EECS598-012 Project Proposal: 3D Point Cloud Completion

Mingyu Yang, Luya Gao, Shiyu Liu
University of Michigan
{mingyuy, mlgao, shiyuliu}@umich.edu

1. Introduction

3D acquisition technologies has been widely involved in the industries of autonomous driving and robotics. In a wide collection of 3D representations, point cloud is well preferable for various computer vision tasks. Compared to voxelized 3D data that cubically grows with the data size, point cloud is a tractable and highly scalable way of depicting either objects or scenes in the 3D space. Along with the explosion of real-world 3D data, such as those captured by LiDAR scans, we notice that most of them are noisy and sparse due to limited sensor resolution and interference (*e.g.* occlusions) among objects. The wider applications of downstream tasks such as 3D object detection and view extrapolation has also created a growing demand for denser and more complete 3D scans. One approach would be taking multiple scans and align them afterwards, but it is expensive and does not guarantee we could cover every angle. It would be very interesting, however, if we could learn a shape prior to help us inpaint an incomplete 3D scan. This is not only more efficient, but also could be applied to more general scenarios where taking scans from different angles is not always possible.

In this course project, we propose to tackle the problem of sparse 3D object scans through 3D point cloud completion. Our method is based on PCN [16], one of the most acknowledged benchmarks in the 3D completion domain. We implemented this architecture with Pytorch and sought to improve this method by experimenting with self-attention layers and advanced loss terms such as classification loss and feature loss.

2. Related Works

The point cloud based 3D shape completion is a surging research area benefited from the pioneering work of PointNet [9] and PointNet++ [10]. Compared with voxelized 3D data, point cloud requires a smaller memory cost. Recent notable studies such as PCN [16] and FoldingNet [15] usually learn a global representation from partial point cloud and generate the complete shape based on this learned feature. Following the same practise, a tree-structured decoder

Figure 1. Data samples from ShapeNet [1].

was proposed in TopNet [11]. To make better use of the incomplete point cloud, PF-NET [7] only recovers the missing points with a multi-scale generating network and an additional adversarial loss. More recently, a style-coded folding network is adopted in [13] to boost the model capacity. Besides, in order to generate visually-pleasing results, the generated point clouds are projected to depth images and is further examined by adversarial discriminators. Despite the supervised learning methods introduced above, there are also recent works [2, 12] that explore unpaired point cloud completion where the ground truth completed point clouds are not provided.

3. Point Completion Network

The first part of our work is to re-implement the PCN method using PyTorch, whose overall diagram is shown in Figure. 2. The encoder is an extended version of PointNet [9] that abstracts the input cloud X as a global feature vector v. It inherits the invariance to permutation and tolerance to nosie from PointNet. Specifically, the encoder consists of two stacked PointNet layers. The first layer consumes m input points and the input P is a $m \times 3$ matrix where each row is the 3D coordinate of a point p_i. Then, each point coordinate p_i are transformed to a k-dimensional point feature f_i using a shared multi-layer perceptron (MLP), which gives us a point feature matrix F. After that, a point-wise maxpooling is performed on F to obtain a k-dimensional global feature g. Similarly, the second PointNet layer takes the concatenation of F and g as input and gives us the final

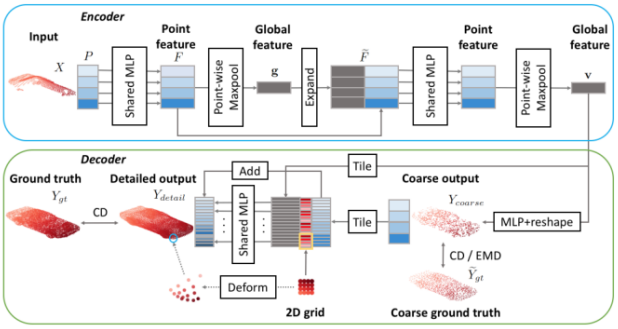

Figure 2. Diagram of PCN architecture in [16]

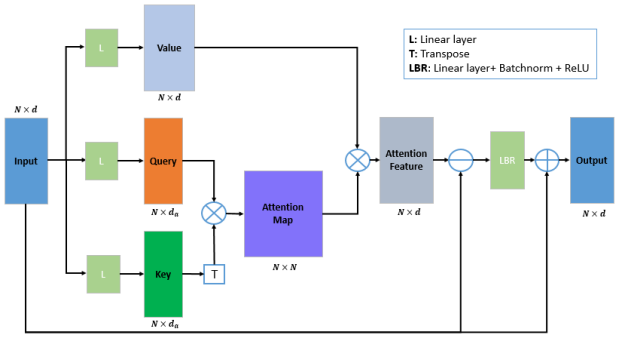

Figure 3. Diagram of the self-attention layers used in our project.

feature vector v.

The decoder of PCN performs multistage point generation where a MLP is use for coarse estimation and a folding-based decoder [15] is used for fine estimation. In the first stage, the coarse estimation Y_{coarse} of s points is generated by feeding the global feature v to a MLP to get a $3s$ output and reshaping the output to a $s \times 3$ matrix. In the second stage, for each point q_i in Y_{coarse}, a patch of $t = u^2$ points is generated in the local coordinates centered at q_i via the folding operation, and transformed into the global coordinates by adding q_i to the output. Then, all s patches are combined to achieve the detailed output Y_{detail} consisting of st points.

In the original PCN paper, the loss function for the coarse estimation Y_{coarse} contains both Chamfer Distance (CD) and Earch Mover's Distance (EMD) [3] whereas the loss function for Y_{detail} only contains CD. However, due to the high computational cost of EMD, we only use CD in our project. An expression of CD between two point clouds $\mathcal{T}$ and $\mathcal{R}$ is given by:

$$CD(\mathcal{T}, \mathcal{R}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \min_{r \in \mathcal{R}} \|t - r\|_2 + \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \min_{t \in \mathcal{T}} \|r - t\|_2$$

4. Modifications on PCN

Although PCN provides us with reasonable point cloud reconstructions, we find two problems that might limit the system performance: 1) The generation of the global feature v is too simple to capture the global structure of the input partial point cloud, 2) The fine estimation Y_{detail} is only guided with CD, which is not enough to generate realistic 3D shapes. To deal with the first problem, we introduce self-attention (SA) layers to better capture the point-point relationship. Besides, we introduce an additional classification loss for the global vector v to encourage object-aware point cloud generation. For the second problem, we propose to project point clouds to the feature space via a pre-trained point cloud auto-encoder. Then we enforce their similarity using MSE loss.

4.1. PCN with self-attention layers

In this project, we utilize the SA layers introduce in Point Cloud Transformer (PCT) [5], which is shown in Figure. 3. In [5], the authors use a stack of such SA layers and achieve the *state-of-the-art* performance in point cloud classification and segmentation. Similar with the transformer in NLP, for each d-dimensional point feature, three vectors are computed: *query*, *key* and *value* through linear layers. Then, the attention weight between any two points can be obtained by matching (dot-producting) their query and key vectors, and the attention feature is defined as the weighted sum of all value vectors with attention weights. Then the attention feature is subtracted by the input feature, passed through a LBR layer and then added with the input through an additional residual path to produce the output. This SA structure is quite suitable for point cloud outputs because 1) the SA operation is inherently permutation-invariant, 2) SA operation is able to learn the relationship between two points that are far from each other in the space.

Then, we modified the PCN encoder by inserting additional SA layers and the structure is shown in Figure. 4. We follow the data processing of PCN encoder to get the feature embeddings for each point. In the meantime, we treat the intermediate point feature $\bar{F}$ as the positional embedding vectors since it contains no inter-point connections. We find that adding such position embedding to each SA layer greatly improves the converging speed during training. After the three consecutive SA layers, we concatenate their outputs as the new point feature $\tilde{F}$ and feed them to another PN layer to get the final global vector v.

4.2. Additional Classification Loss

To make the global feature more powerful, we train an additional classification network that takes the global feature v as input and predict its object category. The motivation is that we think the distribution of points is significantly related to the object categories. For example, the surface of a car tends to be flat and smooth while a table usually contains thin legs and sudden curvature changes at the joints.

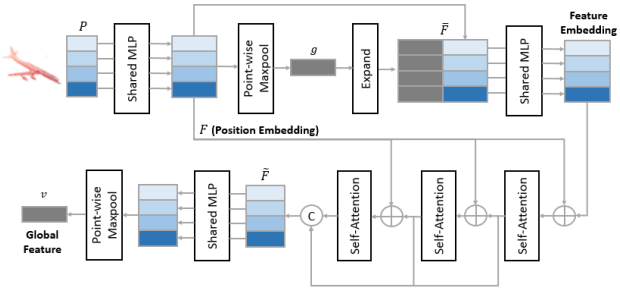

Figure 4. Diagram of the modified PCN encoder with self-attention layers.

Let v_i be the global feature of the ith point cloud and L_i be its one-hot label where the value is one for correct class and is zero otherwise. Suppose the classification network is D, the probability vector p_i for the ith point cloud can be calculated as $p_i = softmax\{D(v_i)\}$. Then, the classification loss L_{cla} is:

$$L_{cla} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} L_{ij}\log(p_{ij})$$

where C is the total number of categories.

4.3. Constraints in Feature Space

Another possible way to emphasize and preserve the global feature during training is to add constraints in feature space on top of the reconstruction loss given by CD. Similar to other losses and criteria we examine, this feature space perceptual loss is also permutation invariant, based on the assumption that the bottleneck feature vector captures high-level information of the input point cloud. In this way, additional supervision at the bottleneck can directly regularize the encoder network.

In our experiment, we first train an autoencoder that aims to reconstruct the ground truth point cloud, then freeze this encoder when training PCN. With the feature vector of a partial point cloud encoded by PCN encoder $E_{pcn}(\cdot)$, we simultaneously encode the corresponding ground truth point cloud with the pretrained encoder $E_{pre}(\cdot)$ and acquire a "ground truth" feature vector. Then the feature space constraints is given by the mean-squared error between the feature vectors:

$$L_{feat} = \text{MSE}(E_{pcn}(P_{partial}), E_{pre}(P_{gt})),$$

where $P_{partial}$ and P_{gt} refer to the input and ground truth point clouds correspondingly.

5. Experiments

5.1. Experimental Setup

Dataset. We train and evaluate the point cloud completion network on the Completion3D [11] dataset, which is a subdivision of the ShapeNet. This dataset contains point clouds of 8 categories, including 28,784 training samples and 800 validation samples. The ground truth point clouds has 2048 points, which is quite sparse and friendly to our project. For each complete point cloud, 8 partial point clouds are sampled from 8 randomly distributed viewpoints.

Evaluation metric. We use the Chamfer Distance (CD) to evaluate the models we reproduce and improved on the validation split of the Completion3D [11] dataset.

Models. As comparison, we train a baseline model following prior works [4, 11] where only a set of fully connected layers is adopted as the point cloud decoder. We also use the PCN model as another baseline model.

For our proposed method, we test three different models: *PCN+self-attn*, *PCN+classification* and *PCN+feature*. The *PCN+self-attn* model simply replaces the PCN encoder with the new encoder proposed in section 4.1. *PCN+classification* adds the additional classification network in section 4.2 to the PCN model. *PCN+feature* adds the feature loss in section to the PCN model. Due to the time limit, we haven't integrate all proposed components.

All models we tested contains a CD loss L_{CD} for both the coarse point cloud and fine point cloud. The loss function for *PCN+classification* is a weighted sum of L_{CD} and L_{cla} while the loss function for *PCN+feature* is a weighted sum of L_{CD} and L_{feat}.

5.2. Quantitative analysis

The quantitative analysis of different point cloud completion methods is shown in Table. 1. Among the three proposed models, *PCN+classification* and *PCN+self-attn* outperforms the two baselines. It can be observed that *PCN+self-attn* achieves the best performance and provides a large improvement against the PCN baseline, demonstrating the power of SA layers. On the contrary, we achieve a worse performance with *PCN+feature* compared to the PCN baseline. However, we think it is too early to say that adding additional feature loss doesn't help. One possible reason for the unsatisfactory result is that the hyperparameters or the training strategy we are using is not optimal and we need more experiments to find the best scheme.

5.3. Qualitative analysis

From the data for evaluation, we draw samples belonging to different categories and visualize the outcome of different models in Figure 5. As illustrated in the figure, *PCN+self-attn* is able to produce the most faithful reconstructions even in cases where the cues on shape and construction barely exists in the input (incomplete) point clouds. Moreover, the model with SA layers can concentrate the points in the recovered point clouds, resulting in more informative constructions to better serve the purpose of point cloud completion such as refining a LiDAR map.

Table 1. Point completion results on Completion3D measured using Chamfer Distance (CD). For each evaluated model, the CD is computed on 2,048 points and multiplied by 10^4.

Methods	Average	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft
Baseline (reproduced)	17.52	5.71	20.18	8.23	20.22	30.60	15.15	26.54	13.53
PCN (reproduced)	16.64	5.13	21.02	8.15	19.68	26.33	14.28	26.51	12.05
PCN + feature	17.46	5.22	22.13	8.26	19.40	30.51	15.66	26.36	12.15
PCN + classification	15.80	4.43	19.49	7.66	18.31	26.54	13.44	25.33	11.22
PCN + self-attn	**13.65**	**3.68**	**18.77**	**7.10**	**16.65**	**19.50**	**12.34**	**22.03**	**9.12**

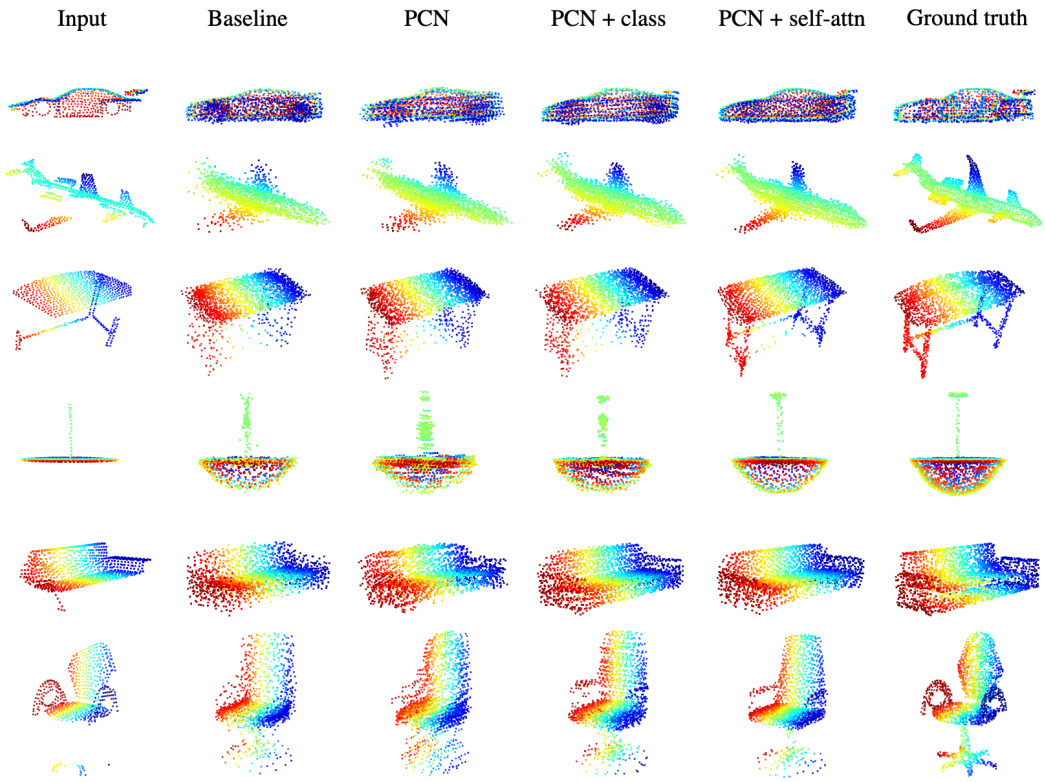

Figure 5. Point cloud completion results on the validation split of Completion3D.

We also notice that there are cases where none of the point completion models we study can produce reconstructions close enough to the ground truth. Some typical failure cases are presented in Figure 6 in the Appendix. By observation, the models are not capable of reconstructing hollow shapes such as the lamp at the top row of Figure 6. When the object is rare in the training set (as illustrated in the middle row), the recovered point cloud will be dissimilar to the ground truth. Nevertheless, the recovery is informative, provided that the partial point cloud is recovered to a sedan rather than a truck. We also find that our models are likely to fail when the input point cloud contains little information about the object (as shown in the last row). In this case, our proposed improvements manage to produce more reasonable reconstructions than the baseline methods.

6. Conclusions

In this work, we re-implemented PCN [16] on PyTorch platform and explored several ways to improve this method. In general, our proposed improvements aim to provide more powerful supervision so as to preserve the global feature of the recovered point cloud that inherits from the input (partial) point cloud. Based on this goal, we present the usage of self-attention layers, adding classification and feature losses. Evaluation results show that the modified methods outperform the baselines from the viewpoints of both Chamfer Distance and visual quality.

References

[1] 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.

[2] Xuelin Chen, Baoquan Chen, and Niloy J Mitra. Unpaired point cloud completion on real scans using adversarial training. *arXiv preprint arXiv:1904.00069*, 2019.

[3] 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.

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

[5] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020.

[6] Tao Hu, Zhizhong Han, and Matthias Zwicker. 3d shape completion with multi-view consistent inference. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 10997–11004. AAAI Press, 2020.

[7] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7662–7670, 2020.

[8] Justin Johnson, Nikhila Ravi, Jeremy Reizenstein, David Novotny, Shubham Tulsiani, Christoph Lassner, and Steve Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses*, pages 1–1. 2020.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017.

[10] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[11] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019.

[12] Xin Wen, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Cycle4completion: Unpaired point cloud completion using cycle transformation with missing region coding. *arXiv preprint arXiv:2103.07838*, 2021.

[13] Chulin Xie, Chuxin Wang, Bo Zhang, Hao Yang, Dong Chen, and Fang Wen. Style-based point generator with adversarial rendering for point cloud completion. *arXiv preprint arXiv:2103.02535*, 2021.

[14] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In *ECCV*, 2020.

[15] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

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

A. Failure Cases

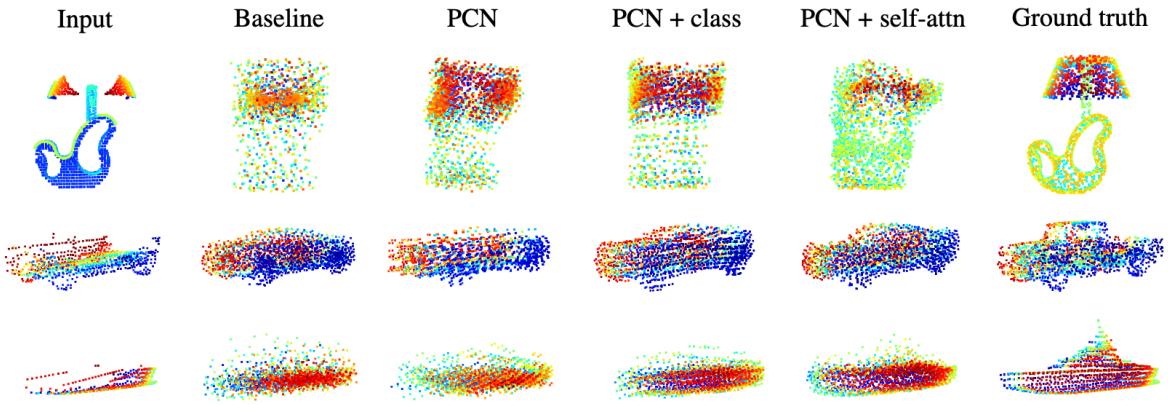

Figure 6. Visualization of failure cases observed in evaluation.

B. Multiview Loss

As pointed out by [14], the distance criteria do not always faithfully reflect the reconstruction quality regarding the preservation of structural correspondence within the point could. Further more, [6] underline that in real-word scenarios it is unlikely to obtain supervision from completed, fine-grained point clouds. Under the aforementioned concerns, we experimented with an adversarial loss term that leverages the consistency held among the 2D projections from multiple view points of a single point cloud by drawing inspiration from [13].

To achieve this, we rendered the completed point cloud and the ground truth point cloud with the same pose and trained a discriminator to distinguish between the two rendered results. To ensure that the completed point cloud has reasonable appearances all around, we rendered each pointcloud with 8 viewpoints and added up the losses. We utilized PyTorch3D's renderer [8] to project the 3D model with the OpenGL orthographic camera. One example of this rendering is shown in Figure 7. We have included this part of the code in our github repo but are still in the process of jointly training the reconstruction loss with this adversarial loss due to limits of computing resources. We hope to present the results in our future work.

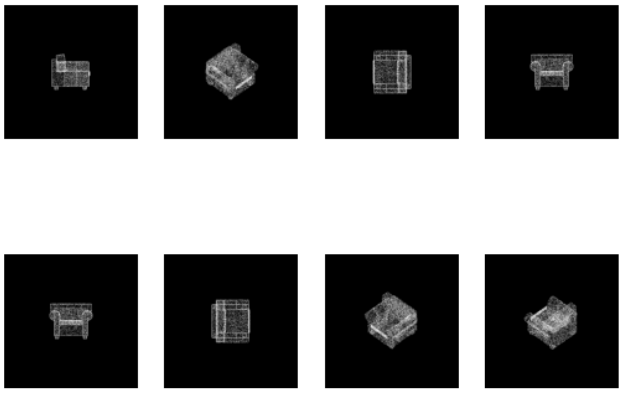

Figure 7. Example of multi-view rendering of a chair pointcloud