SPNR: Generalizable Sparse-Point Neural Rendering

Xuyi Meng[1,2] Jialin Zhang[1,3] Fanbo Xiang[1] Jiayuan Gu[1] Xiaoshuai Zhang [1]
Hao Su[1†]
[1]UC San Diego [2]Nanyang Technological University [3]Tsinghua University

Abstract

Recent advancements on neural volume rendering methods like NeRF [32] have demonstrated impressive ability to produce photo-realistic rendering for complex scenes. In this work, we present SPNR, which encodes sparse point clouds into neural volume representations to produce images with high visual quality. Given sparsely sampled surface points, SPNR generates disentangled density and color volumes and utilizes volumetric rendering to produce view-consistent high-quality images. The training is supervised with adversarial training objectives and the learned model can generalize to point clouds of unseen shapes. Experimentally, we demonstrate that SPNR outperforms previous point-cloud-based neural rendering methods in terms of rendering quality on two datasets (ABO and ShapeNet), and it generalizes to point clouds of objects unseen during training.

Figure 1. SPNR renders sparse input point clouds with material features to high-quality images with detailed geometry and high-frequency textures through neural radiance fields.

1. Introduction

Recent advancements on neural volume rendering methods like NeRF [32] have demonstrated impressive ability to produce photo-realistic rendering for complex scenes. In this work, we present SPNR, which encodes sparse point clouds into neural volume representations to produce images with high visual quality. Given sparsely sampled surface points, SPNR generates disentangled density and color volumes and utilizes volumetric rendering to produce view-consistent high-quality images. The training is supervised with adversarial training objectives and the learned model can generalize to point clouds of unseen shapes. Experimentally, we demonstrate that SPNR outperforms previous point-cloud-based neural rendering methods in terms of rendering quality on two datasets (ABO and ShapeNet), and it generalizes to point clouds of objects unseen during training.

3D point clouds are adopted in a wide range of applications. Most 3D capture devices, such as LiDARs and depth cameras, capture objects as point clouds. Lagrangian physical simulation methods also use point clouds to represent the physical states, and they have been used to simulate many types of materials, including cloth [29], fluid [2], and granular material [46]. For both capturing and simulation, point clouds need to be visualized in their downstream applications. However, rendering point clouds to high-quality images is non-trivial, *especially when the points are sparse*, which is commonly encountered in real applications. For example, 3D sensors such as iPad LiDAR scanner and Intel RealSense cannot capture accurate depths for glossy or translucent surfaces, resulting in missing points in the captured results; many physical simulators run on a tight computation budget, and choose to trade point cloud fidelity for speed. Therefore, a method to render high-quality images from sparse point clouds is a crucial component in 3D capturing and simulation, and it will further enhance downstream applications, including autonomous driving, virtual reality, robotics, and gaming industries.

Existing approaches of point cloud rendering typically fall into 3 categories: splatting, meshing, and volumetric rendering. Point splatting methods render 3D points as elliptical [49, 61] patches or ellipsoids [55]. To mitigate holes and elliptical artifacts resulted from the sparsity of points,

[40] proposes to perform point cloud super-resolution before splatting. Meshing-based methods, including marching cubes, Poisson surface reconstruction, and recent learning-based methods [47, 50] seek to convert point clouds to meshes, which is the most commonly used representation in renderers. Inspired by recent advancements in neural volumetric rendering pioneered by NeRF [32], [60] proposes to convert point clouds to neural volumetric representation and use ray marching to render images. However, none of these works is capable of rendering sparse point clouds with both detailed geometries and textures.

In this work, SPNR, we aim to address the challenge of producing images with high visual quality from **sparse input point clouds in a generalizable way**. First, we take inspiration from Neural Volumes [28] and MVSNeRF [8], which use 3D CNNs to process volumetric information, store it in 3D feature volumes, and render photo-realistic images. Note that MVSNeRF is a generalizable neural radiance estimation framework, and we would like to build our framework by adapting the MVSNeRF architecture and inherit its generalizable learning ability. Since point clouds also naturally reside in 3D space, we similarly encode the input point cloud into 3D feature volumes, process with 3D CNNs, and render with ray marching. Next, we discover that separately encoding geometry and appearance information from the point cloud into two feature volumes leads to much better rendering quality and generalizability. Finally, we identify that, similar to image super-resolution, rendering sparse point clouds to high-quality images is intrinsically ambiguous, and requires hallucinating additional geometry and appearance details beyond the input point positions and colors. To this end, we introduce an adversarial training objective, to encourage our neural renderer to synthesize perceptually reasonable details. To ensure the rendered results are properly conditioned on the input, we devise a method to convert sparse point clouds to 2D images, which are conditional inputs to the image-based discriminator.

To summarize, our main contributions are as follows:

- We propose a novel point-cloud-based neural rendering framework, which consumes sparse (1024) point clouds and can generalize to unseen objects;
- We show that disentanglement in encoding geometry and appearance information can significantly improve visual quality and generalizability;
- We introduce a conditional adversarial training objective to encourage synthesizing perceptually reasonable and high-frequency details despite sparse inputs.

2. Related Work

Neural Rendering. Recent advancements in neural scene representations have enabled high-quality image synthesis through differentiable rendering. Implicit surface-based methods represent 3D geometry as signed distance fields (SDFs) and render with ray marching [35], sphere tracing [27, 43], or by converting SDFs to meshes [24, 42]. Mesh-based methods render meshes through differentiable rasterizers [20] or path tracers [14, 23]. Point-based methods represent scenes as 3D point clouds and render with splatting methods [1, 19, 21, 53]. Very recently, volumetric representations, such as neural radiance field (NeRF) [32] and Neural Volumes (NV) [28], represent scenes with volumetric density and appearance, and can be rendered via differentiable ray marching. NeRF and its many extensions [3, 8, 15, 30, 33, 36, 37, 39, 44, 48, 51, 52, 54, 56, 58] have demonstrated that neural volumetric representations can achieve high-quality novel-view synthesis, relighting, appearance editing, and also efficient differentiable rendering from input RGB images. While most of these methods follow the original NeRF set-up to encode all scene information in a single MLP, [8, 26, 58] divide scenes into discrete voxels with local features; such feature-conditioned neural representations provide the opportunity to learn generalizable rendering networks across scenes. Our work (SPNR) follows a similar methodology. We encode input point clouds into 3D feature volumes for density and color, and utilize these features for subsequent volume rendering. Such architecture allows our framework to render point clouds of objects unseen during training.

3D-aware Generative Image Synthesis. Generative adversarial networks [12] are now capable of synthesizing photo-realistic 2D images [4, 17, 18]. More recently, 3D-aware image synthesis has gained popularity thanks to the advancements in 3D representations and differentiable rendering, since it requires generating multi-view consistent images. The previous methods can mainly be divided into two kinds. Mesh-based methods [11, 25, 45] directly synthesize textured 3D meshes. Voxel-based generative models [10, 13, 59] synthesize objects represented by 3D voxel grids with 3D CNNs. However, due to high memory cost, the resolution for the voxel grids is typically limited. To address this issue, recent works [6, 34, 41] introduced a NeRF-like volumetric rendering module to improve the quality of image synthesized from a low-resolution 3D feature volume. Our method (SPNR) adopts a similar design in 3D feature volumes, but a key difference is that our work synthesizes images conditioned on input sparse point clouds, and we focus on designing better feature conditions from such point cloud inputs.

3. SPNR

3.1. Overview

In this work, we present a point-cloud-based neural renderer, SPNR, which consumes a sparse point cloud along with material information (e.g., color, roughness, metallic values) and renders a 2D image conditioned on the input point cloud. Formally, given a sparse point cloud $P = \{p_i | i = 1 \dots N\} \in \mathbb{R}^{N\times 3}$ and its point-wise material information $M = \{m_i | i = 1 \dots N\} \in \mathbb{R}^{N\times C_M}$, the neural renderer R needs to render a 2D image $I = R(P, M, \Phi)$, where Φ is the camera parameter.

Our neural renderer is based on neural radiance fields and volumetric rendering due to their superior performance on novel view synthesis. However, different from prior works on neural radiance fields (e.g., NeRF [32]) that require per-scene optimization, we aim at a *generalizable* neural renderer that encodes the input, which is a sparse point cloud in our case, to neural radiance fields, even if the input is unseen during training. Such generalizability leads to a plug-and-play neural renderer, while demands learning generalizable representations instead of memorizing a specific scene by neural networks.

Besides, different from classical rendering, where inputs are either densely sampled points or continuous representations (e.g., mesh), our neural renderer takes sparse point clouds as inputs, which only contain low-frequency information, and needs to synthesize high-frequency details. Fig 3 illustrates the ambiguity due to the sparsity. Two visually different objects may correspond to similar sparse point cloud representations. Thus, previous training objectives (e.g., L1/L2 loss), which only match each rendered image with the corresponding ground truth, can be sub-optimal and hurt generalizability.

Fig 2 illustrates the overall pipeline of our SPNR. First, we encode the input point cloud to two disentangled neural feature volumes for geometry and appearance, followed by two separate MLPs to decode neural radiance fields. Then, differentiable ray-marching is applied to neural radiance fields to render the 2D image. During training, we introduce a conditional adversarial objective, to encourage the model to synthesize high-frequency details conditioned on sparse inputs. We will elaborate on the pipeline in the following sections.

3.2. Points to Neural Feature Volumes

To achieve strong generalizability, we use neural feature volumes as intermediate representations to encode neural radiance fields, inspired by MVSNeRF [8]. Concretely, we first voxelize the input sparse point cloud and construct an input feature volume $V \in \mathbb{R}^{D\times D\times D\times C}$, where D is the spatial resolution and C is the feature dimension. For each voxel, we use a PointNet [38] to encode all the points within this voxel and their material properties:

$$V(i) = \text{PointNet}(\{\hat{p}_j, m_j | \text{voxelize}(p_j) = i\}) \quad (1)$$

where i is the voxel index, and $\hat{p}_j$ is a local coordinate of p_j in the voxel. Note that representing coordinates in local frames is critical to learn generalizable representations.

The information is scattered over sparse locations in the input feature volume. Thus, we use a 3D convolutional U-Net, which contains several downsampling and upsamping convolutions as well as skip connections, to generate the dense neural feature volume: $S = \text{U-Net}(V)$. The encoder-decoder structure of U-Net can effectively propagate sparse information contained in the input feature volume. And it can also encode both local and global geometry and appearance information in a hierarchical way, which helps synthesize details.

Moreover, we observe that it is critical to separately encode geometry and appearance, to improve generalizability. Otherwise, neural networks might memorize the correlation between geometry and appearance, instead of conditioning the output on the input, especially its material information. Thus, we propose to use two branches to separately encode geometry and appearance. Concretely, we first use a PointNet to encode input positions to a geometry feature volume $V_{geo}(i)$.

$$V_{geo}(i) = \text{PointNet}_{geo}(\{\hat{p}_j | \text{voxelize}(p_j) = i\}) \quad (2)$$

We use another PointNet to encode both positions and material properties to an appearance feature volume $V_{app}(i)$.

$$V_{app}(i) = \text{PointNet}_{app}(\{\hat{p}_j, m_j | \text{voxelize}(p_j) = i\}) \quad (3)$$

Then, two 3D U-Nets are employed to generate two neural feature volumes S_{geo} and S_{app} seperately:

$$S_{geo} = \text{U-Net}_{geo}(V_{geo});\ S_{app} = \text{U-Net}_{app}(V_{app}) \quad (4)$$

In Sec 5.4, we show that our design to disentangle geometry and appearance enables our neural renderer to generate images conditioning on inputs better. Such disentanglement in encoding geometry and appearance is also shown to be helpful in prior works on 3D-aware generative models like GET3D [11].

3.3. Neural Feature Volume Rendering

The neural feature volume rendering stage of our pipeline refers to the process of rendering the RGB image I_{RGB} from the geometry feature volume S_{geo}, appearance feature volume S_{app}, and camera parameter Φ. First, we apply the *volumetric rendering* technique to render a low-resolution RGB image I^-_{RGB} along with a feature map I^-_F. Then, following EG3D [6], we employ a *super-resolution* module to upsample and refine I^-_{RGB} with the help of I^-_F, to obtain a high-resolution RGB image I_{RGB}. The super-resolution module is introduced to improve the overall efficiency during both training and inference.

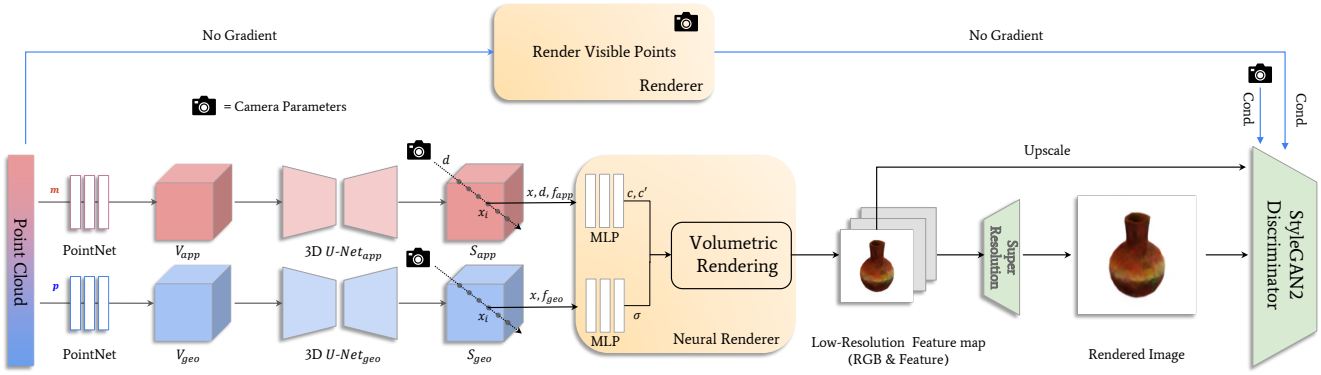

Figure 2. Overall pipeline. We present a point cloud rendering pipeline that first encodes the input point cloud to disentangled geometry and appearance feature volumes, and then renders RGB images with volumetric rendering and super-resolution. Given input points $\{p_i\}$ and associated point material features $\{m_i\}$, they are separately encoded to feature volumes V_{geo} and V_{app} through per-cell PointNets. V_{geo} and V_{app} are then processed with 3D U-Nets to neural feature volumes S_{geo} and S_{app}. Next, we apply differentiable ray-marching to generate 2D low-resolution RGB images and feature maps. For each shading point on a camera ray, an appearance MLP takes the spatial position x, viewing direction d and trilinearly interpolated features of S_{app} to produce radiance c and appearance feature c'. An geometry MLP takes the position and trilinearly interpolated features of S_{geo} to generate a volumetric density. c and c' are then aggregated by density along the ray to generate per-pixel colors and features, which are then passed through a super-resolution CNN to generate the final image. Our pipeline is trained with a StyleGAN2 discriminator that additionally conditions on the input point cloud by taking the 2D projection of the visible part of the point cloud.

Figure 3. The ambiguity caused by sparsity. The original meshes are shown in the top row and the sampled point cloud are correspondingly shown in the bottom. Meshes of different shapes may be sampled into similar point clouds. And for those similarly shaped meshes, there may be loss of texture details when sampled.

Volumetric Rendering Given the geometry feature volume S_{geo} and appearance feature volume S_{app}, for a spatial point x and viewing direction d, we regress the volumetric density σ, radiance c and an additional appearance-related feature vector c' with 2 MLPs, namely MLP$_{geo}$ and MLP$_{app}$:

$$\sigma = \text{MLP}_{geo}(x, f_{geo}), \quad f_{geo} = S_{geo}(x) \tag{5}$$

$$c, c' = \text{MLP}_{app}(x, d, f_{app}), \quad f_{app} = S_{app}(x) \tag{6}$$

where $f_{geo} = S_{geo}(x)$, and $f_{app} = S_{app}(x)$ are the neural features trilinearly interpolated at position x for the feature volumes S_{geo} and S_{app} respectively.

Now σ, c form a density field and a view-dependent radiance field, which allow us to use the classical ray-marching algorithm [31] to compute the pixel color by accumulating radiance values along sampled shading points long the camera ray.

$$c_t = \sum_i T_i (1 - \exp(-\sigma_i \delta_i)) c_i \tag{7}$$

$$T_i = \exp - \sum_{j=1}^{i-1} \sigma_j \delta_j \tag{8}$$

This process is the same as in NeRF [32], where c_t is the target output pixel color. T_i is known as volumetric transmittance at point i along the camera ray. δ_i represents the distance between consecutive shading points. c_i and σ_i are the radiance and density values at point i from evaluating G_{geo} and G_{app} with point position and ray direction. Our pipeline additionally accumulates appearance feature c' along the ray, which provides additional information for the super-resolution step.

$$c'_t = \sum_i T_i (1 - \exp(-\sigma_i \delta_i)) c'_i \tag{9}$$

Super Resolution We follow EG3D [6] to use a 2D CNN to upsample and refine the low-resolution RGB image generated by the differentiable ray marching process. We also adopt dual discrimination, where the concatena-

tion of the bilinearly upsampled low-resolution RGB image Upsample(I^-_{RGB}) and the super-resolved image I^{RGB} is fed into a discriminator for adversarial training. We refer readers to [6] for more details. In short, dual discrimination encourages the consistency between the neural rendering and the super-resolved image, and avoids view-inconsistency artifacts. We extend dual discrimination to condition on the input sparse point cloud, which is described in the next section.

3.4. Conditional Adversarial Training

As discussed in image super-resolution [22], minimizing the pixel-wise reconstruction error can result in lack of high-frequency details and poor perceptual quality despite high signal-to-noise ratios. We have observed the similar phenomenon as our input point cloud is sparse and stands for low-frequency signals sampled from the original. Thus, in addition to the reconstruction loss (e.g., L1 loss), we introduce an adversarial objective to encourage SPNR to synthesize perceptually reasonable details given sparse inputs. Let D denote the discriminator. Our adversarial objective is as follows:

$$L_{adv} = d(D(R(P, M, \Phi)|P, M)) + d(-D(I_{gt}|P, M)) \quad (10)$$

where $d(x) = -\log(1 + \exp(-x))$. Note that a key difference from prior works is that our discriminator needs to be conditioned on the input point cloud and its material information (P, M). Otherwise, the generator (renderer) can ignore input information like colors if the adversarial objective is unconditioned. Our generator is already conditioned on the input point cloud by design.

However, it is unclear how to effectively design an image-based discriminator conditioned on point clouds. In this work, we propose to project the sparse point cloud to a *condition image* I_{cond}. Concretely, we transform the 3D sparse point cloud to the camera frame according to camera parameters, filter out invisible points according to ground-truth depth, and project visible points to the 2D image.

Thus, we manage to extend dual discrimination [6] to be conditioned on our sparse inputs. The concatenation of I_{cond}, Upsample(I^-_{RGB}) and I_{RGB} is fed to the discriminator. The final adversarial objective is as follows:

$$\begin{aligned} L_{adv} =& d(D(R(P, M, \Phi), \text{Upsample}(I^-_{RGB})|I_{cond})) \\ &+ d(-D(I_{gt}, \text{Upsample}(I^-_{gt})|I_{cond})) \end{aligned} \quad (11)$$

4. Implementation Details

Network Architectures The input point cloud is normalized to a unit cube before voxelization and feature extraction. The resolution of input feature volumes is 64^3. The PointNet [38], which is used to extract voxel features from points, consists of a 3-layer MLP with $(64, 128, 256)$ hidden units and a max-pooling layer to aggregate point-wise features. The dimensions of input feature volumes are both 32. The architecture of 3D U-Nets, which are used to output geometry and color feature volumes, is the same as that in MVSNeRF [8], with 5 downsampling and upsampling stages. The dimensions of output neural feature volumes are both 16. The neural radiance field decoders MLP_{geo} and MLP_{app} both consist of 7 layers with (32,32,32,32,32,32,64) hidden units, and output 1-dim density and 32-dim radiance feature.

Volumeric Rendering Our volumetric rendering follows the two-pass importance sampling strategy as in NeRF [32], resulting in 64 equally spaced shading points per ray for the first pass and 64 importance-sampled points per ray for the second pass. Different from NeRF, which only generates a 3-channel radiance value, our renderer produces 29 features in addition to 3-channel radiance to serve as auxiliary inputs to the super-resolution module. Unless otherwise stated, the output resolution is 64×64.

Super Resolution We adopt the same architecture for our super-resolution module as EG3D [6], except that style modulation is not included. Our super-resolution module is trained to upsample the low-resolution rendered result to the target resolution, which is 128x128 for ABO [9] and 200x200 [1] for ShapeNet [7].

Discriminator We modified the discriminator of StyleGAN2 [18] for conditional adversarial training described in Sec 3.4. There are three conditions in our task. First, to prevent degenerate shape solutions as mentioned in [6], we condition the discriminator on camera parameters. This condition injection follows the implementation in [16]. Secondly, to ensure RGB images generated by the super-resolution module do not inpaint view-inconsistent contents in 2D space, we condition the discriminator on the low-resolution image output via volumetric rendering. Thirdly, since the rendering process is conditional in nature, the discriminator should also be conditioned on the input point cloud, which is represented by a condition image illustrated in Sec 3.4.

Training Details To supervise SPNR, we use the L1 loss L_1, the proposed conditional adversarial objective L_{adv}, and R1 regularization $\|\nabla D(I)\|_2^2$. The total loss is $L = 100L_1 + L_{adv} + 0.1\|\nabla D(I)\|_2^2$. The optimizer is Adam, and the learning rates are 0.002 for discriminator and 0.0025 for generator (renderer). We use a batch size of 8 for all experiments. The model is trained on 4 RTX 2080-Ti GPUs.

[1] We first bilinearly upscale the 64×64 rendered result to 100×100, and then apply our super-resolution module.

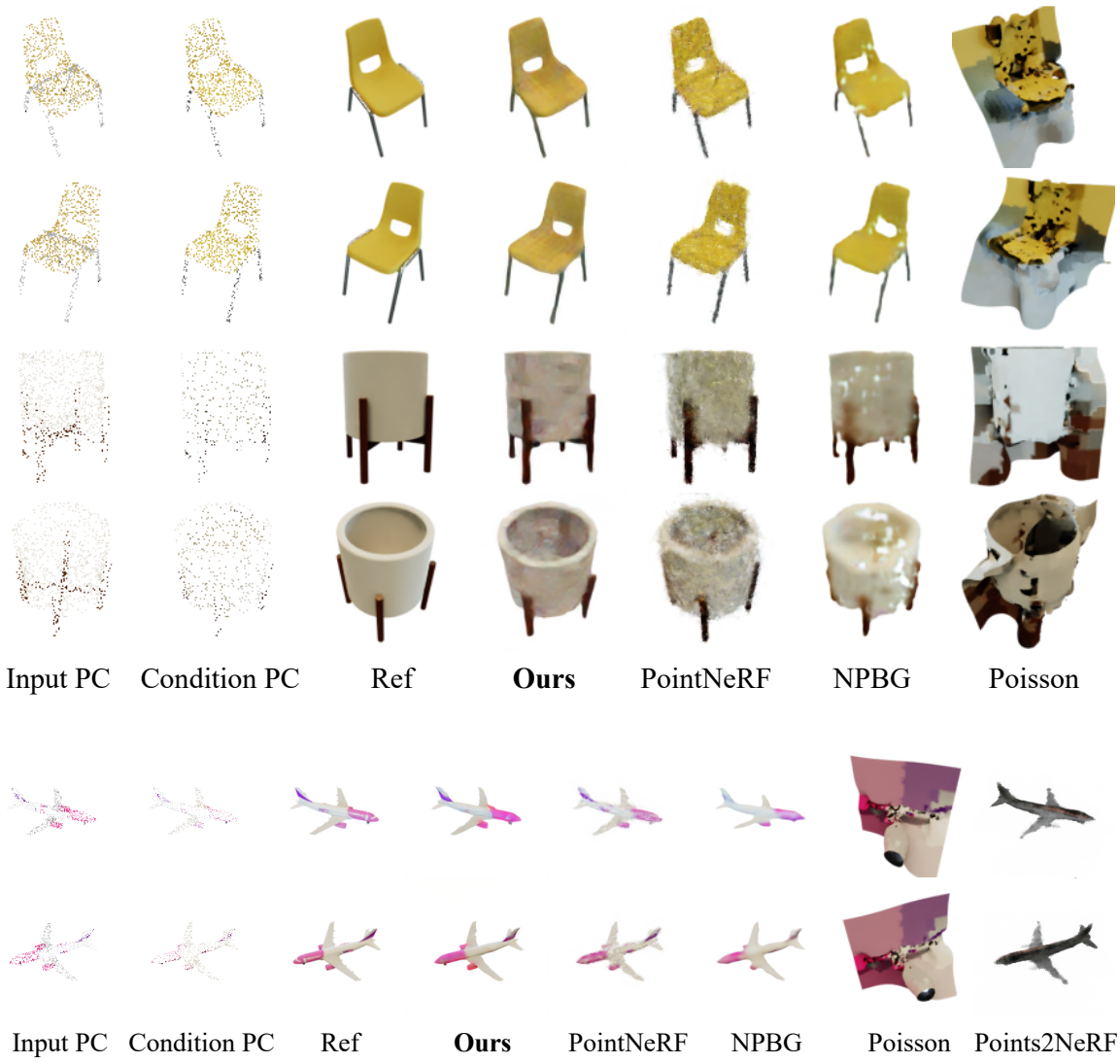

Figure 4. Qualitative comparison of our SPNR with Point-NeRF, NPBG, classical Poisson surface reconstruction, and Points2NeRF, given point cloud input shown in 1st column (only material RGB is shown). The 3rd column shows the refernece image labeled "real" for our GAN loss and used as the target image for other losses. The 2nd column shows the condition image for our conditional discriminator. Points2NeRF fails to converge on the ABO dataset and we only show the ShapetNet airplane result produced from their pretrained model.

5. Experiments

5.1. Dataset

We evaluate SPNR on two datasets: Amazon Berkeley Objects (ABO) [9] and ShapeNet [7]. For each object in a dataset, we sample 1024 points uniformly from the mesh, with material properties including base color, roughness and metallic value. 10 RGB images with white backgrounds are rendered from different viewpoints for training and evaluation. For our SPNR, we also generate corresponding condition images as described in Sec 3.4. We use the Blender Cycles renderer and a fixed environment map as lighting.

The ABO dataset contains 7953 products with artist-designed 3D meshes. It covers 63 classes and features great diversity of geometry and appearance. To enhance the diversity of base colors, we augment each ABO object with 5 variants by transforming its base colors. For each variant, we apply a single random rotation in the RGB space to all base colors. The target resolution is 128×128.

For the ShapeNet dataset, we showcase the airplane category. The target resolution is 200×200 following [60]. Results on other categories (car and chair) are presented in Appendix. We do not apply color augmentation for ShapeNet.

5.2. Metrics and Evaluation Protocol

We split the datasets into training and test sets. For ABO, there are 16161 training objects (including variants) and 200 test objects (without color augmentation). For ShapeNet plane, there are 3627 training objects and 406 test objects.

We report **PSNR** and **LPIPS** [57] on the test split of each dataset. In addition to these metrics measuring whole-image quality, we introduce **Conditional PSNR**, which measures how well the rendered image respects the material properties associated with the input point cloud. It is computed as the average PSNR score over the pixels corresponding to visible input points.

5.3. Comparison with Baselines

We first compare SPNR quantitatively with 2 baseline methods, Neural Point-Based Graphics (NPBG) [1] and Point-NeRF [52], shown in Table 2. For another baseline method, Points2NeRF [60], it fails to converge on the ABO dataset, and we are not able to reproduce the quality reported in their papers due to its weeks-long training time on ShapeNet.

NPBG [1] associates neural descriptors with the point cloud, which needs to be optimized given multiple images of a scene. It rasterizes the input point cloud and associated descriptors to multiple 2D feature images, which are fed into a 2D U-Net to output the final result. The original implementation is not suitable for generalization. Thus, we adapt NPBG by replacing descriptors with material properties. As shown in Fig 4, it either fails to fill in holes due to the sparsity of rasterized images, or can only generate blurry edges. Such limitation is also discussed by the original NPGB paper.

Point-NeRF [52] uses a 3D volumetric rendering pipeline and includes a cross-scene training and per-scene finetuning. We modify their feed-forward network in the cross-scene training stage to fit our setting. During the ray marching process, Point-NeRF aggregates features of nearest input points for each shading point. While this design may perform well for the original setting with dense point clouds, it is not suitable for processing sparse point clouds even with increased neighbor-search radius. As shown in Figure 4, the generated images tend to be blurry noisy.

In addition, we show qualitative comparison with 2 other baselines, mesh rendering with Poisson surface reconstruction with vertex color, and Points2NeRF [60]. As shown in Figure 4, Poisson surface reconstruction fails to produce a reasonable surface on sparse point clouds. This demonstrates that our problem is highly non-trivial for classical algorithms, and a quantitative measure only may not provide much insight. The setting of Points2NeRF is very close to ours. It uses a hypernetwork that takes a sparse point cloud as input and outputs the parameters of a NeRF. However, we are not able to compare with it quantitatively as its training fails to converge on the ABO dataset, and takes too long on ShapeNet. Their pretrained model seems to use a non-standard coordinate system that we could not align with our settings. For qualitative comparison, we use their pretrained model and manually choose views that roughly align with ours.

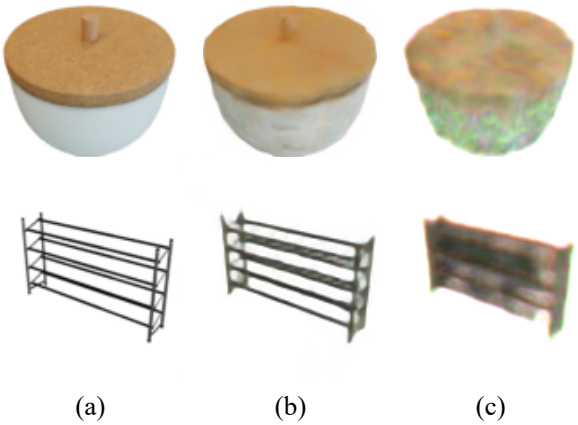

Figure 5. Comparison of reference (a), our 2-branch design (b) and single-branch design (c). The information leak between geometry and appearances hinders the learning of both geometric structures and object colors.

5.4. Ablation Studies

To justify our design choices, we conduct comprehensive ablation studies on the ABO dataset.

First, we show the effectiveness of GAN loss by replacing it with L1 loss. As shown in Figure 6, the pipeline is unable to produce fine structures with L1 supervision only. We speculate that such details are key clues for the discriminator to identify generated images from real ones, while they do not greatly affect the L1 loss.

Next, we justify our geometry/appearance 2-branch design by comparing it to a single-branch pipeline, which uses a single 3D U-Net to produce a 3D volume containing both volumetric density and appearance-related features. For the single-branch pipeline, we double the number of channels in some layers of the 3D U-Net and the MLPs, giving it roughly the same learnable parameters as the 2-branch version. As shown in Table 4 and Figure 5, the 2-branch version produces qualitatively and quantitatively better results than the single-branch one.

5.5. Additional Experiments & Applications

Rendering real-world objects. We demonstrate that SPNR trained on the ABO dataset can transfer directly to render point clouds of real-world objects. To this end, we use objects from the YCB dataset [5], a collection of

	ABO			Car			Airplane			Chair		
Method	PSNR↑	LPIPS↓	cPSNR↑	PSNR	LPIPS	cPSNR	PSNR	LPIPS	cPSNR	PSNR	LPIPS	cPSNR
NPBG [1]	16.7	0.173	13.3	16.3	0.157	8.51	25.6	0.0639	13.8	19.2	0.133	13
Point-NeRF [52]	20.5	0.214	15.2	16.6	0.181	9.33	25.6	0.0879	14.5	18.2	0.193	9.34
Points2NeRF* [60]				20.86			20.45			17.17		
Ours	22.3	0.110	17.0	19.89	0.078		25.4	0.0442	14.4	18.0	0.105	11.2

Table 1. Quantitative comparison. *PSNR values are copied from Points2NeRF paper.

	ABO			Airplane		
Method	PSNR↑	LPIPS↓	cPSNR↑	PSNR↑	LPIPS↓	cPSNR↑
NPBG [1]	16.7	0.173	13.3	25.6	0.0639	13.8
Point-NeRF [52]	20.5	0.214	15.2	25.6	0.0879	14.5
Points2NeRF* [60]	-	-	-	20.45	-	-
Ours	22.3	0.110	17.0	25.4	0.0442	14.4

Table 2. Quantitative comparison. *PSNR values are copied from the number Points2NeRF paper, due to its failure to converge on the ABO dataset and weeks-long training on ShapeNet.

Method	PSNR↑	LPIPS↓
L1 only	18.38	0.143
GAN w/o condition	16.54	0.157
GAN w/o L1	21.16	0.153
Full (L1 + GAN w/ condition)	22.26	0.110

Table 3. Ablation studies of losses of our SPNR on ABO [9].

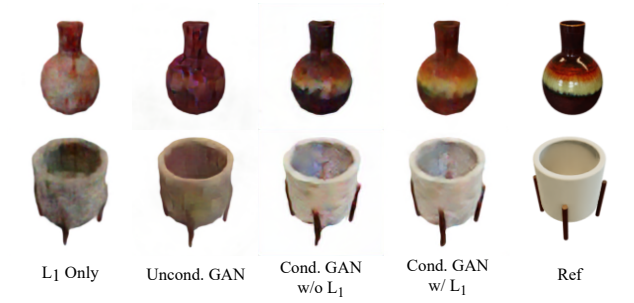

Figure 6. Qualitative comparison for ablation study on loss design.

Method	PSNR↑	LPIPS↓
1-branch	15.62	0.188
2-branch	22.26	0.110

Table 4. Ablation studies for network architecture.

scanned real objects, and uniformly sample 1024 points on their reconstructed surfaces. The results are shown in Figure 7.

Material editing. The conditional design of our pipeline allows changing object appearance by editing the input point cloud color. We demonstrate it in Figure 8. Our disentangled 2-branch framework guarantees that modifying point material features does not change the geometry. And our data augmentation strategy helps the model to adapt to various input material colors, even to unseen geometry-

Figure 7. Generalizable point cloud rendering for real-world YCB objects.

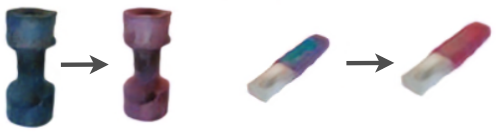

Figure 8. Material editing for real-world YCB objects.

material combinations.

6. Conclusion

In this work, we propose a novel generalizable point-cloud-based neural rendering pipeline SPNR, which takes sparse point clouds as input and renders high-quality images via volumetric rendering. Different from prior works on neural point-based rendering, SPNR can generalize to unseen point clouds without fine-tuning. We show that disentanglement in encoding geometry and appearance can considerably enhance visual quality and generalizability. The proposed conditional adversarial training objective can effectively encourage the neural renderer to synthesize high-frequency and perceptually reasonable details despite sparse inputs.

However, the rendering quality of SPNR still has room to improve. It is currently limited by the resolution of the dense voxel representation. Improving the efficiency of SPNR is our future work.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. June 2019. 2, 7, 8

[2] Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics, 2015. 1

[3] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. NeRD: Neural reflectance decomposition from image collections. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2021. 2

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 2

[5] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research, 2015. 7

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021. 2, 3, 4, 5

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing 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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 5, 6

[8] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2021. 2, 3, 5

[9] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. *CVPR*, 2022. 5, 6, 8

[10] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. Dec. 2016. 2

[11] 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. Sept. 2022. 2, 3

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[13] Philipp Henzler, Niloy Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. Nov. 2018. 2

[14] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022. 2

[15] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciniski, and Andrea Tagliasacchi. CoNeRF: Controllable neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2022. 2

[16] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020. 5

[17] 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

[18] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 2, 5

[19] Georgios Kopanas, Julien Philip, Thomas LeimkÃ¼hler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum*, 40(4):29–43, July 2021. 2

[20] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 2

[21] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. 2

[22] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. Sept. 2016. 5

[23] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. page 1–11, 12 2018. 2

[24] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018. 2

[25] Yiyi Liao, Katja Schwarz, Lars Mescheder, and Andreas Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. Dec. 2019. 2

[26] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields, 2021. 2

[27] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. Nov. 2019. 2

[28] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes. 38:1–14, 2019. 2

[29] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics, 2016. 1

[30] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. 2

[31] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 4

[32] 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 *Computer Vision – ECCV 2020*, pages 405–421. Springer International Publishing, 2020. 1, 2, 3, 4, 5

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 2

[34] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 2

[35] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. Dec. 2019. 2

[36] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2021. 2

[37] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. June 2021. 2

[38] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Dec. 2016. 3, 5

[39] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2021. 2

[40] Riccardo Roveri, A. Cengiz Öztireli, Ioana Pandele, and Markus Gross. Pointpronets: Consolidation of point clouds with convolutional neural networks. 37:87–99, 2018. 2

[41] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis, 2021. 2

[42] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[43] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. June 2019. 2

[44] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. 2

[45] Attila Szabó, Givi Meishvili, and Paolo Favaro. Unsupervised generative 3d shape learning from natural images. Oct. 2019. 2

[46] Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. Multi-species simulation of porous sand and water mixtures. 36:1–11, 2017. 1

[47] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. In *Computer Vision – ECCV 2018*, pages 55–71. Springer International Publishing, 2018. 2

[48] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction, 2021. 2

[49] Yifan Wang, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. 38:1–14, 2019. 1

[50] Chao Wen, Yinda Zhang, Chenjie Cao, Zhuwen Li, Xiangyang Xue, and Yanwei Fu. Pixel2mesh++: 3d mesh generation and refinement from multi-view images. Apr. 2022. 2

[51] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. NeuTex: Neural texture mapping for volumetric neural rendering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. 2

[52] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2022. 2, 7, 8

[53] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. page 1–14, 11 2019. 2

[54] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. 2

[55] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. 32:1–12, 2013. 1

[56] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020. 2

[57] 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. 7

[58] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. Mar. 2022. 2

[59] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. Visual object networks: Image generation with disentangled 3d representation. Dec. 2018. 2

[60] D. Zimny, T. Trzciński, and P. Spurek. Points2nerf: Generating neural radiance fields from 3d point cloud. June 2022. 2, 6, 7, 8

[61] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting, 2001. 1