Point2Pix: Photo-Realistic Point Cloud Rendering via Neural Radiance Fields

Tao Hu[1] Xiaogang Xu[1*] Shu Liu[2] Jiaya Jia[1,2]
[1] The Chinese University of Hong Kong [2] SmartMore
{taohu, xgxu, leojia}@cse.cuhk.edu.hk, liushuhust@gmail.com

Abstract

Synthesizing photo-realistic images from a point cloud is challenging because of the sparsity of point cloud representation. Recent Neural Radiance Fields and extensions are proposed to synthesize realistic images from 2D input. In this paper, we present Point2Pix as a novel point renderer to link the 3D sparse point clouds with 2D dense image pixels. Taking advantage of the point cloud 3D prior and NeRF rendering pipeline, our method can synthesize high-quality images from colored point clouds, generally for novel indoor scenes. To improve the efficiency of ray sampling, we propose point-guided sampling, which focuses on valid samples. Also, we present Point Encoding to build Multi-scale Radiance Fields that provide discriminative 3D point features. Finally, we propose Fusion Encoding to efficiently synthesize high-quality images. Extensive experiments on the ScanNet and ArkitScenes datasets demonstrate the effectiveness and generalization.

1. Introduction

Point cloud rendering aims to synthesize images from point clouds at given camera parameters, which has been frequently utilized in 3D visualization, navigation, and augmented reality. There are many advantages to point cloud representation, such as flexible shape and general 3D prior. However, since point clouds are generally produced by 3D scanners (RGBD or LiDAR) [3, 9, 14] or by Multi-View Stereo (MVS) from images [4, 15, 52], the points are usually sparsely distributed in 3D scenes. Although traditional graphics-based renderers [38, 42, 50, 58] can render point clouds to images without training or finetuning, the quality is not satisfying with hole artifacts and missing details [10].

Recently, Neural Radiance Fields (NeRF) [29] were proposed for 3D representation and high-fidelity novel view synthesis. It employs an implicit function to directly map each point's spatial information (location and direction) to attributes (color and density). However, most NeRF-based methods [23,49,54,55] are scene-specific, thus taking much time to train from scratch for novel scenes in abundant multi-view images, which limits the practical applications.

In this work, we bridge the gap between point clouds and NeRF, thus proposing a novel point cloud renderer, called Point2Pix, to synthesize photo-realistic images from colored point clouds. Compared with most NeRF-based methods [23, 29, 53, 54], ours does not necessarily require multi-view images or fine-tuning procedures for indoor scenes.

First, point clouds are treated as underlying anchors of NeRF. The training process of NeRF is to learn 3D point attributes from given locations. Because there is no mapping ground truth for multi-view images, NeRF-based methods [23, 29] indirectly train their networks with a pixel reconstruction loss. Note that point clouds are exactly made up of points with location and attributes, thus can provide training pairs for the mapping function to conduct supervised learning and improve performance.

Then, point clouds can also improve the efficiency of ray sampling. NeRF-based methods [23, 29, 54] learn the 3D shape and structure from multi-view images, which do not involve geometric prior in novel scenes. Thus, dense uniform sampling [5, 46] or coarse-to-fine sampling [29, 54] were used to synthesize high-quality images. These strategies are inefficient because most of the locations in 3D scenes are empty [23, 31]. Since point clouds represent a relatively fine shape of 3D scenes, the area around existing points deserves more attention. Based on this observation, we propose a point-guided sampling strategy to mainly focus on the local area of points in the point cloud. It can significantly reduce the number of required samples while maintaining decent synthesis accuracy.

Further, point-based networks can provide 3D features prior to subsequent applications, general for novel scenes. Although many methods [7, 18, 36, 37] have been proposed for various point cloud understanding tasks, they are usually designed for existing points. In this work, we propose Multi-scale Radiance Fields, including Point Encoder and MLP networks, to extract multi-scale features for any location in the scene. These 3D point features are discriminative and general, ensuring finetuning-free rendering. Also, in-

*Corresponding author.

spired by recent NeRF-based image generators [19, 22, 57], we render the 3D point features as multi-scale feature maps. Our fusion decoder gradually synthesizes high-resolution images. It can not only fill the possible holes but also improve the quality of rendered images. Our main contributions are summarized as follows.

- We propose Point2Pix to link point clouds with image space, which renders point clouds into photo-realistic images.
- We present an efficient ray sampling strategy and fusion decoder to greatly decrease the number of samples in each ray and the total number of rays, thus accelerating the rendering process.
- We propose Multi-scale Radiance Fields, which extract discriminative 3D prior for arbitrary 3D locations.
- Extensive experiments and ablation studies on indoor datasets demonstrate the effectiveness and generalization of the proposed method.

2. Related Work

In this section, we briefly review the related works, including various point renderers, point-based networks in extracting 3D point features, and NeRF-based synthesis.

2.1. Point-based Rendering

Traditional Point Rendering [38,42,58] is based on computer graphics, which generates images from point clouds by simulating the physical process of imaging, considering geometry [44], material [12], BRDF [16], and lighting [21]. The rendering pipeline is general for arbitrary scenes. But it cannot fill missing points and thus generate vacant pixels.

In the deep learning era, neural-based point renderers [10, 11, 41] made great progress in generating images from point clouds. They first extract multi-scale projected features from point clouds and then use a decoder to generate images. NPBG [11] augments each point by a neural descriptor to encode local geometry and appearance. NPCR [10] represents point features by 3D Convolution Neural Network (CNN), and converts the 3D representation to multi-plane images. Different from these methods, our Point2Pix combines a more discriminative point encoder with the renderer pipeline of NeRF, thus achieving better performance.

2.2. Point-based Networks

Point-based networks have been developed for many years [7, 8, 17, 18, 36, 37]. For general point understanding, PointNet [36] utilizes point-wise Multi-Layer Perception (MLP) and Pooing to extract features for 3D classification and semantic segmentation, while not capturing local structures. PointNet++ [37] introduces hierarchical feature learning to encode local point features. Although 3D CNN can also deal with point cloud data after voxelization, the maximal resolution of 3D volumes is low because of huge memory consumption and slow computation. Thus, sparse 3D CNNs [7, 17, 20, 45] draw more attention. SparseConvNet [18] differs from previous sparse convolutional networks since it uses a rectangular grid, instead of dilating the observation. In this paper, we adopt a more efficient sparse 3D CNN – MinkowskiEngine [7] – as the basic point encoder to extract 3D prior from point clouds.

2.3. NeRF-based Synthesis

NeRF [29] well balances neural network and physical rendering, thus achieving state-of-the-art performance in novel view synthesis. To handle dynamic scenes where objects move, a deformable function is learned to align different frames [24, 33]. For human reconstruction and synthesis, many methods, such as Neural-Body [35], Neural-Actor [25], and Anim-NeRF [6], introduce the parameterized human model SMPL [26] as a strong 3D prior and achieve impressive performance. To generate high-resolution images, methods of [19,22,32,57] first render low-resolution feature maps instead of images, then upsample features to the final images via 2D CNN. NeRF's input is only multi-view images and camera parameters, when combined with 3D prior, such as depth and point cloud, the performance can be further improved [1,13,31,51]. Our model also combines deep learning with physical rendering, while taking more advantage of point clouds as 3D priors to render decent-quality images in general indoor scenes.

3. Our Approach

For a point cloud $\mathbf{P} = \cup_{k=1}^{K}\{\mathbf{p}_k, \mathbf{c}_k\}$ with K points, $\mathbf{p}_k = (x_k, y_k, z_k) \in \mathbb{R}^3$ and corresponding colors $\mathbf{c}_k = (r_k, g_k, b_k) \in \mathbb{R}^3$, our goal is to synthesize a high-fidelity image $\hat{I}$ at the given camera parameter $\mathbf{V}$ via our proposed renderer (Point2Pix) $\mathcal{R}$, formulated as

$$I = \mathcal{R}(\mathbf{P}, V), \tag{1}$$

where V is represented by $H \times W$ rays $\mathbf{r}$. Each ray starts from camera center $\mathbf{o}$ in pixel direction $\mathbf{d}$.

To begin with, we introduce the background knowledge of NeRF in Sec. 3.1. Then, we propose an efficient point-based ray sampling strategy in Sec. 3.2. We also build a network to extract multi-scale 3D prior from point clouds in Sec. 3.3. Finally, we show how to combine the point feature with NeRF to render target images in Sec. 3.4. The overview of our framework can be seen in Fig. 1.

3.1. Preliminary

Given multi-view camera-calibrated images of a scene, NeRF [29] synthesizes high-quality novel view images. It

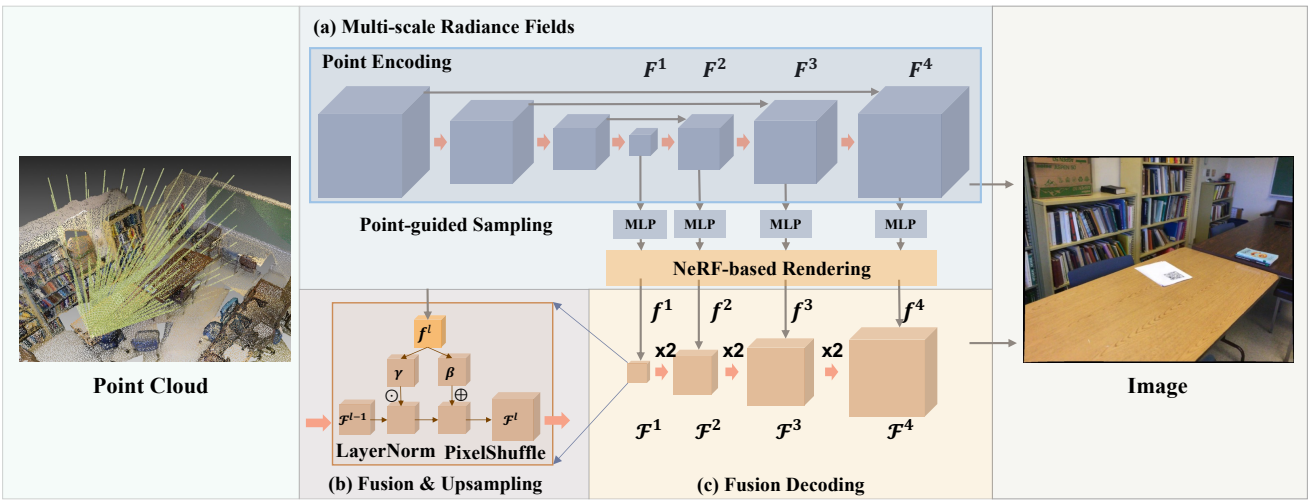

Figure 1. Overview of our proposed Point2Pix. (a) Multi-scale Radiance Fields. For an input point cloud, we first extract multiple 3D feature volumes in four scales. Next, for any queried point, we first linearly interpolate the coarse features from these feature volumes and infer the final features through MLP networks. (b) Fusion and Upsampling. Four 2D feature maps are respectively rendered through NeRF. They are fused with the previous 2D CNN output and then are upsampled by 2 times. (c) Fusion Decoding. We finally design a neural renderer to gradually synthesize target images from the projected feature maps.

mainly consists of ray sampling, implicit function, and volume rendering.

Ray Sampling. Starting from the camera center $\mathbf{o}$, ray sampling is the process that obtains a series of positions $\mathbf{x}_i$ along ray $\mathbf{r}$ with direction $\mathbf{d}$ as

$$\mathbf{x}_i = \mathbf{o} + z_i \cdot \mathbf{d}, \quad i = 1, 2, ..., N, \tag{2}$$

where z_i is the sampling depth and N is the number of samples on each ray.

Implicit Function. An implicit function f_θ is trained as a mapping from each queried location $\mathbf{x}_i$ and direction $\mathbf{d}$ to corresponding colors $\mathbf{c}_i = (r_i, g_i, b_i)$ and density σ_i, as

$$(\mathbf{c}_i, \sigma_i) = f_\theta(\mathbf{x}_i, \mathbf{d}), \tag{3}$$

where f_θ is an MLP network, and θ is its parameter.

Volume Rendering. Each ray (or pixel) color $\hat{\mathbf{c}}$ is calculated via volume rendering [28] as

$$\begin{aligned} \hat{\mathbf{c}} &= \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i, \\ \alpha_i &= 1 - \mathbf{exp}(-\sigma_i \delta_i), \\ T_i &= \mathbf{exp}(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \end{aligned} \tag{4}$$

where δ_j is the distance between neighbor samples along the ray $\mathbf{r}$.

3.2. Point-guided Sampling

According to Eq. (4), increasing the number of samples N along each ray can generate more realistic results [29]. However, the required computing resources and running time will also linearly grow. Our model is based on a point cloud with a relatively fine shape prior. Thus, we propose point-guided sampling to achieve more efficient ray sampling through the guidance of the point cloud.

For any queried point $\mathbf{x}_i$, we first find the nearest neighbour point $\mathbf{p}_i$, then check whether $\mathbf{x}_i$ is located in $\mathbf{p}_i$'s ball area with radius r or not, as

$$\| \mathbf{p}_i - \mathbf{x}_i \|_2 \leq r. \tag{5}$$

If the above condition is satisfied, we treat the queried point $\mathbf{x}_i$ as a valid sample and obtain the point feature in Sec. 3.3. If there is no valid sample along one ray, we adopt uniform sampling from default near to far depth. As illustrated in Fig. 2. Compared with previous uniform and coarse-to-fine sampling [5, 29, 46], our sampling strategy reduces computation and memory costs.

3.3. Multi-scale Radiance Fields

We extract discriminative 3D point and ray features via constructing Multi-scale Radiance Fields, including Point Encoding and NeRF-based Feature Rendering.

Point Encoding. Point Encoding is to output a discriminative 3D point feature for each valid sample $\mathbf{x}_i$. We adopt 3D sparse UNet from the Minkowski Engine (ME) [7] as the backbone of our point encoder. ME is an auto-differentiation library to build a 3D CNN for sparse tensors,

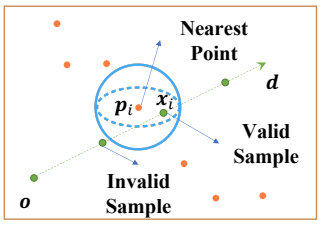

Figure 2. The proposed point-guided sampling. For any queried point $\mathbf{x}_i$, we find its nearest point $\mathbf{p}_i$ in the point cloud. If $\mathbf{x}_i$ is located in the ball area (with radius r) of $\mathbf{p}_i$, it is a valid sample. Invalid samples are omitted to improve sampling efficiency.

that are converted from point clouds. As illustrated in Fig. 1, our point encoder extracts multiple 3D feature volumes from the raw point cloud in L different scales. We select F_l at scale l to construct multi-scale radiance fields.

For each valid sample $\mathbf{x}_i$ at each scale l, we query feature in F_l to obtain the interpolated feature F_i^l and employ an implicit function Φ_l to infer density σ_i^l and final point feature f_i^l for $\mathbf{x}_i$ as

$$(\sigma_i^l, \mathbf{f}_i^l) = \Phi_l(\mathbf{F}_i^l) = \Phi_l(\mathbf{F}^l[\mathbf{x}_i]). \tag{6}$$

NeRF-based Feature Rendering. Then we render the queried 3D point features to 2D feature maps and generates images at different scales. At each feature scale l, we aggregate density σ_i^l and feature $\mathbf{f}_i^l$ to generate 2D feature map $\mathbf{f}^l$ by volume rendering of NeRF [29] as

$$\begin{aligned} \mathbf{f}^l &= \sum_{i=1}^{N} \mathbf{w}_i^l \mathbf{f}_i^l, \\ \mathbf{w}_i^l &= \mathbf{exp}(-\sum_{j=1}^{i-1} \sigma_j^l \delta_j)(1 - \mathbf{exp}(-\sigma_i^l \delta_i)). \end{aligned} \tag{7}$$

So far, we obtained L rendering feature maps $\{\mathbf{f}^l\} \in \mathbb{R}^{C_l \times H_l \times W_l}$, where H_l and W_l respectively represent the feature height and width, and C_l is the number of channels.

3.4. Fusion Decoding

Although we propose an efficient ray sampling strategy to reduce memory consumption, it still requires more than 20 GB GPU memory to render target images with the size of 480×640, as shown in Tab. 4. In addition, there are still many holes to be filled in the 2D-rendered image space. To address these issues, we design a Fusion Decoder as a neural renderer that synthesizes final images from these rendered feature maps $\mathbf{f}^l, l \in [1, L]$ by conditional convolution and upsampling modules.

Fusion. Our conditional convolution is to fuse the previous layer's feature $\mathcal{F}^{l-1}$ with the rendered features $\mathbf{f}^l$, which treats the rendered feature at each scale l as the conditional input. This module is inspired by SPADE [34], while we use Layer Normalization [2]. Specifically, as shown in Eq. (9), for the rendered feature map $\mathbf{f}^l$, we calculate the conditional parameters, including scale γ and bias β, by a *Conv2D* module. Then for feature $\mathcal{F}^{l-1}$ from the previous stage, we normalize it by Layer Normalization and scale it by γ. Finally, the fused feature $\mathcal{F}^l$ is obtained by adding the bias β as

$$\begin{aligned} (\gamma, \beta) &= \mathbf{Conv2D}(\mathbf{f}^l), \\ \mathcal{F}^{l-1} &= \gamma \cdot \mathbf{LayerNorm}(\mathcal{F}^{l-1}) + \beta. \end{aligned} \tag{8}$$

Upsampling. We adopt PixelShuffle [43] as our upsampling modules that upsample the fused feature $\mathcal{F}^{l-1}$ by 2 times at each stage, instead of using bilinear or nearest interpolation. PixelShuffle [43] is frequently adopted in the super-resolution task, which utilizes a convolution layer to extend the channel size and reshape them into the spatial size, as

$$\mathcal{F}^l = \mathbf{Pixelshuffle}(\mathbf{Conv2D}(\mathcal{F}^{l-1}), 2). \tag{9}$$

ToRGB. Finally, we introduce the decoder of the present large-scale generator, like [39], as a post-process to generate the final rendering image $\hat{I}$ for the whole point cloud renderer.

3.5. Loss Function

For the NeRF-based rendering images and neural rendering images, their optimization goals are the ground-truth images with target camera parameters. We employ point cloud loss, NeRF rendering loss, neural rendering loss, and perceptual loss to train the parameters of the proposed point encoder and fusion decoder as

$$\mathcal{L} = \lambda_{pc}\mathcal{L}_{pc} + \lambda_{nr}\mathcal{L}_{nr} + \lambda_{per}\mathcal{L}_{per}, \tag{10}$$

where λ_{pc}, λ_{nr}, and λ_{per} respectively control weights of these losses.

Point Cloud Loss. All points $\mathbf{p}_k$ in raw point clouds provide groud-truth mapping from locations $\mathbf{x}_k$ to densities $\hat{\sigma}_k$ and colors $\hat{\mathbf{c}}_k$. Denoting the queried 3D densities and colors from Point2pix in point $\mathbf{p}_k$ as $\mathbf{c}_k$ and σ_k, the point cloud loss can be represented as

$$\mathcal{L}_{pc} = \sum_{k=1}^{K} (\parallel \hat{\mathbf{c}}_k - \mathbf{c}_k \parallel^2 + \frac{1}{D} \max(0, D - \sigma_k)). \tag{11}$$

We encourage the predicted densities at $\mathbf{p}_k$ to be greater than a threshold D.

Neural Rendering Loss. $\mathcal{L}_{nr}$ is the MSE between rendered images I from fusion decoder and ground truths $\hat{I}$ as

$$\mathcal{L}_{nr} = \parallel \hat{I} - I \parallel_2^2 . \tag{12}$$

Dataset	ScanNet [9]			ARKitScenes [3]		
Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR↑	SSIM↑	LPIPS↓
Pytorch3D [38]	13.62	0.528	0.779	15.21	0.581	0.756
Pix2PixHD [47]	15.59	0.601	0.611	15.94	0.636	0.605
NPCR [10]	16.22	0.659	0.574	16.84	0.661	0.518
NPBG++ [11]	16.81	0.671	0.585	17.23	0.692	0.511
ADOP [41]	16.83	**0.699**	0.577	17.32	0.707	**0.495**
Point-NeRF [51]	**17.53**	0.685	**0.517**	**17.61**	**0.715**	0.508
Point2Pix (Ours)	**18.47**	**0.723**	**0.484**	**18.84**	**0.734**	**0.471**

Table 1. Comparing our method with different point renderers on the ScanNet [9] and ARkitScenes [3] datasets. There is no finetuning process in this experiment, which demonstrates the generalization in novel scenes.

Perceptual Loss. $\mathcal{L}_{per}$ is a frequently used loss in image synthesis, which improves the realism of generation, as

$$\mathcal{L}_{per} = \sum_{l=1}^{L} \parallel \phi(\hat{I}^l) - \phi(I^l) \parallel_1, \tag{13}$$

where $\phi(\cdot)$ means extracting VGG features.

4. Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our proposed method. First, we introduce the indoor datasets and evaluation metrics. Then, we quantitatively and qualitatively compare the proposed method with state-of-the-art point cloud renderers to show our advantages. Next, ablation studies are performed to validate the effect of each proposed module, including point-guided sampling, point encoder, and fusion decoder. Finally, we apply our method to point cloud applications.

4.1. Experimental Settings

Datasets. We perform experiments on indoor datasets containing point cloud and multi-view images, including ScanNet [9] and ARKitScenes [3]. ScanNet [9] is an RGBD scanned dataset, which contains 2.5 million images at different views in 1,513 scenes. The dataset has been annotated with calibrated cameras and colored point clouds. We split the first 1,200 scenes as a training set and the rest as a testing set. ARkitScenes [3] is a 3D indoor-scene understanding dataset, whose scenes are captured by Apple iPad Pro. There are around 5,000 scenes, and we choose the first 4,500 scenes for training and the 500 scenes for testing.

Metrics. We adopt three common metrics, including PSNR, SSIM [48], and LPIPS [56], to evaluate the performance of Point2Pix. They measure the reconstruction accuracy between the rendered images and ground-truth images.

Evaluation. We evaluate the rendering quality of different methods in two aspects, including *non-finetuning* and *finetuning*. The non-finetuning evaluation means directly measuring the rendering quality in the testing datasets. As for the finetuning evaluation, methods can refine their results on specific scenes to improve performance. Since finetuning evaluation in each case usually consumes much resources and time, we randomly choose 8 testing scenes from ScanNet datasets and the same number from the ArkitScene dataset for finetuning evaluation.

Implementation Details. We adopt MinkUnet14A as our Point Encoder. The radius r for point-guided sampling is 0.08 meters. The maximal number N of samples for each ray is 128. We extract feature volumes with $L = 4$ scales. Thus the scales are $\frac{1}{8}, \frac{1}{4}, \frac{1}{2}$, and 1 respectively. The resolution of the final rendered images is 640×480. During training, the initial learning rate is 0.004 with AdamW [27] optimizer. It exponentially decays to 0.0004 till the end (by 500 epochs). We set $\lambda_{pc} = 0.1$, $\lambda_{nr} = 1.0$, and $\lambda_{per} = 0.1$ empirically. The density threshold $D = 10$. We train our model on 4 NVIDIA Titan-V GPUs, and the batch size is 1 for each GPU.

4.2. Comparison with Point Renderers

We first compare our method with different point rendering methods by non-finetuning evaluation. After training, all methods are directly tested in novel scenes. The competitors include graphics-based point cloud renderer Pytorch3D [38], previous neural-based point renderers NPBG [11] and NPCR [10], and image generator Pix2PixHD [47]. For Pix2PixHD, we use it as an image-to-image generator to translate the rendered images from graphics-based point renderer [38] to the ground-truth images. The evaluation results are shown in Tab. 1. Ours achieves significantly higher accuracy than other solutions, which reflects the great advantage in practical applications.

4.3. Comparison with NeRF-based Synthesis

We also compare the proposed method with NeRF-based synthesis in both non-finetuning and finetuning evaluations. In this experiment, we adopt the same coarse-to-fine ray sampling as NeRF-based methods [29, 54] for a fair comparison. The results are illustrated in Tab. 2.

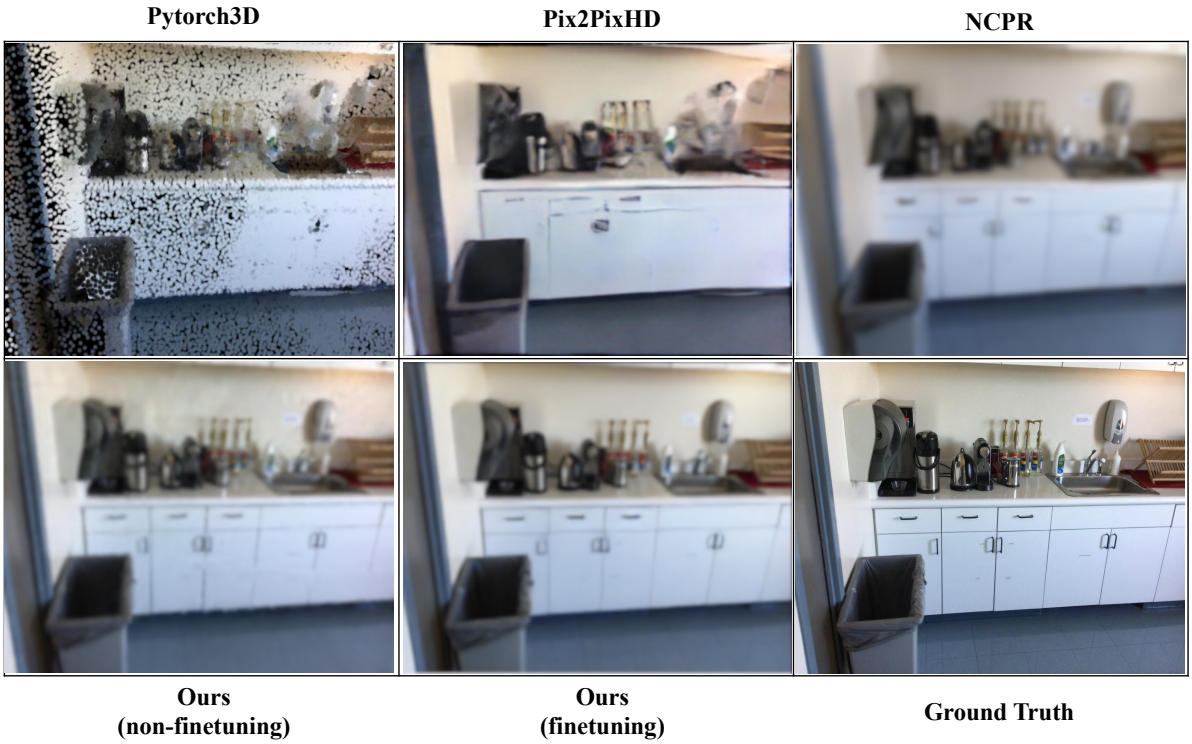

Figure 3. Qualitative comparison between different point renderers on the ScanNet [9].

Method	Time	PSNR(↑)	SSIM (↑)	LPIPS(↓)
Point-NeRF [51]	0 mins	17.53	0.685	0.517
Point2Pix (Ours)	0 mins	**18.47**	**0.723**	**0.484**
NeRF [29]	∼30 hours	21.33	0.788	0.355
NSVF [23]	∼40 hours	22.47	0.791	0.337
PlenOctrees [54]	∼30 hours	22.02	0.795	0.341
Instant-NGP [30]	20 mins	21.94	0.775	0.363
Plenoxels [53]	20 mins	22.35	0.780	0.346
Point-NeRF [51]	20 mins	22.55	0.792	0.336
Point2Pix (Ours)	20 mins	**23.02**	**0.815**	**0.318**

Table 2. Comparing our method with NeRF-based methods on the ScanNet dataset [9]. "Time" means the average finetuning time for all scenes.

To achieve general view synthesis in novel scenes, MVS-NeRF [5] and IBRNet [46] combine image prior with NeRF, while ours combines the point cloud prior. For a fair comparison, we also pre-train these two methods on the same pretraining ScanNet dataset. Our method achieves the highest performance among all. Although NeRF [29], NSVF [23], and PlenOctrees [54] can achieve competitive accuracy, their training time is much longer. When training Instant-NGP [30], Plenoxels [53], Point-NeRF [51], and our Point2Pix in 20 minutes, ours achieves better performance. This experiment demonstrates the advantage of point cloud prior when combined with NeRF.

4.4. Qualitative Comparison

We also qualitatively compare our Point2Pix with other point renderers and NeRF-based synthesis. The visualization is illustrated in Fig. 3 and Fig. 4. The Graphics-based point renderer Pytorch3D [38] usually generates images with holes because of sparse points. Due to missing 3D prior, the generated images are not realistic. Ours achieves the best visual quality, which shows Point2Pix's superiority.

4.5. Ablation Studies

Effect of Point-guided Sampling. To prove the efficiency of our point-guided sampling, we replace our sampling with other strategies used in NeRF-based methods, including uniform [5, 46] and coarse-to-fine sampling [29, 54]. Uniform sampling means uniformly obtaining N points on each ray in near-to-far depth. Coarse-to-fine sampling is proposed by the original NeRF [29]. The coarse stage uniformly samples $\frac{N}{2}$ points and the fine stages samples $\frac{N}{2}$ points according to the probability from the coarse stage.

Our point-guided sampling uniformly samples N points while only inferring the valid samples. The results are shown in Tab. 3. Since our sampling strategy considers the point cloud 3D prior, we significantly decrease the average number of samples and reduce the rendering time and GPU memory.

Effect of Multi-scale Radiance Fields. Previous meth-

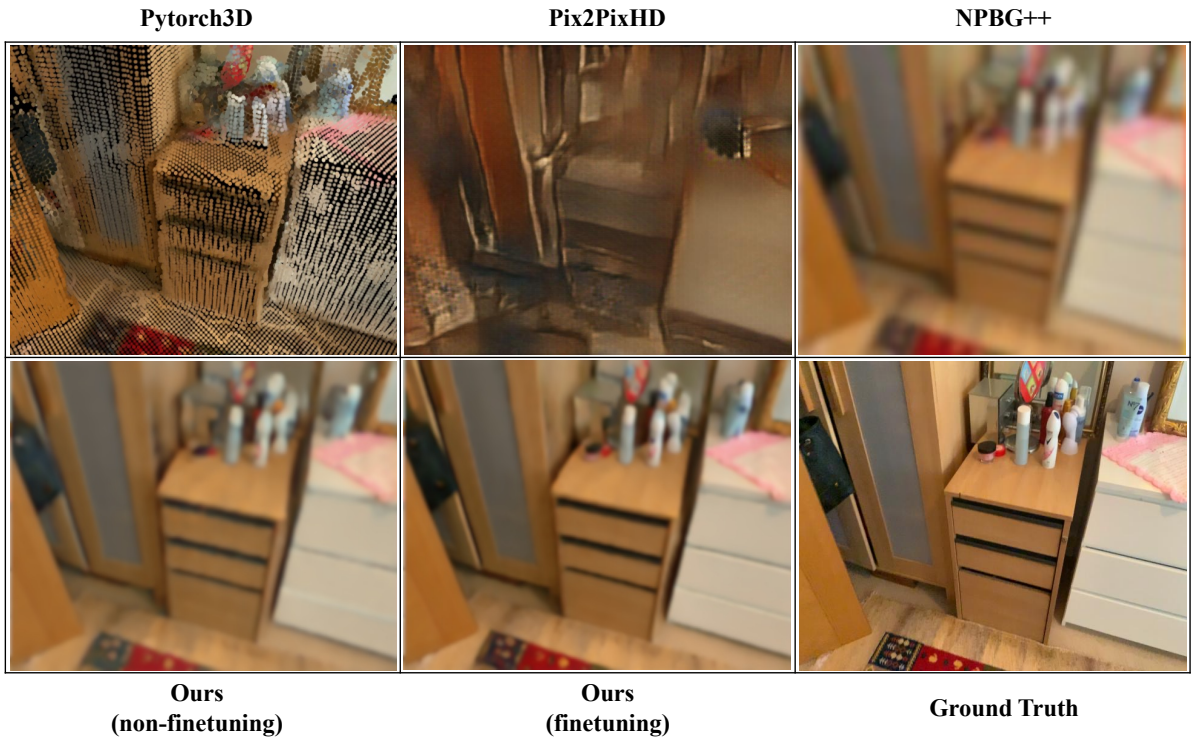

Figure 4. Qualitative comparison between different point renderers and NeRF-based methods on the ArkitScenes [3] dataset.

Ray Sampling	# Val. Sampl.	PSNR (↑)	Render Time (seconds, ↓)	Train Memory (GB, ↓)
Uniform	128	17.96	3.13	8.54
Coarse-to-Fine	128	**18.69**	5.78	9.17
Point-Guide (Ours)	**15.6**	**18.47**	**0.92**	**6.68**

Table 3. Comparison between different ray sampling strategies on ScanNet [9]. In our method, the mean sampling number for each ray is only 15.6, which is significantly less than the others.

ods, like GIRAFFE [32], HeadNeRF [22], StyleNeRF, and CIPS-3D [57], also render feature maps via NeRF, which can effectively reduce the memory and rendering time. However, different from ours, they only adopt a single scale of radiance fields. We validate our Multi-scale Radiance Fields in this experiment and show results in Tab. 4.

We first study the effect of the number of rays. In the condition of a single scale, if we directly render the image at the final resolution by NeRF, the memory consumption is heavy, and the running time is long. Decreasing sampled rays and increasing upsampling scale can promote the synthesis quality and rendering efficiency. With the increasing number of NeRF scales, accuracy further improves, which validates the effectiveness of our multi-scale radiance fields. We finally adopt the combination in the last column to balance the accuracy and time.

Selection of Point Encoder. In our Point2Pix, the Point Encoder is the backbone to provide multi-scale 3D features. In the literature of point cloud analysis, many point-based networks [7, 18, 36, 37] were proposed. We compare different backbones in this experiment.

The candidate networks include PointNet++ [37], SparseConvNet [18], and MinkUnet [7] (MinkUnet14A and MinkUnet34C). We evaluate them by only changing the point encoder, and the results are shown in Tab. 5. PointNet++ [37] consumes the largest memory and takes the longest rendering time, while the final synthesized accuracy is low. MinkUnet achieves the best results and is faster than SparseConvNet [18]. We select MinkUnet14A as our point encoder since it is more efficient than MinkUnet34C.

Effect of Fusion Decoder. Previous neural point encoders usually adopt image-to-image translator [40, 47] to render images from projected feature maps. We conduct this experiment to analyze the effect of our Fusion Decoder. We construct different alternatives by combining different decoder and fusion strategies, as shown in Tab. 6.

The combination between U-Net [40] and concatenation strategy is the most frequently adopted [10, 11, 41], while its performance is not high. When replacing the decoder with PixelShuffle [43], accuracy improves. It shows that the neural renderer does not require large respective fields as U-Net. When the concatenate strategy is replaced with our proposed fusion model, the performance is further improved, showing the rationality of our design.

Effect of Point Cloud Loss. We perform this experiment

# Scales	1	1	1	2	4
# Rays	640×480	320×240	80×60	80×60	80×60
Upsampling	$\times 1$	$\times 2$	$\times 8$	$\times 8$	$\times 8$
PSNR (↑)	17.49	17.86	18.05	18.16	**18.47**
Rendering Time (seconds, ↓)	13.12	3.56	0.85	0.92	0.96
Training Memory (GB, ↓)	22.93	11.12	6.27	6.68	6.95

Table 4. Effect of different combinations in terms of the number of scales, number of rays and scale of upsampling. We choose the combination in the last column, which achieves the best performance and is also efficient.

Point Encoder	Training Memory (GB, ↓)	Rendering Time (seconds, ↓)	PSNR (dB, ↑)
PointNet++ [37]	9.14	3.41	16.53
SparseConvNet [18]	8.18	2.18	18.26
MinkUnet14A [7]	**6.95**	**0.96**	**18.47**
MinkUnet34C [7]	8.26	1.45	18.45

Table 5. Comparison between different point encoders. We adopt MinkUnet14A [7] to extract the basic 3D prior since it achieves accurate synthesized results and is also lightweight.

Decoder	Fusion Strategy	PSNR (↑)	SSIM (↑)	LPIPS (↓)
U-Net [40]	Concatenate	18.04	0.708	0.511
PixelShuffle [43]	Concatenate	18.13	0.712	0.499
PixelShuffle [43]	SPADE (LayerNorm)	**18.47**	**0.723**	**0.484**

Table 6. Comparison between different neural renderers.

λ_{pc}	0.0	0.1	1.0
PSNR (dB, ↑)	18.23	**18.47**	18.30

Table 7. The Effect of point cloud loss.

to validate the effect of point cloud loss. By setting different loss weight λ_{pc}, we obtain the results in Tab. 7. We conclude that point cloud loss indeed promotes the mapping process from point features to 3D attributes, thus improving the performance. Interestingly, larger λ_{pc} does not necessarily achieve better accuracy, which demonstrates the minor difference between point cloud attributes with NeRF's attributes.

4.6. Application: Point Cloud Sampling

Since our point encoder can extract multi-scale point features and predict the density and color attributes, we upsample the raw point cloud by dense sampling and predict corresponding 3D attributes in the nearby area of existing points. As illustrated in Fig. 5, although there are no ground-truth dense points for us to perform supervised learning, our Point2Pix can still in-paint the missing points and insert many details for input point clouds.

5. Conclusion

In this paper, we have proposed a general point renderer, which can be directly utilized to render photo-realistic images in indoor scenes. We introduce the advantages of point cloud representation to NeRF where existing points provide ground truth pairs during training, the point area can guide the ray sampling process and the 3D prior feature can be generalized to novel scenes. We propose Multi-scale Radiance Fields to extract discriminative 3D features, point-guided sampling to efficiently reduce the number of valid samples, and a Fusion Decoder to synthesize realistic images. Experiments and ablation studies demonstrate that our Point2Pix achieves state-of-the-art synthesized performance. Our Point2Pix can also be directly employed to upsample and in-paint the raw point cloud for indoor scenes.

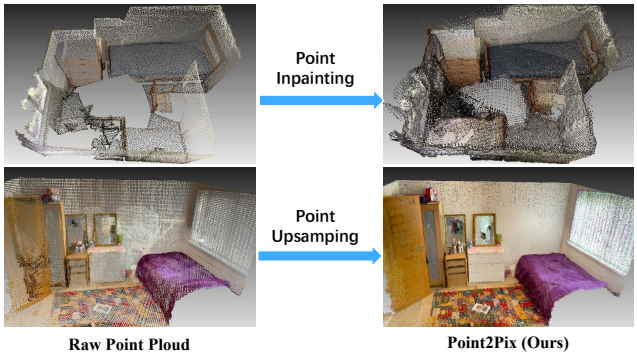

Figure 5. Our Point2Pix application in point cloud in-painting and upsampling. We upsample and fix the missing part of the raw point cloud by random and dense sampling around existing points.

Limitation and Future Work. There are still common limitations in our proposed Point2Pix. First, the overall rendering time is long compared with recent caching-based rendering [53]. Second, apart from indoor scenes, it is still difficult to directly render photo-realistic images for arbitrary environments. In future work, we will accelerate the rendering speed by combining it with other 3D scene representations, such as octrees. We will also extend our present work to more real-world situations, like the human body.

Acknowledgments

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

References

[1] Dejan Azinovic, Ricardo Martin-Brualla, Dan B. Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction. In *CVPR*, 2022. 2

[2] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv*, 2016. 4

[3] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 1, 5, 7

[4] Dan Cernea. OpenMVS: Multi-view stereo reconstruction library. 2020. 1

[5] 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 *ICCV*, 2021. 1, 3, 6

[6] Jianchuan Chen, Ying Zhang, Di Kang, Xuefei Zhe, Linchao Bao, Xu Jia, and Huchuan Lu. Animatable neural radiance fields from monocular rgb videos. *arXiv*, 2021. 2

[7] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1, 2, 3, 7, 8

[8] Ruihang Chu, Yukang Chen, Tao Kong, Lu Qi, and Lei Li. Icm-3d: Instantiated category modeling for 3d instance segmentation. *IEEE Robotics and Automation Letters*, 7(1):57–64, 2021. 2

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1, 5, 6, 7

[10] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *CVPR*, 2020. 1, 2, 5, 7

[11] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *CVPR*, 2020. 2, 5, 7

[12] Paul E. Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques '98, Proceedings of the Eurographics Workshop in Vienna, Austria, June 29 - July 1, 1998*, 1998. 2

[13] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022. 2

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1

[15] Michael Goesele, Brian Curless, and Steven M Seitz. Multi-view stereo revisited. In *CVPR*, 2006. 1

[16] Dan B. Goldman, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE TPAMI*, 2010. 2

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

[18] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv*, 2017. 1, 2, 7, 8

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

[20] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM TOG*, 2018. 2

[21] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *CVPR*, 2017. 2

[22] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *CVPR*, 2021. 2, 7

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 1, 6

[24] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2

[25] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM TOG*, 2021. 2

[26] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM TOG*, 2015. 2

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

[28] Nelson Max. Optical models for direct volume rendering. *IEEE TVCG*, 1995. 3

[29] 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. 1, 2, 3, 4, 5, 6

[30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv*, 2022. 6

[31] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 2021. 1, 2

[32] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2020. 2, 7

[33] 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 *ICCV*, 2021. 2

[34] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 4

[35] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021. 2

[36] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2, 7

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 2, 7, 8

[38] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv*, 2020. 1, 2, 5, 6

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10674–10685. IEEE, 2022. 4

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 7, 8

[41] Darius Rückert, Linus Franke, and Marc Stamminger. ADOP: approximate differentiable one-pixel point rendering. *ACM TOG*, 2022. 2, 5, 7

[42] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011. 1, 2

[43] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 4, 7, 8

[44] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM TOG*, 2006. 2

[45] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018. 2

[46] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021. 1, 3, 6

[47] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 5, 7

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

[49] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF–: Neural radiance fields without known camera parameters. *arXiv*, 2021. 1

[50] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999. 1

[51] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2, 5, 6

[52] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *ECCV*, 2018. 1

[53] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 6, 8

[54] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 1, 5, 6

[55] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NERF++: Analyzing and improving neural radiance fields. *arXiv*, 2020. 1

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

[57] Peng Zhou, Lingxi Xie, Bingbing Ni, and Qi Tian. CIPS-3D: A 3d-aware generator of gans based on conditionally-independent pixel synthesis. *arXiv*, 2021. 2, 7

[58] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv*, 2018. 1, 2