnn-based blind image quality predictor," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, p. 1–14, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8383698/ 1

[13] A.-D. Nguyen, S. Choi, W. Kim, and S. Lee, "A simple way of multimodal and arbitrary style transfer," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, p. 1752–1756. 1

[14] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, Conference Proceedings, pp. 1510–1519. 1, 3, 4

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, Conference Proceedings, pp. 2672–2680. 1

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, Conference Proceedings, pp. 652–660. 1, 3, 9

[17] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, Conference Proceedings, pp. 605–613. 1, 2, 3, 7, 10

[18] Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma, "PointGrow: Autoregressively learned point cloud generation with self-attention," in *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*. Institute of Electrical and Electronics Engineers Inc., oct 2020, pp. 61–70. [Online]. Available: http://arxiv.org/abs/1810.05591 1, 3

[19] E. Insafutdinov and A. Dosovitskiy, "Unsupervised learning of shape and pose with differentiable point clouds," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, Conference Proceedings, pp. 2807–2817. 1, 3, 7

[20] C.-H. Lin, C. Kong, and S. Lucey, "Learning efficient point cloud generation for dense 3D object reconstruction," in *AAAI Conference on Artificial Intelligence*, 2018, Conference Proceedings. 1, 3

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016. 2, 5

[22] H. Kim, H. Lee, W. H. Kang, J. Y. Lee, and N. S. Kim, "SoftFlow: Probabilistic Framework for Normalizing Flow on Manifolds," jun 2020. [Online]. Available: http://arxiv.org/abs/2006.04604 2

[23] S. Choi, A. D. Nguyen, J. Kim, S. Ahn, and S. Lee, "Point Cloud Deformation for Single Image 3d Reconstruction," *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2379–2383, 2019. 2

[24] A. D. Nguyen, S. Choi, W. Kim, and S. Lee, "GraphX-Convolution for Point Cloud Deformation in 2D-to-3D Conversion," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2019, pp. 8627–8636. [Online]. Available: http://arxiv.org/abs/1911.06600https://ieeexplore.ieee.org/document/9008115/ 2, 5, 7, 8

[25] M. Goesele, B. Curless, and S. Seitz, "Multi-View Stereo Revisited," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, vol. 2. IEEE, 2011, pp. 2402–2409. [Online]. Available: papers://90b07e07-903c-45d4-a899-2629f88b6b69/Paper/p2493http://ieeexplore.ieee.org/document/1641048/ 2, 3

[26] A. Hornung and L. Kobbelt, "Robust and efficient photo-consistency estimation for volumetric 3D reconstruction," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3952 LNCS. Springer, Berlin, Heidelberg, 2006, pp. 179–190. [Online]. Available: https://link.springer.com/chapter/10.1007/11744047{_}14 2

[27] K. N. Kutulakos and S. M. Seitz, "Theory of shape by space carving," *International Journal of Computer Vision*, vol. 38, no. 3, pp. 199–218, jul 2000. [Online]. Available: https://link.springer.com/article/10.1023/A:1008191222954 2

[28] S. N. Sinha and M. Pollefeys, "Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. I, 2005, pp. 349–356. 2

[29] M. Lhuillier and L. Quan, "Quasi-dense reconstruction from image sequence," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2351. Springer Verlag, 2002, pp. 125–139. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-47967-8{_}9 2

[30] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, vol. 35, no. 2, pp. 151–173, nov 1999. [Online]. Available: https://link.springer.com/article/10.1023/A:1008176507526 2

[31] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2352, 2002, pp. 65–81. [Online]. Available: http://www.cs.cornell.edu/{~}rdz/ 2

[32] S. Paris, F. X. Sillion, and L. Quan, "A surface reconstruction method using global graph cut optimization," in *International Journal of Computer Vision*, vol. 66, no. 2. Springer, feb 2006, pp. 141–161. [Online]. Available: https://link.springer.com/article/10.1007/s11263-005-3953-x 2

[33] E. Prados and O. Faugeras, *Shape from shading*. Springer, 2006, pp. 375–388. 3

[34] J. Aloimonos, "Shape from texture," *Biological cybernetics*, vol. 58, no. 5, pp. 345–360, 1988. 3

[35] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *UIST'11 - Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, no. 11. New York, New York, USA: ACM Press, 2011, pp. 559–568. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2047196.2047270 3

[36] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009. 3

[37] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 577–584, 2005. 3

[38] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, Conference Proceedings, pp. 1912–1920. 3

[39] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, Conference Proceedings, pp. 1696–1704. 3

[40] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, Conference Proceedings, pp. 82–90. 3

[41] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, Conference Proceedings, pp. 2626–2634. 3, 10

[42] S. Tulsiani, A. A. Efros, and J. Malik, "Multi-view consistency as supervisory signal for learning shape and pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, Conference Proceedings, pp. 2897–2905. 3

[43] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3D mesh models from single rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, Conference Proceedings, pp. 52–67. 3, 4, 7

[44] C. Wen, Y. Zhang, Z. Li, and Y. Fu, "Pixel2Mesh++: Multi-view 3D mesh generation via deformation," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob. Institute of Electrical and Electronics Engineers Inc., aug 2019, pp. 1042–1051. [Online]. Available: http://arxiv.org/abs/1908.01491 3, 7

[45] G. Gkioxari, J. Johnson, J. Malik, and J. Johnson, "Mesh R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob. Institute of Electrical and Electronics Engineers Inc., jun 2019, pp. 9784–9794. [Online]. Available: http://arxiv.org/abs/1906.02739 3, 5, 7, 9, 10

[46] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," jan 2020, pp. 165–174. [Online]. Available: http://arxiv.org/abs/1901.05103 3

[47] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion," pp. 6968–6979, mar 2020. [Online]. Available: http://arxiv.org/abs/2003.01456 3

[48] W. Wang, Q. Xu, D. Ceylan, R. Mech, and U. Neumann, "DISN: Deep implicit surface network for high-quality single-view 3d reconstruction," pp. 492–502, 2019. [Online]. Available: https://github.com/laughtervv/DISN. 3

[49] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Deep level sets: Implicit surface representations for 3D shape inference," *arXiv*, jan 2019. [Online]. Available: http://arxiv.org/abs/1901.06802 3

[50] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS. Springer Science and Business Media Deutschland GmbH, 2020, pp. 405–421. 3

[51] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and Improving Neural Radiance Fields," oct 2020. [Online]. Available: http://arxiv.org/abs/2010.07492 3

[52] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. P. A. Lensch, "NeRD: Neural Reflectance Decomposition from Image Collections," 2020. 3

[53] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Deformable neural radiance fields," 2020. 3

[54] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis," 2020. [Online]. Available: https://github.com/autonomousvision/graf. 3

[55] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987*, vol. 21, no. 4. Association for Computing Machinery, Inc, 1987, pp. 163–169. 3

[56] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 2017, pp. 5100–5109. 3

[57] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, p. 3391–3401. 3, 10

[58] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph Cnn for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, 2019. 3, 7

[59] J. Yin, J. Shen, X. Gao, D. Crandall, and R. Yang, "Graph Neural Network and Spatiotemporal Transformer Attention for 3D Video Object Detection from Point Clouds," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, pp. 1–1, 2021. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/34752380/ 3

[60] Q. Meng, W. Wang, T. Zhou, J. Shen, Y. Jia, and L. Van Gool, "Towards A Weakly Supervised Framework for 3D Point Cloud Object Detection and Annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2021. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/33656990/ 3

[61] Q. Meng, W. Wang, T. Zhou, J. Shen, L. Van Gool, and D. Dai, "Weakly Supervised 3D Object Detection from Lidar Point Cloud," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, jul 2020, vol. 12358 LNCS, pp. 515–531. [Online]. Available: https://arxiv.org/abs/2007.11901v1https://link.springer.com/10.1007/978-3-030-58601-0_31 3

[62] J. Yin, J. Shen, C. Guan, D. Zhou, and R. Yang, "LiDAR-based Online 3D Video Object Detection with Graph-based Message Passing and Spatiotemporal Transformer Attention," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11 492–11 501, apr 2020. [Online]. Available: https://arxiv.org/abs/2004.01389v1 3

[63] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 3, no. January, 2014, pp. 2672–2680. 3

[64] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, jan 2017, pp. 214–223. [Online]. Available: http://arxiv.org/abs/1701.07875 3

[65] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4217–4228, dec 2021. [Online]. Available: https://arxiv.org/abs/1812.04948v3https://ieeexplore.ieee.org/document/8977347/ 3

[66] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," nov 2014. [Online]. Available: https://arxiv.org/abs/1411.1784v1 3

[67] C. L. Li, M. Zaheer, Y. Zhang, B. Póczos, and R. Salakhutdinov, "Point cloud gan," in *Deep Generative Models for Highly Structured Data, DGS@ICLR 2019 Workshop*. International Conference on Learning Representations, ICLR, oct 2019. [Online]. Available: https://arxiv.org/abs/1810.05795v1 3

[68] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *35th International Conference on Machine Learning, ICML 2018*, vol. 1. International Machine Learning Society (IMLS), jul 2018, pp. 67–85. [Online]. Available: https://arxiv.org/abs/1707.02392v3 3

[69] R. Li, X. Li, C. W. Fu, D. Cohen-Or, and P. A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, 2019, pp. 7202–7211. [Online]. Available: https://liruihui.github.io/publication/PU-GAN/ 3

[70] D. Shu, S. W. Park, and J. Kwon, "3D point cloud generative adversarial network based on tree structured graph convolutions," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, 2019, pp. 3858–3867. 3

[71] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351. Springer Verlag, may 2015, pp. 234–241. 4, 5

[72] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 4

[73] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, 2017, pp. 386–396. 5

[74] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, Conference Proceedings, pp. 828–838. 5, 9

[75] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019. 6

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014, Conference Proceedings. 7

[77] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, and H. Su, "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015. 7

[78] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, Conference Proceedings, pp. 628–644. 7, 10

[79] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3D: Dataset and methods for single-image 3D shape modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, Conference Proceedings, pp. 2974–2983. 7, 8, 10

[80] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: An open-source mesh processing tool," in *6th Eurographics Italian Chapter Conference 2008 - Proceedings*, 2008. 7

[81] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A Papier-Mache Approach to Learning 3D Surface Generation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2018, pp. 216–224. 7, 10
[82] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D Mesh Renderer," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, nov 2018, pp. 3907–3916. [Online]. Available: http://arxiv.org/abs/1711.07566 7
[83] L. Jiang, S. Shi, X. Qi, and J. Jia, "GAL: Geometric adversarial loss for single-view 3D-object reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, Conference Proceedings, pp. 802–816. 7
[84] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: A retargetable forward and inverse renderer," *ACM Transactions on Graphics*, 2019. 7
[85] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90. 10
[86] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, "MarrNet: 3D Shape Reconstruction via 2.5D Sketches," *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 541–551, nov 2017. [Online]. Available: http://arxiv.org/abs/1711.03129 10

Duc Nguyen received the B.Eng. degree in automatic control from the Hanoi University of Science and Technology, Vietnam, in 2015. He is currently pursuing Ph.D. at Yonsei University, South Korea. His research interests include image/video analysis, geometric computer vision, machine learning and deep learning.

Seonghwa Choi received the B.S. degree in electronic engineering from Soongsil University, Seoul, South Korea, in 2018. He is currently pursuing the M.S. and Ph.D. degrees with the Multidimensional Insight Laboratory, Yonsei University. His research interests include image and video processing based on the human visual system, computer vision, and machine learning.

Woojae Kim received his B.S. degree in electronic engineering from Soongsil University, Seoul, South Korea, in 2015, M.S. and Ph.D. degrees from Multidimensional Insight Laboratory, Yonsei University, Seoul, South Korea in 2021. He was a research assistant under the guidance of Prof. Weisi Lin with the Laboratory for School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore in 2018. Currently, he joined Samsung Research (SR), Seoul R&D Center, South Korea. His research interests include Deep Learning, Computer Vision, AR/VR Signal Processing, and Image Video Quality Assessment.

Jongyoo Kim is a senior researcher at Microsoft Research Asia. He received his B.S. degree, M.S. degree and Ph.D. degree in Electrical and Electronic Engineering from Yonsei University, Seoul, Korea in 2011, 2013 and 2018, respectively. His research interests include 2D/3D computer vision, computer graphics, and perceptual image/video processing. He was a recipient of the Global PhD Fellowship by National Research Foundation of Korea from 2011 to 2016.

Heeseok Oh received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2010, 2012, and 2017, respectively. He was a senior engineer of the Samsung Electronics, Seoul, Korea. From 2017 to 2022, he was with Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He is currently an assistant professor in Department of Applied AI, Hansung University, Seoul, Korea. His research interests include 2D/3D image and video processing based on human visual system, computer vision, extended reality, and deep generative model.

Jiwoo Kang was born in South Korea, in 1987. He received the B.S. in 2011 from Yonsei University, Seoul, South Korea, in Electrical and Electronic Engineering. He received the M.S. and the Ph. D. in 2019, both at once, through the integrated Ph. D. program, in Electrical and Electronic Engineering, Yonsei University. He worked as a Researcher at University-Industry Foundation in Yonsei University from September 2019 to November 2020, and as a Research Professor at BK21 Y-BASE R&E Institute in Yonsei University from December 2020 to February 2022. He is currently an Assistant Professor at Sookmyung Women's University, Seoul. His research interests include computer vision, computer graphics, artificial intelligence, machine learning, and image analysis.

Sanghoon Lee received the B.S. degree from Yonsei University, South Korea, in 1989, the M.S. degree from the KAIST, South Korea, in 1991, and the Ph.D. degree from The University of Texas at Austin, TX, USA, in 2000. From 1991 to 1996, he was with Korea Telecom, South Korea. From 1999 to 2002, he was with Lucent Technologies, NJ, USA. In 2003, he joined the Department of EE, Yonsei University, as a Faculty Member, where he is currently a Full Professor. His current research interests include image/video processing, computer vision, and graphics. He also served as an Editor for the Journal of Communications and Networks from 2009 to 2015. He was an Associate Editor and Guest Editor of the IEEE Transactions on Image Processing from 2010 to 2014, and 2013, respectively. He was the General Chair of the 2013 IEEE IVMSP Workshop. He has been serving as the Chair of the IEEE P3333.1 Working Group since 2011. He served as an Associate Editor and has been serving as a Senior Area Editor for the IEEE Signal Processing Letters from 2014 to 2018, and since 2018. He was a member of the IEEE IVMSP/MMSP TC from 2014 to 2019/from 2016 to 2021, respectively. He has been serving as an Associate Editor of IEEE Transactions on Multimedia since 2022. He was the Image, Video, and Multimedia TC Chair of APSIPA from 2018 to 2019. He is a BoG member of APSIPA, and also an Editor in Chief of APSIPA News Letters.