A Conditional Generative Adversarial Network for Rendering Point Clouds

Rowel Atienza
University of the Philippines
Electrical and Electronics Engineering Institute, Diliman, Quezon City, Philippines
rowel@eee.upd.edu.ph

Abstract

In computer graphics, point clouds from laser scanning devices are difficult to render into photo-realistic images due to lack of information they carry about color, normal, lighting, and connection between points. Rendering a point cloud after surface mesh reconstruction generally results into poor image quality with many noticeable artifacts. In this paper, we propose a conditional generative adversarial network that directly renders a point cloud given the azimuth and elevation angles of camera viewpoint. The proposed method, called pc2pix, renders point clouds into objects with higher class similarity with the ground truth as compared to images from surface reconstruction. pc2pix is also significantly faster, more robust to noise and can operate on a lower number of points. The code is available at: https://github.com/roatienza/pc2pix.

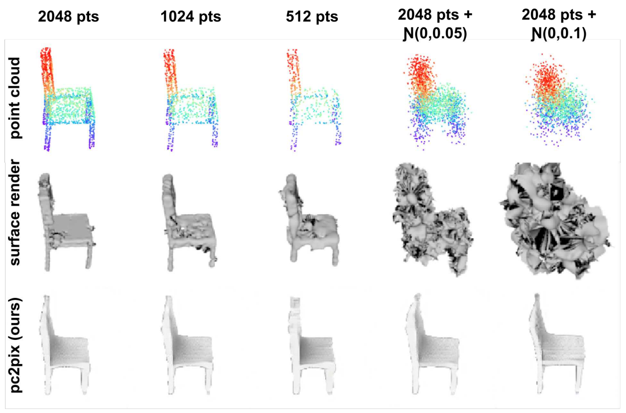

Figure 1. *Top:* Point cloud (original number of points, down sampled and noised), *Middle:* Rendering after surface reconstruction, *Bottom:* Rendering by *pc2pix*.

1. Introduction

A point cloud is a set of points with x, y, and z components representing spatial positions relative to a coordinate system. In laser scanning devices such as LIDAR or Kinect, each point is the intersection where the laser strikes the surface of an object. Rendering a point cloud into an image enables us to take advantage of many computer vision algorithms for analysis.

In computer graphics, a point cloud can be rendered given its consistent polygonal 3D model (mesh) which can be created after surface reconstruction. However, surface reconstruction of point clouds is itself a very hard problem [27] and not completely solved. Berger et al. [3] provided a comprehensive analysis of state of the art in surface reconstruction from point clouds. The algorithms impose a strong prior such as volumetric smoothing [31], structural repetition [36], part composition [30], and polygonal surface fitting [23].

Performing surface reconstruction from point clouds generated by laser scanning devices presents some unique problems. In particular, unlike in multi-view stereo point clouds, there is no information on color and normal of each point that can guide the reconstruction process. Point clouds generated by scanning devices can be sparse, noisy, and may contain outliers. Furthermore, figuring out the connection between points or edges is a difficult problem.

Recently, generative adversarial networks (GANs) exhibited a remarkable performance in synthesizing high resolution realistic images from latent space. Inspired by these results, we theorize that a GAN can render point clouds. In this paper, we propose a conditional GAN called *pc2pix*. Our work is closely related to *points2pix* [19] which was developed at the same time as *pc2pix*. The main difference is *pc2pix* is built on the better performing auxiliary conditional GAN (ACGAN) [24]. ACGAN forces the discriminator to recover the conditional input. As a result, the generator produces images with higher scores. Another difference is *pc2pix* hallucinates color based on the object statistics in the given dataset. *points2pix* relies on an additional background texture input to propose the color of the rendered object.

To illustrate rendering, Figure 1 shows a point cloud of a chair object from ShapeNet [6], the image outputs after surface reconstruction and *pc2pix*. The point cloud is centered at the origin of a sphere with radius 1.0. From the original number of 2048 points, the point cloud was down sampled to 1024 and 512 points. We also experimented on adding Gaussian noise with $\mu = 0$ and $\sigma = 0.05$ and 0.1.

For the surface reconstruction, we used the ball-pivoting algorithm (BPA) after point normal estimation [4]. After mesh surface reconstruction, additional surface refinement techniques such as holes closing and marching cubes (APSS) [18, 9] were applied. From this point, we will call this ensemble of algorithms *surface reconstruction* or *SR*. We used Blender [1] and MeshLab [2] for SR.

Figure 1 demonstrates the robustness of *pc2pix* when the point cloud is subjected to down sampling and noise. The rendered image is also smoother and has less visual artifacts compared to the output after SR. We can also observe how rendering after SR degrades with down sampling and noise.

In summary, our contributions are as follows:

- We present a model, *pc2pix*, that directly renders point clouds. *pc2pix* skips surface reconstruction by inferring the image from a given point cloud and camera viewpoint angles. *pc2pix* rendering process is fast, and robust to noise and down sampling of point clouds. The generated images are also of higher visual quality.

- *pc2pix* can navigate the point cloud encoder latent space. Simple arithmetic operations in the latent space can also be performed. These result into new point cloud configurations that have not been observed before and can be rendered into images by *pc2pix*.

2. Background

Most 3D data acquisition devices generate point clouds which are memory efficient continuous spatial data. The problem is point clouds can not be easily converted to meshes for rendering purposes to due to lack of information about normal and interconnection between points to form graph vertices. In this paper, we propose to address the problem of directly rendering point clouds by using a generative model to infer the image given the camera viewpoint angles.

Recently, generative models demonstrated a remarkable ability to synthesize high-resolution realistic images from latent space: ProGAN [16], SNGAN [21], SAGAN [34] and BigGAN [5]. These networks recommended improvements in training and network configuration on the original GAN [8] and its implementation using deep convolutional neural networks, DCGAN [26]. Most of the tricks are built upon improving training stability that impose 1-Lipschitz constraint [2] and gradient penalty [10] on the discriminator weights.

[1]https://www.blender.org

[2]http://www.meshlab.net

A GAN is made of two networks, generator and discriminator. The concatenation of these two networks is called an adversarial network. GANs find the equilibrium by alternately training the discriminator and generator to minimize their respective loss functions, Eq. 1 and 2 [8]:

$$\mathcal{L}^{(D)} = -\mathbb{E}_x \log \mathcal{D}(x) - \mathbb{E}_z \log(1 - \mathcal{D}(\mathcal{G}(z))) \quad (1)$$

$$\mathcal{L}^{(G)} = -\mathbb{E}_z \log \mathcal{D}(\mathcal{G}(z)) \quad (2)$$

where $z \in \mathbb{R}^{d_z}$ is the latent code drawn from distribution $\mathcal{N}(0,1)$ or $\mathcal{U}(-1,1)$.

We can force GANs to generate an image with specific attributes. CGAN [20], ACGAN[24], and cGANs[22] generate images given a specific label or discrete-valued side information. For example, ACGAN forces the discriminator to reconstruct the side information using an auxiliary network. The ACGAN discriminator and generator loss functions are shown in Eq. 3 and 4:

$$\begin{aligned}\mathcal{L}^{(D)} = &-\mathbb{E}_x \log \mathcal{D}(x|y) - \mathbb{E}_z \log(1 - \mathcal{D}(\mathcal{G}(z|y))) \\ &- \mathbb{E}_x \log \mathcal{P}(c|x) - \mathbb{E}_z \log \mathcal{P}(c|\mathcal{G}(z|y))\end{aligned} \quad (3)$$

$$\mathcal{L}^{(G)} = -\mathbb{E}_z \log \mathcal{D}(\mathcal{G}(z|y)) - \mathbb{E}_z \log \mathcal{P}(c|\mathcal{G}(z|y)) \quad (4)$$

where y is the side information (e.g. class). $\mathcal{P}(c|\cdot)$ is the predicted class conditional probability. Related to ACGAN is cGANs which uses a projection layer in the discriminator for improved generator performance. However, the loss functions of ACGAN are more transparent and interpretable than cGANs.

GANs that perform image to image translation such as *pix2pix* [14] and *CycleGAN* [37] are not suitable for rendering since point clouds of two different objects can have the same appearance on certain viewpoints resulting to ambiguity. For example, the point clouds of two different chairs may appear indistinguishable from the side view.

Following the recommendations made by the large-scale analysis of state of the art GANs [17] on losses, architecture, normalization, and regularization, *pc2pix* builds on SNGAN. To condition the image to generate from a point cloud, we borrow the concept of side information and reconstruction from ACGAN. We use ResNet [11, 12] architecture on both generator and discriminator. Spectral normalization is applied on the discriminator. Batch normalization is used only on the generator.

In the following sub sections, the state of the art generator evaluation metrics are put in the context of rendering point clouds.

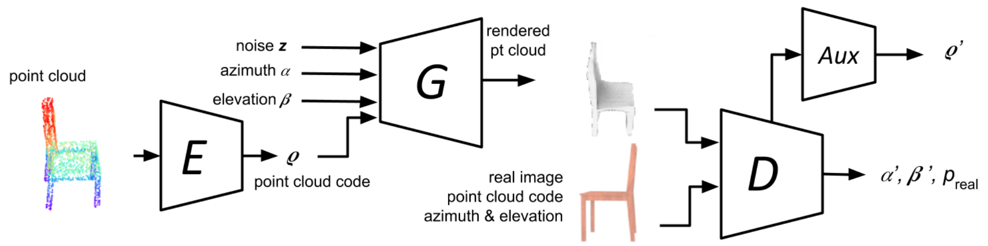

Figure 2. *pc2pix* is a conditional GAN. The point cloud is encoded into a low-dimensional code. The side information is made of point cloud code and normalized azimuth and elevation angles of camera viewpoint. The side information is recovered in the discriminator network.

2.1. Image Similarity

Image similarity could be a good measure of the performance of a generator for rendering point clouds. Given the expected image as rendered from the CAD model of an object, the similarity with the images generated by surface reconstruction rendering and generator can be obtained. Generally, the higher the similarity, the better.

SSIM [32] is an established measure for measuring similarity between two images. Other measures of image similarity include MS-SSIM [33], CW-SSIM [29], and Perceptual Similarity [35]. SSIM breaks down the similarity measure into three independent components with equal weights: 1) *Luminance*, 2) *Contrast*, and 3) *Structure*. Luminance is a measure of image intensity due to illumination and reflectance of the scene. Contrast is also a measure of intensity but after subtracting the mean intensity from the image. Structural information is the structure of the objects in the scene independent of the average luminance and contrast. Given that point clouds have no color and intensity information, we argue that the most important component for our purposes is structure since it directly compares inferred object geometry in the scene.

2.2. Frèchet Inception Distance

Inception Score (IS) was proposed by [28] to measure the performance of a generator. The main criticism against IS is it is not a proper measure of distance [17]. It is also challenging to use IS for measuring image similarity to benchmark rendering algorithms.

Frèchet Inception Distance (FID) [13] proposes to address the concerns on IS. In FID, the features of generated and real samples are projected on an Inception V3 network layer. Assuming the features can be modelled by multivariate Gaussians, $X_r = \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g = \mathcal{N}(\mu_g, \Sigma_g)$ for real and generated feature distributions respectively:

$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \quad (5)$$

The lower the FID, the closer is the generated samples to the real samples. The proponent of FID argues that FID is closer to human judgment. Furthermore, the presence of noise and artifacts in the image reduces the FID and consequently, the similarity. Because of these, FID is a good performance metric of the rendering algorithm.

3. Generative Rendering

Figure 2 shows *pc2pix*. We implement a ResNet architecture on both generator G and discriminator D. The side information is reconstructed from the discriminator via auxiliary networks A_ρ, A_α, and A_β where ρ is point cloud latent code, α is the azimuth angle, and β is the elevation angle. Conditioning the generator to use side information enables it to generate an image that mimics the rendering process. *pc2pix* discriminator and generator loss functions are shown in Eq. 6 and 7:

$$\begin{aligned}\mathcal{L}^{(D)} = &-\mathbb{E}_x \log \mathcal{D}(x|\rho,\alpha,\beta) \\ &- \mathbb{E}_z \log(1-\mathcal{D}(\mathcal{G}(z|\rho,\alpha,\beta))) \\ &+ \sum_{i=1}^{3}\Big\{\mathbb{E}_x\mathcal{R}(c_i|x) + \mathbb{E}_z\mathcal{R}(c_i|\mathcal{G}(z|\rho,\alpha,\beta))\Big\} \quad (6)\end{aligned}$$

$$\begin{aligned}\mathcal{L}^{(G)} = &-\mathbb{E}_z \log \mathcal{D}(\mathcal{G}(z|\rho,\alpha,\beta)) \\ &+ \sum_{i=1}^{3}\mathbb{E}_z\mathcal{R}(c_i|\mathcal{G}(z|\rho,\alpha,\beta)) \quad (7)\end{aligned}$$

where $c_1 = \rho$, $c_2 = \alpha$, and $c_3 = \beta$. $\mathcal{R}(c_i|\cdot) = -\log \mathcal{P}(c_i|\cdot)$. $\mathcal{R}(c_i|\cdot)$ can be interpreted as a reconstruction error. In our experiments, the distance of the camera from the origin of the point cloud coordinate system is kept constant (i.e. the camera is moved along the surface of a sphere with a constant radius).

3.1. Point Cloud Autoencoder

One of the inputs to *pc2pix* is the raw point cloud with dimensions $N \times 3$ where N is the number of points and the value 3 refers to the x, y, and z coordinates. This is

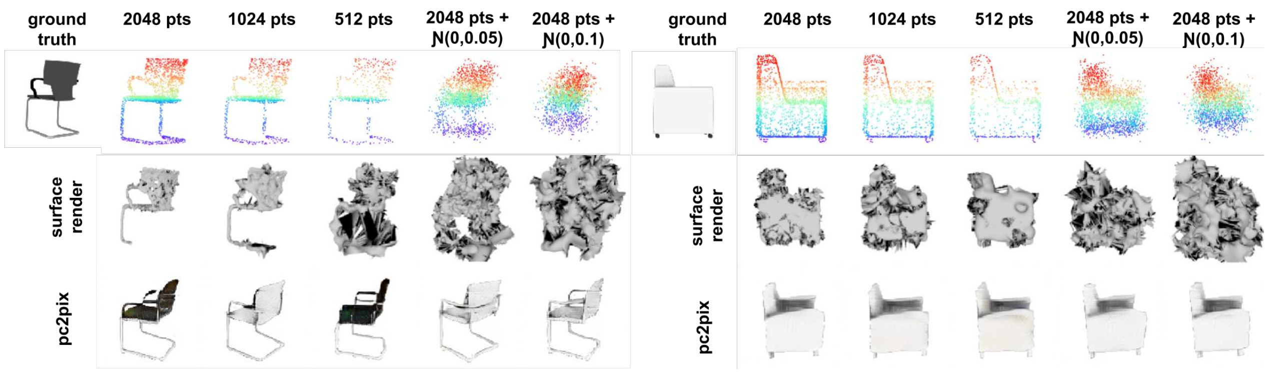

Figure 3. Examples of point cloud rendering by *pc2pix* and surface reconstruction on chair dataset.

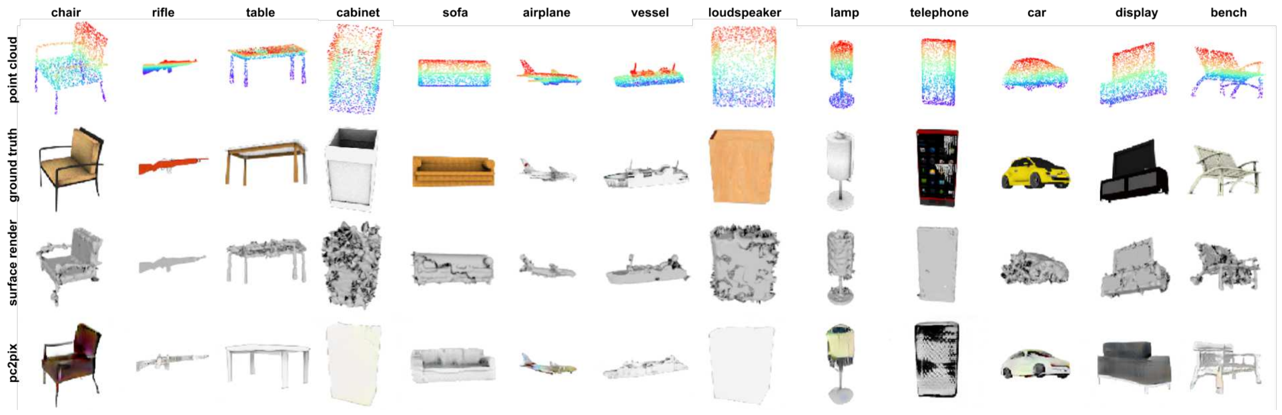

Figure 4. Examples of point cloud rendering by *pc2pix* and surface reconstruction on 13 classes.

Table 1. *pc2pix* generator. $ResBlock$ is a ResNet block with Batch Normalization (BN).

Generator Layers
$z \in \mathbb{R}^{128} \sim \mathcal{U}(-1,1), \rho \in \mathbb{R}^d, \alpha \in \mathbb{R}^1, \beta \in \mathbb{R}^1$
$x = Concat(z, \rho, \alpha, \beta)$
$x = Dense(x, 4 \times 4 \times 1024, ReLU)$
$x = Reshape(x, (4, 4, 1024))$
$x = ResBlock(x, (4, 4, 1024) \rightarrow (8, 8, 1024))$
$x = ResBlock(x, (8, 8, 1024) \rightarrow (16, 16, 512))$
$x = ResBlock(x, (16, 16, 512) \rightarrow (32, 32, 256))$
$x = ResBlock(x, (32, 32, 256) \rightarrow (64, 64, 128))$
$x = ResBlock(x, (64, 64, 128) \rightarrow (128, 128, 64))$
$x = BN - ReLU(x)$
$x = Conv2D(x, (128, 128, 64) \rightarrow (128, 128, 3), tanh)$
$x \in \mathbb{R}^{128 \times 128 \times 3}$

encoded into a low-dimensional code $\rho \in \mathbb{R}^d$ by E. We implemented improvements on the base PointNet [25] and l-GAN [1] autoencoder using wide 1D convolution with skip connections. After training, the encoder weights are frozen and used in *pc2pix* to generate point cloud latent code vectors.

3.2. Rendering GAN

The GAN architecture of *pc2pix* is shown in Tables 1 and 2. Both generator and discriminator use ResNet blocks.

Table 2. *pc2pix* discriminator. $ResBlockSN$ has no BN but uses Spectral Normalization (SN). $DenseSN$ is $Dense$ with SN.

Discriminator Layers
$x \in \mathbb{R}^{128 \times 128 \times 3}$
$x = ResBlockSN(x, (128, 128, 3) \rightarrow 64, 64, 64))$
$x = ResBlockSN(x, (64, 64, 64) \rightarrow (32, 32, 128))$
$x = ResBlockSN(x, (32, 32, 128) \rightarrow (16, 16, 256))$
$x = ResBlockSN(x, (16, 16, 256) \rightarrow (8, 8, 512))$
$\rho = Flatten(x)$
$\rho = DenseSN(\rho, 256, ReLU)$
$\rho = DenseSN(\rho, 1024, ReLU)$
$\rho = DenseSN(\rho, 1024, ReLU)$
$\rho = DenseSN(\rho, 1024, ReLU)$
$\rho = DenseSN(\rho, d, linear) \in \mathbb{R}^d$
$x = ResBlockSN(x, (8, 8, 512) \rightarrow (4, 4, 1024))$
$x = ResBlockSN(x, (4, 4, 1024) \rightarrow (4, 4, 1024))$
$x = ReLU(x)$
$x = GlobalSumPooling2D(x)$
$p_{real} = DenseSN(x, 1, sigmoid) \in \mathbb{R}^1$
$\alpha = DenseSN(x, 1, sigmoid) \in \mathbb{R}^1$
$\beta = DenseSN(x, 1, sigmoid) \in \mathbb{R}^1$

Only the generator uses batch normalization while spectral normalization helps to stabilize the discriminator training. We generalize ACGAN to include continuous side information (ρ, α, β) as conditioning inputs to the generator. The side information is recovered using auxiliary networks.

We use binary cross entropy loss for the discriminator

Table 3. SSIM and its components. SSIM is against rendered chairs using ShapeNet CAD model. The closer to 1.0, the better.

	2048 PTS		1024 PTS		512 PTS		2048 PTS + N(0,0.05)		2048 PTS + N(0,0.1)	
METRIC	SR	PC2PIX	SR	PC2PIX	SR	PC2PIX	SR	PC2PIX	SR	PC2PIX
SSIM	0.70	0.64	0.67	0.64	0.63	0.64	0.46	0.63	0.30	0.63
LUMINANCE	0.95	0.94	0.95	0.94	0.94	0.94	0.92	0.94	0.88	0.94
CONTRAST	0.87	0.79	0.85	0.79	0.82	0.79	0.69	0.79	0.53	0.79
STRUCTURE	0.77	0.77	0.75	0.77	0.73	0.78	0.65	0.77	0.65	0.77

Table 4. FID scores of rendered chairs. FID is measured against rendered chairs using ShapeNet CAD model. The lower, the better.

	2048 PTS		1024 PTS		512 PTS		2048 PTS + N(0,0.05)		2048 PTS + N(0,0.1)	
METRIC	SR	PC2PIX	SR	PC2PIX	SR	PC2PIX	SR	PC2PIX	SR	PC2PIX
FID	94.0	31.5	133.6	34.0	183.8	36.7	293.0	33.0	294.2	39.0

output and MSE for the point cloud auxiliary network. The normalized azimuth and elevation angles use relative angular error $L_\theta = \text{atan2}\,\frac{\sin \delta_\theta}{\cos \delta_\theta}$, where $\delta_\theta = \theta_{gt} - \theta_{pred}$ and $\theta = \alpha, \beta$. The training optimizer is Adam with learning rate $1e-4$ and $2e-4$ for the generator and discriminator respectively. $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The discriminator and generator are alternately trained. The batch size is 32.

4. Experimental Evaluation

In our experiment, we used approximately 44k CAD models belonging to 13 classes from the ShapeNet dataset [6]. To generate the point cloud, each object was axis aligned and centered around a unit sphere. The surface was uniformly sampled for $N = 2048$ points [1]. We used the train-val-test split and rendering of CAD models made by [15]. Twenty 128×128 pixels RGB images were rendered after positioning the camera on random azimuth and elevation angles. The azimuth range is from $-180°$ to $+180°$ while the elevation range is from $-40°$ to $+40°$. Each point cloud can be rendered at the average of 0.02sec using *pc2pix* and 0.18sec after SR on an NVIDIA GTX 1060. *pc2pix* is $9x$ faster than rendering after SR.

4.1. Chair Dataset

Our initial evaluation involved training *pc2pix* on a single class. We chose *chair* given its advantage in terms of diversity and sample size. The autoencoder with 32-dim latent code was trained for 30 epochs by optimizing the Chamfer Distance loss [7]. Afterward, the conditional GAN of *pc2pix* was trained for 120 epochs (around 4 days on NVIDIA GTX 1080Ti) using the side information from the encoder and azimuth and elevation angles of the camera viewpoint. The performance evaluation was done on the test split.

Figure 3 shows sample rendered images after SR and using *pc2pix*. For reference, we also included the point cloud oriented on a given camera viewpoint and the ground truth rendered image. Qualitatively, we can observe the advantages of *pc2pix* in terms of similarity and robustness against down sampling and noise. There are many instances wherein the rendered images after SR are not recognizable (e.g. 1-seater sofa).

As shown in Table 3, in terms of SSIM structure component, *pc2pix* has the same performance as the render after SR. However, *pc2pix* remains robust against down sampling and presence of noise. The luminance and contrast are highly arguable measures of similarity in the context of rendering of point clouds without color and normal information.

Table 4 shows the similarity measure of rendered image versus ground truth using FID. *pc2pix* has a significant performance advantage compared to rendering after SR. Similar to its qualitative performance, *pc2pix* is robust against down sampling and noise.

4.2. Multiple Classes Dataset

After validating *pc2pix* on *chair* dataset, we trained the point cloud autoencoder with 512-dim latent code and rendering GAN on all 13 classes. Figure 4 shows the point cloud and rendered images from ground truth CAD model, SR, and *pc2pix*. Generally the images produced by *pc2pix* have a much better visual quality in terms of smoothness and texture. On some objects, even the regularity in the pattern of color is predicted correctly. For example, cars have uniform body color, tinted windows and windshields, and black tires.

The visual quality is reflected on the FID values as shown on Table 5. *pc2pix* has a significantly higher FID on all classes except on *rifle* and *lamp* categories. On *rifle*, the small difference on FID may be considered not significant. Both scores are low. However, *pc2pix* has poor performance on *lamp* category. This could be attributed to the small sample size (1.8k) and significant design variations (e.g. chandelier, study lamp, beside lamp, lamp post, etc) in the lamp objects. In contrast for example to cars with many samples (6k) and have very little intricate design variations (e.g. sedan, SUV, van, and truck). For the case of *chair* object, the FID has 2.7 improvement in value when all 13 classes were used. This may be attributed to additional features learned from similar structures like sofa and table.

Table 5. FID scores of 13 object classes from ShapeNet. The lower, the better.

AIRPLANE		BENCH		CABINET		CAR		CHAIR		DISPLAY		LAMP	
SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix
54.6	13.6	93.0	43.6	188.8	57.9	203.1	40.0	94.0	28.8	148.4	121.7	51.6	112.1
LOUDSPEAKER		**RIFLE**		**SOFA**		**TABLE**		**TELEPHONE**		**VESSEL**		**AVERAGE**	
SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix	SR	pc2pix
193.6	78.2	26.3	29.0	135.6	41.8	127.2	39.7	103.7	94.4	67.4	43.0	114.4	57.2

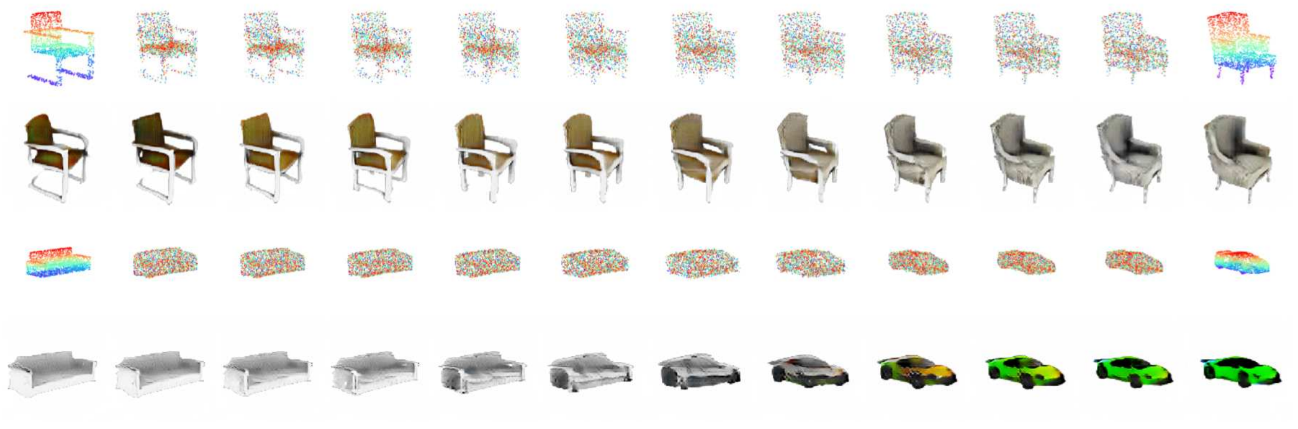
Figure 5. Rendering interpolated point clouds.

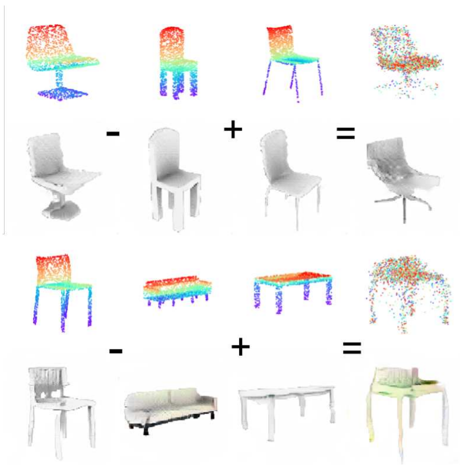

Figure 6. Rendering latent code arithmetic operation.

Another observation that can be made on outputs of the two rendering algorithms is that SR can generate good quality images whenever the point clouds are dense. This can be seen on visual quality of airplane, rifle, and lamp. *pc2pix* outputs improve with quantity of the training data. It suffers when there is dataset imbalance.

We experimented on latent interpolation on point cloud code. Figure 5 shows interpolation of point cloud for both single and two classes of objects. For example, we can interpolate between two *chair* designs and we can see how point clouds vary from left to right. Figure 5 shows that for each point cloud interpolation, *pc2pix* renders a meaningful image of a *chair*. Between two different objects, we can observe how the point cloud and its corresponding rendered image morph from *sofa* to *car*.

Arithmetic operation in latent space of point cloud is illustrated in Figure 6. Subtracting two point cloud latent code vectors and adding to a third results into another meaningful point cloud with some features added or removed. For example, subtracting a chair with four legs from a chair with a single leg but with a wide base support structure and adding to another chair with four thinner legs results into a chair with a single leg and four thin radial supports structure. The second example in Figure 6 shows that subtracting a sofa from a chair and adding the difference to a table results into a another point cloud that looks like a hybrid sofa-chair object.

5. Conclusion

Rendering point clouds is a challenging problem. We proposed a conditional GAN *pc2pix* to directly render point clouds. The results showed significant performance improvement compared to rendering point clouds after SR. *pc2pix* can also perform interpolation and arithmetic operation in point cloud latent space to synthesize new point clouds that can be directly rendered into images.

Acknowledgement

Supported in part by DOST-ERDT FRDG and CHED-PCARI Project IIID-2016-005.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ICML*, 2017.

[3] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. State of the art in surface reconstruction from point clouds. In *EUROGRAPHICS star reports*, volume 1, pages 161–185, 2014.

[4] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *ICLR*, 2019.

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

[7] Haoqiang Fan, Hao Su, and Leonidas 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*, pages 605–613, 2017.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[9] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 26, page 23. ACM, 2007.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[15] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems*, pages 365–376, 2017.

[16] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018.

[17] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *ICML*, 2018.

[18] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

[19] Stefan Milz, Martin Simon, Kai Fischer, and Maximillian Pöpperl. Points2pix: 3d point-cloud to image translation using conditional generative adversarial networks. *arXiv preprint arXiv:1901.09280*, 2019.

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[21] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018.

[22] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *ICLR*, 2018.

[23] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy*, pages 2353–2361, 2017.

[24] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *ICML*, 2017.

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016.

[27] Fabio Remondino. From point cloud to surface: the modeling and visualization problem. *International Archives of photogrammetry, Remote Sensing and Spatial Information Sciences*, 34, 2003.

[28] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[29] Mehul P Sampat, Zhou Wang, Shalini Gupta, Alan Conrad Bovik, and Mia K Markey. Complex wavelet structural similarity: A new image similarity index. *IEEE Transactions on Image Processing*, 18(11):2385–2401, 2009.

[30] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)*, 31(6):180, 2012.

[31] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. In *ACM Transactions on Graphics (TOG)*, volume 28, page 71. ACM, 2009.

[32] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[33] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. IEEE, 2003.

[34] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[35] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018.

[36] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. Graph.*, 29(4):94–1, 2010.

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.