UHDNeRF: Ultra-High-Definition Neural Radiance Fields

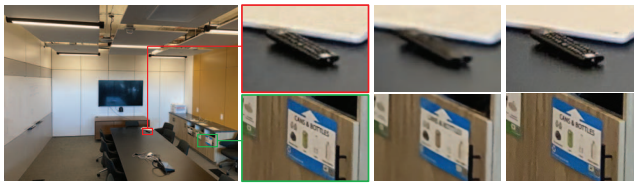
Quewei Li[1], Feichao Li[1], Jie Guo[1,†] and Yanwen Guo[1,†]
[1]National Key Lab for Novel Software Technology, Nanjing University, China
{queweili, feichaoli}@smail.nju.edu.cn, {guojie, ywguo}@nju.edu.cn

Abstract

We propose UHDNeRF, a new framework for novel view synthesis on the challenging ultra-high-resolution (e.g., 4K) real-world scenes. Previous NeRF methods are not specifically designed for rendering on extremely high resolutions, leading to burry results with notable detail-losing problems even though trained on 4K images. This is mainly due to the mismatch between the high-resolution inputs and the low-dimensional volumetric representation. To address this issue, we introduce an adaptive implicit-explicit scene representation with which an explicit sparse point cloud is used to boost the performance of an implicit volume on modeling subtle details. Specifically, we reconstruct the complex real-world scene with a frequency separation strategy that the implicit volume learns to represent the low-frequency properties of the whole scene, and the sparse point cloud is used for reproducing high-frequency details. To better explore the information embedded in the point cloud, we extract a global structure feature and a local point-wise feature from the point cloud for each sample located in the high-frequency regions. Furthermore, a patch-based sampling strategy is introduced to reduce the computational cost. The high-fidelity rendering results demonstrate the superiority of our method for retaining high-frequency details at 4K ultra-high-resolution scenarios against state-of-the-art NeRF-based solutions.

1. Introduction

Novel view synthesis, which aims to generate images at new views given a sparse set of observed images, is a long-standing problem in computer graphics and vision. Very recently, Neural Radiance Fields (NeRF) [28] have demonstrated great success in this task for learning to represent 3D scenes with implicit volumetric representation. Several following works [29, 10, 3, 18, 23, 54, 33] then improve this method in different aspects. However, previous NeRF-based methods are typically designed for training images up to 1K resolution. With the development of modern display devices, ultra-high-resolution, *e.g.*, 4K high-definition format, has become a standard for recording/displaying images and videos. Unfortunately, even though trained on 4K images, these methods cannot reproduce fine-grained details, leading to burry results with severe detail-losing problems, as shown in Fig. 1 (c).



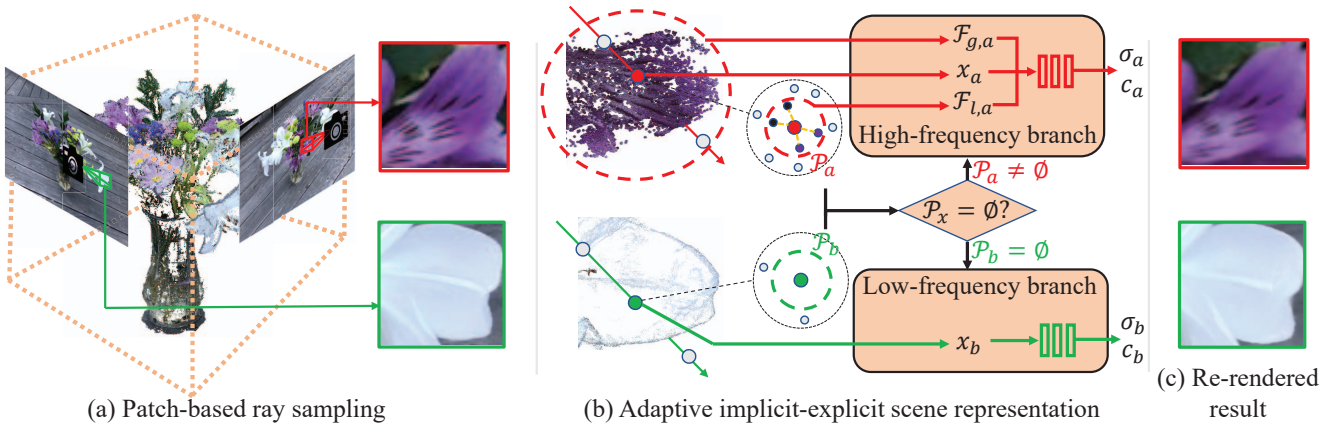
(a) Ours (b) Closeups (c) Instant NGP (d) GT

Figure 1. Visualization of our result with resolution 4032×3024 (a). We zoom in on the fine-grained details to compare our method with a modern NeRF-based method, *i.e.*, Instant NGP [29] (b)-(d).

The main difficulty preventing a NeRF model to generate ultra-high-resolution images is the mismatch between the high-resolution inputs and the low-dimensional volumetric representation (determined by the density of samples). NeRF needs to increase the sampled locations over the whole scene to produce higher-resolution renderings. For instance, to generate a high-quality result of resolution 4032×3024, more than 12 million pixels need to be rendered. Besides, thousands of MLP evaluations should also be conducted for each camera ray, which requires several hours to render a single image. Obviously, achieving 4K renderings by simply increasing the number of samples is impractical, considering the intolerable rendering time.

To reduce the long training and inference time, some fast rendering methods are proposed [10, 15, 52]. These methods commonly adopt a hybrid scene representation by utilizing explicit data structures, such as voxel grids [8, 24] or point clouds [50], to cache the scene properties and synthesize novel views by the fast query. However, reconstructing dense voxels/points to store an extremely high-resolution 3D scene requires a huge memory cost, which is also unacceptable. In short, it is not trivial to convert a NeRF model into its ultra-high-definition version by either dense sam-

†Corresponding authors.



(a) Patch-based ray sampling (b) Adaptive implicit-explicit scene representation (c) Re-rendered result

Figure 2. Overview of UHDNeRF. We sample locations over the implicit-explicit scene representation with a patch-based ray sampling strategy (a). A sample is adaptively fed into different branches according to the surrounding scene composition (whether there exist point clouds or not) (b). We leverage volume rendering techniques to integrate these samples into patches (c).

pling or utilizing a dense data cache.

In this paper, we introduce UHDNeRF, a novel NeRF-based framework that supports ultra-high-resolution view synthesis, realizing high-frequency reproduction as shown in Fig. 1 (a) and (b). We achieve this with an *adaptive implicit-explicit scene representation* by combining the implicit neural radiance fields with a sparse point cloud initialized from COLMAP [36]. Unlike previous methods utilizing explicit data structure to cover the whole scene, we adjust the point cloud by placing more points at high-frequency regions while less at low-frequency areas. The use of the sparse point cloud significantly reduces the memory overhead. Furthermore, as the generation of the sparse point cloud is independent of the NeRF volume, the implicit and explicit representations in our framework are complementary. On the one hand, the implicit volume learns to represent the low-frequency properties of the whole scene. On the other hand, the sparse point cloud is treated as a series of anchor points indicating fine-grained details. With the high-frequency information provided by the point cloud, we boost the representational ability of a NeRF model in preserving subtle details during ultra-high-resolution rendering.

As shown in Fig. 2, our UHDNeRF consists of two branches based on a *frequency separation strategy*. We adaptively feed a sample into one of the branches according to the composition of the surrounding scenes. That is, given any sampled location, if it lies in regions without a point cloud (the green sample), the low-frequency branch is selected, and our UHDNeRF goes back to the purely implicit volumetric representation. Otherwise, the sample is fed into the high-frequency branch (the red sample), where we additionally consider the surrounding point cloud to regress the scene properties at that location. To better explore the high-frequency information embedded in the sparse point cloud, we generate a global structure feature and a local point-wise feature for each sample in this branch. Moreover, to reduce the computational cost, we introduce a patch-based ray sampling strategy which notably reduces the number of investigated points in a batch. Combining the implicit volume and the sparse point cloud lead to ultra-high-definition renderings with rich details.

To summarize, our main contributions are:

- an ultra-high-definition NeRF-based framework to achieve high-fidelity 4K rendering results with an adaptive implicit-explicit scene representation,
- a frequency separation strategy by using a two-branch configuration to significantly reduce memory and computational costs during extremely high-resolution rendering, and
- a global structure feature, a local point-wise feature, and a patch-based ray sampling strategy to efficiently explore the high-frequency information embedded in the sparse point cloud.

2. Related work

Novel view synthesis has been extensively studied in the past decades. One popular class of methods uses geometry-based solutions by reconstructing a 3D proxy, which includes meshes [42, 14, 38], point clouds [1, 35, 17, 26, 43], volumes [13, 16, 32], multiplane images [22, 27, 40, 55, 44], and depth maps [7, 34, 49], and then rendering images from target views. However, the accuracy of the reconstructed geometry limits these methods. Another category leverages image-based techniques [6, 19, 11, 42, 9] for view synthesis but often requires a large amount of training data. With the emergence of neural radiance fields [28], the implicit neural representation has demonstrated remarkable

success in representing complex 3D scenes, even on sparse training images.

There is a rich literature on NeRF extensions to improve NeRF's performance in different aspects, including accelerating rendering speed [10, 12, 15, 18, 33, 52], reducing training costs [4, 41], achieving anti-aliasing [2, 3], reconstruction without camera poses [23, 47], *etc*. NeRF has also been extended to enable various applications such as relighting [54, 39, 20], 3D reconstruction [5, 48, 53], editing [51, 25], dynamic scenes [21, 31], and generation [37, 30]. Among them, some works also explore the use of the hybrid scene representation. Liu et al. [24] proposed the neural sparse voxel fields that learn to encode local properties of the scene into a set of voxels. Plenoxels [8] used a sparse voxel grid to store opacity and spherical harmonic coefficients of the scene. Muller et al. [29] constructed a multiresolution hash table of trainable feature vectors to encode the scene properties. Nevertheless, these methods are designed for fast rendering, and the explicit component is just used to store the features provided by the NeRF model. Besides, all the above methods are developed on low-resolution training images, which is far not enough to meet the demand for ultra-high-resolution rendering.

Relevant to our work, Point-NeRF [50] also utilizes point clouds to model a neural radiance field. However, this approach relies heavily on the dense point cloud that it has to add new points to cover all the missing areas in the original point cloud. During regression, Point-NeRF treats all the samples equally by aggregating a fixed number of neighboring points for each sampled location to obtain the final estimates. This method is impractical for ultra-high-resolution scenarios since it requires intolerable memory and computational costs. In contrast, our method only requires a sparse point cloud to achieve high-fidelity results by using the adaptive implicit-explicit scene representation with a frequency separation strategy, which is much more efficient in training and inference.

Image super-resolution, *i.e.*, recovering a high-resolution image from a single low-resolution, has been widely studied recently [58, 59, 60]. Inspired by the 2D single image super-resolution approaches, some methods [45, 46] adopt the super-resolution strategy by combining the NeRF representation with the convolutional neural networks (CNNs). That is, a NeRF model is used to reconstruct the low-resolution result, and a CNN model then resolves the result to a higher resolution. However, these methods still struggle with reproducing fine-grained details at ultra-high resolutions when the details are already lost in the relatively low-resolution inputs. Instead, the proposed UHDNeRF is directly trained on ultra-high-resolution images that there is no information decay at the input side.

3. UHDNeRF

In this section, we detail our framework to reconstruct the ultra-high-definition neural radiance fields. We start with some preliminaries of NeRF and discuss its limitation in representing extremely high-resolution scenes (Sec. 3.1). Then, we introduce the novel adaptive implicit-explicit scene representation for modeling scenes at ultra-high resolutions (Sec. 3.2). To better explore information embedded in the sparse point cloud, we extract the global structure and local point-wise features from the queried 3D points for each sampled location (Sec. 3.3). The network architectures and training details are introduced in Sec. 3.4.

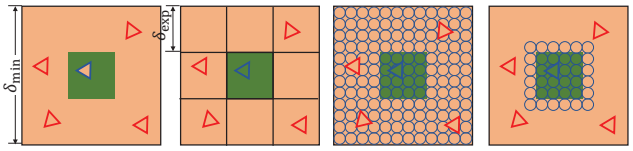
3.1. Background

Review of NeRF. NeRF represents a 3D scene as a continuous volumetric function that maps a 5D coordinate, *i.e.*, the 3D location $\boldsymbol{x} = (x, y, z)$ and 2D viewing direction $\boldsymbol{d} = (\theta, \phi)$, to the properties of a scene, *i.e.*, the volume density $\sigma(\boldsymbol{x})$ and view-dependent emitted radiance $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{d}) = (r, g, b)$. NeRF is typically parameterized by multilayer perceptrons (MLPs) $f : (\boldsymbol{x}, \boldsymbol{d}) \mapsto (\boldsymbol{c}, \sigma)$. To render the color $\boldsymbol{C}(\boldsymbol{r})$ of an image pixel, NeRF samples a set of locations along the camera ray $\boldsymbol{r} = \boldsymbol{r} + t\boldsymbol{d}$ from the camera center $\boldsymbol{o}$ through the pixel. The predicted densities and colors are then used to approximate the volume rendering integral with the numerical quadrature discussed in Max's work [61]:

$$\boldsymbol{C} = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \boldsymbol{c}_i, \tag{1}$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$ denotes the accumulated transmittance along the ray before reaching i, and $\delta_i = t_{i+1} - t_i$ represents the distance between two adjacent samples. The NeRF model is optimized by minimizing the mean squared errors (MSE) between predicted pixel colors and ground truths.

Limitation on high-resolution inputs. Previous NeRF methods struggle with handling ultra-high-resolution scenarios mainly due to the mismatch of the high-resolution input images and the low-dimensional volumetric representation. The volume resolution is determined by the density of samples. For convenience, we assume that the volume's resolution (*i.e.*, the voxel size) is roughly equal to the minimum distance $\delta_{\min}$ between two samples, and the resolution to support 4K synthesis is $\delta_{\exp}$. During inference, samples located in the same voxel would regress to the same scene properties, which are then used to integrate the pixel values in 2D images. Consequently, when $\delta_{\min} > \delta_{\exp}$, a sample would regress to the low-frequency properties even though it locates in a high-frequency region (see the sample with



(a)Implicit σ-volume (low resolution) (b) Implicit σ-volume (high resolution) (c) Hybrid scene representation (dense point cloud) (d) Our adaptive implicit-explicit representation

Figure 3. Comparison among different scene representations. We visualize a 3D voxel grid as a 2D square for simplification. We use the triangle to represent a sampled location and the circle to denote a 3D point.

blue edges in Fig. 3 (a)). This is because the low-frequency component dominates this voxel.

A simple strategy is to increase the sampling number to make $\delta_{\min} \leq \delta_{\exp}$ so that the high-frequency component occupies a separate voxel as shown in Fig. 3 (b). However, this is relatively expensive when dealing with extremely high-resolution scenarios. For instance, NeRF [28] typically renders a 1008×756 image for around 1 minute. Hence, for rendering a 4032×3024 image ($16\times$ in 2D space) to achieve the comparable visual quality as its 1K format, NeRF empirically needs to sample at least $64\times$ locations (in 3D space) over the scene, which may take more than one hour to reconstruct a single image. This inefficiency impedes the practical applications of NeRF.

3.2. Adaptive implicit-explicit scene representation

One option to reduce the long rendering time mentioned above is to utilize explicit data structures, such as the dense point cloud to enable fast rendering as shown in Fig. 3 (c). Though high-frequency information can be preserved in this way, the huge memory overhead makes this approach struggle with ultra-high-resolution scenarios. Out of this consideration, we model a complex real-world scene with the adaptive implicit-explicit scene representation, *i.e.*, the combination of an implicit volume and a sparse point cloud.

As illustrated in Fig. 3 (d), the sparse point cloud only covers the high-frequency regions instead of the whole scene, greatly reducing the memory cost. To achieve this, we initialize the point cloud with COLMAP [36] taking multiview images as input. Besides, we calculate the edge maps of the observed images. Then, we project the point cloud to each view and only retain the points around the edges. It is practical to realize ultra-high-resolution rendering using the sparse point cloud. On the one hand, the subtle details can be conveniently preserved in a point cloud format. Compared to the volumetric representation, which is limited by the voxel size, a 3D point can locate at an arbitrary position. On the other hand, though struggling with reproducing fine-grained details, the implicit volume can reconstruct low-frequency properties without the information provided by the point cloud.

Under the implicit-explicit scene representation, a sampled location adaptively regresses to the scene properties according to the surrounding scene composition. As shown in Fig. 2 (b), given any sampled location $\boldsymbol{x}$, we query its neighboring points $\mathcal{P}_x$ within a certain radius R. Note that $\mathcal{P}_x$ might be empty since the point cloud only covers the high-frequency regions. In this case, we believe that $\boldsymbol{x}$ locates in low-frequency areas, and we predict the scene properties of $\boldsymbol{x}$ with only the implicit volumetric representation. Otherwise, the surrounding region of $\boldsymbol{x}$ may probably contain high-frequency details that both $\boldsymbol{x}$ and the surrounding point cloud are used for regression.

In particular, the regression module Φ of our framework contains two branches Φ_H and Φ_L. A sampled location $\boldsymbol{x}$ is adaptively fed into one of the branches depending on $\mathcal{P}_x$, *i.e.*, whether $\mathcal{P}_x$ is empty or not. We first use an MLP Φ_σ to estimate the volume density σ at $\boldsymbol{x}$ by

$$(\sigma, \mathcal{F}_\sigma) = \Phi_\sigma(\boldsymbol{x}), \tag{2}$$

where $\mathcal{F}_\sigma$ is a 64-dimensional feature vector encoding the geometry information.

The two branches are then used to regress the view-dependent radiance $\boldsymbol{c}$ defined as

$$\boldsymbol{c} = \begin{cases} \Phi_H(\mathcal{F}_\sigma, \boldsymbol{d}, \mathcal{F}_x) & \text{if } \mathcal{P}_x \neq \emptyset \\ \Phi_L(\mathcal{F}_\sigma, \boldsymbol{d}) & \text{otherwise,} \end{cases} \tag{3}$$

where $\boldsymbol{d}$ is the viewing direction and $\mathcal{F}_x$ is the point cloud feature of $\boldsymbol{x}$. Finally, the estimated densities σ and radiances $\boldsymbol{c}$ are used to render the color $\boldsymbol{C}$ of a pixel with volume rendering [28]. With the adaptive regression process, our UHDNeRF avoids the time consuming dense sampling and the expensive dense point cloud reconstruction.

3.3. Point feature generation

To fully explore the information embedded in the sparse point cloud, we divide the point cloud feature $\mathcal{F}_x$ into a global structure feature $\mathcal{F}_{g,x}$ that represents 3D-aware structure information, and a local point-wise feature $\mathcal{F}_{l,x}$ that encodes the neighboring point features to the sampled location.

Global structure feature. We extract the global feature from all the points $\mathcal{P}$ investigated in a training epoch. As $\mathcal{P}$ may contain several groups of points representing different scene structures, it is unreasonable to generate a unified global feature from $\mathcal{P}$. Therefore, we cluster $\mathcal{P}$ according to the distance among all the points. For any sampled location $\boldsymbol{x}$, $\boldsymbol{x}$ belongs to one of the clusters determined by its nearest neighboring point. We denote all the points in this cluster as $\mathcal{P}'$. As shown in Fig. 4, we first conduct neural processing for each 3D point in $\mathcal{P}'$ with a PointNet-like

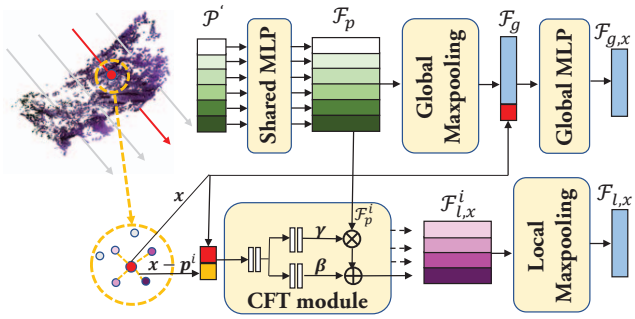


Figure 4. Pipeline of generating the global structure feature and the local point-wise feature.

neural network [56], generating a group of neural point features $\mathcal{F}_p$. A global maxpooling operation is then performed on $\mathcal{F}_p$ to obtain the global structure feature $\mathcal{F}_g$. Finally, we use a global MLP that takes $\mathcal{F}_g$ and $\boldsymbol{x}$ as input to predict a specific global feature $\mathcal{F}_{g,x}$ at this location.

Local point-wise feature. We extract the local point-wise feature $\mathcal{F}_{l,x}$ of $\boldsymbol{x}$ from its k neighboring points $\mathcal{P}_x$ within a radius R. One straightforward way to generate $\mathcal{F}_{l,x}$ is aggregating the features regressed from $\mathcal{P}_x$ using weighted averaging methods such as standard inverse distance weighting. However, by doing so, the high-frequency features may be weakened or erased. Considering this, we learn to directly predict the re-weighted point feature $\mathcal{F}_{l,x}^{i\in[1,k]}$ of each neighboring point $\boldsymbol{p}^i$ with a conditional feature transfer (CFT) module defined as

$$\begin{aligned}\mathcal{F}_{l,x}^i &= \mathrm{CFT}(\mathcal{F}_p^i, \boldsymbol{p}^i, \boldsymbol{x}|\boldsymbol{\gamma}, \boldsymbol{\beta}) \\ &= \boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{x}-\boldsymbol{p}^i) \otimes \mathcal{F}_p^i \oplus \boldsymbol{\beta}(\boldsymbol{x}, \boldsymbol{x}-\boldsymbol{p}^i),\end{aligned} \tag{4}$$

where $\mathcal{F}_p^i$ denotes the point feature of $\boldsymbol{p}^i$, $\boldsymbol{\gamma}$ is the scaling operation vector and $\boldsymbol{\beta}$ is the shifting operation vector, $\otimes$ and $\oplus$ represent the element-wise multiplication and addition operation, respectively. Specifically, we leverage $\boldsymbol{x}$ and the relative position $\boldsymbol{x}-\boldsymbol{p}^i$ to estimate the weighted operators, *i.e.*, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. These operators are then applied to the original point feature $\mathcal{F}_p^i$, generating the re-weighted feature $\mathcal{F}_{l,x}^i$. The usage of $\boldsymbol{x}-\boldsymbol{p}^i$, instead of $\boldsymbol{p}^i$ directly, makes the CFT module invariant to point translation for better generalization. After that, we utilize a maxpooling operation to obtain the local point-wise feature $\mathcal{F}_{l,x}$. Fig. 4 illustrates the whole pipeline.

To reduce the computational cost, we introduce a patch-based ray sampling strategy to replace the pixel-wise random sampling in NeRFs. By doing so the neighbours of different sampled locations are highly overlapping and hence the number of queried 3D points is notably reduced. We detail this in the ablation study (Sec. 4.3).

3.4. Implementation

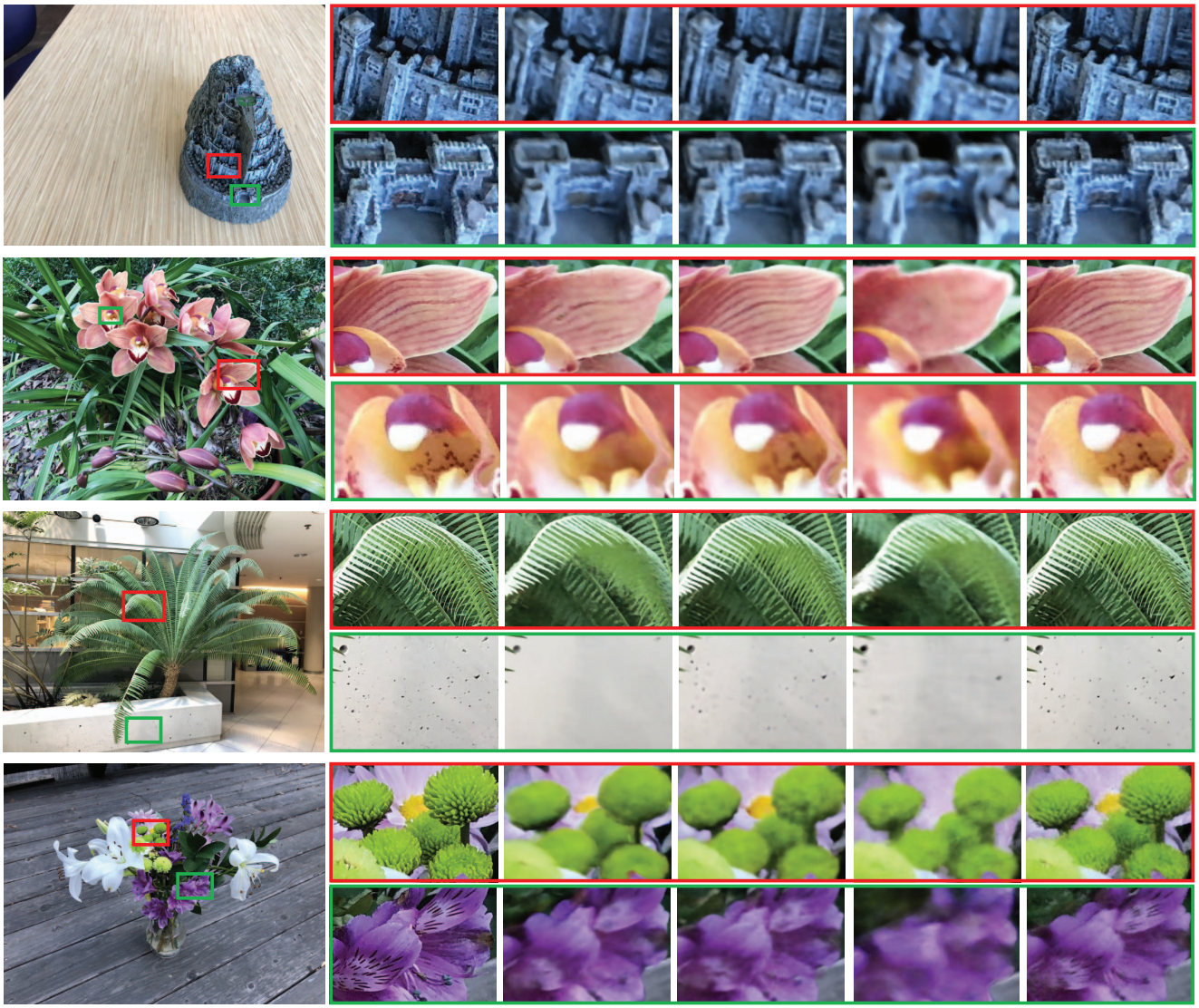
Network architectures. Our framework is comprised of a series of MLPs. Each hidden layer is followed by a ReLU activation. The point processing network contains 4 fully connected layers with 256 channels to encode the 6-channel points (3D location and the 3-channel color) into the 64-dimensional point features. The global MLP has 2 fully connected layers with 64 channels. The CFT module contains three sub-MLPs, each containing 2 fully connected layers with 64 channels. The first MLP outputs a 64-dimensional feature. This feature is then fed into the subsequent MLPs to predict the 64-dimensional $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, respectively. Both the global structure feature and the local point-wise feature are 64-dimensional. The regression module Φ also contains three sub-MLPs with the same structure as the CFT module. We use positional encoding [28] to expand the 3D locations from 3 to 63 and the viewing directions from 3 to 27. The MLPs Φ_H and Φ_L use the sigmoid function as the output activation. Except for Φ_H and Φ_L, other networks have no output activation.

Training details. To enable stable training and convergence, we first optimize the implicit volumetric representation separately with random sampling to initialize the scene geometry and reconstruct the low-frequency radiance of the scene. We utilize the bitfield [29] to skip ray marching steps in empty space. The pre-trained neural radiance field is also used to prune outliers in the point cloud by masking out the points with low estimated densities. After that, the neural radiance field is combined with the sparse point cloud for joint optimization. We train the adaptive implicit-explicit scene representation in a patch-based manner. We split the observed images into 100×100 patches and randomly selected a patch for training in each epoch. The whole framework is trained with the reconstruction loss $\mathcal{L}_{rec} = |\boldsymbol{C} - \boldsymbol{C}^*|$ between the predicted color $\boldsymbol{C}$ and the ground truths $\boldsymbol{C}^*$.

We implement UHDNeRF with PyTorch and train it on a single NVIDIA V100 GPU. We use the Adam optimizer, and the learning rate is 5e-4. We first train the implicit NeRF model for 50k epochs, which takes approximately 1 hour. After training, we optimized the adaptive implicit-explicit scene representation for 300k epochs, which takes $5 \sim 6$ hours. We query 4 nearest neighboring points for each sampled location within the radius $R = 1e^{-4}L$ where L is the maximum distance of the given scene.

4. Results

In the following, we evaluate our UHDNeRF on complex real-world scenes with 4K ultra-high-resolution captured images. We select the LLFF dataset [62] which consists of 8 forward-facing scenes with training views between 20



(a) Input (b) GT (c) MipNeRF-360 (d) Instant NGP (e) NeRF-SR (f) Ours

Figure 5. Comparison with state-of-the-art methods on the facing-forward fortress, orchids, and fern datasets (the first three rows) and the realistic 360° vasedeck dataset (the last row).

and 62. Furthermore, we conduct experiments on the more challenging realistic 360° vasedeck dataset [28], which covers the whole upper hemisphere with only 116 images. All the image resolution is 4032×3024.

4.1. Comparison with previous methods

We compare our method with three classes of methods. The first category represents scenes with purely implicit neural representation, in which we choose MipNeRF-360 [3] as a representative. For a fair comparison, we experiment on it with two settings: the original configuration of the method and an enhanced configuration by doubling the sample number of each ray. The second category adopts a hybrid scene representation. We compare with Instant NGP [29], which utilizes a voxel grid to indicate occupied spaces and a hash table to store scene properties. Similarly, we conduct experiments on it with the original and enhanced configurations by doubling both the voxel grid dimension and the hash table size. The third class combines NeRF with super-resolution techniques by upscaling the NeRF outputs with CNNs, in which NeRF-SR [45] is selected. We train all the methods on 4K resolution.

Fig. 5 compares all the methods (the enhanced versions are applied for visualization if used). While NeRF-based methods naturally support novel view synthesis at arbitrary scales, they struggle with ultra-high-resolution reconstructions even though trained on 4K images due to insufficient representational ability. As seen, although we enhance

Table 1. Quantitative comparison on LLFF and vasedeck datasets. We use MipNeRF-360+ and Instant NGP+ to denote the corresponding enhanced versions. The best metrics are highlighted in bold.

Dataset	Method	Metric			Inference times (s)	Cache memory (GB)
		PSNR↑	SSIM↑	LPIPS↓		
LLFF	MipNeRF-360 [3]	24.73	0.777	0.409	119.75	31.49
	MipNeRF-360+	24.79	0.784	0.393	245.98	31.49
	Instant NGP [29]	27.05	0.815	0.365	5.83	11.51
	Instant NGP+	27.63	0.821	0.357	8.11	20.68
	NeRF-SR [45]	23.67	0.726	0.574	240.88	29.36
	Ours	**29.03**	**0.834**	**0.325**	12.17	13.32
	Ours (w/o global)	28.52	0.823	0.341	11.06	12.42
	Ours (w/o local)	27.98	0.825	0.352	10.50	12.31
Vasedeck	MipNeRF-360 [3]	24.02	0.692	0.520	623.33	31.48
	Instant NGP [29]	24.93	0.690	0.547	7.28	11.74
	Instant NGP+	25.39	0.703	0.535	11.98	21.03
	NeRF-SR [45]	23.15	0.641	0.629	301.30	30.11
	Ours	**25.76**	**0.712**	**0.508**	20.61	15.23
	Ours (w/o global)	25.60	0.699	0.517	18.40	13.35
	Ours (w/o local)	25.52	0.705	0.526	13.03	12.71

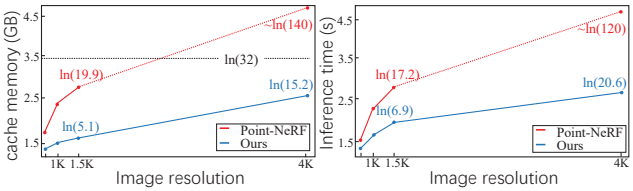


Figure 6. Comparison with Point-NeRF on cache memory and inference time at different image resolutions. We use logarithm of each value to draw curves for better visualization.

MipNeRF-360 and Instant NGP, many subtle details are still lost in the rendering results, such as the structures on the fortress and the textures on the orchids. The detail-losing problem becomes more visible when handling the realistic 360° vasedeck dataset. NeRF-SR also tends to produce burry results since the super-resolution strategy cannot reproduce fine-grained details already lost in the low resolutions. In comparison, our method achieves high-fidelity rendering on ultra-high-resolution scenarios, successfully generating results of resolution 4032×3024 and preserving fine-grained details as much as possible.

We further conduct a quantitative analysis of these methods in Table 1. We adopt PSNR, SSIM, and LPIPS as the metrics, as in [3, 28]. The inference time and cache memory are also provided as references for a comprehensive evaluation. As seen, our method ranks first among these methods on all the evaluation metrics. Note that by doubling the number of samples, the performance of MipNeRF-360 is improved slightly. However, its inference time also increases significantly (more than $2\times$ slower). Besides, the enhanced MipNeRF-360 would lead to memory limit exceeded if applied to the vasedeck dataset. Obviously, it is impractical to improve the model's representational ability with only the dense sampling strategy. As for Instant NGP, the improvement of the enhanced version by doubling the grid dimension and hash table size is negligible compared to the huge increase of cache memory (nearly $2\times$ overhead). Thanks to the proposed adaptive implicit-explicit representation and the frequency separation strategy, our UHDNeRF achieves superior performance on synthesizing 4K ultra-high-resolution results with acceptable memory cost and inference speed, and outperforms existing methods both qualitatively and quantitatively.

Table 2. Quantitative comparison with Point-NeRF on 1K LLFF dataset.

	PSNR↑	SSIM↑	LPIPS↓
Point-NeRF	20.11	0.486	0.550
Ours	30.17	0.893	0.172

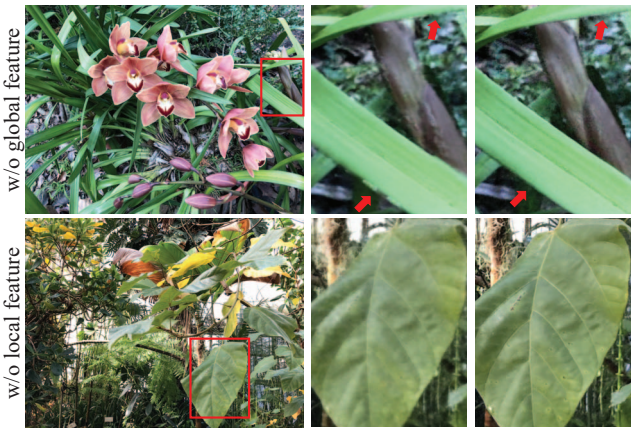
4.2. Comparison with Point-NeRF

Point-NeRF [50] also combines NeRF with the point cloud to realize high-quality rendering. However, this method is built upon a dense point cloud which is not achievable for the 4K ultra-high-resolution scenarios. To illustrate this, we train Point-NeRF and our method with a series of downscaling images in the vasedeck dataset. The input resolutions for training are 504×378 (0.5K), 1008×756 (1K), and 1344×1008 ($\sim$1.5K), respectively.

In Fig. 6 we report the cache memory and the infer-

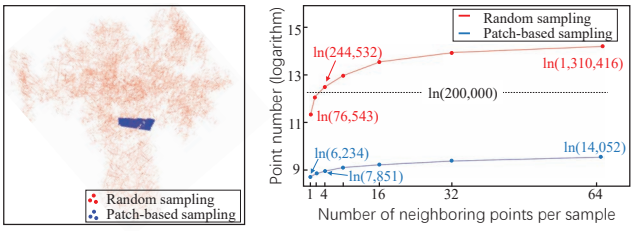
(a) Point-NeRF (b) Ours

Figure 7. Qualitative comparison with Point-NeRF on 1K fern and room datasets.



(a) Input (b) Variants (c) Our complete

Figure 8. Comparison between our complete model (c) and two variants (b), *i.e.*, the model trained without the global structure feature or the local point-wise feature.



(a) Sampled locations (b) The number of the investigated points

Figure 9. Comparison between two sampling strategies. All the samples are guaranteed to have neighboring points.

ence time at different resolutions. According to the growing trend of the curves, Point-NeRF requires intolerable memory overhead for current devices if it is applied to 4K images. Thanks to the use of the sparse point cloud and the patch-based sampling strategy, our method needs much less cache memory. Furthermore, our method achieves a much faster rendering speed with the frequency separation strategy. That is, most of the samples locate in low-frequency regions (The ratio of samples with/without neighbors is about $1:5$) without the need for point cloud processing.

We further conduct quantitative and qualitative comparisons on the downscaled 1K LLFF dataset. Unlike synthesis datasets, real-world scenarios always contain backgrounds. However, it is impractical to densely reconstruct both the foreground and background with limited views of images. As Point-NeRF only samples in the regions containing a point cloud, there are large holes in the rendered results where the point cloud is missing, as pointed out by green arrows in Fig. 7 (a). Furthermore, Point-NeRF relies heavily on the dense point cloud that each sampled location must query 8 nearest point neighbors for prediction. As the point cloud for a complex real-world scene is always imperfect (contains noises and outliers), the rendering quality is also decreased (see the red boxes in Fig. 7). As shown in Table 2, our method achieves much better metrics than Point-NeRF on the downscaled LLFF dataset (we mask out the holes in Point-NeRF when computing PSNR for a fair comparison). These experiments also demonstrate that our method is general to low-resolution scenarios (*e.g.*, 1K resolution) even though we focus on ultra-high-definition rendering.

4.3. Ablation studies

Effectiveness of the point cloud features. We verify the effectiveness of the point cloud features by removing the global structure feature and the local point-wise feature from our complete method, respectively. As shown in Fig. 8, without using the global structure feature, some artifacts may occur around the boundaries of the objects. This is because the point cloud is so sparse that a sampled location cannot obtain reliable boundary information with only the neighboring points. With the global feature providing a much larger field-of-view 3D-aware information, our complete method produces clearer and more plausible results in object boundaries.

As for the variant model removing the local point-wise feature, the fine-grained details are lost in the rendering results. The reason is that the global structure feature cannot provide high-frequency information for reproducing subtle details. Our complete method demonstrates that the local point-wise feature plays an important role in detail-preserving when dealing with extremely high-resolution rendering.

We further conduct a quantitative evaluation of different variants of our method. The quantitative results of Table 1 show that our complete model surpasses other variant models in all the metrics.

Sampling strategy. During training, NeRF selects camera rays randomly to optimize the radiance fields in every epoch. This random sampling strategy makes samples far away from each other. Consequently, the number of investigated 3D points in a batch is extremely large, and it would be highly burdened to process these point features.

Out of this consideration, we opt for the patch-based ray sampling strategy. By doing so, all the samples in a batch are constrained within a conical frustum. As a result, their neighbors are highly overlapping. As shown in Fig. 9 (a), we sample 200K locations with the random and patch-based sampling strategies, respectively. For each sample, we query its k neighboring points with k from 1 to 64. As shown in Fig. 9 (b), the number of investigated points is greatly reduced with the patch-based sampling strategy. Furthermore, there is no significant increase in the point number with k increased from 1 to 64, which means a small k value is already enough to provide complete information about the surrounding point clouds. This avoids the high computational cost of handling a large amount of neighboring points for each sample.

5. Conclusion and future work

We have proposed UHDNeRF for novel view synthesis on 4K ultra-high-resolution real-world scenes. The key to our work is an adaptive implicit-explicit scene representation with a frequency separation strategy. The scene representation is comprised of an implicit volume that learns to reconstruct the low-frequency properties over the whole scene and an explicit sparse point cloud for generating high-frequency details. A sample adaptively regresses to its scene properties based on the surrounding scene composition. To efficiently extract high-frequency information embedded in the sparse point cloud, we adopt a patch-based ray sampling strategy in each batch and generate a global structure feature and a local point-wise feature from the investigated points for any sample located in high-frequency regions. Extensive experiments on 4K scenarios demonstrate the superiority of our method in providing extremely high-resolution rendering results with rich texture details.

While our method achieves state-of-the-art performance, it suffers from some limitations. Notably, as our method relies on a sparse point cloud for subtle detail reproduction, it requires more cache memory than the original NeRF model. Besides, despite preserving much more details than the state-of-the-art NeRF solutions, our method still cannot retain all the details in the original captured images, especially for the challenging realistic 360° scenes. This is probably due to the excessively sparse training views. We hope this would be solved in the future by incorporating few-shot techniques [63, 64] into our framework.

6. Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China (62032011, 61972194), the Natural Science Foundation of Jiangsu Province (No. BK20211147), and the Key Scientific Project of Nanjing (No. 202209003).

References

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

[2] 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.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022.

[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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021.

[6] Paul 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 9*, pages 105–116. Springer, 1998.

[7] 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.

[8] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[9] Duan Gao, Guojun Chen, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deferred neural lighting: free-viewpoint relighting from unstructured photographs. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[10] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021.

[11] 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.

[12] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.

[13] 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.

[14] Ronghang Hu, Nikhila Ravi, Alexander C Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12528–12537, 2021.

[15] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12902–12911, 2022.

[16] 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.

[17] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, volume 40, pages 29–43. Wiley Online Library, 2021.

[18] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 254–270. Springer, 2022.

[19] 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.

[20] Quewei Li, Jie Guo, Yang Fei, Feichao Li, and Yanwen Guo. Neulighting: Neural lighting for free viewpoint outdoor scene relighting with unconstrained photo collections. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[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 *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 178–196. Springer, 2020.

[23] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[24] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[25] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5773–5783, 2021.

[26] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6878–6887, 2019.

[27] 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.

[28] 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 *European conference on computer vision*, pages 405–421. Springer, 2020.

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

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

[31] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

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

[33] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021.

[34] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 623–640. Springer, 2020.

[35] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[36] Johannes L Schonberger, Enliang Zheng, Marc Pollefeys and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016.

[37] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020.

[38] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8028–8038, 2020.

[39] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.

[40] 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.

[41] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[42] 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.

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

[44] 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.

[45] Chen Wang, Xian Wu, Yuan-Chen Guo, Song-Hai Zhang, Yu-Wing Tai, and Shi-Min Hu. Nerf-sr: High quality neural radiance fields using supersampling. In

Proceedings of the 30th ACM International Conference on Multimedia, pages 6445–6454, 2022.

[46] Zhongshu Wang, Lingzhi Li, Zhen Shen, Li Shen, and Liefeng Bo. 4k-nerf: High fidelity neural radiance fields at ultra high resolutions. *arXiv preprint arXiv:2212.04701*, 2022.

[47] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[48] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619, 2021.

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

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

[51] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021.

[52] 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.

[53] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *Advances in Neural Information Processing Systems*, 34:29835–29847, 2021.

[54] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *arXiv preprint arXiv:2106.01970*, 2021.

[55] 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.

[56] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[57] Karen Simonyan and Andrew Zisserman Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[58] Xiaozhong Ji, Yun Cao, Ying Tai, Chengjie Wang, Jilin Li, and Feiyue Huang Real-world super-resolution via kernel estimation and noise injection. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 466–467, 2020.

[59] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021.

[60] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings ofthe IEEE/CVF International Conference on Computer Vision*, pages 1905–1914, 2021.

[61] Nelson L. Max Optical models for direct volume rendering. In *IEEE Trans. Vis. Comput. Graph*, 1995.

[62] 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. In *ACM Transactions on Graphics*, 2019.

[63] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[64] Mijeong Kim, Seonguk Seo, and Bohyung Han Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022.