This WACV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

MFNeRF: Memory Efficient NeRF with Mixed-Feature Hash Table

Yongjae Lee
Arizona State University
Tempe, AZ 85281
ylee298@asu.edu

Li Yang
University of North Carolina at Charlotte
Charlotte, NC 28223
lyang50@uncc.edu

Deliang Fan
Arizona State University
Tempe, AZ 85281
dfan@asu.edu

Abstract

*Recently, neural radiance fields (NeRFs) have shown remarkable performance in generating photorealistic novel views in 3D modeling. The traditional NeRF typically requires extensive training and long rendering times, inspiring many recent works to utilize efficient data structures, such as feature grids, to ease the computational burden of multilayer perceptron networks. However, Those approaches require storing features in dense grids that demands a substantial amount of memory space, resulting in a notable memory bottleneck. Consequently, it leads to a significant increase in training time. To address this issue, in this work, we propose MFNeRF, a memory-efficient NeRF framework that employs a mixed-feature hash table to improve memory efficiency and reduce training time while maintaining reconstruction quality. Specifically, we first design a mixed-feature hash encoding method to adaptively mix parts of multi-level feature grids and map them to a single hash table. Following that, in order to obtain the correct index of a grid point, we further develop an index transformation method that transforms indices of an arbitrary-level grid to those of a canonical grid. Extensive benchmarking against the state-of-the-art methods, including InstantNGP, TensoRF, and DVGO, indicates that our MFNeRF achieves faster training and rendering times on the same GPU hardware, with a significantly smaller memory, while maintaining similar or even higher reconstruction quality. Compared to the InstantNGP-Big model, our method could achieve 89% improvement in the figure of merit, defined in terms of PSNR*FPS/MB.*

1. Introduction

Representing 3D scenes has gained significant recognition across various industries. In the past, the traditional approach involved breaking down free surfaces into primitive triangles, also known as polygons [5]. However, this method incurs high cost, so has to simplify the scene to reduce the cost, which, in turn, leads to a sacrifice in the reconstruction quality. Recently, the neural radiance field (NeRF) [17] have made significant advancements in both the quality and efficiency of synthesizing novel views. It employs multilayer perceptrons (MLPs) to model the colors and densities of points in space based on their 3D coordinates. Motivated by NeRF, a large body of methods under various constraints has been introduced [6, 13, 16, 24, 31].

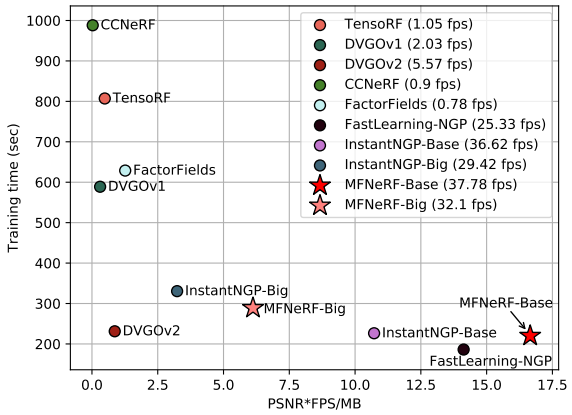


Figure 1. Comparison to state-of-the-art methods with ours in terms of memory efficiency and training time on the Synthetic NeRF dataset [17]. The PSNR*FPS/MB metric quantifies how efficiently a method encodes and renders a 3D scene relative to the unit memory size. Our MFNeRF represents scenes using fewer parameters, enhanced by sharing features across multiresolution grids. Notably, MFNeRF-Base requires only 66.5% of the memory size compared to InstantNGP-Base while achieving improvements in rendering quality, frames per second (FPS), and training time.

In general, NeRF encompasses two ways to encode 3D scenes: implicit and explicit representations. The implicit representation uses deep, wide MLPs to learn the continuous radiance field [16, 17]. It is crucial to provide sufficiently large inputs to encode high-dimensional and complex radiance fields into MLP networks. One common approach is positional encoding [17, 28], which decomposes

low-dimensional spatial and directional inputs into a set of high-dimensional frequency parameters, enabling MLP networks to learn complex shapes and view-dependent light effects. To process high-dimensional inputs effectively, the structure of MLP networks needs to be both deep and wide. Additionally, due to the infinite and continuous nature of the space, a vast number of queries to large MLP networks are required to model every direction through every point. As a result, both training and rendering times are typically very long.

To address the above issue, explicit representation-based methods [6, 13, 24, 31] have been proposed, which abstract and manage scenes as features. In practice, these methods mainly utilize feature grids to represent spatial information through voxel coordinates. With rich feature representation, this design allows the use of small MLPs, leading to training and rendering time reductions. However, these methods need to maintain the large and dense feature grids, suffering from enormous GPU memory consumption. To address the memory bottleneck challenge, feature compression methods have been proposed [3, 18]. A representative work, InstantNGP [18], proposes a multiresolution hash table to compress the features of each grid at different resolutions into smaller-sized hash tables. However, having more features in the grid demands significantly larger memory allocations and frequent access to bigger hash tables, as shown in our analysis of InstantNGP-Base and -Big (see Fig. 1), which utilize hash table sizes of 2^{20} and 2^{22}, respectively. In real-world applications, this memory bottleneck can become a critical issue when learning on large scenes, such as buildings and cities [25].

To address the memory bottleneck challenge, following the explicit representation approach, we propose MFNeRF, a memory-efficient NeRF framework featuring a novel **mixed-feature (MF) hash encoding** method to reduce memory usage and training time while preserving reconstruction quality. The proposed MF hash encoding maps multiple feature grids into a shared MF hash table, enabling the learning of mixed coarse and fine abstracted features among grids. As a consequence, MFNeRF significantly compresses the conventional multiresolution hash feature encoding, leading to a great reduction in model size and training time while maintaining high rendering quality (e.g., MFNeRF-Big outperforms InstantNGP-Big with a 47% smaller model size and a 13% reduction in training time, as shown in Tab. 1). To summarize, our contributions are as follows:

- We propose MFNeRF, a memory-efficient NeRF framework designed to reduce parameter size and training time while keeping high reconstruction quality. To achieve this, we propose a *mixed-feature hash encoding* method to adaptively mix partial feature grids into a single hash table.
- To ensure accurate feature vector retrieval from the shared hash table, we introduce an *index transformation* technique that standardizes grid indices from distinct grid spaces to a shared grid space.
- We conduct extensive experiments to benchmark MFNeRF against state-of-the-art (SOTA) methods, including TensoRF [3], DVGO [22], CCNeRF [26], FastLearning [32], FactorFields [4], MERF [21], and InstantNGP [18]. Our MFNeRF method achieves great training and rendering time reductions on the same GPU hardware while maintaining similar or higher rendering quality. For instance, MFNeRF-Big setup (hash table size of 2^{22}) achieves around 13% training time reduction, 9% rendering time reduction, and 42% memory reduction compared to InstantNGP-Big (same setup) on the same GPU, with an average rendering quality improvement of 0.10 in peak signal-to-noise ratio (PSNR) on the synthetic NeRF [17]. This translates to an 89% improvement in the figure of merit, defined as rendering quality (PSNR) * rendering speed (frame per second) / memory size (MB), as details shown in Fig. 1.

2. Related work

2.1. Neural Scene Representations

To represent 3D scenes, NeRF [17] encodes a scene into a couple of MLP networks, using multiple posed images. Inspired by NeRF, numerous works [1, 7, 12] have been proposed for novel view synthesis tasks. As NeRF struggles to learn scenes when provided with inaccurate pose information or impaired images, earlier works investigate approaches to train a clear scene from blurry images. For instance, Deblur-NeRF [14] optimizes a separate neural network that generates rays to simulate camera motion and defocus blurs, enabling NeRF to capture and represent deblurred scenes. Different from Deblur-NeRF, NAN [19] focuses on generating clean images with the same view from a burst of noisy images that have relatively small camera movements. Ev-NeRF [7] further shows that NeRF can be directly optimized using event camera data instead of the conventional posed RGB images.

2.2. Efficient Radiance Field Rendering Algorithms

More recently, to address the slow training and long rendering times of NeRF, recent works have explored the utilization of explicit representation for storing and accessing features to encode the shape and color information of scenes [6, 13, 15]. NSVF [13] employs an octree to store optimized scene features. This approach supports the gradual subdivision of voxels and the pruning of empty cells, allowing increased scene resolution and improved rendering quality. DVGO [22] adaptively increases the resolu-

tion of the feature grid during training. In contrast, KiloNeRF [20] uses a numerous amount of tiny MLPs instead of a single large MLP to dramatically reduce rendering time. SNeRG [6] and MERF [21] extract view-dependent and color features from a trained NeRF, bake them into a 3D texture atlas, and use a relatively small MLP to synthesize novel views with the baked features. RT-NeRF [11] performs fast rendering by back-projecting the feature grid onto the render target image plane instead of using ray marching. A recent seminal work, InstantNGP [18], maps multiresolution grids that encode a scene at various levels of abstraction onto a relatively small memory space using a hash function, allowing for the rapid learning of a plausible scene in seconds. Additionally, TensoRF [3] decomposes a deep feature grid into vectors and matrices, resulting in improved storage efficiency.

Recently, 3D Gaussian Splatting [8] (3DGS) has been proposed to represent scenes with 3D Gaussians that maintain the desirable characteristics of continuous volumetric radiance fields while minimizing computation in empty spaces. However, achieving the desired level of fidelity in the rendered scene necessitates propagating a colossal number of Gaussians during the training process, often resulting in substantial memory overhead, with real-world scenes demanding approximately 1 GB of 3D Gaussian data. Therefore, this work focuses on efficient NeRF research and leaves the comparison with 3DGS for future work.

3. Preliminary

3.1. Scene Representation of NeRF

NeRF [17] uses MLP networks to learn the radiance field of arbitrary scenes. After training on a set of images taken from different viewpoints, NeRF can estimate both the volume density σ and RGB color $\mathbf{c}$ of the radiance field as a continuous function. Thus, it is able to reconstruct a photorealistic view from an unseen pose using the differentiable volume rendering technique.

Specifically, given a camera pose and pixel coordinates, NeRF first generates a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ that marches from camera origin $\mathbf{o} \in \mathbb{R}^3$ in direction $\mathbf{d} \in [-\pi, \pi]^2$ to the pixel. Given each sampled point $\mathbf{x} \in \mathbb{R}^3$ on the ray, the MLP $F_\Phi : (\mathbf{x}, \mathbf{d}) \mapsto (\sigma, \mathbf{c})$ infers the density $\sigma \in \mathbb{R}$ of the point $\mathbf{x}$ and RGB color $\mathbf{c} \in \mathbb{R}^3$ corresponding to the ray direction $\mathbf{d}$. Taking into account the transparency α_i of each point $\mathbf{x}_i$, the final color $\hat{C}(\mathbf{r})$ for the ray (i.e., the color of the pixel) is determined by a weighted sum of the colors of the M sampled points along the ray $\mathbf{r}$, which can be formulated as:

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} \{ \prod_{j<i} (1 - \alpha_j) \} \alpha_i \mathbf{c}_i, \\ \alpha_i &= 1 - \exp(-\sigma_i \delta_i), \end{aligned} \tag{1}$$

where δ_i is the distance between successive sampled points x_i and x_{i+1}, and i and j indicate the order of the sampled points.

3.2. Multiresolution Hash Encoding

To enhance training efficiency without compromising reconstruction quality, prior NeRF methods [13,15,18,22] opt to encode the radiance field using feature grids with small MLP network instead of training a large MLP on the entire scene.

InstantNGP [18], as a representative work of this method, compresses multiple feature grids at different resolutions into dedicated hash tables to ensure smoothness and reduce uninformative grid points. Specifically, it utilizes two types of trainable parameters: encoding parameters Θ, stored in the hash tables, and weight parameters Φ for the NeRF Renderer. The encoding parameters are arranged into L hierarchical grids at distinct resolutions, where each grid point contains an F-dimensional feature vector. Furthermore, every feature vector for a given grid resides in its dedicated l-th hash table, which has a maximum capacity of T. The resolution ratio b between successive grids is defined as follows:

$$b = (N_{max}/N_{min})^{1/(L-1)}, \tag{2}$$

where $L > 1$, and N_{max} and N_{min} are the maximum resolution and minimum resolution of grids, respectively. Hence, the resolution of l-th grid is given by:

$$N_l = \lfloor N_{min} \cdot b^{l-1} \rfloor. \tag{3}$$

Given a sampled point $\mathbf{x}$ and a viewing direction $\mathbf{d}$, InstantNGP identifies the indices $\mathrm{I}^{\mathcal{G}_l} = (i, j, k)$ of the eight voxel corners surrounding $\mathbf{x}$ at each grid $\mathcal{G}_l$, resulting in a total of $8 \cdot L$ corner points identified for $\mathbf{x}$. Then, the spatial hash $h(\cdot)$ [27] is applied to each index $\mathrm{I}^{\mathcal{G}_l}$ to retrieve the associated feature vector from the dedicated hash table:

$$h(\mathrm{I}^{\mathcal{G}_l}) = (i \cdot \pi_1 \oplus j \cdot \pi_2 \oplus k \cdot \pi_3) \mod \mathbf{T}, \tag{4}$$

where $\oplus$ is the bit-wise XOR operator, $\mathbf{T}$ is the hash table size, and π_i denotes a prime number (i.e., $\pi_1 = 1$, $\pi_2 = 2,654,435,761$, and $\pi_3 = 805,459,861$). Subsequently, the eight F-dimensional feature vectors are trilinearly interpolated, resulting in $f_l \in \mathbb{R}^F$. Then, the L feature vectors are concatenated into $\mathbf{y} = [f_1; ...; f_L]$, which serves as input to the NeRF Renderer along with the viewing direction $\mathbf{d}$. The NeRF Renderer consists of a density MLP m_d and a color MLP m_c, parameterized by Φ_d and Φ_c, respectively (i.e., $\Phi = [\Phi_d; \Phi_c]$). The density σ and color $\mathbf{c}$ estimations can be expressed as:

$$\begin{aligned} f_c &= m_d(\mathbf{y}; \Phi_d), \\ \sigma &= f_c[0], \\ \mathbf{c} &= m_c(f_c, \mathbf{d}; \Phi_c), \end{aligned} \tag{5}$$

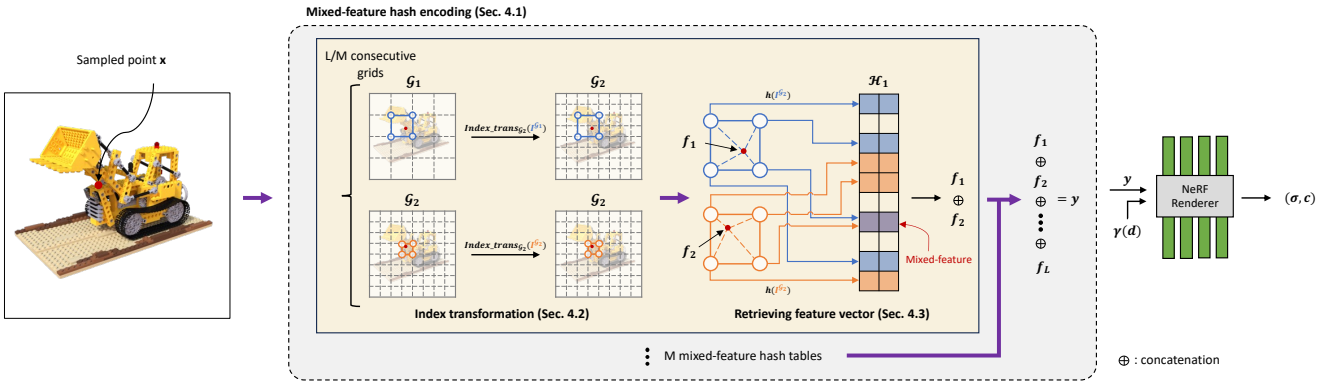


Figure 2. Overview of MFNeRF in 2D representation. Given a sampled point, MFNeRF employs mixed-feature hash encoding (Sec. 4.1) to share scene features across consecutive grids, allowing for mixing fine and coarse details. Indices from lower grids are remapped to the highest resolution grid in the grid group sharing the same MF hash table, allowing grids to reference identical hash entries for congruent scene coordinates, as exemplified by the lower-right corners of the orange and blue voxels in the figure. After interpolating feature vectors at the voxel corners, the NeRF Renderer processes these along with the positional embedded viewing direction $\gamma(\mathbf{d})$ to determine point $\mathbf{x}$'s density σ and color $\mathbf{c}$. Our method could be simply extended to three dimensions as well.

where f_c represents the color features produced by the density MLP m_d.

3.3. Limitations of Multiresolution Hash Encoding

Although multiresolution hash encoding enables faster learning compared to conventional NeRF [17], several limitations remain. First, it inefficiently uses multiple dense feature grids to represent vast empty spaces and occupied spaces. Due to the uniformity of the object's interior, voxels inside the object across different grids are prone to contain similar feature information. This also applies to empty spaces, resulting in many voxels across the grids having similar feature information. Second, the independent feature storage (i.e., separate hash tables) for each feature grid leads to inefficiencies, as features with possibly similar values are stored at different resolution levels for a point in multiple memory areas. Since each feature grid covers the same scene volume, they capture similar feature information for the same point. This tendency becomes stronger as the grids become adjacent. For example, the spatially same points $\mathbf{x}$ (although they have different indices) in $\mathcal{G}_l$ and in $\mathcal{G}_{l+1}$ would have nearly indistinguishable feature information because the consecutive grids capture similar levels of detail in the scene. Moreover, because the feature grids encode scene information in dedicated hash tables, the maximum hash table size T and feature resolution levels L crucially affect the learning capacity of InstantNGP. This indicates that higher values of T and L are essential for effectively reconstructing high-quality scenes with more intricate details. However, this results in a substantial number of encoding parameters, often orders of magnitude larger than the MLP's weight parameters. For instance, in the typical InstantNGP configuration with T set to 2^{19} and L set to 16, the number of encoding parameters reaches 12.6 M, which is **1,191**$\times$ larger than the weight parameters (10 k). This extensive memory usage due to large encoding parameters likely leads to a notable increase in computer's cache miss ratio, consequently resulting in significantly longer training time, especially when T exceeds the computer's cache size [18]. These limitations pose challenges for practical implementation, especially in real-world scenarios such as large-scale view synthesis or deployment on memory-limited edge devices.

4. MFNeRF

In this section, we present MFNeRF, a memory-efficient NeRF framework with a novel **mixed-feature hash table** design to reduce memory usage and training time while preserving high reconstruction quality.

As shown in the overview of MFNeRF in Fig. 2, we propose mixed-feature hash encoding to mix multiple grids into a shared hash table, enabling the learning of combined coarse and fine abstracted features in a memory-efficient manner (Sec 4.1). Within this novel design, we develop an **index transformation** method to convert indices from lower-level grids to their corresponding indices in higher-level grids (Sec 4.2). Once the transformed indices are obtained, the feature vectors in various resolution feature grids can be retrieved from the shared MF hash table (Sec 4.3). The detailed techniques are introduced in the following sections.

4.1. Mixed-Feature Hash Encoding

First of all, taking advantage of the multiresolution feature encoding, we construct L feature grids $\{\mathcal{G}_l\}_{l=1}^{L}$ with varying resolutions. Our goal is to map these L feature

grids onto a smaller number of hash tables. These tables are referred to as mixed-feature hash tables (MF hash tables). For instance, as shown in Fig. 2, the feature grids $\mathcal{G}_1$ and $\mathcal{G}_2$, which have different resolutions, are mapped to a shared hash table $\mathcal{H}_1$. In practice, we build M MF hash tables $\{\mathcal{H}_m\}_{m=1}^{M}$, where $M < L$. Each feature grid has a different resolution, with the finest and coarsest resolutions determined by N_{min} and N_{max}, respectively. The resolution of each next-level grid, relative to its consecutive previous-level grid, is scaled by the factor b as defined in Eq. (2).

Moreover, to ensure a balanced mapping of feature grids across MF hash tables, we assign an identical L/M number of grids to each MF hash table. However, randomly mapping feature grids to the same hash table can increase the likelihood of hash collisions, hindering the grids' ability to effectively learn the scene. To mitigate this issue, we map every set of L/M consecutive grids to one MF hash table. The reason behind this is that the adjacent grids are likely to capture a similar distribution of the scene. For example, when two grid points from different grids have the exact same spatial coordinates within the scene, the features at these points are highly similar and interchangeable. Consequently, a point on one grid can be accurately approximated by a point from an adjacent grid at a similar coordinate within the scene.

By doing so, our proposed mixed-feature hash encodes multiple feature grids into a shared hash table, reducing feature redundancy. For example, in a case with $L = 4$ and $M = 2$, the first two grids, $\mathcal{G}_1$ and $\mathcal{G}_2$, are assigned to the first MF hash table $\mathcal{H}_1$, whereas the last two grids, $\mathcal{G}_3$ and $\mathcal{G}_4$, are associated with the second MF hash table $\mathcal{H}_2$. As a consequence, MFNeRF allows similar features from different grids to be stored in the same entry of a shared mixed-feature hash table, leading to memory efficiency while preserving feature quality.

4.2. Index Transformation

To mix multiple grids with different levels into one hash table, we develop an index transformation method to transform the current index space of these grids to the index space of the finest grid for each hash table with the corresponding grid groups. Specifically, since voxel size varies depending on grid resolution, grid points at identical spatial positions in different grids might have distinct indices. For example, in Fig. 2, both the lower-right corners of the orange and blue boxes are located at the same position within the Lego scene. However, their indices are $(2, 2)$ in grid $\mathcal{G}_1$ and $(4, 4)$ in grid $\mathcal{G}_2$. When hashed, these indices map to different entries in the MF hash table despite representing the same spatial location. To address this, we employ an index transformation function $\mathit{Index_trans}_{\mathcal{G}_j}(\cdot)$ to translate indices in an arbitrary-level grid $\mathcal{G}_i$ into those in the j-th level grid $\mathcal{G}_j$ prior to hashing.

$$\begin{aligned} \mathit{Index_trans}_{\mathcal{G}_j}(\mathrm{I}^{\mathcal{G}_i}) &= \lfloor \mathrm{I}^{\mathcal{G}_i} \times N_j / N_i \rfloor \\ &= \mathrm{I}^{\mathcal{G}_j}, \end{aligned} \tag{6}$$

where $N_{\mathcal{G}_i}$ and $N_{\mathcal{G}_j}$ are the resolutions of i-th and j-th grids, respectively, calculated by Eq. (3). With this function, we obtain the indices in the finest grid among those sharing the same MF hash table. Using consistent grid space indices when accessing the MF hash table ensures that multiple grids reference the same feature for a specific location.

4.3. Retrieving Feature Vector

Once the transformed indices are obtained, they are hashed to find the entries in the defined hash table by using the spatial hash function described in Eq. (4). The resulting eight F-dimensional feature vectors undergo trilinear interpolation to produce $\{f_l\}_{l=1}^{L}$. These L feature vectors are concatenated as $\mathbf{y} = [f_1; ...; f_L]$ and then fed into the NeRF Renderer (Eq. (5)) alongside the viewing direction $\mathbf{d}$, yielding the density and color values of point $\mathbf{x}$. By doing so, the MF hash table consolidates partial grids within a single hash table, allowing the proposed MFNeRF to learn mixed features of coarse and fine abstraction, which leads to a significant reduction in feature size.

4.4. Training

The proposed MFNeRF is trained by using a set of posed images. Specifically, we jointly optimize both encoding and weight parameters by calculating the L2 loss between the rendered color $\hat{C}(\mathbf{r})$ and the ground truth color $C(\mathbf{r})$. The loss function is defined as:

$$\mathit{Loss} = \sum_{\mathbf{r} \in \mathcal{R}} || \hat{C}(\mathbf{r}) - C(\mathbf{r}) ||_2^2, \tag{7}$$

where $\mathcal{R}$ is a batch of camera rays randomly sampled from all the pixels in the train image set.

In summary, the whole training process can be illustrated as follows: given a sampled point $\mathbf{x}$ and a viewing direction $\mathbf{d}$, MFNeRF first obtains the voxel's eight corner indices $\{\{\mathrm{I}_k^{\mathcal{G}_l}\}_{k=1}^{8}\}_{l=1}^{L}$ at each grids. Before hashing the indices, they are transformed to the corresponding highest-resolution grid space by using the index transformation function. After this transformation, corresponding feature vectors are retrieved from the MF hash tables. The eight feature vectors $\{\mathrm{I}_k^{\mathcal{G}_l}\}_{k=1}^{8}$ corresponding to the voxel's eight corner points are then trilinearly interpolated, resulting in one feature vector f_l for l-th level grid. Then, L interpolated feature vectors are concatenated (i.e., $\mathbf{y} = [f_1; ...; f_L]$) and forwarded to the NeRF Renderer along with the viewing direction $\mathbf{d}$ (Eq. (5)).

Table 1. Quantitative comparison of SOTA methods on Synthetic NeRF [17] (averaged over 8 scenes). **Bold values** indicate the highest results, and underlined values indicate the second highest.

	MB↓	PSNR↑ (dB)	SSIM↑	Train↓ (sec)	FPS↑	FOM: PSNR*FPS/MB↑
TensoRF [3]	71.79	**33.20**	**0.963**	807.13	1.05	0.49
DVGOv1 [22]	209.18	31.93	0.957	588.82	2.03	0.31
DVGOv2 [23]	205.30	31.90	0.956	231.31	5.57	0.87
CCNeRF [26]	997.77	31.22	0.947	988.23	0.90	0.03
FactorFields [4]	**20.44**	33.09	0.961	628.81	0.78	1.26
FastLearning* [32]	54.98	30.66	0.949	**186.16**	25.33	14.13
InstantNGP-Base [18]	112.63	32.98	0.960	226.44	36.62	10.72
InstantNGP-Big [18]	300.09	33.01	0.960	330.57	29.42	3.24
MFNeRF-Base	74.91	33.03	0.960	220.19	**37.78**	**16.66**
MFNeRF-Big	173.63	33.11	0.961	289.03	32.10	6.12

* We used the authors' official code and hyperparameters but got different values than reported.

Table 2. Quantitative comparison of SOTA methods on MipNeRF360 [2] (averaged over 7 scenes). **Bold values** indicate the highest results, and underlined values indicate the second highest.

	MB↓	PSNR↑ (dB)	SSIM↑	Train↓ (sec)	FPS↑	FOM: PSNR*FPS/MB↑
DVGOv2 [23]	1640.53	25.42	0.694	737.29	0.58	0.01
MERF* [21]	242.29	26.35	**0.779**	12250.00**	**119.00*****	**12.94**
InstantNGP-Base [18]	170.27	26.70	0.742	474.13	4.61	0.72
InstantNGP-Big [18]	436.67	26.90	0.765	752.69	3.64	0.22
MFNeRF-Base	**122.57**	26.73	0.735	**466.87**	4.61	1.01
MFNeRF-Big	261.99	**26.97**	0.760	672.04	3.80	0.39

* We copied values of MERF from their paper by averaging over 7 scenes.
** They reported that they trained MERF for an average of 2 hours using eight V100 GPUs.
*** This FPS is the averaged value of outdoor scenes from MipNeRF360 dataset on an RTX 3090.

5. Experiments

5.1. Experimental Setup

We conduct experiments on the four popular 3D scene datasets: Synthetic NeRF [17], MipNeRF360 [2], Tanks&Temples [10], and BlendedMVS [30]. The Synthetic NeRF dataset includes well posed images for eight different synthetic scenes. MipNeRF360 dataset encompasses seven publicized, unbounded indoor/outdoor real scenes. We divide every 8th image into the test set and downsample images with resolutions ranging from 1k to 1.6k, following [2]. For details of other datasets, please refer to the supplementary material.

To evaluate performance, we used popular metrics: peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) [29], model size (MB), training duration in seconds, and rendering speed in frames per second (FPS).

We selected SOTA baselines for grid-based NeRF: DVGO (Python implementation [22] and CUDA implementation [23]), TensoRF [3], CCNeRF [26], FactorFields [4], FastLearning [32], MERF [21], and InstantNGP [18]. In the following paragraphs, we refer to InstantNGP with hash table sizes T of 2^{20} and 2^{22} as InstantNGP-Base and InstantNGP-Big, respectively.

Implementation details We adopt the Adam optimizer [9] and decay the learning rate from 2e-2 to 2e-4 using cosine decay function. All synthetic scenes are trained with the same hyperparameters. We empirically select the number of channels in the color MLP's hidden layer as 128. For unbounded scenes from MipNeRF360 [2], we use scene-specific settings. We train MFNeRF for 20k iterations on both synthetic and unbounded scenes using a single NVIDIA RTX A6000 GPU. We implement the mixed-feature hash encoding upon the Tiny CUDA neural networks code base [1] and develop training pipeline upon the Python implementation [2] of InstantNGP [18]. For detailed configurations, please refer to the supplementary material. The code is available at https://github.com/nfyfamr/MFNeRF.

Based on our ablation studies, we set the number of mixed-feature hash tables to $M = 8$ as the default for comparison with SOTA methods. Similar to InstantNGP, we also use the suffixes -Base and -Big to indicate the proposed MFNeRF with maximum hash table size of $T = 2^{20}$ and $T = 2^{22}$, respectively.

[1] https://github.com/NVlabs/tiny-cuda-nn
[2] https://github.com/kwea123/ngp_pl

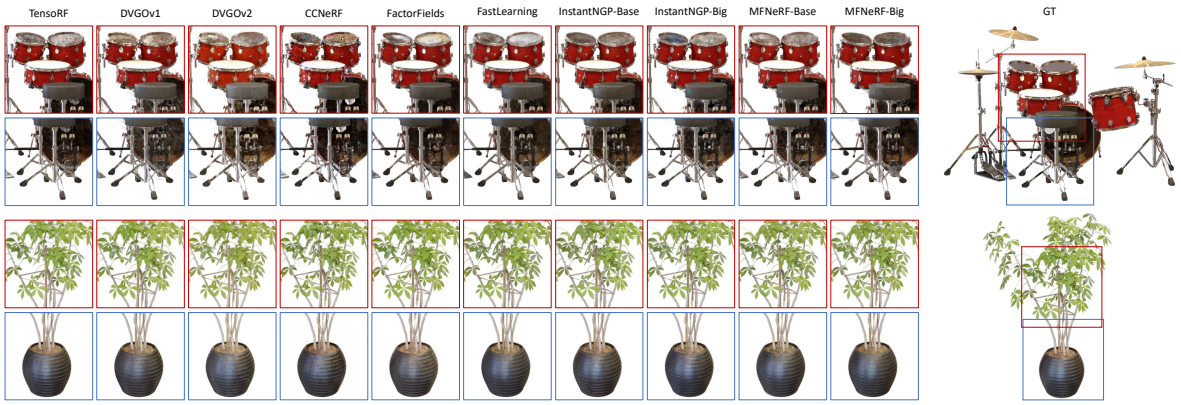


Figure 3. Qualitative comparison on drums and ficus from Synthetic NeRF [17].

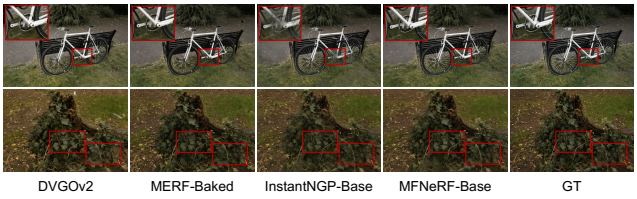


Figure 4. Qualitative result on bicycle and stump scenes from MipNeRF360 [2].

5.2. Experimental Results

To evaluate the performance of our proposed MFNeRF, we first conduct experiments on the Synthetic NeRF [17] in comparison to SOTA neural network-based NeRF methods and summarize the results in Tab. 1. First of all, compared to TensoRF which owns highest PSNR, MFNeRF-Base achieves $\sim$ 3.7$\times$ training time reduction and remarkably, $\sim$ 36$\times$ FPS improvement with little PSNR degradation. Compared to both DVGOv1 and DVGOv2, MFNeRF-Base ($T = 2^{20}$) and MFNeRF-Big ($T = 2^{22}$) learn the scenes similar of faster and exhibit higher average PSNR and SSIM. In addition, compared to the more related InstantNGP, MFNeRF-Base and MFNeRF-Big consistently achieve better PSNR with less memory, training time and faster rendering compared to InstantNGP-Base and InstantNGP-Big, respectively. More importantly, compared to all these prior NeRF methods, the MFNeRF-Base achieves the best figure of merit (FOM) defined in terms of PSNR*FPS/MB.

We also conduct the experiments on the more challenging MipNeRF360 [2] scenes. Due to the unavailability of source codes or hyperparameters for some baseline methods, we mainly compare the proposed MFNeRF with DVGOv2, MERF, and InstantNGP. As the results shown in Tab. 2, MFNeRF-Base and MFNeRF-Big surpass DVGOv2, exhibiting average PSNR improvements by $1.31/1.55$, respectively, alongside training time reductions of $36.68\%/8.85\%$. Although MFNeRF-Base achieves a higher PSNR by 0.38 with a model size roughly 49% smaller, it scored lower than MERF in terms of FOM due to FPS. Note that MERF's fast FPS primarily results from the use of baking and deferred rendering techniques, which are orthogonal method to our memory-efficient feature grid architecture. Additionally, MFNeRF's substantially faster training speed ($\sim$ 26$\times$) makes it more preferable for real-world applications than MERF. In comparisons with InstantNGP of T counterpart, MFNeRF-Base and MFNeRF-Big consistently showcase superior reconstruction, with PSNR enhancements by $0.03/0.07$, and training time savings of $7.2s/80.6s$, respectively.

Fig. 3 and Fig. 4 present qualitative comparisons between various NeRF methods on test images. When examining the drum's surface and legs, it is noticeable that MFNeRF produces smoother and more accurate shading compared to InstantNGP. Furthermore, MFNeRF renders multiple leaves more effectively, especially when compared to CCNeRF and FactorFields, which displays unusual specular light on edges and surfaces. In the bike scene, our method avoids the foggy appearance observed in other methods, as evidenced in the bike gear and leaves on stumps. Additional rendering results on the other scenes are included in the supplementary material.

5.3. Ablation Study

The effectiveness of the number of shared hash tables While sharing a single hash table among all grids (i.e., $M = 1$) can significantly reduce memory usage, this approach may also impair learning performance due to an excessively constrained memory space. To assess the effectiveness of varying the number of shared hash tables M, we conducted experiments by doubling M from 1 to 8, while

Table 3. Ablation study of MFNeRF-Base with different numbers of mixed-feature hash tables M on Synthetic NeRF [17] (averaged over 8 scenes). Note that $M = 16$ is equivalent to InstantNGP-Base.

M	MB↓	PSNR↑ (dB)	SSIM↑	LPIPS↓	Train↓ (sec)	FPS↑
1	40.35	32.70	0.957	0.056	197.69	41.19
2	48.35	32.72	0.958	0.054	209.83	40.70
4	56.73	32.78	0.958	0.053	210.47	39.27
8	74.91	33.03	0.960	0.051	220.19	37.78
16	112.63	32.98	0.960	0.050	226.44	36.62

Table 4. Ablation study of MFNeRF with different maximum hash table sizes T on Synthetic NeRF [17] (averaged over 8 scenes).

T	MB↓	PSNR↑ (dB)	SSIM↑	LPIPS↓	Train↓ (sec)	FPS↑
2^{20}	74.91	33.03	0.960	0.051	220.19	37.78
2^{21}	109.63	33.02	0.960	0.049	248.66	35.19
2^{22}	173.63	33.11	0.961	0.048	289.03	32.10
2^{23}	294.20	33.16	0.961	0.047	351.40	27.93

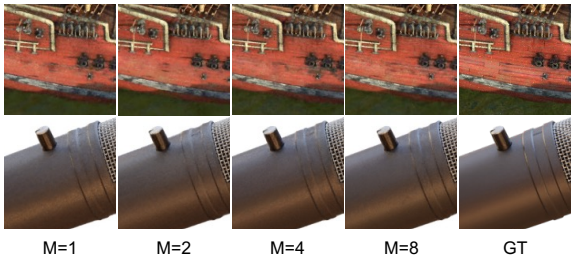


Figure 5. Rendering results with different values of M for the ship and mic scenes demonstrate that fewer tables capture less detailed scene information. We fixed $L = 16$ and $T = 2^{20}$ for all variants.

keeping the maximum hash table size $T = 2^{20}$ and number of grids $L = 16$ constant on the Synthetic NeRF [17]. As shown in Tab. 3, we observe that with an increase in M, there is a corresponding increase in model size along with improvements in PSNR, SSIM, LPIPS, and fps. As anticipated, a larger model size results in longer training and rendering times, indicating that a greater number of parameters can be a bottleneck for training and rendering efficiency. Additionally, we find an improvement in quality with an increased number of hash tables M, as demonstrated in Fig. 5. Particularly, when the scene is extremely compressed into just one hash table (i.e., $M = 1$), MFNeRF surprisingly still generates reasonable quality. This observation supports our assumption, as stated in Section 4.1, that consecutive feature grids can share highly similar features.

The effectiveness of different hash table size We investigate how our mixed-feature hash encoding method performs with different maximum hash table sizes. To do this, we set $L = 16$ and $M = 8$, and varied T from 2^{20} to 2^{23}. As shown in Tab. 4, when sufficient trainable parameters are provided, the reconstruction quality improves in terms of all metrics, including PSNR, SSIM, and LPIPS. Similar to our previous ablation study regarding the number of hash tables M, we also observe that the training duration becomes longer and the fps decrease as the model size increases. This shows the issue that accessing a large memory area could be a potential memory bottleneck.

6. Conclusion

In this paper, we propose a novel framework for neural radiance field (NeRF) that employs a mixed-feature hash table to improve memory efficiency while maintaining reconstruction quality. To achieve that, we first propose a mixed-feature hash encoding method to map feature grids with different resolutions to shared hash tables. Moreover, to ensure accurate feature storage and retrieval, we design an index transformation technique that standardizes grid indices across different feature grids. By sharing the hash table, our method saves memory space for storing scene features and enhances the training speed by preventing multiple feature grids from redundantly learning similar features. In the experiment, we observe that our proposed method achieves significant reduction of training time and higher rendering quality with fewer trainable parameters compared to the prior InstantNGP method.

Limitation While MFNeRF succeeds to reduce memory usage and maintain high rendering quality, it has limitations. The compact MLPs are effective for handling simple view-dependent lighting but may struggle with complex lighting effects. It also fails to resolve shape-radiance ambiguity, leading to challenges in learning accurate density distributions, particularly on texture-less, bumpy surfaces. Additionally, the limited resolution of the feature grid restricts the encoding of fine details. Lastly, MFNeRF's reliance on volume rendering demands substantial GPU resources, making it less preferable to run the model on low-resource devices such as mobile VR.

Acknowledgments

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) contract number 140D0423C0076. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

References

[1] J T Barron, B Mildenhall, M Tancik, P Hedman, R Martin-Brualla, and P P Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, Los Alamitos, CA, USA, 10 2021. IEEE Computer Society. 2

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, New Orleans, LA, USA, 6 2022. IEEE. 6, 7

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 333–350, Cham, 2022. Springer Nature Switzerland. 2, 3, 6

[4] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Dictionary Fields: Learning a Neural Basis Decomposition. *ACM Trans. Graph.*, 42(4), 7 2023. 2, 6

[5] Donald Hearn. *Computer graphics, C version*. Pearson Education India, 1997. 1

[6] 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 (ICCV)*, pages 5875–5884, 10 2021. 1, 2, 3

[7] Inwoo Hwang, Junho Kim, and Young Min Kim. Ev-NeRF: Event Based Neural Radiance Field. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 837–847. IEEE, 2023. 2

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4):1–14, 7 2023. 3

[9] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv, 2014. 6

[10] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):1–13, 7 2017. 6

[11] Chaojian Li, Sixu Li, Yang Zhao, Wenbo Zhu, and Yingyan Lin. RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '22, New York, NY, USA, 2022. Association for Computing Machinery. 3

[12] C Lin, W Ma, A Torralba, and S Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5721–5731, Los Alamitos, CA, USA, 10 2021. IEEE Computer Society. 2

[13] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15651–15663. Curran Associates, Inc., 2020. 1, 2, 3

[14] L Ma, X Li, J Liao, Q Zhang, X Wang, J Wang, and P V Sander. Deblur-NeRF: Neural Radiance Fields from Blurry Images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12851–12860, Los Alamitos, CA, USA, 6 2022. IEEE Computer Society. 2

[15] Julien N P Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive Coordinate Networks for Neural Scene Representation. *ACM Trans. Graph.*, 40(4), 7 2021. 2, 3

[16] R Martin-Brualla, N Radwan, M M Sajjadi, J T Barron, A Dosovitskiy, and D Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7206–7215, Los Alamitos, CA, USA, 6 2021. IEEE Computer Society. 1

[17] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 405–421, Cham, 2020. Springer International Publishing. 1, 2, 3, 4, 6, 7, 8

[18] 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, 7 2022. 2, 3, 4, 6

[19] N Pearl, T Treibitz, and S Korman. NAN: Noise-Aware NeRFs for Burst-Denoising. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12662–12671, Los Alamitos, CA, USA, 6 2022. IEEE Computer Society. 2

[20] C Reiser, S Peng, Y Liao, and A Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14315–14325, Los Alamitos, CA, USA, 10 2021. IEEE Computer Society. 3

[21] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-Efficient Radiance Fields for Real-Time View Synthesis in Unbounded Scenes. *ACM Trans. Graph.*, 42(4), 7 2023. 2, 3, 6

[22] C Sun, M Sun, and H Chen. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5449–5459, Los Alamitos, CA, USA, 6 2022. IEEE Computer Society. 2, 3, 6

[23] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved Direct Voxel Grid Optimization for Radiance Fields Reconstruction. arXiv, 2022. 6

[24] T Takikawa, J Litalien, K Yin, K Kreis, C Loop, D Nowrouzezahrai, A Jacobson, M McGuire, and S Fidler. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11353–

11362, Los Alamitos, CA, USA, 6 2021. IEEE Computer Society. 1, 2

[25] M Tancik, V Casser, X Yan, S Pradhan, B P Mildenhall, P Srinivasan, J T Barron, and H Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8238–8248, Los Alamitos, CA, USA, 6 2022. IEEE Computer Society. 2

[26] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable NeRF via Rank-residual Decomposition. In S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, and A Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 14798–14809. Curran Associates, Inc., 2022. 2, 6

[27] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, and Markus Gross. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *VMV'03: Proceedings of the Vision, Modeling, Visualization*, volume 3, pages 47–54, Munich, 2003. 3

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I Guyon, U Von Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30 of *NIPS'17*, page 6000–6010. Curran Associates, Inc., 2017. 1

[29] Zhou Wang, A C Bovik, H R Sheikh, and E P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 4 2004. 6

[30] Y Yao, Z Luo, S Li, J Zhang, Y Ren, L Zhou, T Fang, and L Quan. BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1787–1796, Los Alamitos, CA, USA, 6 2020. IEEE Computer Society. 6

[31] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5732–5741. IEEE Computer Society, 10 2021. 1, 2

[32] Wenyuan Zhang, Ruofan Xing, Yunfan Zeng, Yu-Shen Liu, Kanle Shi, and Zhizhong Han. Fast Learning Radiance Fields by Shooting Much Fewer Rays. *IEEE Transactions on Image Processing*, 32:2703–2718, 5 2023. 2, 6