

---

# Points2NeRF: Generating Neural Radiance Fields from 3D point cloud

---

D. Zimny, T. Trzciński, P. Spurek  
 Faculty of Mathematics and Computer Science,  
 Jagiellonian University, Kraków, Poland  
 przemyslaw.spurek@gmail.com

## Abstract

Contemporary registration devices for 3D visual information, such as LIDARs and various depth cameras, capture data as 3D point clouds. In turn, such clouds are challenging to be processed due to their size and complexity. Existing methods address this problem by fitting a mesh to the point cloud and rendering it instead. This approach, however, leads to the reduced fidelity of the resulting visualization and misses color information of the objects crucial in computer graphics applications. In this work, we propose to mitigate this challenge by representing 3D objects as Neural Radiance Fields (NeRFs). We leverage a hypernetwork paradigm and train the model to take a 3D point cloud with the associated color values and return a NeRF network's weights that reconstruct 3D objects from input 2D images. Our method provides efficient 3D object representation and offers several advantages over the existing approaches, including the ability to condition NeRFs and improved generalization beyond objects seen in training. The latter we also confirmed in the results of our empirical evaluation.

## 1 Introduction

3D registration devices, such as LIDARs and depth cameras, have become ubiquitous across a multitude of contemporary computer vision applications, including autonomous driving [54] or robotic manipulation [15]. They capture the surrounding 3D visual data as point clouds - structures that require large memory and computational costs to be processed and rendered. To increase the efficiency of processing the point clouds, current approaches transform them into regular 3D voxel grids or collections of images [42, 52]. This, however, increases the memory footprint of object representations and leads to reduced fidelity of the resulting visualization by omitting color information required, *e.g.* for computer graphics applications.

Recently introduced approaches attempt to mitigate those challenges by representing 3D objects as meshes [38, 41, 47, 50] or voxels [5, 30], rather than 3D point clouds, which makes them easier to be rendered. Unfortunately, the former approaches assume that the resulting meshes are topologically coherent with spheres which reduces the generality of this approach. In contrast, the latter methods assume voxelized training datasets and are hard to generalize beyond objects provided during training.

This paper addresses all the above problems by representing objects as Neural Radiance Fields (NeRF) [25]. NeRF represents a static scene as a continuous 5D function that outputs the radiance emitted in each direction  $(\theta, \phi)$  at each point  $(x, y, z)$  in space, and a density at each point which acts like a differential opacity controlling how much radiance is accumulated by a ray passing through  $(x, y, z)$ . The vanilla NeRF method optimizes a deep fully-connected neural network to represent this function by regressing from a single 5D coordinate  $(x, y, z, \theta, \phi)$  to a single volume density and view-dependent RGB color. In this work, we introduce a Points2NeRF<sup>1</sup> method that leverages a

---

<sup>1</sup>We make our implementation available at <https://github.com/gmum/points2nerf>.

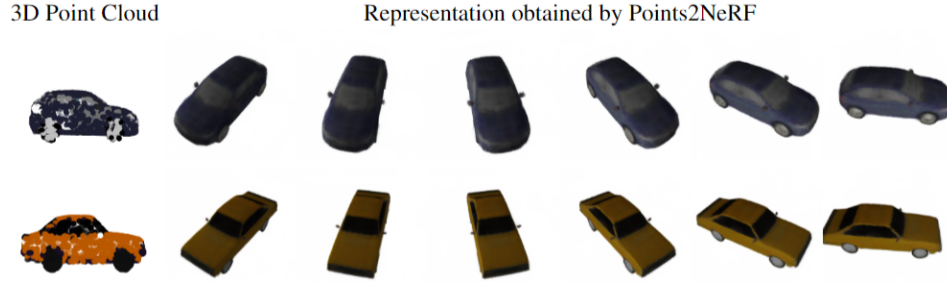


Figure 1: Our Points2NeRF approach takes a 3D point cloud with the associated color values and returns the weights of a NeRF network that reconstructs 3D objects with high fidelity and coherent coloring.

hypernetworks architecture [11] to produce a NeRF representation of a 3D point cloud, as Fig. 1 shows. The main intuition behind our approach is to train a hypernetwork that takes a 3D point cloud with colors as an input and returns the parameters of the target network – NeRF. As a result, we obtain a set of network parameters that can be interpreted as a continuous parametrization of the 3D objects. Furthermore, the resulting NeRFs are easy to render and can be converted to meshes with colors. Last but not least, this formulation allows for conditioning NeRF models and offers improved generalization beyond 3D objects seen during training.

To summarize, our contributions of this work are the following:

- We propose a new method dubbed Points2NeRF which adapts a hypernetwork framework to the NeRF architecture, and hence allows to produce Radiance Fields from 3D point cloud.
- Our approach enables conditioning NeRFs, which, in turn, allows to generalize the model beyond 3D objects seen in training.
- Lastly, our method offers a generative model that can represent 3D objects as NeRF parameters in a continuous manner, enabling interpolation within the object space.

## 2 Related works

3D objects can be represented by using various techniques, including voxel grids [6, 9, 20, 51], octrees [12, 33, 44, 49], multi-view images [2, 21, 42], point clouds [1, 8, 31, 32, 56], geometry images [35, 36], deformable meshes [9, 36, 47, 56], and part-based structural graphs [19, 57].

Mentioned above representations are discrete, which is a substantial limitation in real-life applications. Alternatively, we can represent 3D objects as a continuous function [7]. In practice implicit occupancy [5, 23, 29], distance field [24, 28] and surface parametrization [55, 38, 41, 4] models use a neural network to parameterize a 3D object. We do not have a fixed number of voxels, points, or vertices in such a case, but we represent shapes as a continuous function.

In [23] the occupancy networks represent the 3D surface as a continuous decision boundary of a deep classifier. In [5, 29] the authors propose an implicit decoder (IM-NET) which uses a binary classifier that takes a point coordinate concatenated with a feature vector encoding a shape and outputs an inside or outside label.

DeepSDF [28] represents the surface of the object by a continuous volumetric field. Points represent the distance to the surface boundary, and the sign indicates whether the region is inside or outside. Such representation parameterizes a shape’s border as a classification boundary.

In [38, 41] the authors propose HyperCloud model that uses a hypernetwork to output weights of a generative network to create 3D point clouds instead of generating a fixed size reconstruction. One neural network is trained to produce a continuous representation of an object. In [55] authors propose to use a conditioning mechanism to produce a flow model which transfers gaussian noise into a 3D object.

These models are limited by their requirement of access to ground truth 3D geometry. Recently works relaxed this requirement of ground truth 3D shapes by using only 2D images. In [26] authors present 3D occupancy fields. The numerical method is used to find the surface intersection for each ray. In [37] propose neural network which produces feature vector and RGB color at each continuous 3D coordinate, and propose a differentiable rendering function consisting of a recurrent neural network.

Above models are limited to simple shapes with low geometric complexity, resulting in over smoothed renderings. To solve such a problem NeRF [25] model was proposed. NeRF represents a static scene as a continuous 5D function that outputs the radiance emitted in each direction at each point and a density at each point which acts like a differential opacity controlling how much radiance is accumulated by a ray passing through the point.

The NeRF method is a state-of-the-art solution for representing 3D objects. However, there are many different generalizations of the model for static [25] as well as dynamic scene [14].

Information from point clouds was used in NeRF across different applications. In [53] authors present a novel neural scene representation Point-NeRF that models a volumetric radiance field with a neural point cloud. In NeuS [48], authors propose to represent a surface as the zero-level set of a signed distance function (SDF) and develop a new volume rendering method to train a neural SDF representation. In consequence, we obtain a novel neural surface reconstruction method. In [27] authors introduce Neural Point Light Fields that represent scenes implicitly with a light field living on a sparse point cloud. In our work, we use point cloud as a conditional for producing NeRF representation.

Most of such models are trained on a single scene. NeRF-VAE [18] is a 3D scene generative model. In contrast to NeRF, such a model considers shared structure across scenes. Unfortunately, the model was trained only on simple scenes containing geometric figures. In our paper, we present Points2NeRF, which is trained on an extensive data set and can transform a 3D point cloud to NeRF representation.

### 3 Points2NeRF: generating Neural Radiance Fields from 3D point cloud

In this section, we present our Points2NeRF model for building NeRF representations of 3D point clouds. To that end, we leverage three main components, described below: the auto-encoder architecture, the NeRF representation of a 3D static scene, and the hypernetwork training paradigm. The main intuition behind our approach is the construction of an auto-encoder, which takes as an input 3D point cloud and generates weight of the target network – NeRF. For effective NeRF training, our model requires a set of 2D images, on top of the 3D point clouds, all of which is captured by 3D registration devices.

**Hypernetwork** Hypernetworks, introduced in [11], are defined as neural models that generate weights for a separate target network solving a specific task. The authors aim to reduce the number of trainable parameters by designing a Hypernetwork with a smaller number of parameters. Making an analogy between Hypernetworks and generative models, the authors of [34], use this mechanism to generate a diverse set of target networks approximating the same function.

In the context of 3D objects, various methods make use of a hypernetwork to produce a continuous representation of objects [30, 39, 38, 40, 41]. HyperCloud [38] represent 3D point cloud as a classical NLP while in [41] by a Continuous Normalizing Flow [10]. In the case of [30] authors model voxel representation by hypernetwork architecture.

**Auto-encoders for 3D Point Clouds** In our approach, we use hypernetwork paradigm to aggregate information from 3D point cloud representation and produce the weight of NeRF architecture. Moreover, such a solution allows us to create a high-resolution model of the 3D point cloud.

In Points2NeRF hypernetwork is an auto-encoder type architecture for the 3D point cloud. Let  $\mathcal{X} = \{X_i\}_{i=1,\dots,n} = \{(x_i, y_i, z_i, r_i, g_i, b_i)\}_{i=1,\dots,n}$  be a given dataset containing point clouds with colors. The first three elements encode the position while the last three encode an RGB color. The basic aim of autoencoder is to transport the data through a typically, but not necessarily, lower-dimensional latent space  $\mathcal{Z} \subseteq \mathbb{R}^D$  while minimizing the reconstruction error. Thus, we search for an encoder  $\mathcal{E} : \mathcal{X} \rightarrow \mathcal{Z}$  and decoder  $\mathcal{D} : \mathcal{Z} \rightarrow \mathcal{X}$  functions, which minimizes the reconstruction error

between  $X_i$  and its reconstructions  $\mathcal{D}(\mathcal{E}X_i)$ . We use a permutation invariant encoder that is based on PointNet architecture [31] and a modified decoder to produce weight instead of row points. For point cloud representation, the crucial step is to define proper reconstruction loss that can be used in the auto encoding framework. In our approach, we use NeRF cost function.



Figure 2: Reconstruction of object obtained by Points2NeRF.

**Neural Radiance Fields (NeRF)** NeRF [25] achieves state-of-the-art results for synthesizing novel views of complex scenes. The algorithm represents a scene using a fully-connected architecture. On the input NeRF takes 5D coordinate (spatial location  $\mathbf{x} = (x, y, z)$  and viewing direction  $(\theta, \psi)$ )

and whose output is an emitted color  $\mathbf{c} = (r, g, b)$  and volume density  $\sigma$ . Since volume rendering is naturally differentiable, the only input required to optimize NeRF is a set of images with known camera poses.

In practice, NeRF express direction as a 3D Cartesian unit vector  $\mathbf{d}$ . We approximate this continuous 5D scene representation with an MLP network  $F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$  and optimize its weights  $\Theta$  to map from each input 5D coordinate to its corresponding volume density and directional emitted color. The MLP  $F_\Theta$  first processes the input 3D coordinate  $\mathbf{x}$  with 8 fully-connected layers, and outputs  $\sigma$  and a 256-dimensional feature vector. This feature vector is then concatenated with the camera ray’s viewing direction and passed to one additional fully-connected layer that outputs the view-dependent RGB color. The architecture of NeRF is relatively simple but uses many weights. Therefore Hypernetwork must generate many parameters. Therefore we use the chunking mechanism [46]. In practice, we generate weight to a single layer by conditioning our hypernetwork by the index of layers we would like to produce.

The cost function of NeRF is inspired by classical volume rendering [13]. We render the color of any ray passing through the scene. The volume density  $\sigma(\mathbf{x})$  can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location  $\mathbf{x}$ . The expected color  $C(\mathbf{r})$  of camera ray  $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$  with near and far bounds  $t_n$  and  $t_f$  is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), \mathbf{d}) dt, \text{ where } T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right).$$

The function  $T(t)$  denotes the accumulated transmittance along the ray from  $t_n$  to  $t$ , i.e., the probability that the ray travels from  $t_n$  to  $t$  without hitting any other particle. In practice, this continuous integral is numerically estimated using quadrature. We use a stratified sampling approach where we partition  $[t_n, t_f]$  into  $N$  evenly-spaced bins and then draw one sample uniformly at random from within each bin:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right].$$

We use these samples to estimate  $C(\mathbf{r})$  with the quadrature rule discussed in the volume rendering review by Max [22]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \text{ where } T(t) = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right).$$

where  $\delta_i = t_{i+1} - t_i$  is the distance between adjacent samples. This function for calculating  $\hat{C}(\mathbf{r})$  from the set of  $(\mathbf{c}_i, \sigma_i)$  values trivially differentiable.

We then use the volume rendering procedure to render the color of each ray from both sets of samples. Our loss is simply the total squared error between the rendered and true pixel colors

$$\mathcal{L} = \sum_{\mathbf{r} \in R} \|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2 \quad (1)$$

where  $R$  is the set of rays in each batch, and  $C(\mathbf{r})$ ,  $\hat{C}(\mathbf{r})$  are the ground truth and predicted RGB colors for ray  $\mathbf{r}$  respectively. Contrary to the original paper of NeRF, we use only single architecture.

**Points2NeRF** In our model we use such three components: hypernetwork, autoencoder and NeRF. The Points2NeRF model uses Hypernetwork to output weights of generative network to create NeRF representation from 3D point cloud. More specifically, we present parameterization of the 3D objects as a function  $F_\Theta : \mathbb{R}^5 \rightarrow \mathbb{R}^4$ , which given location  $(x, y, z)$  and viewing direction  $(\theta, \psi)$  returns a color  $\mathbf{c} = (r, g, b)$  and volume density  $\sigma$ . Roughly speaking, instead of producing 3D objects, we would like to produce many neural networks (a different neural network for each object) that model them.

We have one neural network architecture that uses different weights for each 3D object in practice. The target network (NeRF) is not trained directly. We use a Hypernetwork  $H_\Phi : \mathbb{R}^3 \supset X \rightarrow \Theta$ , which for an point-cloud  $X \subset \mathbb{R}^3$  returns weights  $\Theta$  to the corresponding target network (NeRF)  $F_\Theta$ . Thus, a point cloud  $X$  is represented by a function

$$F((x, y, x, \theta, \psi); \Theta) = F((x, y, x, \theta, \psi); H_\Phi(X)).$$



Figure 3: From NeRF representation, we can extract 3D mesh with color. Mesh representation of objects produced by Points2NeRF.

To use the above model, we need to train the weights  $\Phi$  of the hypernetwork. For this purpose, we minimize the NeRF cost function over the training set consisting of pairs: point clouds and 2D images of the object. More precisely, we take an input point cloud  $X \subset \mathbb{R}^6$  (the first three elements encode the position while the last three encode an RGB color) and pass it to  $H_\Phi$ . As a result, the hypernetwork returns weights  $\Theta$  to the target network  $F_\Theta$ . Next, the set of 2D images is compared to the renderings generated by the target network  $F_\Theta$ . As a hypernetwork, we use a permutation invariant encoder that is based on PointNet architecture [31] and a modified decoder to produce weight instead of raw points. The architecture of  $H_\Phi$  consists of: an encoder ( $\mathcal{E}$ ) which is a PointNet-like network that transports the data to lower-dimensional latent space  $\mathcal{Z} \in \mathbb{R}^D$  and a decoder ( $\mathcal{D}$ ) (fully-connected network), which transfers latent space to the vector of weights for the target network. In our framework hypernetwork  $H_\Phi(X)$  represents our autoencoder structure  $\mathcal{D}(\mathcal{E}X)$ . Assuming  $H_\Phi(X) = \mathcal{D}(\mathcal{E}X)$ , we train our model by minimizing the cost function given by equation (1).

We only train a single neural model (hypernetwork), which allows us to produce various functions at test time.

**Mesh representation** In the paper, we represent 3D objects as Neural Radiance Fields. Such representation has few advantages over the classical one. In particular, we can obtain 3D mesh with colors.

Thanks to volume density  $\sigma$ , we obtain voxel representation. We can predict an inside/outside category for points from grid  $(x, y, z)$ . Then we can render objects via the iso-surface extraction method such as Marching Cubes. When we have mesh representation, we can predict the color for all vertices. By using colors in vertices of mesh, we can add colors to the faces of the graph. We present 3D objects with colors in Fig. 3.

**Generative model** In our model, we use autoencoder architecture in hypernetwork. Therefore it is easy to construct a generative model.

Autoencoder-based generative model is a classical auto-encoder model with a modified cost function, which forces the model to be generative, i.e., ensures that the data transported to the latent space comes from the prior distribution (typically Gaussian) [16, 45, 17]. Thus, to construct a generative auto-encoder model, we add to its cost function to measure the distance of a given sample from the prior distribution.

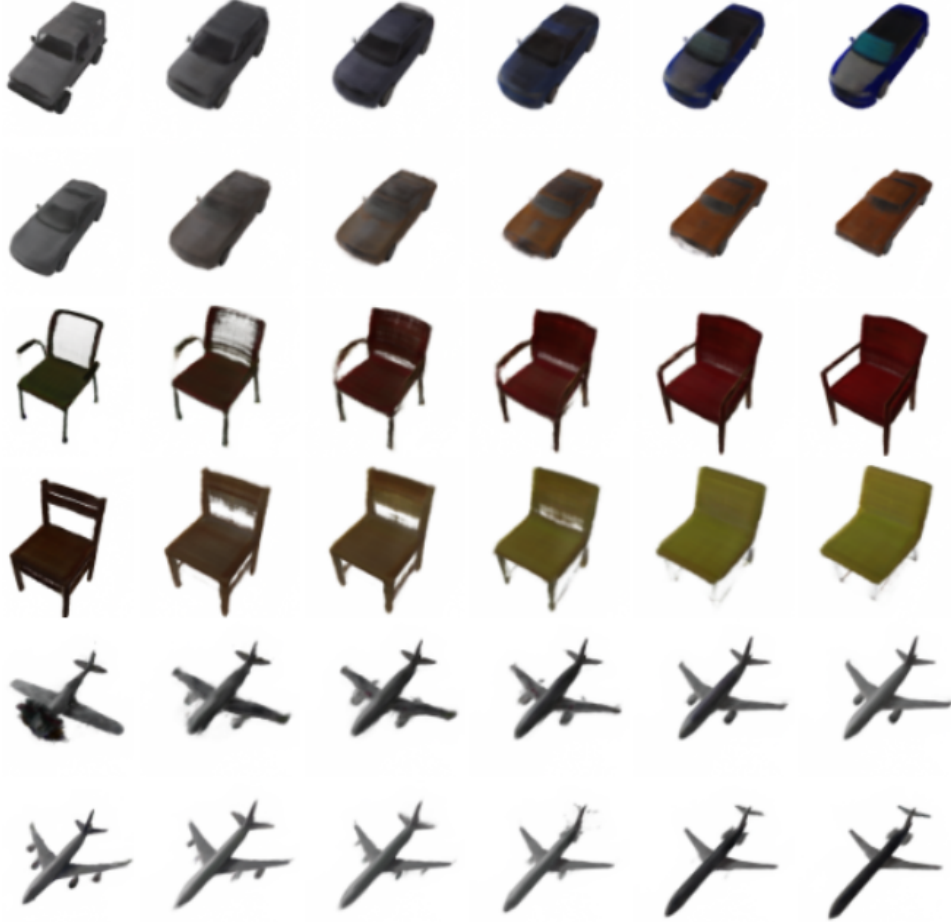


Figure 4: Interpolations between elements produce by Generative Points2NeRF.

Variational Auto-encoders (VAE) are generative models that are capable of learning approximated data distribution by applying variational inference [16]. To ensure that the data transported to latent space  $\mathcal{Z}$  are distributed according to standard normal density. We add the distance from standard multivariate normal density. By adding Kullback–Leibler divergence to our cost function to obtain a generative model, which we cold Generative Points2NeRF. In Fig. 5 we present samples an in Fig. 4 interpolations obtained by Generative Points2NeRF.

## 4 Experiments

In this section, we describe the experimental results of the proposed model. Up to our knowledge, it is the first model which obtains translation from 3D point cloud to Neural Radiance Fields therefore it is hard to compare our results to other algorithms. In the first subsection, we show that our model produces high-quality NeRF representations of the objects by comparing our model with classical NeRF dedicated to static scenes. In the second one, we compare our model by using voxel representation obtained from NeRF.

	Ours, test set	Ours, train set	NeRF
planes	20.45	24.83	27.66
cars	20.86	28.14	
chairs	17.17	23.90	

Figure 6: Comparison between our model trained on ShapeNet data and classical NeRF trained on 8 different objects. Five random images per object from set were taken and mean was calculated. While NeRF used 25 test images.





Figure 5: Object generated by Generative Points2NeRF.

**Methodology** We used **ShapeNet** dataset to train our model. First, we sampled 2048 colored points from each object from three categories: cars, chairs, planes. For each object: fifty 200x200 transparent background images, from random camera positions taken from upper hemisphere were rendered.

**Point to NeRF evaluation** We compare metric reported by NeRF called PSNR (*peak signal-to-noise ratio*) which is used to measure image reconstruction effectiveness. In practice, this comparison can be seen as unfair since NeRF introduces two networks fine and coarse[25] which highly improves rendering of a small detailed elements whereas we only use one coarse model with taking 256 points along each ray.

Additionally, notice that NeRF authors report PSNR only on 8 objects from different dataset and measure it on unseen views from objects that were trained. In our method we compare Points2NeRF reconstructions from training set objects as well as objects from test set. To compute that metric, we take 5 random images related to each cloud of point for that object and compare it with its' reconstruction, see Tab. 6.

We achieve comparable results only for one category but as it was previously mentioned, Points2NeRF could compress many NeRF networks with relatively small loss of information. It is also worth to remember, that our model could reach those PSNR values after three weeks of training for a few thousands of object, where one classical NeRF for one object can take up to two days.

**Voxel representation** NeRF networks, which are output of our hypernetwork can describe occupancy of a given point in 3D space. With use of marching cubes algorithm we can obtain mesh reconstruction for a given point cloud.

To compare reconstruction with original mesh we use Chamfer Distance defined as distance between two cloud of points  $P_1$  and  $P_2$  such that:

$$CD(P_1, P_2) = \frac{1}{2|P_1|} \sum_{p_1 \in P_1} \max_{p_2 \in P_2} d(p_1, p_2) + \frac{1}{2|P_2|} \sum_{p_2 \in P_2} \max_{p_1 \in P_1} d(p_1, p_2) \quad (2)$$

Additionally, we use F-Score metric between two cloud of points with some threshold  $t$  defined as:

$$\text{F-Score}(P_1, P_2, t) = \frac{2\text{Recall Precision}}{\text{Recall} + \text{Precision}}. \quad (3)$$



Category	PointConv[29]	ONet[23]	ConvONet[43]	POCO[3]	Ours
cars	0.577	0.747	0.849	0.946	<b>0.995</b>
planes	0.562	0.829	0.965	<b>0.994</b>	0.952
chairs	0.618	0.730	0.939	0.985	<b>0.992</b>

Table 1: F-Score comparison between our model and other models. We trained our model on 2048 colored points and sampled on 3000 while other methods used 3000 to train and test.

For F-Score and Chamfer Distance calculation, we sample random 3000 points from both original and reconstructed mesh, and we used threshold  $t = 0.01$  to find matching points for F-Score.

Even though our loss function was not directly related to mesh but to image reconstruction, we were able to achieve competitive results, see Tab 1 and Tab 2.

Category	PointConv[29]	ONet[23]	ConvONet[43]	POCO[3]	Ours
cars	1.49	1.04	0.75	0.41	<b>0.16</b>
planes	1.40	0.64	0.34	<b>0.23</b>	0.91
chairs	1.29	0.95	0.46	0.33	<b>0.15</b>

Table 2: Chamfer Distance (multiplied by  $10^2$ ) comparison between our model and other methods. We trained our model on 2048 colored points and sampled on 3000 while other methods used 3000 to train and test.

## 5 Conclusions

In this work, we presented a novel approach to generating NeRF representation from 3D point clouds. Our model leverages a hypernetwork paradigm and NeRF representation of the 3D scene. Points2NeRF take a 3D point cloud with the associated color values and return the weights of a NeRF network that reconstructs 3D objects from 2D images. Such representation gives several advantages over the existing approaches. First of all, we can add a conditioning mechanism to NeRFs that allows to control the process of creating a generative model. Secondly, we can quickly obtain mesh representation with colors, which is a challenging task in 3D object rendering.

**Limitations** The main limitation of Points2NeRF is the computational cost required to build and store NeRF architecture. In practice, hypernetwork must produce many output weights, and consequently, the training time takes up significant chunk of time.

## References

- [1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.
- [2] A. Arsalan Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1511–1519, 2017.
- [3] A. Boulch and R. Marlet. POCO: Point convolution for surface reconstruction. *arXiv:2201.01831*, 2022.
- [4] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 364–381. Springer, 2020.
- [5] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

- [6] 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 *European conference on computer vision*, pages 628–644. Springer, 2016.
- [7] E. Dupont, Y. W. Teh, and A. Doucet. Generative models as distributions of functions. *arXiv preprint arXiv:2102.04776*, 2021.
- [8] 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*, pages 605–613, 2017.
- [9] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016.
- [10] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.
- [11] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.
- [12] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017.
- [13] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.
- [14] K. Kania, K. M. Yi, M. Kowalski, T. Trzciński, and A. Tagliasacchi. Conerf: Controllable neural radiance fields. *arXiv preprint arXiv:2112.01983*, 2021.
- [15] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409, 2015.
- [16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [17] S. Knop, P. Spurek, J. Tabor, I. Podolak, M. Mazur, and S. Jastrzębski. Cramer-wold auto-encoder. *Journal of Machine Learning Research*, 21, 2020.
- [18] A. R. Kosior, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokrá, and D. J. Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *International Conference on Machine Learning*, pages 5742–5752. PMLR, 2021.
- [19] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.
- [20] Y. Liao, S. Donne, and A. Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.
- [21] C.-H. Lin, C. Kong, and S. Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [22] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.
- [23] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.
- [24] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019.

- [25] 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 *European conference on computer vision*, pages 405–421. Springer, 2020.
- [26] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.
- [27] J. Ost, I. Laradji, A. Newell, Y. Bahat, and F. Heide. Neural point light fields. *arXiv preprint arXiv:2112.01473*, 2021.
- [28] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.
- [29] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020.
- [30] M. Proszewska, M. Mazur, T. Trzciński, and P. Spurek. Hypercube: Implicit field representations of voxelized 3d models. *arXiv preprint arXiv:2110.05770*, 2021.
- [31] 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*, pages 652–660, 2017.
- [32] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.
- [33] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. Octnetfusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision (3DV)*, pages 57–66. IEEE, 2017.
- [34] A.-S. Sheikh, K. Rasul, A. Merentitis, and U. Bergmann. Stochastic maximum likelihood optimization via hypernetworks. *arXiv preprint arXiv:1712.01141*, 2017.
- [35] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.
- [36] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6040–6049, 2017.
- [37] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.
- [38] P. Spurek, S. Winczowski, J. Tabor, M. Zamorski, M. Zieba, and T. Trzcinski. Hypernetwork approach to generating point clouds. In *International Conference on Machine Learning*, pages 9099–9108. PMLR, 2020.
- [39] P. Spurek, A. Kasymov, M. Mazur, D. Janik, S. Tadeja, u. Struski, J. Tabor, and T. Trzciński. Hyperpocket: Generative point cloud completion. *arXiv preprint arXiv:2102.05973*, 2021.
- [40] P. Spurek, S. Winczowski, M. Zięba, T. Trzciński, and K. Kania. Modeling 3d surface manifolds with a locally conditioned atlas. *arXiv preprint arXiv:2102.05984*, 2021.
- [41] P. Spurek, M. Zieba, J. Tabor, and T. Trzcinski. General hypernetwork framework for creating 3d point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [42] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

- [43] J. Tang, J. Lei, D. Xu, F. Ma, K. Jia, and L. Zhang. Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6504–6513, 2021.
- [44] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.
- [45] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.
- [46] J. Von Oswald, C. Henning, J. Sacramento, and B. F. Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.
- [47] 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)*, pages 52–67, 2018.
- [48] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.
- [49] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong. Adaptive o-cnn: A patch-based deep representation of 3d shapes. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.
- [50] C. Wen, Y. Zhang, C. Cao, Z. Li, X. Xue, and Y. Fu. Pixel2mesh++: 3d mesh generation and refinement from multi-view images. *arXiv preprint arXiv:2204.09866*, 2022.
- [51] 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 *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 82–90, 2016.
- [52] 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*, pages 1912–1920, 2015.
- [53] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann. Point-nerf: Point-based neural radiance fields. *arXiv preprint arXiv:2201.08845*, 2022.
- [54] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.
- [55] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019.
- [56] Y. Yang, C. Feng, Y. Shen, and D. 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.
- [57] C. Zhu, K. Xu, S. Chaudhuri, R. Yi, and H. Zhang. Scores: Shape composition with recursive substructure priors. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018.