Point As Gaussian: Forward Point Cloud Rendering

Zibo Zhao[1,2*] Wen Liu[3] Xin Chen[2] Xianfang Zeng[2] Shiyu Liu[1]
Gang Yu[2] Shenghua Gao[1†]
[1]ShanghaiTech University [2]Tencent PCG, China [3]DeekSeek-AI

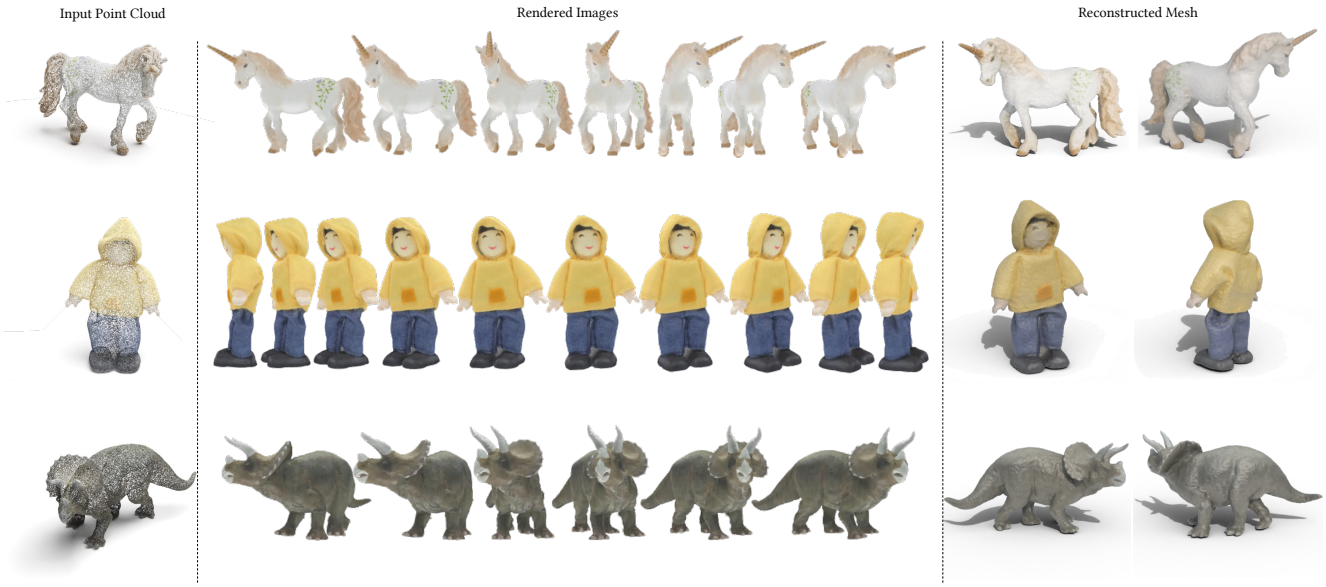

Figure 1. Given a point cloud as input (left column), PAG can recover its radiance field and render photo-realistic images (middle columns). The results support image-based reconstruction algorithms for obtaining meshes (right column).

Abstract

Point clouds, being sparse samples from a surface, inherently produce blur and holes in rendered images. Although previous methods tackle hole-filling by representing point clouds in 3D neural representation and utilizing neural decoders or neural radiance fields for rendering, they lack a unified representation of global and local information, restricting sensitivity to texture changes and rendering quality. To address this limitation, we present Point As Gaussian (PAG), which integrates a hybrid neural radial basis function (H-NRBF) to enable the neural network to capture both local and global features of the point cloud, consequently achieving hole-filling and improving the rendering quality of local details. Moreover, drawing inspiration from recent 3D Gaussian Splatting, we adopt 3D Gaussian as the expression of the radiance field predicted by our model, allowing the model to concentrate on learning point-based features. Extensive experiments on the synthetic dataset ShapeNet and the scanned dataset Google Scanned Objects demonstrate that our model can render the input point cloud into a photo-realistic image without additional optimization or fine-tuning. Additionally, our method offers an alternative approach for mesh reconstruction from point clouds by rendering images from point clouds and subsequently utilizing image-based reconstruction algorithms.

1. Introduction

Point cloud rendering, which strives to generate photo-realistic images, is in high demand due to the widespread

[1]∗Work was partially done while Zibo Zhao was a Research Intern with Tencent PCG.
[2]†Corresponding author.

use of point primitives for representing 3D shapes, offering versatility and efficiency. This technique finds numerous applications across various domains, such as visualizing real scanned or LIDAR point clouds, augmented and virtual reality systems, and the film and video game industry.

Point cloud rendering remains challenging due to the sparse nature of point clouds as discrete surface samples. With the emergence of deep learning approaches, early learning-based methods [17, 18, 43] typically project the point cloud into a feature map and use a neural decoder to render images. Although this improves hole-filling, artifacts persist as the feature map inherits the sparsity of points, and the texture quality is unsatisfactory. The introduction of neural representations has led to new methods for point cloud rendering. Previous works employ a global latent code [26] to represent the 3D shape but often produce rendering results with limited detail and exhibit constrained generalization capabilities. Recent approaches utilize voxel [5], triplane [7], or triple volume [10] encoding for entire point clouds and leverage 3D convolutional neural networks [28] to enhance local information modeling. These approaches achieve impressive results in hole-filling and significantly improve texture details. However, challenges remain as compressing points into voxel or triplane representations can lead to further information loss, making it difficult to restore the texture and shape of the surface.

Recent works, such as 3DS2V [48] and Michelangelo [49], have employed global latent sets (GLS) based on learnable query tokens to model the global features of point clouds. These models directly learn global features from positionally embedded points through cross-attention modules, bypassing additional compression or downsampling. Although these methods effectively restore the geometry of the surface, they face difficulties in recovering the surface's radiance field, which is more complex and exhibits more abrupt changes than geometry. To address this challenge, we designed a local neural radial basis function (L-NRBF) that combines with GLS to form a hybrid neural radial basis function (H-NRBF). H-NRBF can better model both local and global information of point clouds. Specifically, for each point, we identify the point with the most significant difference among its nearest neighbors and use it as an additional feature, modeling the abrupt changes in local patches. Through cross-attention, the global information from the GLS can be integrated with local information, enabling the prediction of a high-quality radiance field.

In addition to effectively representing the information in point clouds, radiance field and rendering techniques are also essential for point cloud rendering. Early methods often utilized neural network decoders to generate final images, and several high-quality neural decoders were proposed [1, 12, 13]. However, these approaches primarily considered 2D feature maps and did not directly model 3D information, resulting in reduced perceptual quality for the color information of the point cloud and rendering blurred images. With the growing popularity of Neural Radiance Fields (NeRF) [23], recent methods predominantly produce images through volume rendering based on predicted neural radiance fields [10, 11, 44]. Although these approaches better utilize 3D information, recovering a continuous neural radiance field from point clouds remains challenging. We observed the recent 3D Gaussian Splats (3DGS) [16], which effectively leverages point-primitives by modeling 3D Gaussian kernels on discrete points, naturally corresponding to the information represented by Hybrid Neural Radial Basis Functions (H-NRBF). Furthermore, by adopting 3D Gaussian Splats as the expression of radiance field, the model only needs to predict the corresponding 3D Gaussian parameters for recovering the radiance fields rather than a continuous neural radiance field.

In summary, our contributions can be outlined: 1) This paper introduces PAG, which leverages a meticulously designed H-NRBF to represent the input's local and global features, thereby enhancing the model's ability to predict superior radiance fields. 2) Once trained, PAG can perform forward translation of points to 3D Gaussians without additional optimization or fine-tuning. 3) Extensive experiments conducted on the ShapeNet [4] and Google Scanned Objects datasets [6] demonstrate that our method can render photo-realistic images from point clouds. 4) PAG offers an alternative solution for reconstructing meshes from point clouds [15].

2. Related Work

We briefly review previous literature of Point cloud rendering, 3D neural representation, and radiance field from computer graphics and computer vision communities. Comprehensive elaborations refer to the surveys [30, 40, 41].

2.1. Point Cloud Rendering

The common **graphics-based approach** for rendering point cloud is rasterization, which projects points onto the 2D plane [20]. However, due to the sparsity of point clouds, holes exist between points, leading to rendering flaws [8]. Classical splatting algorithms place small discs at each point to fill holes [29]. However, the point position varies from different input point clouds and further results in difficulty estimating the shape of discs [37].

Recent **neural decoder-based methods** [3, 22] propose integrating neural networks to enhance point cloud rendering. NPBG [2] rasterizes the point cloud as 2D feature maps and renders final images via neural networks. The neural decoder reduces the rendering artifacts, and ADOP [36] further disentangles the rendering process with exposure time and white balance. To enhance the generalizability of neural decoder-based methods, NPBG++ [32] extended neural

decoder-based methods from per-scene optimization to support general input. However, these approaches still struggle with photo-realistic rendering due to the aliasing caused by the neural networks [14].

Moreover, by applying **volume rendering techniques**, neural radiance field (NeRF) [23] achieves rendering photo-realistic images. Recently, PointNeRF [44] has combined point cloud with volume rendering, which extracts point features of any sampling point via KNN (K Nearest Neighbors) to enhance rendering quality. Since PointNeRF requires a per-scene optimization, TriVol [10] proposes a general solution, where they encode point clouds as triple slim volumes with sparse 3D UNet. Rendering point clouds by volume rendering facilitates the image's quality.

2.2. 3D Neural Representation

Representing point clouds and 3D shapes with neural primitives is significant for neural networks performing downstream tasks, e.g., rendering [35], reconstruction [45], and generation [9].

DVR [25] encodes 3D shapes as a **global latent code**, which is lightweight but limited for rendering high-quality images. In contrast, DeepVoxels [38] embeds shapes as 3D **voxel grids** of features. However, voxel-based neural representation suffers from computation costs. Thus, ConvOcc [28] devised the **tri-plane** representation, employing three orthogonal feature planes for decoding more details in high computation efficacy. Recently, inspired by radial basis function, 3DILG and 3DS2V [47, 48] propose another technique for representing point clouds as **irregular latent grid** and **latent set**. Furthermore, Michelangelo [49] proposes **learnable latent set** for representing point clouds via leveraging learnable queries. Although these methods preserve information for high-quality shape reconstruction, such rough modelings are still limited to capturing and recovering the fine details of surface textures.

2.3. Radiance Field

Recently, significant breakthroughs have been made in radiance field methods. These advancements can be broadly categorized into two types: point-based radiance fields and neural radiance fields.

Point-based radiance field. Seminal work extends point primitives to 3D Gaussian and introduces EWA (elliptical weighted average) filtering for anti-aliasing [50, 51]. With the development of differentiable rendering techniques, DSS [46] proposes a plug-and-play module in deep-learning architectures, and Pulsar [19] devised a sphere-based differentiable renderer for fast rasterization. Notably, a recent breakthrough in 3D Gaussian Splatting (3DGS) [16] introduces a tile-based differentiable rasterizer and dramatically enhances the rendering speed and quality. However, 3DGS requires a per-scene optimization, and leveraging its power to improve the rendering capabilities of forward models remains explored.

Neural radiance field. NV [21] and SRN [39] propose differentiable ray marching to introduce backward mapping rendering to deep learning frameworks. Although these methods have impressive results, rendering complex shapes and high-resolution images remains challenging. NeRF improves the rendering quality by importance sampling and positional encoding, and InstantNGP [24] speeds up by leveraging multi-resolution hash grid and tiny multi-layer perceptrons.

3. Approach

Effectively capturing global and local information of the input point cloud is crucial for recovering the radiance field for rendering. To address this, we propose the hybrid neural radial basis function (H-NRBF) and develop Point as Gaussian (PAG) based on it. Specifically, PAG consists of an H-NRBF encoder and a radiance field decoder. The H-NRBF encoder represents the global and local information of the point cloud, while the radiance field decoder reconstructs the radiance fields and renders images under given viewpoints. Given that the color and position of an input point cloud provide a natural initialization for 3D Gaussian Splat, and 3D Gaussian Splatting (3DGS) [16] offers an efficient differentiable rasterizer for 3D Gaussian-represented scenes, we adopt 3DGS as the radiance field in our framework.

3.1. Preliminary

Since PAG harnesses the power of 3DGS, we briefly explain the 3DGS in the following.

3DGS is defined by a 3D Gaussian set, and each 3D Gaussian contains a covariance matrix Σ centering at a point (mean) μ in the world space:

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{1}$$

Then, the covariance matrix is represented with a scaling matrix S and a rotation matrix R to ensure it is positive and semi-definite,

$$\Sigma = RSS^TR^T \tag{2}$$

The scaling and rotation matrix can trivially convert to a 3D vector $s \in \mathbb{R}^3$ and a quaternion vector $r \in \mathbb{R}^4$.

Following previous works [17], 3DGS leverages point-based α-blending for computing the color of each pixel C in 2D images. It blends N ordered points overlapping the pixel,

$$C = \sum_{n=1}^{N} c_n \alpha'_n \prod_{j=1}^{n-1}(1-\alpha'_j) \tag{3}$$

where c_n is the color of each point represented with spherical harmonic [33] and the blending weight α'_n is determined

by evaluating the 2D projection of a 3D Gaussian, which is then multiplied by the opacity α of each point.

To succinctly summarize, 3DGS parameterizes a point in point clouds via five learnable vectors: 1) a position μ, 2) a color c, 3) a scaling vector s, 4) a quaternion vector r, and 5) an opacity α.

3.2. Point As Gaussian

We devise H-NRBF to represent the input point cloud and predict the parameters of 3DGS. The following describes the H-NRBF, H-NRBF encoder, and radiance field decoder.

3.2.1 Hybrid Neural Radial Basis Function

H-NRBF adapts latent sets to be suitable for predicting the radiance field. While latent sets effectively represent and reconstruct surface points or signed distance functions, their performance in predicting radiance field parameters is often suboptimal due to the smoother nature of shape compared to texture. Therefore, texture functions necessitate more refined local modeling for accurate prediction. Subsequently, the H-NRBF integrates a proposed local neural radial basis Function (L-NRBF) with the global latent sets (GLS) for learning texture functions of the input point cloud.

The L-NRBF models the information of a local patch in the point clouds. Given a center point P_o with position x_o and color c_o from an arbitrary local patch of an input point cloud and the center point's spatially nearest K points are denoted as $\{x_k, c_k\}_{k=1}^{K}$, where x_k is the position, and c_k is the color. An L-NRBF on the local patch $\mathcal{F}_{L-NRBF}(P_o)$ is written as

$$\mathcal{F}_{L-NRBF}(P_o) = \max(\mathcal{F}_{NN}(x_c - x_k, c_c - c_k)_{k=1}^{K}) \quad (4)$$

where $\mathcal{F}_{NN}$ is a neural network. Since the dissimilarity is determined by the distance of textural and spatial space, L-NRBF is sensitive to the texture and geometry of the local patch on point clouds.

For the GLS, we follow learnable latent sets $\{z_i\}_{i=1}^{L}$ in Michelangelo [49] and keep the latent vector $z_i \in \mathbb{R}^d$ in a coordinate-free structure, where L is the number of latent vectors and d is the dimension of latent vectors. We denote GLS $\mathcal{F}_{GLS}$ with the cross-attention operation as:

$$\mathcal{F}_{GLS}(P_o) = \sum_{i=1}^{L} v(z_i) \frac{e^{q(P_o)^T k(z_i)/\sqrt{d}}}{Z(P_o, \{z_i\}_{i=1}^{L})} \quad (5)$$

where $Z(P_o, \{z_i\}_{i=1}^{L}) = \sum_{i=1}^{L} e^{q(P_o)^T k(z_i)/\sqrt{d}}$ is a normalizing factor. Therefore, our H-NRBF $\mathcal{F}_{H-NRBF}$ comprises the L-NRBF, and GLS is the following learnable function composition:

$$\mathcal{F}_{H-NRBF}(P_o) = (\mathcal{F}_{GLS} \circ \mathcal{F}_{L-NRBF})(P_o) \quad (6)$$

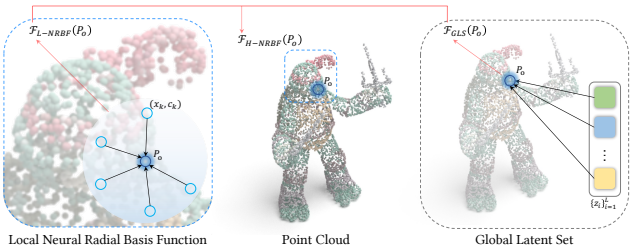

Figure 2. Illustration of hybrid neural radial basis function (H-NRBF). H-NRBF represents a center point P_o of input point cloud based on its nearest neighbors and global features (middle). We design local neural radial basis function (L-NRBF) for modeling its dissimilarity from nearest neighbors in local patch (left). Our model characterizes the global feature with global latent set (GLS) $\{Z_i\}_{i=1}^{L}$ based on learnable query tokens (right).

3.2.2 H-NRBF Encoder

Two components compose the H-NRBF encoder: a global encoder for representing point clouds to global latent sets and a patch aggregator for capturing local latent sets. Given that input point cloud $P \in \mathbb{R}^{M\times 6}$ contains M points, each point P_i has a coordinate and color $\{x_i \in \mathbb{R}^3, c_i \in \mathbb{R}^3\}$. We utilize a linear layer to project the Fourier positional encoded point clouds P to the input $X \in \mathbb{R}^{M\times d}$ of global encoder and a cross-attention layer to inject the position and color information of the point cloud into the learnable queries $Q \in \mathbb{R}^{L\times d}$, where L is the length of learnable queryies. Benefiting from the cross-attention operation, our model can directly extract information from the input point cloud without additional compression or transformation operations. To enhance the model's ability to encode the point cloud into global latent sets $Z \in \mathcal{R}^{L\times d}$, we incorporate a UNet-ViT following the cross-attention layer, consisting of self-attention layers and skip connections.

At the same time, we devise the patch aggregator to extract local latent sets. Specifically, we apply the K-nearest neighbor algorithm for each point P_i in the point cloud P and extract the local latent set following the L-NRBF in quation 4, where $\mathcal{F}_{NN}$ is a PointNet-like module [31].

3.2.3 Radiance Field Decoder

Taking the global and local latent sets, and given point cloud P as input, the decoder aims to predict a radiance filed for rendering high-quality images. Following equation 6, the points are converted to the 3D Gaussians with additional tiny projection layers. Since the point already provides an initialization for the position μ of 3D Gaussian, our model predicts an offset based on the coordinates x of points by the position head $\mathcal{F}_{\mu}$:

$$\mu = x + \mathcal{F}_{\mu}(\mathcal{F}_{H-NRBF}(P)), \quad (7)$$

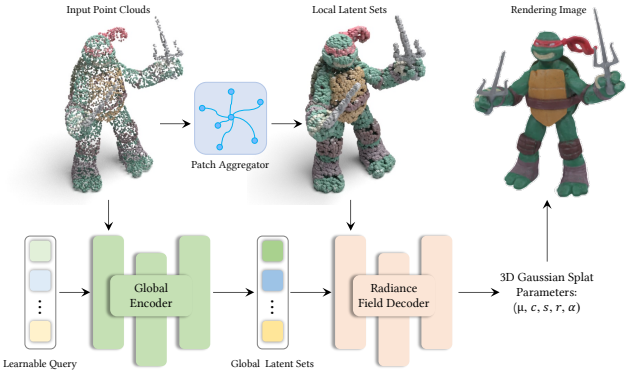

Figure 3. Pipeline of Point as Gaussian (PAG). PAG comprises a global encoder, a patch aggregator, and a radiance field decoder. Given a point cloud, the PAG model employs a learnable query to represent the positional embedded point cloud as global latent sets through the global encoder. Simultaneously, the patch aggregator extracts local latent sets based on the position and color of points from the input. Subsequently, the radiance field decoder predicts the parameters of the 3D Gaussian Splats based on both local and global latent sets. Ultimately, PAG renders images from point clouds with the predicted 3D Gaussian Splats under specified viewpoints.

The color of 3D Gaussian is predicted with the color head $\mathcal{F}_c$ and we use the sigmoid as activation function to make sure $c \in [0, 1)$:

$$c = \text{sigmoid}(\mathcal{F}_c(\mathcal{F}_{H-NRBF}(P))) \tag{8}$$

The decoder predicts the opacity α in a similar way with by utilizing the opacity head $\mathcal{F}_\alpha$:

$$\alpha = \text{sigmoid}(\mathcal{F}_\alpha(\mathcal{F}_{H-NRBF}(P))) \tag{9}$$

We follow a similar approach in 3DGS for predicting the rest scaling vector s and quaternion vector q:

$$s = \exp(\mathcal{F}_s(\mathcal{F}_{H-NRBF}(P))) \tag{10}$$

$$q = \text{norm}(\mathcal{F}_q(\mathcal{F}_{H-NRBF}(P))) \tag{11}$$

where $\mathcal{F}_s$ is the scaling head and $\mathcal{F}_q$ is the quaternion head.

3.3. Training Objectives

We optimize PAG via three objectives, rendering loss $\mathcal{L}_{rgb}$, LPIPS loss $\mathcal{L}_{LPIPS}$, and total variance regularizer on the scaling vector $\mathcal{L}_{scaling}$. Following 3DGS [16], the rendering loss is a weighted sum of an $\mathcal{L}_1$ loss and a D-SSIM term:

$$\mathcal{L}_{rgb} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM} \tag{12}$$

During the training process, we leverage a total variance regularizer for the scaling vector to prevent the 3D Gaussian from becoming dramatically elongated, which would result in a significant amount of noise in the subsequent point cloud-rendered images, reducing their quality. The regularizer for a scaling vector $s = [s_1, s_2, s_3]$ is:

$$\mathcal{L}_{scaling} = \text{mean}\left(|s_0 - s_1| + |s_1 - s_2| + |s_2 - s_0|\right) \tag{13}$$

To summarize, the overall objectives for supervising PAG is

$$\mathcal{L} = \lambda_{rgb}\mathcal{L}_{rgb} + \lambda_{LPIPS}\mathcal{L}_{LPIPS} + \lambda_{scaling}\mathcal{L}_{scaling} \tag{14}$$

4. Experiments

4.1. Implementations

PAG comprises a cross-attention encoder, a UNet-ViT, and a cross-attention decoder. Each transformer layer consists of 12 heads and 64 dimensions for each head, Layer Normalization, Feed-Forward Network with 3072 dimensions, and GELU activation. The learnable query embeddings contain 512 tokens in 768 dimensions. We use an AdamW-based optimizer with a 1e-4 learning rate. Moreover, the hyperparameter in Equation 12 is $\lambda = 0.2$, and in Equation 14, it is $\lambda_{rgb} = 1, \lambda_{LPIPS} = 2$, and $\lambda_{scaling} = 0.05$. The framework is implemented with PyTorch [27] and trained on 8 Tesla V100 GPUs.

4.2. Settings

Dataset. We validate PAG on ShapeNet-Car, ShapeNet-Ensembled, and Google Scanned Object. **ShapeNet-Ensembled** (ShapeNet-Core V2) [4] contains about 50,000 synthetic objects in 55 categories, and **ShapeNet-Car** is the 'Car' category within ShapeNet-Ensembled, including 3000 objects. **Google Scanned Object** [6] provides about 1,000 objects in 17 categories. Notably, all data in Google Scan Object are from the real world.

Metrics. Following TriVol and PBNG, we employ PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index Measure), LPIPS (Learned Perceptual Image Patch Similarity) and FID (Frechet Inception Distance). For PSNR and SSIM, a higher value indicates a better result, and the reverse is true for the other metrics.

Baselines. We compare PAG with five baselines. **Pytorch3D** [34], replacing points with circulars, and rendering point clouds by rasterization. **NPBG++** [32], a neural decoder-based method. **TriVol** [10] employs NeRF to render images of point clouds. Besides, **L-NRBF-MLP** and **Triplane-Transformer** are two natural baselines. L-NRBF-MLP utilizes a PointNet-like module [31] to extract features of a point from its adjacents and converts point clouds to 3D Gaussians by MLPs. Triplane-Transformer leverages a learnable triplane within transformer architecture for encoding the point cloud and predicts 3D Gaussians by querying points from the point cloud in the triplane. Both two methods render images via splatting.

	ShapeNet-Car				ShapeNet-Ensembled				Google Scanned Object			
	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Pytorch3D [34]	25.97	0.924	0.087	124.70	24.89	0.900	0.120	96.21	25.66	0.890	0.107	109.38
Triplane-Transformer	29.30	0.975	0.041	44.78	28.45	0.962	0.065	46.85	28.78	0.953	0.069	76.47
L-NRBF-MLP	28.59	0.971	0.041	40.97	28.11	0.960	0.076	53.51	27.53	0.952	0.063	62.20
NPBG++ [32]	30.69	0.974	0.055	91.50	28.83	0.962	0.082	67.14	31.87	0.965	0.065	80.16
TriVol [10]	32.11	0.980	0.043	48.07	30.43	0.968	0.077	58.43	33.90	0.973	0.055	55.22
Ours	33.36	0.987	0.023	22.24	33.38	0.982	0.043	28.26	34.72	0.984	0.031	26.60

Table 1. Quantitative evaluation on rendering images. We compare PAG with baselines on ShapeNet-Car (single-category synthetic dataset), ShapeNet-Ensembled (cross-category synthetic dataset), and Google Scanned Object (cross-category real scanned dataset). The numerical results indicate that our model outperforms the others. (↑ indicates that a higher value represents a better performance, and ↓ indicates that a lower value represents a better performance)

Figure 4. Qualitative comparison of rendering images. The results demonstrate that our model overwhelms the baselines regarding image quality. Upon zooming in, it becomes evident that the point cloud rendering produced by PAG has no holes and exhibits superior detail in patterns and characters. Moreover, the fine-grain texture comparison further indicates the effectiveness of our H-NRBF modeling.

GLS	L-NRBF	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
✔		26.11	0.924	0.088	97.03
	✔	27.53	0.952	0.063	62.20
✔	✔	**34.72**	**0.984**	**0.031**	**26.60**

Table 2. Quantitative ablation study on the hybrid neural radial basis function (H-NRBF). To validate the effectiveness of each module in H-NRBF, they are deactivated in turn, with a tick indicating the use of the corresponding module. The results reveal that using global latent sets (GLS) or local neural radial basis function (L-NRBF) only achieves lower performance than the last row, which uses all modules. This finding underscores the effectiveness of our proposed model and the significant contribution of each module to the overall performance.

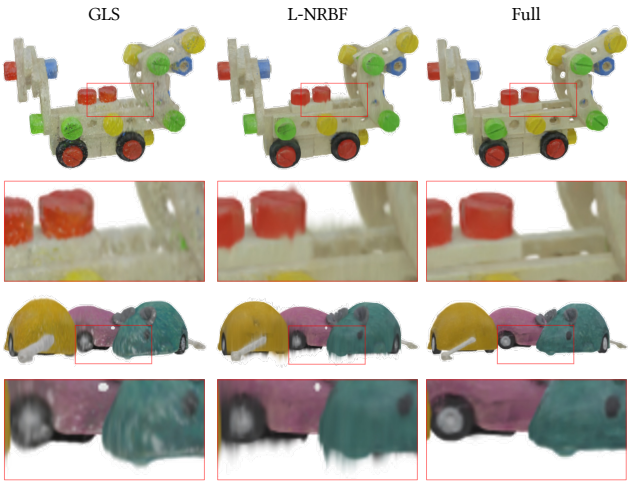

Figure 5. Qualitative ablation study on the hybrid neural radial basis function (H-NRBF). The first column shows the rendering from the model using global latent sets (GLS) only, which shows precise contours, but the texture is blurry. In contrast, the second column shows the results from the model using the local neural radial basis function (L-NRBF) only, which has better texture details, but the contour is blurry. The last row shows that using H-NRBF leads to precise contour and high-quality texture.

4.3. Experimental Comparisons

We evaluate PAG against baseline methods on ShapeNet-Car, ShapeNet-Ensembled, and Google Scanned Object datasets. Generally, we adhere to conventional settings, sampling 100k points from each object as input. The quantitative results are presented in the Table 1. These results demonstrate that our model surpasses the baselines in rendering point clouds from single-category synthetic, cross-category synthetic, or cross-category real-world scanned datasets. A comparison of the qualitative results in the Figure 4 reveals that the Triplane-Transformer can predict a reasonably accurate radiance field overall. Conversely, the L-NRBF-MLP can predict results with better details, albeit with less sensitivity to global contours. The NPBG++ method, which employs a neural decoder, has almost resolved the rendering holes in the results, but noticeable artifacts persist. Benefiting from the local modeling of triple volume and the rendering approach of NeRF, TriVol performs well in most cases. However, blurring occurs when rendering textures such as complex objects or scanned point clouds. PAG, leveraging the hybrid neural radial basis function (H-NRBF) to represent point clouds, allows for rendering images with precise contours and realistic textures.

4.4. Ablations

To assess the effectiveness of the proposed module in PAG, we conducte ablation experiments on the H-NRBF. Additionally, we examine the impact of the distribution difference between training and testing data on the model's performance.

Effectiveness of the hybrid neural radial basis function (H-NRBF). Experiments were conducted on the Google Scanned Objects dataset, with the results reported in the Table 2. The first row depicts the model's performance when only utilizing GLS (global encoder and naive cross-attention decoder). In contrast, the second row indicates the performance when only employing L-NRBF (L-NRBF with a tiny multi-layer perceptrons decoder). These performances exhibit a significant loss compared to the third row, which applies H-NRBF in the PAG, thereby demonstrating the effectiveness of H-NRBF. The Figure 5 presents a visualization of the ablation study. The first column displays results using only global latent sets, the second column presents results with only L-NRBF (Local Neural Radial Basis Function), and the final column illustrates results from the complete PAG method. Comparing the first and second columns, it becomes apparent that when only GLS is used, the model has a better global perception, and the generated 3D Gaussian Splat can render images with precise contours. However, it lacks local information modeling, resulting in defective texture details. The second column exhibits better details thanks to L-NRBF's local information modeling, but the absence of global information results in blurred boundaries of the rendered objects. In contrast to the first two columns, the complete PAG method benefits from the fusion of global and local information, maintaining precise contours of the rendered target while enhancing the texture details.

Impact of data distribution. To better validate the capability of PAG, we conducted ablation experiments on the training data. Specifically, we trained the model using ShapeNet-Car, ShapeNet-Ensembled, and Google Scanned Objects datasets and tested it on ShapeNet-Car. The re-

	ShapeNet-Car			
	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
ShapeNet-Car	**33.36**	**0.987**	**0.023**	**22.23**
ShapeNet-Ensembled	32.71	0.984	0.028	43.75
Google Scanned Object	30.87	0.977	0.043	74.83

Table 3. Ablation study on the dataset. We train the model on three datasets and evaluate its performance on the ShapeNet-Car dataset. The results indicate that the performance of the PAG model is comparable across these different training datasets, demonstrating the stability of our model.

sults are shown in the table 3, where the first row represents training and testing using ShapeNet-Car. The second and third rows represent training on ShapeNet-Ensembled and Google Scanned Objects and testing on ShapeNet-Car. The first row in the table indicates that the model achieves the best performance when the training and testing data distributions are the same. The performance decay from the first to the third row demonstrates that a gap between the training and testing data distributions can hurt the model's performance, but not significantly.

4.5. Applications

Since PAG can render photo-realistic images from input point clouds, our method can be combined with recent image-based reconstruction algorithms [42, 45] for mesh reconstruction, circumventing challenging steps in traditional methods, such as estimating normal vectors. The first two rows of the Figure 6 demonstrate that the Poisson reconstruction results in meshes exhibiting artifacts, including white spots in the texture, holes, or protruding bumps. The third row demonstrates that the meshes reconstructed using our approach exhibit relatively better surface details when the two methods are comparable.

5. Conclusion and Discussion

In conclusion, this paper addresses the challenge of point cloud rendering by devising a novel method called Point as Gaussian (PAG). We propose the hybrid neural radiance basis functions (H-NRBF) approach to model both local and global information of point clouds. Our method successfully translates a point cloud into 3D Gaussian Splats without optimizing or fine-tuning. Extensive results demonstrate the strong performance of the PAG model in various scenarios. Furthermore, the PAG model offers an alternative solution for mesh reconstruction from point clouds.

PAG successfully performs forward translation from point clouds to 3D Gaussian Splats, laying the groundwork for a new approach to 3D generation. By using point primitives to express the generated model and then transforming the generated point cloud into 3D Gaussian Splats through PAG, a new method is established. Additionally, the generation of point clouds is more lightweight than the generation of mesh or signed distance functions, putting less demand on computational resources. Consequently, this opens up a promising avenue for future research, with substantial scope to delve deeper into developing efficient, resource-friendly 3D generation methods. This line of investigation holds significant potential for advancing the field of computer graphics and beyond.

Figure 6. Visualizations of mesh reconstruction. Benefiting from the high-quality renderings, we utilize image-based mesh reconstruction algorithms to obtain mesh from input point clouds. Since our method bypasses some challenges, e.g., normal estimation, in some cases, our model produces mesh with better geometry and colors than Poisson reconstruction.

. Appendix

In this appendix, we present the model parameters A, additional analysis B, and experiments C to clarify further the performance of the methods examined in the main paper. By comparing each method's characteristics, we aim to provide a more comprehensive understanding of their strengths. Additionally, we explore the effect of the number of points utilized in training and testing, illuminating the connection between point density and the overall performance of the models.

A. Implementation Details

This section describes the model parameters for the Point As Gaussian (PAG) method. The PAG model consists of a cross-attention encoder with a learnable query token of length 512 and dimension 768, a five-layer UNet-ViT, and a cross-attention decoder. Each transformer layer within the model comprises 12 heads and 64 dimensions for each head, Layer Normalization, a Feed-Forward Network with 3072 dimensions, and a GELU activation function. The positional embedding frequency for coordinates is set to 12, while for colors, it is set to 11. Additionally, for the local neural radial basis function (L-NRBF), the number of nearest neighbors is set to 24.

B. Attribute analysis

This section outlines the baselines used for comparison with our proposed method in the main text, the attribute is illustrated in Table 4.

The point renderer in **PyTorch3D** replaces each point with a disc rendered with varying weight values, effectively handling cases with smooth shape and texture, allowing for uniform disc fill between points. However, it encounters difficulties with more complex objects.

The **Triplane-Transformer**, which employs a learnable triplane and encodes point clouds with cross-attention, offers high-quality global information, as evident in the visual results. Nonetheless, querying features with point coordinates in the triplane may not capture detailed texture information, leading to blurry renderings.

Conversely, **L-NRBF-MLP** is a natural comparison for Point As Gaussian (PAG), finding the nearest neighbors for each point in the point cloud and composing the most dissimilar one with it as local patch embeddings. The model utilizes a two-layer MLP for predicting the radiance field. L-NRBF-MLP better characterizes local information but lacks global information, resulting in images with finer details but poor contours.

NPBG++ employs a 2D UNet as a neural renderer for rendering final images, projecting point clouds into feature maps on a 2D plane under a given viewpoint. It focuses on modeling local patch information but struggles with rendering complex shapes or textures due to the scarcity of modeling 3D information.

TriVol, a state-of-the-art method, models local information by quantizing the point cloud into a voxel grid and then employing a 3D UNet to extract information. It predicts a neural radiance field for rendering final images. Unlike NPBG++, the 3D UNet in TriVol models 3D information of point clouds and volume rendering further utilizes this information to predict the final color of pixels. However, as point clouds are discrete, recovering the neural radiance field from point primitives is challenging.

In such cases, 3D Gaussian Splat naturally outperforms the neural radiance field as it can better utilize point primitives. Furthermore, our proposed method, Points as Gaussians (PAG), combines the advantages of the methods above. It employs global latent sets to encode global information of point clouds and a local neural radial basis function to characterize local information, resulting in superior performance.

C. Point Ablation

We investigate the impact of varying the number of point clouds used for training and testing, specifically 100k, 50k, 8192, 4096, and 2048, on the model's performance. The results are illustrated in Figure 7. Considering multiple metrics, it is evident that the model achieves optimal performance when training and testing utilize a more significant number of points (100k or 50k). Moreover, the model can generate reasonable results when training and testing employ a similar number of point clouds. However, when trained with more points but tested with fewer points, the model underperforms, which might be attributed to the inability of the L-NRBF obtained from such training to extract features from two distant points in space, ultimately failing to produce satisfactory results.

	Primitive	Encoding	Neural Representation	Decoding	Render Technique
Pytorch3D [34]	Points	Identity	-	Identity	Rasterization
Triplane-Transformer	Points	Cross Attention	Triplane	Grid Sample	Splatting
L-NRBF-MLP	Points	MLP	-	KNN	Splatting
NPBG++ [32]	Points	2D UNet	Multi-scale Feature Maps	Identity	Neural Decoding
TriVol [10]	Voxels	3D UNet	Triple Volumes	Trilinear Interpolation	Volume Rendering
Ours	Points	Cross Attention	Hybrid RBF	Cross Attention	Splatting

Table 4. We present the attributes of the baselines used for comparison with our proposed method. PyTorch3D, replacing points with discs for rendering, effectively handles smooth shapes and textures but struggles with complex objects. Triplane-Transformer, employing a learnable triplane and cross-attention, offers high-quality global information but lacks detailed texture information. L-NRBF-MLP, a natural comparison for Point As Gaussian (PAG), finds nearest neighbors for each point and composes local patch embeddings but has poor contours due to its lack of global information. NPBG++ uses a 2D UNet as a neural renderer, focusing on local patch information but struggling with rendering complex shapes or textures due to its 2D feature mapping approach. TriVol, a state-of-the-art method, models local information by quantizing point clouds into a voxel grid and employing a 3D UNet but faces challenges in recovering the neural radiance field from point primitives due to the discrete nature of point clouds. Our proposed method, Points as Gaussians (PAG), combines the advantages of these methods by employing global latent sets, a local neural radial basis function, and the 3D Gaussian Splat, resulting in superior performance.

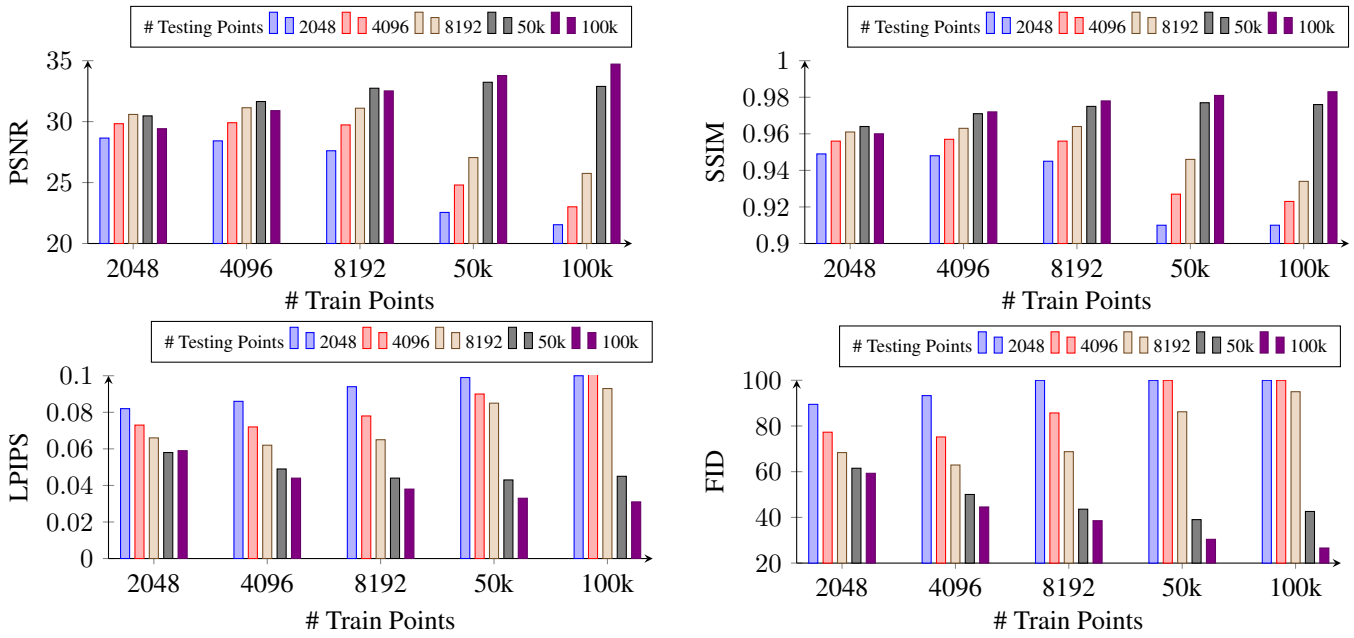

Figure 7. **Ablation study on the number of points.** We study and report the impact of varying the number of point clouds used for training and testing (100k, 50k, 8192, 4096, and 2048) on the model's performance. The results show that optimal performance is achieved with more significant points (100k or 50k) and similar training and testing point counts. However, when trained with more points and tested with fewer points, the model underperforms due to the L-NRBF's inability to extract features from distant points in space.

References

[1] Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)*, 40(3):1–21, 2021. 2

[2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 696–712. Springer, 2020. 2

[3] Giang Bui, Truc Le, Brittany Morago, and Ye Duan. Point-based rendering enhancement via deep learning. *The Visual Computer*, 34:829–841, 2018. 2

[4] 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, 5

[5] Qimin Chen, Zhiqin Chen, Hang Zhou, and Hao Zhang. Shaddr: Interactive example-based geometry and texture generation via 3d shape detailization and differentiable rendering. In *SIGGRAPH Asia 2023 Conference Papers*, 2023. 2

[6] 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 *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. 2, 5

[7] 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. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022. 2

[8] Jeffrey P Grossman and William J Dally. Point sample rendering. In *Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998 9*, pages 181–192. Springer, 1998. 2

[9] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing implicit shapes through part aware generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–20, 2022. 3

[10] Tao Hu, Xiaogang Xu, Ruihang Chu, and Jiaya Jia. Trivol: Point cloud rendering via triple volumes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20732–20741, 2023. 2, 3, 5, 6

[11] Tao Hu, Xiaogang Xu, Shu Liu, and Jiaya Jia. Point2pix: Photo-realistic point cloud rendering via neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8349–8358, 2023. 2

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2

[13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 2

[14] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021. 3

[15] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, page 0, 2006. 2

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 5

[17] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, pages 29–43. Wiley Online Library, 2021. 2, 3

[18] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022. 2

[19] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021. 3

[20] Marc Levoy and Turner Whitted. The use of points as a display primitive. 1985. 2

[21] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 3

[22] Gal Metzer, Rana Hanocka, Raja Giryes, Niloy J Mitra, and Daniel Cohen-Or. Z2p: Instant visualization of point clouds. In *Computer Graphics Forum*, pages 461–471. Wiley Online Library, 2022. 2

[23] 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

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 3

[25] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas 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. 3

[26] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019. 2

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[28] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas 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. 2, 3

[29] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000. 2

[30] Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T Barron, Amit H Bermano, Eric Ryan Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, et al. State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*, 2023. 2

[31] 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. 4, 5

[32] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15969–15979, 2022. 2, 5, 6

[33] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, 2001. 3

[34] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 5, 6, 2

[35] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 3

[36] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14, 2022. 2

[37] Dominik Sibbing, Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Sift-realistic rendering. In *2013 International Conference on 3D Vision-3DV 2013*, pages 56–63. IEEE, 2013. 2

[38] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 3

[39] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[40] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, pages 701–727. Wiley Online Library, 2020. 2

[41] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, pages 703–735. Wiley Online Library, 2022. 2

[42] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3295–3306, 2023. 8

[43] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 2

[44] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2, 3

[45] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, New York, NY, USA, 2023. Association for Computing Machinery. 3, 8

[46] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 3

[47] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. *Advances in Neural Information Processing Systems*, 35:21871–21885, 2022. 3

[48] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4), 2023. 2, 3

[49] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, BIN FU, Tao Chen, YU Gang, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2, 3, 4

[50] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3

[51] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 3