TriVol: Point Cloud Rendering via Triple Volumes

Tao Hu [1]* Xiaogang Xu[1]* Ruihang Chu [1] Jiaya Jia[1,2]
[1] The Chinese University of Hong Kong [2] SmartMore
{taohu,xgxu,rhchu,leojia}@cse.cuhk.edu.hk

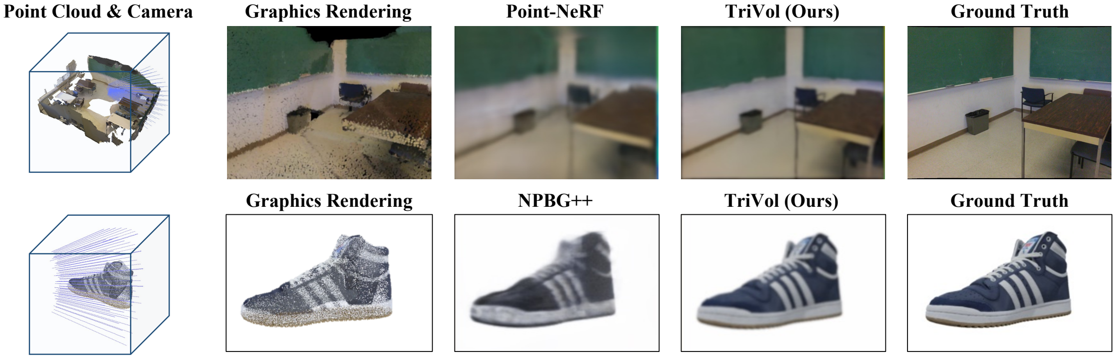

Figure 1. Given the colored point cloud of a category-specific scene or object, our TriVol can render photo-realistic images.

Abstract

Existing learning-based methods for point cloud rendering adopt various 3D representations and feature querying mechanisms to alleviate the sparsity problem of point clouds. However, artifacts still appear in rendered images, due to the challenges in extracting continuous and discriminative 3D features from point clouds. In this paper, we present a dense while lightweight 3D representation, named TriVol, that can be combined with NeRF to render photo-realistic images from point clouds. Our TriVol consists of triple slim volumes, each of which is encoded from the point cloud. TriVol has two advantages. First, it fuses respective fields at different scales and thus extracts local and non-local features for discriminative representation. Second, since the volume size is greatly reduced, our 3D decoder can be efficiently inferred, allowing us to increase the resolution of the 3D space to render more point details. Extensive experiments on different benchmarks with varying kinds of scenes/objects demonstrate our framework's effectiveness compared with current approaches. Moreover, our framework has excellent generalization ability to render a category of scenes/objects without fine-tuning. The source code is available at https://github.com/dvlab-research/TriVol.git.

1. Introduction

Photo-realistic point cloud rendering (without hole artifacts and with clear details) approaches can be employed for a variety of real-world applications, *e.g.*, the visualization of automatic drive [6, 8, 23, 29], digital humans [1, 19, 27], and simulated navigation [5, 11, 43]. Traditional point cloud renderers [36] adopt graphics-based methods, which do not require any learning-based models. They project existing points as image pixels by rasterization and composition. However, due to the complex surface materials in real-world scenes and the limited precision of the 3D scanners, there are a large number of missing points in the input point cloud, leading to vacant and blurred image areas inevitably as illustrated in Fig. 1.

In recent years, learning-based approaches [1, 10, 35, 45, 49] have been proposed to alleviate the rendering problem in graphics-based methods. They use a variety of querying strategies in the point cloud to obtain continuous 3D features for rendering, *e.g.*, the ball querying employed in the PointNet++ [34] and the KNN-based querying in Point-NeRF [45]. However, if the queried position is far away from its nearest points, the feature extraction process will usually fail and have generalization concerns. To guarantee accurate rendering, two groups of frameworks are further proposed. The first group [1, 35, 38] projects the features of all points into the 2D plane and then trains 2D neural networks, like UNet [37], to restore and sharpen the images. However, since such a 2D operation is individual for dif-

*Equal Contribution.

ferent views, the rendering outcomes among nearby views are inconsistent [17, 40], *i.e.*, the appearance of the same object might be distinct under different views. To overcome the artifact, physical-based volume rendering [24] in Neural Radiance Fields (NeRF) [26] without 2D neural rendering is an elegant solution [40]. The second group [10, 47] applies a 3D Convolutional Network (ConvNet) to extract a dense 3D feature volume, then acquires the 3D feature of arbitrary point by trilinear interpolation for subsequent volume rendering. Nevertheless, conducting such a dense 3D network is too heavy for high-resolution 3D representation, limiting their practical application. In summary, to the best of our knowledge, there is currently no lightweight point cloud renderer whose results are simultaneously view-consistent and photo-realistic.

In this paper, we propose a novel 3D representation, named TriVol, to solve the above issues. Our TriVol is composed of three slender volumes which can be efficiently encoded from the point cloud. Compared with dense grid voxels, the computation on TriVol is efficient and the respective fields are enlarged, allowing the 3D representation with high resolution and multi-scale features. Therefore, TriVol can provide discriminative and continuous features for all points in the 3D space. By combining TriVol with NeRF [26], the point cloud rendering results show a significant quantitative and qualitative improvement compared with the state-of-the-art (SOTA) methods.

In our framework, we develop an efficient encoder to transform the point cloud into the *Initial TriVol* and then adopt a decoder to extract the *Feature TriVol*. Although the encoder can be implemented with the conventional point-based backbones [33, 34], we design a simple but effective grouping strategy that does not need extra neural models. The principle is first voxelizing the point cloud into grid voxels and then re-organizing the voxels on each of three axes individually, which is empirically proven to be a better method. As for the decoder, it can extract the feature representation for arbitrary 3D points. Hence, we utilize three independent 3D ConvNet to transfer each volume into dense feature representations. Due to the slender shape of the volumes, the computation of the 3D ConvNet is reduced. Also, the 3D ConvNet can capture more non-local information in the grouped axis via a larger receptive field, and extract local features in the other two directions.

With the acquired dense *Feature TriVol*, the feature of any 3D points can be queried via trilinear interpolation. By combining the queried features with the standard NeRF [26] pipeline, the photo-realistic results are achieved. Extensive experiments are conducted on three representative public datasets (including both datasets of scene [9] and object [4, 12]) level, proving our framework's superiority over recent methods. In addition, benefiting from the discriminative and continuous feature volume in TriVol, our framework has a remarkable generalization ability to render unseen scenes or objects of the same category, when there is no further fine-tuning process.

In conclusion, our contributions are three-fold.

- We propose a dense yet efficient 3D representation called TriVol for point cloud rendering. It is formed by three slim feature volumes and can be efficiently transformed from the point cloud.
- We propose an effective encoder-decoder framework for TriVol representation to achieve photo-realistic and view-consistent rendering from the point cloud.
- Extensive experiments are conducted on various benchmarks, showing the advantages of our framework over current works in terms of rendering quality.

2. Related Works

2.1. 3D Representation

3D representation is very important when analyzing and understanding 3D objects or scenes. There are several important 3D representations, including point cloud, dense voxels [10], sparse voxels [7], Multi-Plane Images (MPI), triple-plane (triplane) [3, 13, 28, 40], multi-plane [25, 39, 41], and NeRF [26], designed from different tasks. A point cloud is a set of discrete data points in space representing a 3D object or scene. Each point location has a coordinate value and could further contain the color. The point cloud is an efficient 3D representation that is usually captured from a real-world scanner or obtained via Multi-view Stereo (MVS) [14]. Each voxel in the dense voxels represents a value on a regular grid in the 3D space. By using interpolation, the continuous 3D features of all 3D positions can be obtained. The sparse voxels [7, 15, 16] are the compressive representation of the dense voxels since only parts of the voxels have valid values. The triplane representation [3, 40] is also the simplification of dense voxels, obtained by projecting the 3D voxels to three orthogonal planes. The MPI [10, 25, 41] represents the target scene as a set of RGB and alpha planes within a reference view frustum. Moreover, NeRF [26] is a recently proposed implicit 3D representation, which can represent the feature of any input 3D coordinate with an MLP. The MLP maps the continuous input 3D coordinate to the geometry and appearance of the scene at that location. We propose TriVol as a new 3D representation of the point cloud and demonstrate its advantages in point cloud rendering.

2.2. Point-based Rendering

Point cloud rendering can be implemented with graphics- and learning-based approaches. The points are projected to the 2D plane via rasterization and composition in the graphics-based algorithms [36]. The learning-based methods [7, 19, 20, 45] design various strategies to compensate the missing information from the sparse point cloud. For example, ME [7] first conducts sparse ConvNet to extract features for existing points, then computes the features of arbitrary 3D points by ball querying in the local space. Obviously, most of the points in the whole 3D space have no meaningful features. Point-NeRF [45] makes use of multi-view images to enhance the features of the input point cloud, formulating the sparse 3D feature volume and then querying any point features by K Nearest Neighbors (KNN) [34]. Also, quite a few learning-based methods [1, 31, 32, 35, 38, 44, 48] project the point cloud onto the 2D plane and utilize the 2D networks to recover the hole artifacts caused by the point cloud's discrete property. For instance, NPBG [1] renders the point cloud with learned neural features in multiple scales and sets a 2D UNet for refinement. Furthermore, several approaches construct 3D feature volumes for rendering [10], *e.g.*, NPCR [10] uses 3D ConvNet to obtain 3D volumes from point clouds and produce multiple depth layers to synthesize novel views.

3. Approach

We aim to train a category-specific point renderer $\mathcal{R}$ to directly generate photo-realistic images I (the image height and width are denoted as H and W) from the colored point cloud P, given camera parameters (intrinsic parameter K and extrinsic parameters R and t). When rendering novel point clouds of the same category, no fine-tuning process is required. The rendering process can be represented as

$$I = \mathcal{R}(P|R, t, K), \tag{1}$$

where P is usually obtained from MVS [2, 14], LiDAR scanners [9], or sampled from synthesized mesh models. In this section, we first encode the point cloud as the proposed TriVol, then utilize three 3D UNet to decode it into the feature representation. Finally, we combine NeRF [26] by querying point features from TriVol at sampled locations to render the final images. An overview of our method is illustrated in Fig. 2.

3.1. TriVol Representation

Grid Voxels. To begin with, we voxelize sparse point cloud P into grid voxels V with shape $\mathbb{R}^{C\times S\times S\times S}$, where S is the resolution of each axis, and C is the number of feature channels. Since V is a sparse tensor, directly querying within V will only get meaningless values for most of the 3D locations, leading to the vacant areas in the rendered images. Therefore, the critical step is transforming the sparse V into a dense and discriminative representation. One approach is employing a 3D encoder-decoder, *e.g.*, 3D UNet. Nevertheless, such a scheme is not efficient to represent a high-resolution space and render fine-grained details. The reason is two-fold: 1) conducting 3D ConvNet on V requires a lot of computations and memory resources, leading to a small value of S [10]; 2) the sparsity of V impedes the feature propagation since regular 3D ConvNet only has a small kernel size and receptive field.

From grid voxels to TriVol. To overcome above two issues of V, we propose TriVol, including V_x, V_y, V_z, as a novel 3D representation. As illustrated in Fig. 2, each item in TriVol is a thin volume whose resolution of one axis is obviously smaller than S, and the others are the same as S. As a consequence, the number of total voxels is reduced for lightweight computations. Note that our TriVol is different from triple-plane representations that are employed in existing works, *e.g.*, ConvOnet [30] and EpiGRAF [40]. Their point features are projected to three standard planes, thus much 3D information might be lost [30].

3.1.1 Encoder for Initial TriVol

We first encode the input point cloud into the *Initial TriVol* ($\{\bar{V}_x, \bar{V}_y, \bar{V}_z\}$). This process can be completed by existing point-cloud-based networks, such as PointNet [33], PointNet++ [34], and Dense/Sparse ConvNet [7, 15]. Nevertheless, these networks bring an additional and heavy computation burden. Instead, we design an efficient strategy without an explicit learning model.

The main step in our encoder is the x-grouping, y-grouping, and z-grouping along different axes. The procedure can be denoted as $\{\bar{V}_x, \bar{V}_y, \bar{V}_z\} = E(V)$, where $E = \{E_x, E_y, E_z\}$. Specifically, to obtain the slim volumes of the x-axis, we first divide V into $G \times S \times S$ groups along the x-axis, thus each group contains $N = S/G$ voxels. Then we concatenate all N voxels in each group as one new feature voxel to obtain $\bar{V}_x \in \mathbb{R}^{(C\cdot N)\times G\times S\times S}$, where $C \cdot N$ is the number of feature channels for each new voxel. $\bar{V}_y$ and $\bar{V}_z$ are encoded by the similar grouping method but along y and z axis, respectively. Therefore, E can be formulated as

$$\begin{aligned}
\bar{V}_x &= E_x(V) \in \mathbb{R}^{(C\cdot N)\times G\times S\times S} \\
\bar{V}_y &= E_y(V) \in \mathbb{R}^{(C\cdot N)\times S\times G\times S} \\
\bar{V}_z &= E_z(V) \in \mathbb{R}^{(C\cdot N)\times S\times S\times G}
\end{aligned} \tag{2}$$

Our encoder is simple and introduces two benefits. Firstly, we can set the different sizes of G and S to balance the performance and computation. When $G \ll S$, huge computing resources are not required, compared with grid voxels V. Thus, we can increase the resolution S to model more point cloud details. Secondly, since the voxels in the same group share the identical receptive field, the receptive field on the grouped axis is amplified N times, allowing the

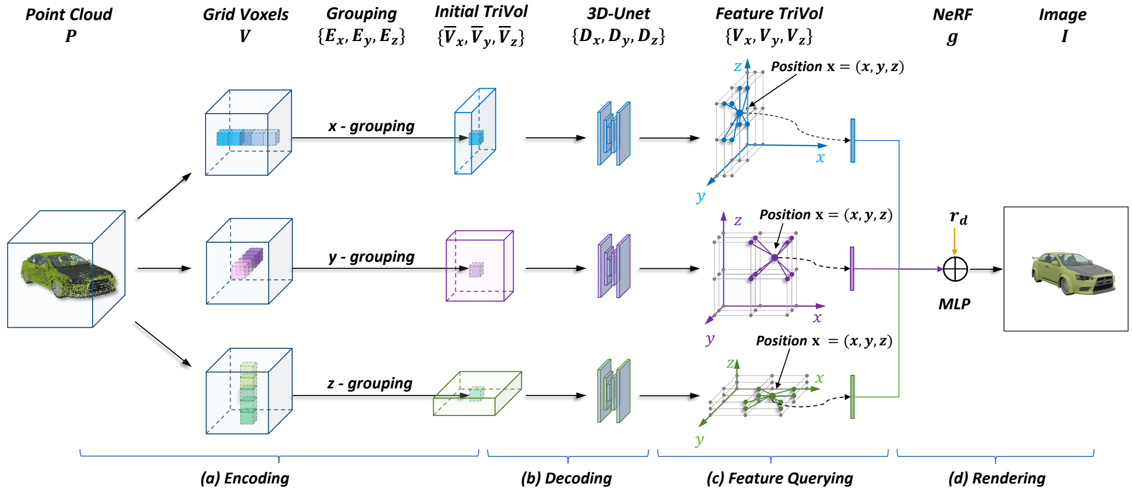

Figure 2. Overview of the proposed TriVol for point cloud rendering. **(a) Encoding**: Input point cloud is encoded to our *Initial TriVol* along x, y, and z axis; **(b) Decoding**: Each volume is decoded to dense feature volume via a unique 3D UNet; **(c) Feature Querying**: Any point's feature is queried by the trilinear interpolation in the *Feature TriVol*; **(d) Rendering**: We combine the queried point feature with NeRF to render the final image.

subsequent decoder to capture global features on the corresponding axis without using a large kernel size. For instance, in volume $\bar{V}_x$, we can extract non-local features on the x-axis and local features on the y and z axis.

3.1.2 Decoder for Feature TriVol

After obtaining the *Initial TriVol* $\{\bar{V}_x, \bar{V}_y, \bar{V}_z\}$ from the encoder, we utilize 3D ConvNet to decode them as the *Feature TriVol* $\{V_x, V_y, V_z\}$. Our *Feature TriVol* decoder consists of three 3D UNet [37] modules $D = \{D_x, D_y, D_z\}$. Each 3D UNet can acquire non-local features on the grouped axis with the amplifying receptive field and can extract local features on the ungrouped two axes that preserve the standard local receptive field. The decoding procedure can be represented as

$$V_x = D_x(\bar{V}_x),\ V_y = D_y(\bar{V}_y),\ V_z = D_z(\bar{V}_z), \qquad (3)$$

where $V_x \in \mathbb{R}^{F\times G\times S\times S}$, $V_y \in \mathbb{R}^{F\times S\times G\times S}$, and $V_z \in \mathbb{R}^{F\times S\times S\times G}$, and F denotes channel number of *Feature TriVol*. Although three 3D UNets are required, the small number of G still makes it possible for us to set a large resolution S without increasing computing resources (verified in Sec. 4), resulting in realistic images with rich details.

3.2. TriVol Rendering

The encoder and decoder modules have transformed the sparse point cloud into a dense and continuous *Feature TriVol*. Therefore, the feature of any 3D location can be directly queried by the trilinear interpolation in $\{V_x, V_y, V_z\}$. Finally, the rendering images can be obtained from the point cloud by following feature querying and volume rendering pipeline of NeRF [26].

3.2.1 Feature Querying

The feature querying consists of point sampling along the casting ray and feature interpolation in the TriVol.

Point sampling. Given the camera parameters $\{R, t, K\}$, we can calculate a random ray with camera center $\mathbf{r}_o \in \mathbb{R}^3$ and normalized direction $\mathbf{r}_d \in \mathbb{R}^3$, we adopt the same coarse-to-fine sampling strategy as NeRF [26] to collect the queried points $\mathbf{x} \in \mathbb{R}^3$ along the ray, as

$$\mathbf{x} = \mathbf{r}_o + z \cdot \mathbf{r}_d, \quad z \in [z_n, z_f], \qquad (4)$$

where z_n, z_f are the near and far depths of the ray.

Querying. For *Feature TriVol* $\{V_x, V_y, V_z\}$ and a queried location $\mathbf{x}$, we first utilize trilinear interpolation to calculate 3 feature vectors: $V_x(\mathbf{x}) \in \mathbb{R}^F$, $V_y(\mathbf{x}) \in \mathbb{R}^F$, and $V_z(\mathbf{x}) \in \mathbb{R}^F$ as shown in Fig. 2. Then, we concatenate them as the final feature $F(\mathbf{x})$ for location $\mathbf{x}$, as

$$F(\mathbf{x}) = V_x(\mathbf{x}) \oplus V_y(\mathbf{x}) \oplus V_z(\mathbf{x}), \qquad (5)$$

where $\oplus$ is the concatenation operation.

3.2.2 Volume Rendering

Implicit mapping. For the queried feature of all points on the ray, we set a Multi-Layer Perceptron (MLP) as an implicit function g to map interpolated feature $F(\mathbf{x})$ to their densities $\sigma \in \mathbb{R}_+$ and colors $c \in \mathbb{R}^3$, with the view direction $\mathbf{r}_d$ as the condition, as

$$\sigma, \mathbf{c} = g(F(\mathbf{x}), \mathbf{r}_d). \qquad (6)$$

Rendering. The final color of each pixel $\hat{\mathbf{c}}$ can be computed by accumulating the radiance on the ray through the pixel

Methods	3D Representation	Feature Extraction	Feature Query	Rendering
NPBG++ [35]	Point Cloud	2D UNet	-	Graphics & CNN
Dense 3D ConvNet	Grid Voxels	3D UNet	Trilinear Interpolation	NeRF
Sparse 3D ConvNet	Sparse Voxels	3D Sparse UNet	Ball Query	NeRF
Point-NeRF [46]	Point Cloud	MLP	KNN	NeRF
Ours	Triple Volumes (TriVol)	3D UNet	Trilinear Interpolation	NeRF

Table 1. Compare different point cloud renderers in 3D representation, feature extraction, feature query, and rendering strategy.

using volume density [24]. Suppose there are M points on the ray, such a volume rendering can be described as

$$\begin{aligned} \hat{\mathbf{c}} &= \sum_{i=1}^{M} T_i \alpha_i \mathbf{c}_i, \\ \alpha_i &= 1 - \mathbf{exp}(-\sigma_i \delta_i), \\ T_i &= \mathbf{exp}(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \end{aligned} \tag{7}$$

where T_i represents volume transmittance, δ_j is the distance between neighbor samples along the ray $\mathbf{r}$.

3.3. Loss Function

Our model is only supervised by the rendering loss, which is calculated by the mean square error between rendered colors and ground-truth colors, as

$$\begin{aligned} \{\hat{\mathbf{c}}_1, ..., \hat{\mathbf{c}}_{H\times W}\} &= \mathcal{R}(P|R, t, K), \\ \mathcal{R} &= \{D_x, D_y, D_z, g\}, \\ \mathcal{L} &= \|\hat{\mathbf{c}}_i - \bar{\mathbf{c}}_i\|_2^2, \end{aligned} \tag{8}$$

where $I = \{\hat{\mathbf{c}}_1, ..., \hat{\mathbf{c}}_{H\times W}\}$ are the rendered colors, $\{\bar{\mathbf{c}}_1, ..., \bar{\mathbf{c}}_{H\times W}\}$ are the ground-truth colors.

4. Experiments

4.1. Datasets

We evaluate the effectiveness of our framework with the TriVol representation on object-level and scene-level datasets. For the object level, we use both synthesized and real-world scanned datasets, including **ShapeNet** [4] and **Google Scanned Objects** (GSO) [12]. ShapeNet is a richly-annotated and large-scale 3D synthesized dataset. There are about 51,300 unique 3D textured mesh models, and we choose the common *Car* category for evaluation. GSO dataset contains over 1000 high-quality 3D-scanned household items, and we perform experiments on the category of *shoe*. The point clouds in object-level datasets can be obtained by 3D mesh sampling [21], and ground-truth rendered images are generated by Blender [18] under random camera poses. For the scene level, we conduct the evaluation on the **ScanNet** [9] dataset, which contains over 1500 indoor scenes. Each scene is constructed from an RGBD camera. We split the first 1,200 scenes as a training set and the rest as a testing set.

4.2. Implementation Details

We set the volume resolution S to 256, and the number of groups G to 16. The decoders (D_x, D_y, and D_z) do not share the weights. The number of layers in NeRF's MLP g is 4. For each iteration, we randomly sample 1024 rays from one viewpoint. The number of points for each ray is 64 for both coarse and fine sampling. The resolutions $H \times W$ of rendered images on the ScanNet [9] dataset are 512 × 640, and the other datasets are 256 × 256. AdamW [22] is adopted as the optimizer, where the learning rate is initialized as 0.001 and will decay to 0.0001 after 100 epochs. We train all the models using four RTX 3090 GPUs. For more details, please refer to the supplementary file.

4.3. Baselines

Besides the comparison with existing point cloud rendering methods, *e.g.*, NPBG++ [35], ADOP [38] and Point-NeRF [46], several important baselines are also combined with NeRF [26] to demonstrate the effectiveness of our framework. They can be described as follows:

- **Voxels-128**: This baseline generates dense voxels with 3D dense UNet. However, due to its constrained efficiency and the memory limitation of the computation resource, the voxel resolution is set as 128^3.
- **Sparse ConvNet**: MinkowskiEngine [7] is a popular sparse convolution library. We make use of its sparse 3D UNet, *i.e.*, MinkUnet34C [7], as the baseline. Its feature querying is performed by a ball query.
- **ConvOnet [30]**: ConvOnet converts the point cloud features into a triple-plane representation for 3D reconstruction. We replace its occupancy prediction with NeRF module to achieve point cloud rendering.

Note that Voxels-128 has the same channel number as TriVol. One alternative baseline is the dense voxels with high resolution, e.g., 256, and reduced channel number for efficiency. However, the minor channel number will impede the formulation of discriminative feature volumes, and such an alternative baseline has been proved to have terrible rendering results empirically. The differences between our framework and these baselines are summarized in Tab. 1. We adopt the metrics of PSNR, SSIM [42], and LPIPS [50] for evaluation.

	ScanNet [9]			ShapeNet [4]			Google Scanned Objects [12]		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Graphics Renderer [36]	13.62	0.528	0.779	19.24	0.814	0.182	23.14	0.829	0.153
Sparse-ConvNet [7]	15.27	0.646	0.602	22.16	0.836	0.159	25.36	0.868	0.128
NPCR [10]	16.22	0.659	0.574	23.41	0.855	0.136	27.73	0.892	0.107
ConvONet [30]	16.43	0.665	0.584	24.25	0.848	0.122	28.17	0.917	0.093
ADOP [38]	16.83	0.699	0.577	24.96	0.857	0.129	29.06	0.922	0.089
NPBG++ [35]	16.81	0.671	0.585	25.32	0.874	0.120	29.42	0.929	0.081
Point-NeRF [46]	17.53	0.685	0.517	**25.73**	**0.897**	**0.107**	**29.84**	**0.938**	**0.069**
Voxels-128	**17.65**	**0.694**	**0.538**	25.53	0.872	0.116	29.41	0.926	0.078
Ours	**18.56**	**0.734**	**0.473**	**27.22**	**0.927**	**0.084**	**31.24**	**0.961**	**0.045**

Table 2. Quantitative comparison for point cloud rendering accuracy between ours and the state-of-the-art methods as well as baselines on the ScanNet [9], ShapeNet [4], and Google Scanned Objects [12] datasets.

4.4. Evaluate Scene-Level Rendering

In this experiment, we compare ours with baselines, and SOTA methods [10, 35, 46] on the ScanNet dataset [9]. For each scene, we sample 100k points on the provided textured mesh as the colored point cloud. Quantitative and qualitative results are presented in Tab. 2 and Fig. 3, respectively.

The point clouds at the scene level usually have missing points and parts, which causes hole artifacts in graphics-based rendering images, as shown in the left column of Fig. 3. The traditional voxel-based method could not generate high-quality images due to its low-resolution representation. There are many artifacts in NPBG++ [35], since the 2D-CNN-based rendering only employs the limited 2D information, not 3D context, to remove the discrete issues, leading to imprecise and unrealistic results when there is no fine-tuning stage. Moreover, the view-inconsistent shortcomings of 2D-CNN-based approaches will be demonstrated in the supplementary file. Furthermore, Point-NeRF [46] could not recover the missing parts. The reason is that the KNN strategy will usually fail if the nearest neighbor points are far from the queried points [34]. Ours performs the best in complementing the missing areas and enhancing local details, thus could render photo-realistic images. Moreover, as shown in Tab. 2, our framework performed better than others by large margins on all metrics. The performance of our method demonstrates the better generalization ability of TriVol compared with the SOTA methods and baselines.

4.5. Evaluate Object-Level Rendering

We also conduct the evaluation of rendering at the object level. Experiments are conducted on the ShapeNet *Car* dataset and the Google Scanned Object *Shoe* dataset. We uniformly sample 100k points for each 3D model. Different from the scene level, most of the point cloud at the object level is relatively dense. Therefore, all methods can achieve higher metrics than the results at the scene level. The quantitative results are shown in Tab. 2, our framework still has the best performance over others, showing the discriminative 3D representation of TriVol. Moreover, the qualitative evaluations are displayed in Figs. 4 and 5. Although there is no fine-tuning process, our method produces rendering results with richer and more precise details than baselines. For example, as shown in Fig. 4, the car wheel structures could be accurately rendered with our approach, and are blurry at baselines' results. The realistic rendering performance demonstrates the effectiveness of our framework in general category-specific point cloud rendering. Please refer to the supplementary file for more object-level comparisons within various categories.

Encoder	Memory (GB)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
PointNet [33]	4.78	18.03	0.698	0.521
PointNet++ [34]	5.23	18.26	0.707	0.505
Sparse ConvNet [7]	2.15	18.50	0.719	0.494
Grouping (Ours)	**0**	**18.56**	**0.734**	**0.473**

Table 3. The effect of different encoders on the ScanNet Dataset. The decoder of these methods are all the same three 3D UNet.

4.6. Ablation Study

In the subsequent experiments, we study the influence of different modules and parameters in our method.

4.6.1 The Effect of Encoder

In this experiment, we explore the performance of different encoders in transforming a point cloud to our *Initial TriVol*. The baselines are the encoders containing the point-based networks, including PointNet [46], PointNet++ [34], and Sparse ConvNet [7]. After the point feature extraction in baselines, we pool the point features to our TriVol's shape. The results are shown in Tab. 3. We note that our grouping strategy does not require additional memory while performing the best, showing the strength in employing the grouping mechanism in the TriVol encoder.

4.6.2 The Effect of TriVol Resolution

One advantage of our method is that high-resolution TriVol can be obtained with lightweight computation through dense 3D ConvNets. In this experiment, we study the influence of TriVol resolution, including the group size G and

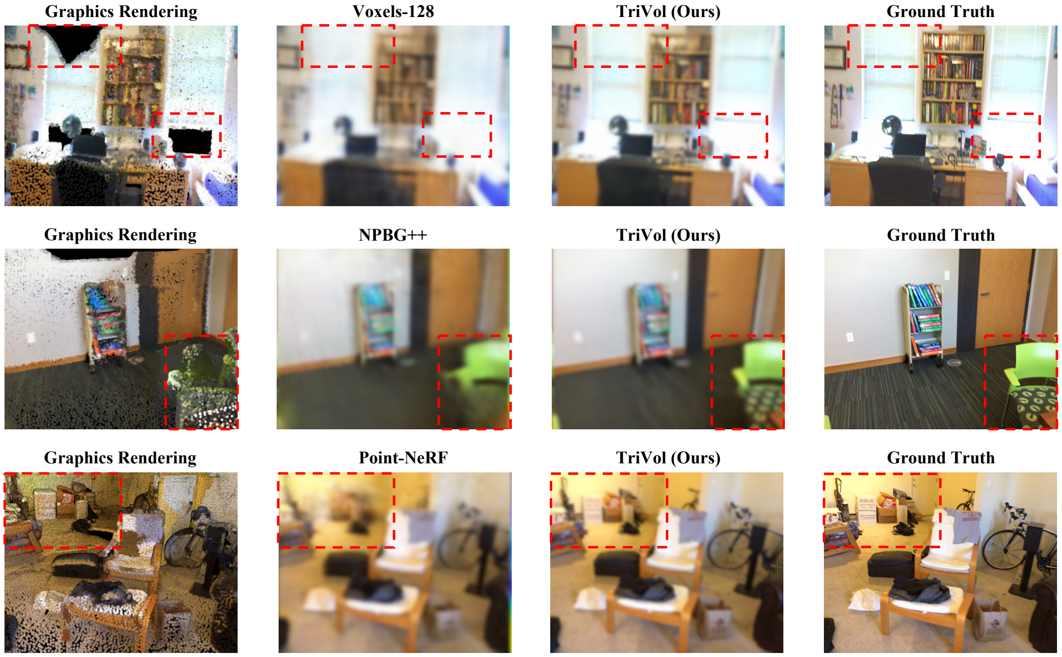

Figure 3. Qualitative point cloud rendering comparison between ours and SOTA methods and baselines on the ScanNet dataset.

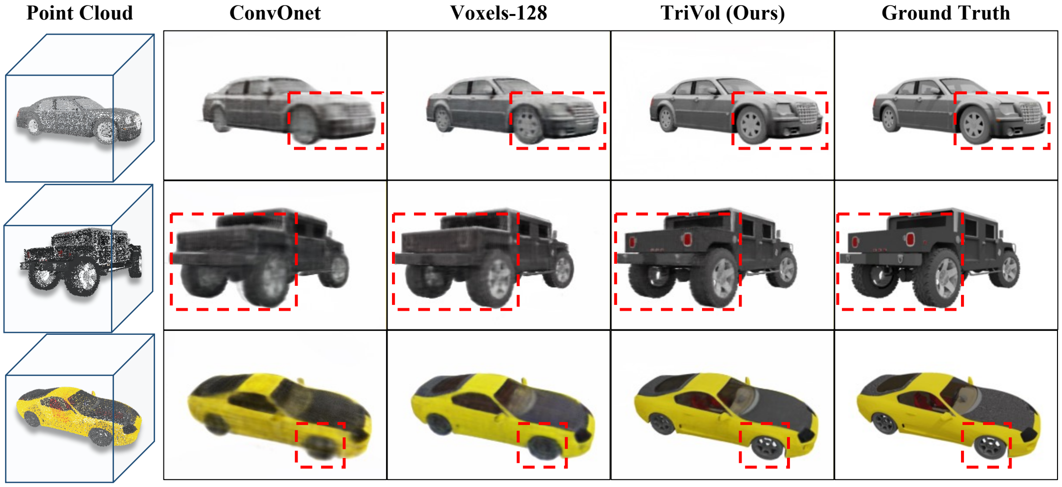

Figure 4. Qualitative point cloud rendering comparison between ours and SOTA methods and baselines on the ShapeNet dataset.

volume resolution S. The experimental results are shown in Tab. 4, where "Number" indicates the number of volumes in each 3D representation. Thus, the top two rows are the baselines of dense voxels with the resolution of 128^3 and 256^3, respectively. The dense voxel with resolution 128^3 requires a tremendous amount of computation resources in terms of GPU Memory and FLOPS. The voxels of 256^3 even return an Out-Of-Memory (OOM) issue. Moreover, we noted that the rendering accuracy with these dense voxels is not satisfying enough. On the other hand, the bottom four rows in Tab. 4 explore the effect of our framework with different combinations between group sizes and volume resolutions. Note that different combinations' memory costs are lower than the dense voxel with 128^3 resolution. The superiority in the efficiency of TriVol can also be observed from the metric of FLOPS. Moreover, TriVol has an much better accuracy in rendering metrics. Therefore, we finally adopt $G = 16$, $S = 256$ to balance the computing resources and accuracy, which has been indicated in Sec. 4.2.

4.6.3 The Effect of Point Number

In the real-world application, the testing scenes or objects might contain a visibly different point number from the

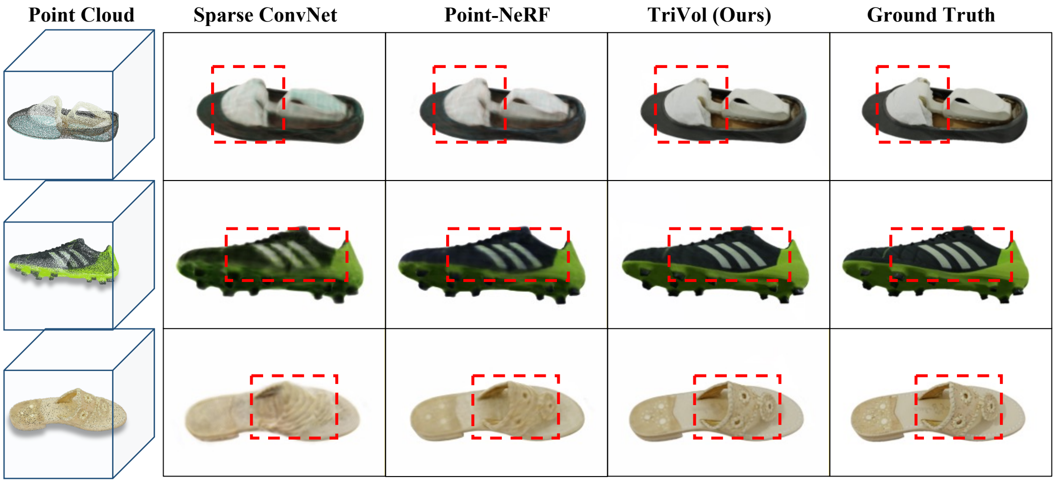

Figure 5. Qualitative point cloud rendering comparison between ours and SOTA methods and baselines on the GSO dataset.

Number	G	S	GPU Memory $(GB, \downarrow)$	FLOPS $(G, \downarrow)$	PSNR (↑)
1	-	128^3	13.83	167.91	25.53
1	-	256^3	OOM	-	-
3	8	64^2	2.11	9.34	22.35
3	16	128^2	3.13	74.69	24.84
3	16	256^2	8.47	103.48	**27.22**
3	32	256^2	9.89	197.55	**27.38**

Table 4. The effect of TriVol Resolution on the ShapeNet Dataset.

Methods	Training Points	Testing Points	PSNR (↑)
Point-NeRF [46]	10k	10k	23.83
	100k	10k	22.75
	100k	100k	25.73
TriVol (Ours)	10k	10k	**25.31**
	100k	10k	**27.01**
	100k	100k	**27.22**

Table 5. The effect of the number of points during training and testing on the ShapeNet dataset.

training data, leading to performance degradation. Therefore, in this experiment, we set different point numbers for the training and evaluation data, as shown in Tab. 5. When the point number during the training and testing are the same, similar to Point-NeRF [46], increasing the input point number can improve the rendering accuracy. Our method achieves the best performance with different numbers of input points. When the number of testing points is far less than the train points, our approach shows a more robust performance compared with Point-NeRF [46] since TriVol is a general discriminative and continuous 3D representation regardless of the point cloud's sparse/dense degree. Note that although the point number is changed, TriVol's resolution is fixed in this experiment. Hence, our TriVol is more robust than Point-NeRF [46] when the points' sparsity changes.

5. Conclusion

In this paper, we analyze the limitations of present point-based rendering methods, including the hole artifacts for graphics-based methods, the view inconsistency for the 2D neural-based approaches, and the inefficiency of existing 3D neural-based strategies. We propose a novel rendering framework by designing a lightweight and continuous 3D representation TriVol. The *Initial TriVol* is obtained from the point cloud with a simple axis grouping mechanism, and the *Feature TriVol* is inferred by efficient computation in 3D UNets. The feature querying for all 3D locations can be completed via the trilinear interpolation in TriVol. The TriVol-based framework can represent high-resolution volumes and adaptive capture both local and non-local information for NeRF-based volume rendering. Extensive experiments are conducted on both scene- and object-level benchmarks, showing our framework can be utilized in rendering images with clear details and without holes artifacts.

Limitation and future work. Although high-quality rendering images were obtained with TriVol from point clouds, it is still very challenging for our method to render the scene where an extremely large number of points are missing. This is technically a 3D scene synthesis task and requires a pre-trained 3D generator trained on a large-scale dataset to synthesize more missing points. We plan to solve it in future work based on the proposed TriVol representation.

Acknowledgments

This work is partially supported by Shenzhen Science and Technology Program KQTD20210811090149095.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020. 1, 3

[2] Dan Cernea. OpenMVS: Multi-view stereo reconstruction library. *City*, 2020. 3

[3] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 2

[4] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arxiv*, 2015. 2, 5, 6

[5] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM TOG*, 2013. 1

[6] Yun Chen, Frieda Rong, Shivam Duggal, Shenlong Wang, Xinchen Yan, Sivabalan Manivasagam, Shangjie Xue, Ersin Yumer, and Raquel Urtasun. Geosim: Realistic video simulation via geometry-aware composition for self-driving. In *CVPR*, 2021. 1

[7] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. 2, 3, 5, 6

[8] Tiago Cortinhal, Fatih Kurnaz, and Eren Erdal Aksoy. Semantics-aware multi-modal domain translation: From lidar point clouds to panoramic color images. In *ICCV*, 2021. 1

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 2, 3, 5, 6

[10] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *CVPR*, 2020. 1, 2, 3, 6

[11] Alexandre Devaux and Mathieu Brédif. Realtime projective multi-texturing of pointclouds and meshes for a realistic street-view web navigation. In *Proceedings of the 21st International Conference on Web3D Technology*, 2016. 1

[12] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *ICRA*, 2022. 2, 5, 6

[13] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022. 2

[14] Michael Goesele, Brian Curless, and Steven M Seitz. Multi-view stereo revisited. In *CVPR*, 2006. 2, 3

[15] Ben Graham. Sparse 3d convolutional neural networks. In *BMVC*, 2015. 2, 3

[16] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arxiv*, 2017. 2

[17] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. StyleNeRF: A style-based 3d-aware generator for high-resolution image synthesis. In *ICLR*, 2021. 2

[18] Roland Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.6*. Focal Press, 2010. 5

[19] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Comput. Graph. Forum*, 2021. 1, 3

[20] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *CVPR*, 2021. 3

[21] Davi Lazzarotto and Touradj Ebrahimi. Sampling color and geometry point clouds from shapenet dataset. *arxiv*, 2022. 5

[22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[23] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 1

[24] Nelson Max. Optical models for direct volume rendering. *IEEE TVCG*, 1995. 2, 5

[25] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 2019. 2

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3, 4, 5

[27] Phong Nguyen-Ha, Nikolaos Sarafianos, Christoph Lassner, Janne Heikkilä, and Tony Tung. Free-viewpoint rgb-d human performance capture and rendering. In *ECCV*, 2022. 1

[28] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Unsupervised learning of efficient geometry-aware neural articulated representations. In *ECCV*, 2022. 2

[29] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *CVPR*, 2022. 1

[30] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 3, 5, 6

[31] Sergey Prokudin, Michael J Black, and Javier Romero. Smplpix: Neural avatars from 3d human models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021. 3

[32] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *CVPR*, 2020. 3

[33] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2, 3, 6

[34] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 2, 3, 6

[35] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. NPBG++: Accelerating neural point-based graphics. In *CVPR*, 2022. 1, 3, 5, 6

[36] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D deep learning with PyTorch3D. *arxiv*, 2020. 1, 3, 6

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *MICCAI*, 2015. 1, 4

[38] Darius Rückert, Linus Franke, and Marc Stamminger. ADOP: approximate differentiable one-pixel point rendering. *ACM TOG*, 2022. 1, 3, 5, 6

[39] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 2

[40] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. *arxiv*, 2022. 2, 3

[41] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 2

[42] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 2004. 5

[43] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *CVPR*, 2019. 1

[44] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 3

[45] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *CVPR*, 2022. 1, 3

[46] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *CVPR*, 2022. 5, 6, 8

[47] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. In *CVPR*, 2022. 2

[48] Ilya Zakharkin, Kirill Mazur, Artur Grigorev, and Victor Lempitsky. Point-based modeling of human clothing. In *ICCV*, 2021. 3

[49] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. *arXiv*, 2022. 1

[50] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5