X-NeRF: Explicit Neural Radiance Field for Multi-Scene 360° Insufficient RGB-D Views

Haoyi Zhu
Shanghai Jiao Tong University
Shanghai, China
hyizhu1108@gmail.com

Abstract

Neural Radiance Fields (NeRFs), despite their outstanding performance on novel view synthesis, often need dense input views. Many papers train one model for each scene respectively and few of them explore incorporating multi-modal data into this problem. In this paper, we focus on a rarely discussed but important setting: can we train one model that can represent multiple scenes, with 360° insufficient views and RGB-D images? We refer insufficient views to few extremely sparse and almost non-overlapping views. To deal with it, X-NeRF, a fully explicit approach which learns a general scene completion process instead of a coordinate-based mapping, is proposed. Given a few insufficient RGB-D input views, X-NeRF first transforms them to a sparse point cloud tensor and then applies a 3D sparse generative Convolutional Neural Network (CNN) to complete it to an explicit radiance field whose volumetric rendering can be conducted fast without running networks during inference. To avoid overfitting, besides common rendering loss, we apply perceptual loss as well as view augmentation through random rotation on point clouds. The proposed methodology significantly out-performs previous implicit methods in our setting, indicating the great potential of proposed problem and approach. Codes and data are available at `https://github.com/HaoyiZhu/XNeRF`.

1. Introduction

Neural Radiance Fields (NeRFs) [29] have aroused significant research interest recently, which usually implicitly encode scenes using coordinate-based multi-layer perceptrons (MLPs) and have a wide range of applications such as novel view synthesis [1, 7, 29, 44, 50, 51]. A lot of follow-up work makes efforts to improve and extend NeRF [29] in various ways from convergence and rendering speed [7, 11, 23, 39] to dynamic scenes [10, 21, 46], etc. Some methods [39, 48, 49] utilize explicit structures to gain huge performance improvement but they still directly encode scenes in learnable network parameters.

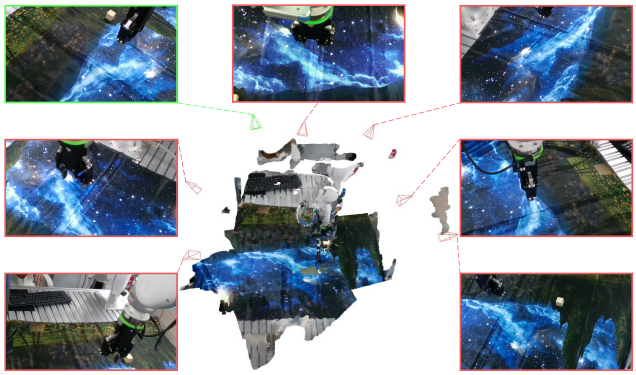

Figure 1. **An illustration of our problem setting.** The center shows an incomplete scene captured by a few low-cost RGB-D cameras. The small square cones around the scene represent the locations and directions of cameras and their corresponding RGB images are shown in surrounding rectangles. Among them, the red are seen views for training while the green one denotes the unseen view for testing. The insufficient views are very sparse with less than 10% to 20% overlapping with each other, making the problem extremely hard.

Despite the exceptional performance in lots of scenarios, most of NeRF-like methods need a lot of densely captured views when training, making them hard or expensive to apply to practice. Although some work [7, 50] has studied few-view training, their usual applicable scenarios require views with small perspective changes and large overlapping. What's more, most methods usually train a model for only one scene given the implicit modeling, making it difficult to apply them to massive scenes. Finally, with the rapid development of hardwares, depth data is increasingly available. But most current NeRF-related work only take RGB modality as input. How to utilize the depth information for better rendering deserves more exploration.

To this end, in this paper we aim to propose a methodology which allows a single model to (i) deal with multiple

scenes, (ii) with insufficient views that are 360° around the scenes and (iii) incorporate the depth data for better rendering. Fig. 1 displays an example of our setting. To tackle this hard setting, we propose an explicit neural radiance field (X-NeRF), which can take RGB-D images as inputs. Different from other NeRF-like approaches implicitly mapping coordinates to colors and densities, we explicitly modeling this problem as a completion task. The intuition behind comes from the observation that given a few RGB-D input images, a large part of the scene is actually known. In other words, plenty of information is already available initially, so that we only have to learn a general scene-irrelevant completion relation. Since the network is designed to encode a general completion mapping rather than a specific scene, we can naturally deal with the multi-scene problem.

Specifically, the input RGB-D images are converted to sparse colorful point clouds and quantized to sparse tensors on which we can directly apply Minkowski Engine [4] to operate. We adopt a 3D sparse generative CNN to construct and complete the explicit neural radiance fields. Our backbone applies a UNet-like [34] encoder-decoder structure with multi-stage generative transposed convolution and pruning layers in the decoder. To avoid overfitting on seen views, besides common rendering loss, we also apply perceptual loss with patch-wise sampling as well as view augmentation through random rotation on point clouds. Volumetric rendering with post-activation is used. By shooting and querying a ray from a pixel , the accumulated color and depth of it can be rendered.

Extensive experiments demonstrate that the proposed task is extremely challenging for existing methods while our approach can handle it well. We first compare our approach with DS-NeRF [7], an advanced NeRF-based work that also supports depth supervision, and DVGO [39] which is a state-of-the-art NeRF-like method utilizing explicit structures, on single scene experiments. To be fair, we add depth supervision to DVGO [39]. Then we compare X-NeRF with some recent NeRF-related work that supports multi-scene training such as pixelNeRF [50] and IBRNet [44] (depth supervision is also added). The results state clearly that X-NeRF is robust with multi-scene 360° insufficient views and can produce reliable novel view predictions. Our work outperforms previous methods on the extreme setting, indicating that X-NeRF can be applied to practice in a low-cost manner as we can train one model for many scenes while the inference process is quite lightweight.

2. Related Work

Novel view synthesis. To synthesize a novel view image given a set of images is a classic and long-standing task. Rendering methods can be mainly divided into image-based or model-based. Image-based methods [9, 15, 44] directly learn the transformation on image level such as warping or interpolation, which are typically more computational efficient. However, they need reference views during inference and the number and density of reference images may influence the rendering quality greatly. Model-based methods [16, 18, 32, 35, 42] express scenes as high dimensional representations and apply physically meaningful models such as optical model [27] to render the novel view images. There are various forms to represent scenes. Earlier works apply lumigraph [2, 12] and light fields [5, 19, 20, 36] to directly interpolate on input images. Nevertheless, they need exceedingly dense inputs which is totally unaffordable in many applications. Other methods utilize explicit representations such as mesh [6, 40, 43, 45] to deal with sparse inputs. However, mesh-based approaches cannot work well with gradient-based optimization due to discontinuities and local minima. Recently, many deep learning based methods employ CNNs to construct multi-plane images (MPIs) [8, 22, 28, 38, 41, 53] for forward-facing captures. There are also approaches that encode scenes as volumetric representations [16, 18, 28, 41, 42, 53], but they often struggles with complex and large-scale scenes.

Neural Radiance Fields. NeRFs have aroused great interest and achieved huge success in novel view synthesis task in recent years. A classic NeRF [29] learns a direct mapping from coordinates to corresponding textures such as color and density, implicitly encoding a scene in MLPs. Since proposed, people have extended NeRF [29] to a lot of variants with different characteristics including editable [17, 24, 47], fast inference and/or training [7, 11, 23, 39], deformable [30, 33], unconstrained images [3, 26], etc. Some recent work [39, 48, 49] introduces explicit structures to gain great performance enhancement, which indicates that the implicit MLP architecture is not necessarily the key to success. Nevertheless, despite the explicit voxel grid structures, they are actually still essentially an implicit modeling, as they still encode the scene information in learnable parameters. The implicit modeling makes NeRF-based methods hard to freely generalize on multi-scene cases. Though some work such as [50] claims that they have the ability to deal with multi-scene task, they in practice can only process multiple small objects or multiple similar simulated scenes. Moreover, when it comes to the extreme situation proposed in this paper that the input views are insufficient, which means the input views are extremely sparse but 360° around the real scenes and have almost no overlapping (often less than 10% to 20%), the implicit modeling easily overfits to a trivial solution due to its little constraints on the scene structure. Some approaches can deal with few inputs such as [7, 50], but their applicable scenario is mostly forward-facing captures which is actually still not sparse enough.

Multi-modal RGB-D data. Nowadays, with the rapid development of hardware devices, depth modal is becoming

increasingly common and available. It is often much more affordable and cheaper to capture few insufficient RGB-D views than to capture tens or hundreds times more RGB views as long as we tolerate some reasonable depth errors.[1] Therefore, it is significantly important to process RGB-D inputs and accurately render novel view depth images. Synthesizing perfect novel views from only insufficient 360° RGB images is probably ill-posed, but with the additional depth modal, the problem is wholly solvable. Deng et al. [7] already make use of depth information, and shows depth knowledge can greatly avoid overfitting on seen views as well as benefit the convergence and performance of NeRFs.

3. Preliminaries

NeRF-based methods take as inputs a set of images of different views and implicitly map 3D coordinates to densities σ and colors $\mathbf{c}$, encoding a particular scene into network parameters: $f(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$. Usually $\mathrm{Sigmoid}$ is acted on $\mathbf{c}$ and ReLU or $\mathrm{Softplus}$ is acted on σ.

Given a particular camera pose P, to render the corresponding 2D image pixels, we first emit rays r in the direction d from the projection center o of the camera to the pixels. Then N ordered query points on r between the predefined near and far planes are sampled and fed into model to obtain their densities and colors $\{(\sigma_i, \mathbf{c}_i)_{i=1}^N\}$, so that we can integrate them using the optical model proposed by [27] to derive the rendered pixel color $\hat{C}(r)$:

$$\alpha_i = 1 - \exp\left(-\sigma_i \delta_i\right) \quad 1 \le i \le N \,, \tag{1a}$$

$$T_i = \prod_{j=1}^{i-1} \left(1 - \alpha_j\right) \quad 1 \le i \le N \,, \tag{1b}$$

$$\hat{C}(r) = \left(\sum_{i=1}^{N} T_i \alpha_i c_i\right) + T_{\mathrm{K}+1} c_{\mathrm{bg}} \,, \tag{1c}$$

where α_i represents the termination probability at point i and the accumulated transmittance to point i is denoted by T_i. δ_i is the sampling step size, i.e., the distance to the adjacent sampled point on a ray. c_{bg} is a pre-defined background color, usually either 0 or 1. Depth rendering is similar to color rendering, which can be given by:

$$\hat{D}(r) = \sum_{i=1}^{N} T_i \alpha_i d_i \,, \tag{2}$$

where d_i is the distance from the ray's origin to point i.

4. Method

In this section, we introduce Explicit Neural Radiance Field (X-NeRF), an explicit representation for novel view synthesis from multi-scene 360° insufficient-view RGB-D images. Instead of implicitly constructing scenes in neural network parameters, we consider a fully explicit methodology using a 3D sparse generative CNN to learn a general scene-irrelevant completion relationship. In the following, we first describe our explicit modeling methodology (Sec. 4.1), then the detailed model architecture of X-NeRF (Sec. 4.2) and finally our volumetric rendering process (Sec. 4.3) as well as optimization functions (Sec. 4.4).

4.1. Explicit Modeling

Existing NeRF-like methods are all essentially implicit, though some such as [39, 48, 23] take use of explicit structures. In spite of their promising performance in many situations, implicit models struggle with three challenges. The first is that when there are insufficient, i.e. extremely sparse and almost non-overlapping, seen views, they tend to overfit to a trivial solution since they have no constraints nor priors on the scene structures. The second is that it is hard for them to naturally process multiple scenes using one model as they directly encode the scene information in model parameters. A few existing NeRF-based methods that support multi-scene learning either need reference views [44, 50] whose number and density have impact on the rendering effect, or employ independent explicit structures before a shared MLP [23] whose space and time costs linearly increase with the rise of scene number. The third is that they usually lack the ability to fuse RGB with depth images which are increasingly popular and available at present.

To this end, we propose a fully explicit approach that can tackle the challenging problem of novel view synthesis from multi-scene 360° insufficient seen views, and can naturally fuse RGB and depth modals. We consider the problem as an explicit completion task motivated by the fact that given a few RGB-D views around, we can easily get a large part of the location and color information of points in a specific space, and what we need to do is to complete the whole space, i.e. the explicit neural radiance field. Therefore, our network can be modeled as a completion mapping function:

$$\begin{aligned} &f_\theta : \{\mathbf{x}_{\mathrm{in}i}, \mathbf{c}_{\mathrm{in}i}\}_{i=1}^{i=N} \rightarrow \left\{\mathbf{x}_{\mathrm{out}\,j}, \mathbf{k}_j, \sigma_j\right\}_{j=1}^{j=M} \,, \\ &\text{where } \mathbf{k} = (k_\ell^m)_{\ell:0\le\ell\le\ell_{\max}}^{m:-\ell\le m} \,. \end{aligned} \tag{3}$$

Here f_θ represents our neural network with learnable parameters θ, which is a completion mapping from N input RGB-D points consisting of input point cloud coordinates $\{\mathbf{x}_{\mathrm{in}i} \in \mathbb{R}^3\}_{i=1}^{i=N}$ and colors $\{\mathbf{c}_{\mathrm{in}i} \in \mathbb{R}^3\}_{i=1}^{i=N}$, to M explicit neural radiance field points where each $\sigma \in \mathbb{R}$ denotes a scalar opacity while we apply $\mathbf{k}$, a vector of spherical harmonic (SH) coefficients, to express output color information similar to [48, 49]. Each $k_\ell^m \in \mathbb{R}^3$ is a set of 3 coefficients for RGB channels. It has been discussed in [48] that spherical harmonics of degree 2 is enough, which re-

[1] The average salary in U.S. reaches $53,490 per year and the price of an Intel RealSense RGB-D camera is only $297.17.

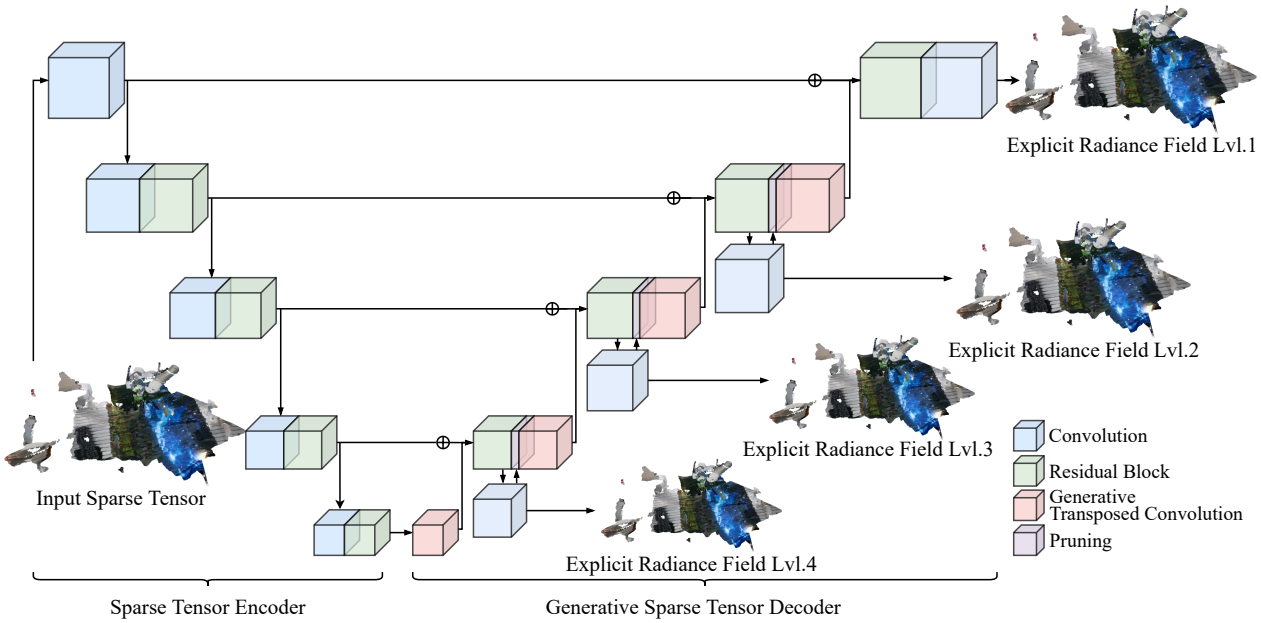

Figure 2. **Overview model architecture of X-NeRF.** We use a 3D sparse generative convolutional neural network to accomplish our fully explicit completion modeling. Given a sparse tensor representing a specific incomplete scene, we use a encoder-decoder structure similar to UNet to complete and map it to an explicit neural radiance field. The encoder only operates on existing coordinates to save cost when extracting features. Then in the multi-stage decoder, generative transposed convolutional layers with pruning layers are applied to produce novel points. Each stage gives an output with different resolution, helping to stabilize the training process.

quires 9 coefficients per color channel for a total of 27 harmonic coefficients per voxel, and we follow their setting. The SHs enable the output colors $\mathbf{c}_{\text{out}}$ to be view-dependent by querying the SH functions $Y_\ell^m : \mathbb{S}^2 \mapsto \mathbb{R}$ given its corresponding view direction $\mathbf{d}$:

$$\mathbf{c}_{\text{out}}(\mathbf{d};\mathbf{k}) = \text{Sigmoid}\left(\sum_{\ell=0}^{\ell_{\max}} \sum_{m=-\ell}^{\ell} k_\ell^m Y_\ell^m(\mathbf{d})\right) . \quad (4)$$

We note here that SHs are vital to novel view synthesis under insufficient-view condition. More detailed discussions are illustrated in Sec. 4.5 and Fig. 5.

4.2. Model Architecture

Processing RGB-D data is a classic multi-modal problem. In this paper, we convert multi-view RGB-D inputs to colorful point clouds, and voxelize them into Minkowski sparse tensors [4] so that we can directly apply operations like convolution and transposed convolution on them.

As shown in Fig. 2, we apply an encoder-decoder architecture with skip connections, similar to the structure of UNet [34]. The sparse tensor encoder is mainly used for extracting spatial and local features, which is composed of some convolution and residual blocks, while the generative sparse tensor decoder consisting of generative transposed convolution, pruning and residual blocks mainly plays a role of up-sampling and generating new points. The decoder is designed to have multi-stage outputs with increasing resolutions. The outputs of every stage participate in loss computation, which is inspired by the coarse-to-fine design in most of NeRF-related methods.

The encoder only operates on known coordinates. In other words, no new coordinates are generated during encoding. When decoding, we apply generative transposed convolutional layers followed by pruning layers to upsample and complete the whole radiance field. A simple low-dimension schematic diagram of the encoding-decoding pipeline is illustrated in Fig 3. As we can see, the function of pruning is to remove redundant points to save computational resources as well as make the output more accurate. Details for generative transposed convolution layer can be found in [4, 13].

The multi-stage design of decoder is also beneficial to decide which points to keep or to prune. Specifically, in each stage, we prune a point P_i if its corresponding output termination probability α_i is too small and it is too far away from the input point set:

$$P_i \text{ is pruned if } \alpha_i \leq \tau_\alpha \text{ and } \min_{P_j \in \mathcal{C}_{\text{in}}} \text{dist}(P_i, P_j) \leq \tau_{\text{dist}} , \quad (5)$$

where τ_α and τ_{dist} are two hyper-parameters; $\mathcal{C}_{\text{in}}$ denotes the coordinate set of input point cloud; $\text{dist}(\cdot,\cdot)$ represents the operation that computes the Euclidean distance between two points. The intention of the second distance term is

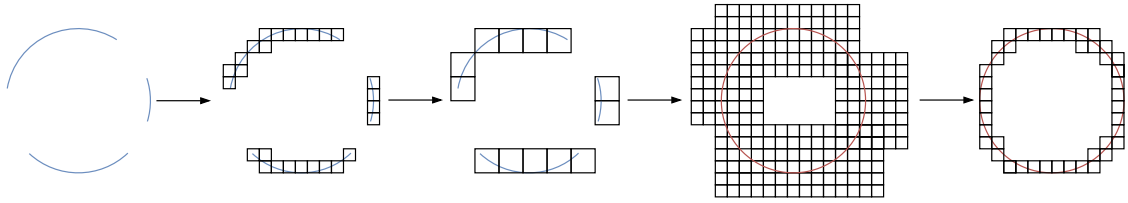

Figure 3. **A 2-D sketch about the encoding-decoding process.** Given an input surface, we first voxelize it into a sparse tensor which is fed into convolutional encoders. The encoders only operate on existing locations and generate no new coordinates. After that, generative transposed convolutional decoders are applied to up-sample and generate new coordinates so as to complete the scene structure. Finally, optional pruning layers are utilized to remove unnecessary points.

that points too far away are not likely to belong to the complete scene. In the practice, we discover that the pruning operation can reduce memory consumption and has little impact on performance, but it may slightly affect the execution speed, so we leave it as an optional choice.

4.3. Volumetric Rendering

Given a few RGB-D images, we can project and voxelize them into a sparse tensor $\mathscr{T}_{\text{in}}$ consisting of a set of coordinates $\mathcal{C}_{\text{in}}$ and the corresponding features $\mathcal{F}_{\text{in}}$:

$$\mathscr{T}_{\text{in}} = (\mathcal{C}_{\text{in}}, \mathcal{F}_{\text{in}}) \,, \tag{6a}$$
$$\mathcal{C}_{\text{in}} = \{x_i, y_i, z_i\}_{i=1}^{i=N} \,, \tag{6b}$$
$$\mathcal{F}_{\text{in}} = \{\mathbf{c}_{\text{in}i}\}_{i=1}^{i=N} \,, \tag{6c}$$

where $\mathbf{c}_{\text{in}i} \in \mathbb{R}^3$ represents the 3-channel RGB colors.

Let's denote our X-NeRF model as f_θ, then we can get an output expanded sparse tensor that contains the information of the completed scene, as discussed in Eq. 3:

$$\mathscr{T}_{\text{out}} = f_\theta(\mathscr{T}_{\text{in}}) = (\mathcal{C}_{\text{out}}, \mathcal{F}_{\text{out}}) \,, \tag{7a}$$
$$\mathcal{C}_{\text{out}} = \{x_j, y_j, z_j\}_{j=1}^{j=M} \,, \tag{7b}$$
$$\mathcal{F}_{\text{out}} = \{\mathbf{k}_j, \ddot{\sigma}_j\}_{j=1}^{j=M} \,, \tag{7c}$$

where $\ddot{\sigma}_j \in \mathbb{R}$ denotes the raw density before activation.

Sun et al. [39] has shown that post-activation, i.e. implementing activation function after interpolation operation, is the best choice for volumetric rendering and we follow this setting. As discussed in Sec. 3, given view directions $\mathbf{d}$ and shooting rays $\mathbf{r}$ corresponding to specific 2D pixels, we first interpolate $\mathscr{T}_{\text{out}}$ on $\mathbf{r}$ to obtain the SHs $\mathbf{k}$ and raw densities $\ddot{\sigma}_{\mathbf{r}}$ on them, and then use the shifted $\mathrm{Softplus}$ mentioned in Mip-NeRF [1] to acquire the corresponding densities:

$$\sigma_{\mathbf{r}} = \log(1 + \exp(\ddot{\sigma}_{\mathbf{r}} + b)) \,, \tag{8}$$

where the shift b is a hyper-parameter. The RGB colors on each ray point in radiance field can be gotten through Eq. 4. After that, we can apply Eq. 1 and Eq. 2 to render the 2D pixel colors $\hat{C}(\mathbf{r})$ and depths $\hat{D}(\mathbf{r})$.

4.4. Optimization

Since all the operations in our pipeline is differentiable, we can optimize X-NeRF through gradient decent. To combat the overfitting issue arising from insufficient views, loss function composed of the following parts is applied.

Rendering Loss. Given a set of RGB-D inputs and camera poses $\mathbf{P}$, the rendering loss is given by the mean squared error (MSE) between ground-truth and rendered outputs:

$$\begin{aligned}
\mathcal{L}_{\text{render}} &= \mathcal{L}_{\text{color}} + \lambda_D \mathcal{L}_{\text{depth}} \,, \\
\mathcal{L}_{\text{color}} &= \frac{1}{|\mathcal{R}(\mathbf{P})|} \sum_{r \in \mathcal{R}(\mathbf{P})} \|\hat{C}(r) - C(r)\|_2^2 \,, \\
\mathcal{L}_{\text{depth}} &= \frac{1}{|\mathcal{R}_{\text{VD}}(\mathbf{P})|} \sum_{r \in \mathcal{R}_{\text{VD}}(\mathbf{P})} \|\hat{D}(r) - D(r)\|_2^2 \,,
\end{aligned} \tag{9}$$

where $\mathcal{R}(\mathbf{P})$ is the set of rays of $\mathbf{P}$ while $\mathcal{R}_{\text{VD}}(\mathbf{P}) \subseteq \mathcal{R}(\mathbf{P})$ contains rays with valid depths. λ_D is a hyper-parameter.

Perceptual Loss. Besides the vanilla MSE loss, we further add a perceptual loss considering per-pixel MSE error contains no global or high level context information, which may not guide the model to the right road. In practice, lower MSE does not necessarily mean better human perceptual quality. We display intuitive and simple examples in Fig 4 for the above two cases. To this end, motivated by many image generation works, we adopt the perceptual loss [52] to avoid falling into a trivial solution:

$$\mathcal{L}_{\text{percep}} = \sum_{l=1}^{L} \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \odot \left(\hat{y}_{hw}^l - \hat{y}_{0hw}^l\right) \right\|_2^2 \,, \tag{10}$$

where $\hat{y}^l, \hat{y}_0^l \in \mathbb{R}^{H_l \times W_l \times C_l}$ are the channel-dimension unit-normalized l-th layer feature stack of rendered and reference image patches, which is extracted from L layers of a fixed pre-trained neural network such as VGG [37]. Note that to use the perceptual loss we need to sample rays in an image patch level.

Combining the above, we can get an overall loss:

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{render}} + \lambda_{\text{percep}} \mathcal{L}_{\text{percep}} \,, \tag{11}$$

where λ_{percep} is a weighting hyper-parameter.

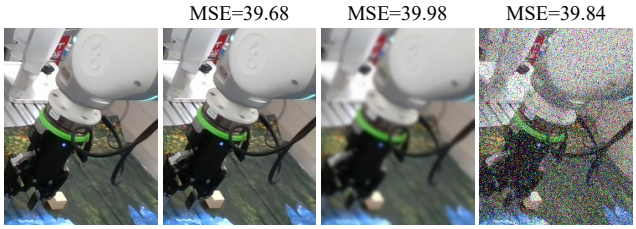

Figure 4. **Illustrations on the ambiguity of MSE loss.** The three generated examples have similar MSE values, but their perceptual quality varies greatly. From left to right are ground truth, shifted by 2 pixels, Guassian blurred and with random noise.

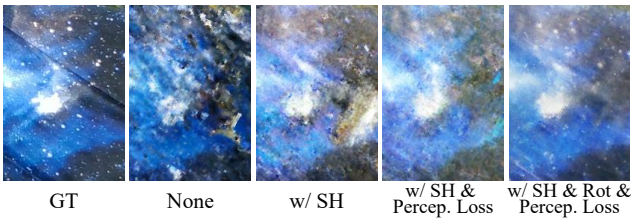

Figure 5. **Effectiveness of SH, perception loss and view augmentation through random rotation on novel view.** We can see that all of them can markedly enhance the synthesis quality.

Lastly, as mentioned in Sec. 4.2, we have multi-stage outputs, so the final total loss is:

$$\mathcal{L}_{\text{total}} = \sum_{s} \lambda^{s}_{\text{stage}} \mathcal{L}^{s}_{\text{overall}} \,, \tag{12}$$

where $\lambda^{s}_{\text{stage}}$ is the weight coefficient of stage s.

4.5. View Augmentation

We utilize spherical harmonics to make rendered colors view-independent (see Sec. 4.1). Nevertheless, due to the sparsity and small quantity of training views, we observe in the experiments that the SH coefficients are not fitted well on novel view directions. To handle this limitation, we apply random rotation augmentation on input point clouds to manually simulate unseen view directions. We find that this simple operation can significantly improve the quality of rendered unseen view images. Fig. 5 shows the effectiveness of implementing SH, perception loss and view augmentation through random rotation on point clouds.

4.6. Fast Inference

Most fully implicit coordinate-based NeRF methods struggle with the rendering efficiency because they have to run networks repeatedly on each position on each ray of given camera poses. However, X-NeRF can just save the explicit scene representations $\mathscr{T}_{\text{out}}$ so that only rendering operations such as interpolation and integral that are highly parallelizable needed to be done during inference. The inference complexity is thus reduced a lot since no neural network is needed to be run.

5. Experiments

5.1. Dataset

Since the setting of multi-scene insufficient 360° RGB-D views has never been discussed before, we collect a new dataset for this challenging task. We use 7 RGB-D cameras to capture 6 seen and 4 novel scenes, in which a robot arm is doing different tasks in different environments. The views are extremely sparse with large angle transformations. The overlapping among views is less than 10% to 20%. One example is shown in Fig. 1. Among 7 views, one is left for testing while the other 6 are training views. Please refer to the supplementary material for more details.

5.2. Implementation Details

All experiments are conducted on a single NVIDIA A100 GPU. We apply PyTorch [31] and Minkowski Engine [4] to build our sparse network. Among all single scene experiments and the multi-scene experiment, we keep the same hyper-parameters. A simple ResNet14 [14] of 3D sparse version is employed as our backbone. When voxelizing the input point clouds, we set the voxel size as 4×10^{-3}. We choose AdamW [25] as our optimizer. In each batch, we sample 2 random image patch of size 40×40 for all 6 training views, which is equivalent to a total ray batch size of $6 \times 2 \times 40 \times 40 = 19200$. We train our models for 240 epochs with an initial learning rate of 10^{-3}, and the learning rate is divided by 10 at 120th and 200th epoch. See supplementary material for detailed hyper-parameter setups.

5.3. Comparison Experiments

In this section, we compare proposed X-NeRF with state-of-the-art implicit NeRF-related work on our extremely challenging 360° insufficient RGB-D view dataset. Note that besides common metrics for RGB novel view synthesis, we also evaluate depth error since in our setting of insufficient RGB-D input views, the rendering quality of depth counts for much. We adopt mean squared error in meters in valid areas as depth metric since low-cost depth cameras may have some invalid values.

Single Scene Comparisons. Considering our data and models are RGB-D, we first compare with DS-NeRF [7], a state-of-the-art implicit NeRF-based method that also allows depth inputs. Furthermore, we also compare with DVGO [39], a state-of-the-art NeRF-based approach that utilizes explicit voxel grid structures. Unfortunately, the above two methods do not support multi-scene training, so we compare with them on single scene. DVGO [39] originally does not support depth supervision. For fair comparison, we also add depth supervision to it. The quantitative metrics on novel view can be found in Tab. 1. We can see that X-NeRF significantly out-performs the two methods on each single scene especially on depth error, which means

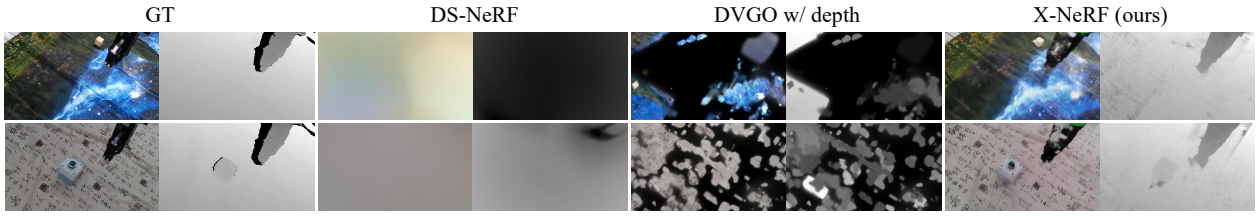

Figure 6. **Single scene qualitative results on scene 1-2.** From top to bottom, each line shows rendered RGB and depth images on novel view of different methods. Obviously, our proposed X-NeRF performs significantly better than the other two implicit methods.

	Scene 1				Scene 2				Scene 3			
	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth
	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓
DS-NeRF [7]	0.891	6.65	0.267	87.54	0.714	15.60	**0.537**	86.55	0.797	8.02	0.011	82.85
DVGO [39]	0.735	8.93	0.100	68.46	0.738	9.47	0.205	55.13	0.776	9.62	0.156	69.81
DVGO [39] w/ depth	0.726	9.46	0.124	66.16	0.723	10.03	0.220	55.69	0.764	10.22	0.170	69.21
X-NeRF	**0.521**	**17.39**	**0.414**	**0.257**	**0.505**	**16.38**	0.457	**0.356**	**0.452**	**17.83**	**0.477**	**1.66**
	Scene 4				Scene 5				Scene 6			
	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth
	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓
DS-NeRF [7]	0.698	7.07	0.093	87.56	0.701	12.00	0.748	86.14	0.754	8.63	0.463	80.04
DVGO [39]	0.651	11.84	0.576	51.67	0.729	11.89	0.540	58.10	0.740	7.36	0.226	59.71
DVGO [39] w/ depth	0.643	11.96	0.560	54.69	0.730	12.05	0.532	58.38	0.762	7.51	0.196	61.51
X-NeRF	**0.397**	**17.73**	**0.754**	**0.367**	**0.431**	**18.58**	**0.812**	**0.485**	**0.471**	**18.19**	**0.593**	**0.269**

Table 1. **Quantitative results on each single scene.** We use three common RGB metrics, namely LPIPS (using pre-trained VGG, lower is better) and PSNR/SSIM(higher is better). The depth error is evaluated by mean squared error in valid area, whose unit is meter%.

that X-NeRF succeeds in avoiding to overfit to a trivial solution. Moreover, DVGO [39] does not have a significant improvement on novel view after adding depth supervision, indicating that the key factor is not the depth loss but the modeling methodology. The qualitative results of scene 1-3 are displayed in Fig. 6 and all results can be found in supplementary material. It is obvious that implicit methods fails to generalize well on novel view facing insufficient training views and large view gaps.

Multi-Scene and Cross-Scene Comparisons. As mentioned above, X-NeRF is able to deal with multi-scene representation due to our explicit completion modeling. There is a little work that can handle multi-scene task. PixelNeRF [50] combines image features from 2D CNNs with NeRF so that it can train on multi-scene. Therefore, we re-train pixelNeRF [50] concurrently on 6 scenes for 3000 epochs to compare our work with it on the multi-scene performance. IBRNet [44] is an image-based rendering approach which applies an MLP and a ray transformer to learn a generic view interpolation function. We finetune IBRNet [44] using its pre-trained weights for 60000 iterations. Since the two methods both need reference views when rendering, we use all the 6 seen views as reference images during evaluation. From the quantitative results in Tab. 2 and the qualitative results of scene 3-6 in Fig. 8 (all results can be found in supplementary material), we can see that pixelNeRF [50] completely overfit to a trivial solution,

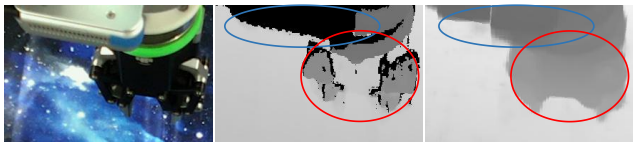

Figure 7. **Depth completion effect.** Examples of ground truth RGB image, ground truth depth image and predicted depth image are shown from left to right. The invalid depth values are represented in black color and marked with a blue ellipse. The red ellipse circles out the area where the depth camera has errors. We can see that the rendering result can not only complete the missing area but can also correct the mistakes.

and X-NeRF again beat both the two methods. If we compare Tab. 2 with Tab. 1, the multi-scene version X-NeRF is even better than the single-scene version, indicating that X-NeRF has a powerful generalization capacity. We also do cross-scene comparisons on novel view of two novel scenes. The results indicate that X-NeRF has a powerful cross-scene performance, which proves that X-NeRF learns a completion mapping with good generalization ability.

Complexity Comparisons. We further compare the inference time per image, training time and the model size among the multi-scene methods, which is shown in Tab. 3. We can find that X-NeRF requires less training time. X-NeRF also has a comparable inference time and model size with IBRNet [44], much better than pixelNeRF [50].

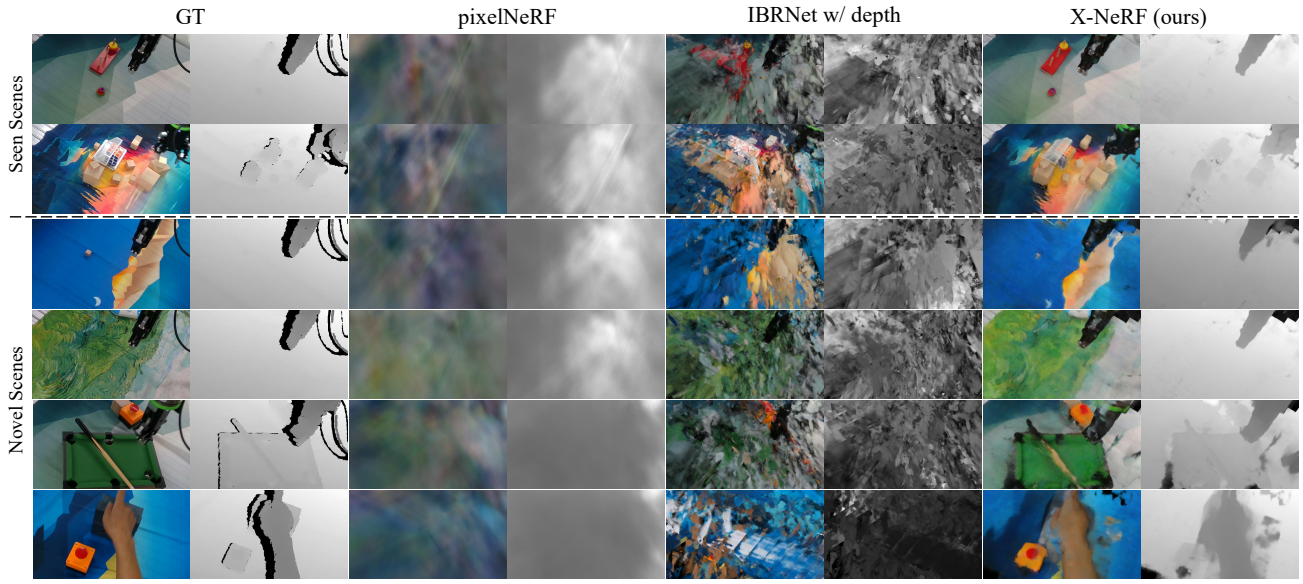

Figure 8. **Multi-scene and cross-scene qualitative results.** The first 2 rows show scene 5-6 and the last 4 rows show the 4 novel scenes.

	Scene 1				Scene 2				Scene 3				Scene 4			
	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth
	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓
pixelNeRF [50]	0.802	11.72	0.333	25.76	0.792	16.75	0.507	22.45	0.785	13.14	0.359	25.59	0.770	12.68	0.576	21.71
IBRNet [44]	0.646	14.37	0.283	9.30	0.673	16.43	0.316	12.42	0.631	16.89	0.336	12.38	0.642	13.95	0.533	8.31
IBRNet [44] w/ depth	0.620	14.85	0.336	11.48	0.657	16.63	0.368	16.79	0.617	16.65	0.351	12.05	0.631	14.67	0.552	8.96
X-NeRF (ours)	**0.534**	**18.25**	**0.440**	**0.0971**	**0.587**	**17.59**	**0.520**	**0.144**	**0.484**	**18.05**	**0.476**	**0.888**	**0.397**	**18.65**	**0.752**	**0.138**
	Scene 5				Scene 6				Novel Scene 1				Novel Scene 2			
	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth
	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓
pixelNeRF [50]	0.707	14.03	0.714	24.28	0.808	11.72	0.477	19.62	0.885	11.74	0.546	23.02	0.722	15.22	0.388	24.24
IBRNet [44]	0.633	18.49	0.613	8.83	0.656	13.45	0.365	5.38	0.633	14.06	0.455	9.42	0.638	16.33	0.294	12.36
IBRNet [44] w/ depth	0.678	14.52	0.512	12.68	0.649	13.74	0.356	5.96	0.628	13.55	0.463	8.72	0.648	14.36	0.261	14.67
X-NeRF (ours)	**0.408**	**18.71**	**0.817**	**0.346**	**0.476**	**17.69**	**0.605**	**0.152**	**0.452**	**18.66**	**0.603**	**1.70**	**0.549**	**17.93**	**0.445**	**1.82**
	Novel Scene 3				Novel Scene 4				Overall Avg. on Seen Scenes				Overall Avg. on Novel Scenes			
	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth	RGB Metrics			Depth
	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓	LPIPS↓	PSNR↑	SSIM↑	Err%↓
pixelNeRF [50]	0.679	14.57	0.630	28.72	0.748	13.24	0.580	22.74	0.778	13.34	0.494	23.23	0.759	13.69	0.536	24.68
IBRNet [44]	0.692	13.44	0.418	7.26	0.713	11.95	0.340	10.45	0.647	15.60	0.408	9.44	0.669	13.95	0.377	9.87
IBRNet [44] w/ depth	0.696	13.60	0.379	10.17	0.724	11.50	0.320	13.62	0.642	15.18	0.412	11.32	0.674	13.25	0.356	11.80
X-NeRF (ours)	**0.512**	**17.17**	**0.656**	**1.70**	**0.520**	**17.57**	**0.633**	**1.67**	**0.486**	**18.19**	**0.582**	**0.661**	**0.508**	**17.83**	**0.584**	**1.72**

Table 2. **Quantitative results on multi-scene.** We train each model on 6 scenes simultaneously, then report their performance on each scene as well as the overall average scores. We also evaluate the models on two novel scenes on novel view synthesis.

Method	#Params	Training	Inference
pixelNeRF [50]	28.2M	>10 days	∼41s
IBRNet [44]	**8.9M**	∼6 days	∼18s
X-NeRF (ours)	22.1M	∼**3.5 days**	∼**11s**

Table 3. **Complexity comparisons.** We report model size, training time (including pre-training) and inference time per image.

5.4. Completion Ability

We further study X-NeRF's completion ability. The result in Fig. 7 clearly illustrates that X-NeRF are robust enough to complete the missing areas and correct the wrong values caused by low-cost depth cameras.

6. Conclusion

We propose a novel and extremely challenging task that to synthesize novel view RGB-D images given only insufficient seen views on multiple scenes. The problem is useful in low-cost environment and may expand practical usage scenarios of NeRF-related work. A new dataset is collected for the new problem, on which our proposed fully explicit methodology X-NeRF greatly out-performs existing methods.

Acknowledgement

Thanks for the selfless help and valuable advice from Hao-Shu Fang and Prof. Cewu Lu!

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001.

[3] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Feng Ying, Xuan Wang, and Jue Wang. Hallucinated neural radiance fields in the wild, 2021.

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[5] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012.

[6] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996.

[7] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021.

[8] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019.

[9] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5515–5524, 2016.

[10] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.

[11] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. pages 14346–14355, 2021.

[12] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[13] JunYoung Gwak, Christopher Choy, and Silvio Savarese. Generative sparse detection networks for 3d single-shot object detection. In *European conference on computer vision*, pages 297–313. Springer, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[16] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8356–8364, 2020.

[17] Zhang Jiakai, Liu Xinhang, Ye Xinyi, Zhao Fuqiang, Zhang Yanshun, Wu Minye, Zhang Yingliang, Xu Lan, and Yu Jingyi. Editable free-viewpoint video using a layered neural representation. In *ACM SIGGRAPH*, 2021.

[18] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[19] Anat Levin and Fredo Durand. Linear view synthesis using a dimensionality gap light field prior. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1831–1838. IEEE, 2010.

[20] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[21] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

[22] Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *European Conference on Computer Vision*, pages 178–196. Springer, 2020.

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[24] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields, 2021.

[25] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018.

[26] 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 *CVPR*, 2021.

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

[28] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[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 *The European Conference on Computer Vision (ECCV)*, 2020.

[30] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

[33] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[35] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016.

[36] Lixin Shi, Haitham Hassanieh, Abe Davis, Dina Katabi, and Fredo Durand. Light field reconstruction using sparsity in the continuous fourier domain. *ACM Transactions on Graphics (TOG)*, 34(1):1–13, 2014.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[38] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019.

[39] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *arXiv preprint arXiv:2111.11215*, 2021.

[40] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[41] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020.

[42] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[43] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *European conference on computer vision*, pages 836–850. Springer, 2014.

[44] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. pages 4690–4699, 2021.

[45] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000.

[46] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021.

[47] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *International Conference on Computer Vision (ICCV)*, October 2021.

[48] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021.

[49] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

[50] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. pages 4578–4587, 2021.

[51] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[53] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.