Plenoptic PNG: Real-Time Neural Radiance Fields in 150 KB

Jae Yong Lee *[1] Yuqun Wu[1] Chuhang Zou[2] Derek Hoiem[1] Shenlong Wang[1]
{lee896, yuqunwu2, dhoiem, shenlong}@illinois.edu zouchuha@amazon.com

[1]University of Illinois at Urbana-Champaign [2]Amazon Inc.

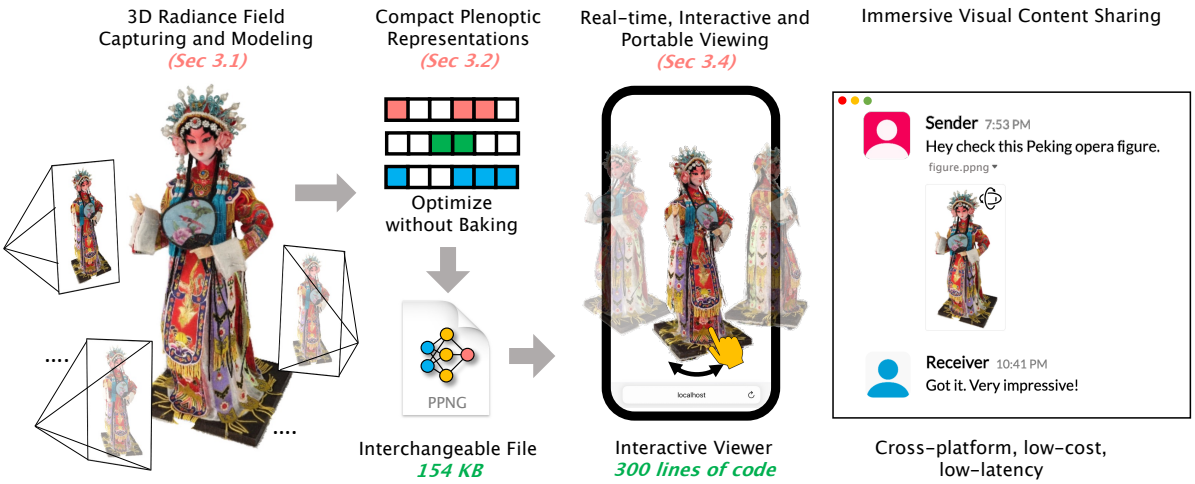


Figure 1. Given multi-view images as input, Plenoptic PNG (PPNG) encodes the parameters of the free-viewpoint radiance field into a compact, interchangeable file as small as 154 KB in just a few minutes. On the user side, the PPNG file can be decoded into a WebGL-compatible shader and 3D texture within 100 ms, and renders on lightweight devices in real-time. PPNG opens up potential new applications, allowing for easier capture and sharing of photo-realistic visuals across platforms.

Abstract

The goal of this paper is to encode a 3D scene into an extremely compact representation from 2D images and to enable its transmittance, decoding and rendering in real-time across various platforms. Despite the progress in NeRFs and Gaussian Splats, their large model size and specialized renderers make it challenging to distribute free-viewpoint 3D content as easily as images. To address this, we have designed a novel 3D representation that encodes the plenoptic function into sinusoidal function indexed dense volumes. This approach facilitates feature sharing across different locations, improving compactness over traditional spatial voxels. The memory footprint of the dense 3D feature grid can be further reduced using spatial decomposition techniques. This design combines the strengths of spatial hashing functions and voxel decomposition, resulting in a model size as small as 150 KB for each 3D scene. Moreover, PPNG features a lightweight rendering pipeline with only 300 lines of code that decodes its representation into standard GL textures and fragment shaders. This enables real-time rendering using the traditional GL pipeline, ensuring universal compatibility and efficiency across various platforms without additional dependencies. Our results are available at: https://jyl.kr/ppng

1. Introduction

Capturing and viewing immersive content has become easier than ever. Recent progress in approaches like Neural Radiance Fields (NeRF) and Gaussian Splatting have enabled users to capture 3D content from mobile devices. The commercialization of XR devices, such as the Apple Vision Pro and Oculus, has enhanced the viewing experiences of photorealistic immersive visual content. However, challenges persist in efficiently storing, transmitting, and browsing this content across various devices. In this work, we pursue a novel approach to model and encode free-viewpoint 3D content into a file as small as a PNG photo, making viewing and interacting with it as easy as browsing videos on various devices like mobile phones, laptops, or AR/VR glasses.

There are three key desiderata to democratize immersive, photorealistic 3D content. First, the model size must be small to avoid degrading user experience in instant messaging and web browsing. Second, viewing and interaction should be universal, not relying on specialized dependencies or hardware. Third, rendering and interaction must be smooth and real-time. Despite significant progress in this field, current approaches fail to meet all these criteria. For instance, NeRFs and their variants may be compact, but many are not real-time and depend on specialized pack-

ages such as CUDA-based neural volume renderers. Explicit methods like Gaussian Splats and NeRF baking are fast and versatile, but explicit geometry requires substantial storage. Table 1 summarizes these methods and their alignment with the desired criteria.

To achieve this, we propose Plenoptic Portable Neural Graphics, a novel framework that is highly compact and fast to render and train, providing a free-view portable network graphics object. At the core of our framework is a novel, compact scene representation and a real-time, cross-platform GL-compatible render. Unlike coordinate-based MLPs [2, 3, 35] or spatial voxel grids [8, 37, 56], our method leverages an explicit 3D voxel feature grid indexed by the sinusoidal encoding of the spatial coordinate. This new spectrally indexed volume enables feature sharing across different spatial locations, improving compactness over spatial voxels. The model size of this dense 3D voxel feature grid can be further reduced through tensor-rank decomposition. Consequently, this representation design combines the best aspects of spatial hashing function [37] and voxel decomposition [8] approaches, resulting in a size as small as 154 KB. Additionally, we develop a novel, lightweight, and real-time rendering pipeline that can decode Plenoptic PNG representation instantly into standard GL textures and shaders, and render with OpenGL pipeline, making it universally viewable on any platform without additional dependencies.

Our experiments demonstrate that Plenoptic PNG surpasses baselines with a significantly reduced model size, as small as 154 KB — 100 times smaller than previous memory-efficient methods. We also show that Plenoptic PNG achieves the best balance between training speed, rendering quality, and model size among all real-time Web-ready NeRF methods, producing a widely accessible, realistic, and efficient interchangeable file format for immersive 3D media. We invite the reader to view our project page in the supplementary file, where we render 8 neural scenes simultaneously in real-time on a webpage at 1.2 MB in total. Our key contributions are:

- We present a novel neural scene model that encodes multiple views into an extremely compact tensor representation indexed by Fourier encoding, showing significant model size reduction compared to prior work.
- We develop a lightweight rendering pipeline that can instantly decode the Plenoptic PNG representation into standard GL textures and shaders, and render it in real-time in WebGL, making it viewable and interactable on any platform.

2. Related works

Our goal is to encode multiple 2D images of a 3D scene into an extremely compact representation that can be rendered from custom viewpoints in real-time across various platforms. Our method relates most closely to real-time neural radiance field methods, and we draw inspiration from 3D and neural compression.

Table 1. **Comparison of various NeRF methods.** Previous implicit neural representations tend to have a smaller memory footprint but suffer from relatively lower speed and are incompatible with web renderers. Explicit approaches enjoy real-time speed and GL compatibility but require a large model size. Our method is the first of its kind to achieve a kilobyte-level model size and satisfies all the criteria.

Method	Real-time (FPS: > 30 Hz)	Web-Ready (GL native)	Memory-Efficient (Model Size: < 5 MB)	Fast Training (Time: < 15 min)
NeRF	✗	✗	5 MB	hours
Plenoxel	✓	✓	778 MB	11.4 min
Plenoctree	✓	✓	1976 MB	hours
DIVeR	✓	✗	67.8 MB	hours
SNeRG	✓	✓	86.8 MB	hours
TensoRF(CP)	✗	✗	3.9 MB	25.2 min
Wavelet-NeRF (4-DWT)	✗	✓	710 KB	23 min
Instant NGP	✓	✗	32 MB	5 min
Compact-NGP	✓	✓	357 KB	12 min
VQRF	✗	✗	1.4 MB	8 min
MobileNeRF	✓	✓	125.8 MB	hours
BakedSDF	✓	✓	382 MB	hours
MERF	✓	✓	120 MB	hours
Re-Rend	✓	✓	199 MB	hours
Gauss. Spl.	✓	✓	67.3 MB	7.6 min
PPNG-3 (Ours)	✓	✓	32.7 MB	5 min
PPNG-2 (Ours)	✓	✓	2.5MB	10 min
PPNG-1 (Ours)	✓	✓	151 KB	13 min

Real-time Neural Radiance Field (NeRF). NeRF [35] has emerged as one of the most promising and widely adopted novel view synthesis methods. NeRF represents the 3D scene with coordinate-based multi-layer perceptrons (MLPs) and achieves high-quality rendering through volume rendering. Despite its compactness, the original NeRF suffers from slow training and rendering. Feature volume-based approaches [8, 26, 37, 56] encode the scene with a dense feature grid, which leads to faster rendering and training. Rendering speed can be further accelerated using sparse volumetric data structures [17, 32, 34, 44, 53, 57]. Additionally, methods like those in [28, 52] optimize the volumetric sampling process to increase rendering speed during inference. However, most neural volume rendering methods still require a high-capacity GPU and specialized volume renderer, which limits their applicability.

An appealing alternative is jointly learning explicit geometry, such as points and meshes, and baking appearance features like opacity and view-dependent color onto the geometric surfaces [11, 21, 39, 41, 55]. Such approaches often align with real-time graphics pipelines like OpenGL, making them accessible across various devices. However, most explicit geometry-based neural rendering methods suffer from memory inefficiency and slow training speeds. Very recently, Gaussian splatting [26] has emerged as an exception, striking the best balance between speed, quality, and training speed. Nevertheless, the file sizes for Gaussian Splatting scenes still range from tens to hundreds of MBs.

Volume Compression. Compressing volumetric 3D data has long been a challenge in graphics. The key lies in designing both a compact 3D representation and an expressive encoding. Numerous approaches explore efficient data structures like octrees coupled with entropy coding [18, 24, 25, 42] to reduce redundancies in 3D data.

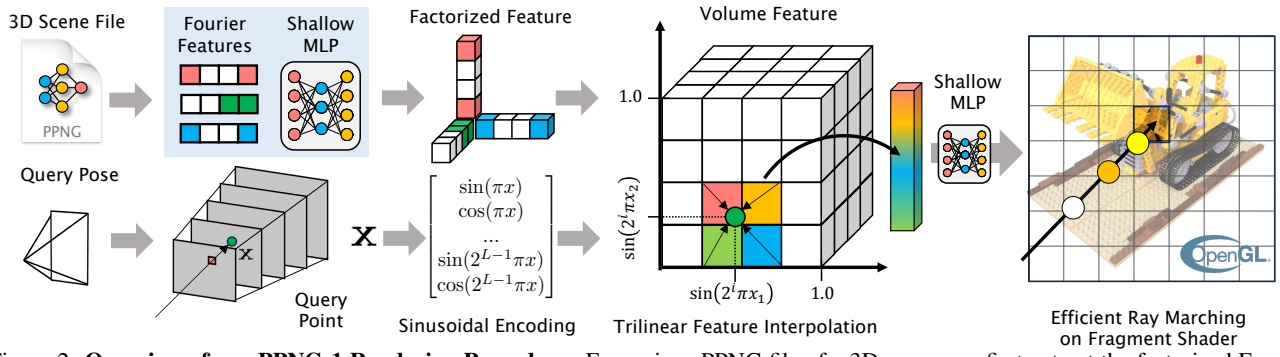


Figure 2. **Overview of our PPNG-1 Rendering Procedure:** For a given PPNG file of a 3D scene, we first extract the factorized Fourier features and the shallow MLP weights (top-left). The factorized Fourier features are then composed to construct a dense Fourier-indexed feature grid (middle). In the rendering stage, for each query point we compute the sinusoidal positional encoding to extract the corresponding feature from the Fourier-indexed voxel grid. The feature vectors, spanning across the spectrum for both sine and cosine at each frequency, are then concatenated. These features are subsequently passed into the fragment shader, which employs a shallow MLP for inferring color and density and applies ray matching to determine the final pixel color.

Block-based coding schemes, commonly used in volumetric compression [1, 12], can be further optimized through data filtering [22, 23]. Recent works also leverage various tools, such as the Karhunen-Loève transform [49], autoencoders [5, 50, 51], and wavelet transform [14], to compress nodes in tree structures. While these compressed explicit representations suit traditional graphics, they fall short in rendering photorealistic images from free viewpoints. With Gaussian Splatting [26] emerging as an alternative to photorealistic representation, some of the most recent works explore anchor / hash-grid based compression [10] and distributing smaller number of samples [16]. These Gaussian Splatting variants show impressive compression ratio, yet still use an order of magnitude larger model size compared to the implicit models.

Neural Field Compression. As implicit representations [35, 38] show promise in graphics, many works aim to reduce memory footprint while maintaining high accuracy [4, 19, 33]. Inspired by pioneering light field work [29], real-time light-field compression approaches distills a compact representation from NeRF and achieved reduced memory footage and real-time rendering on mobile devices [7, 20]. However, this process incurs high training costs and cannot yet achieve KB-level compression. Drawing on its widespread use in graphics, NGLoD [46] learns a sparse octree with continuous levels of detail (LOD). Various approaches using learning and handcrafted codebooks have also been proposed to compress neural fields, such as wavelets [40], vector-quantized feature codebooks [19, 30, 33], learning-based feature indexing mapping [45, 47], multi-scale look-up tables [13], adaptable rank [58] binarization [43], and neural image compression [31]. While showing promising results, they often require additional memory for storing codebooks, use extra decoding time or takes additional time to optimize / render. Moreover, despite their small size, such compressed neural field representations usually need specialized CUDA-dependent renderers and decoders [37, 47] or decodes into large size [8, 40] (i.e, when fully composed to dense voxel grid of size $O(500^3)$ to avoid feature composition for faster rendering) which makes it hard to run on light-weight devices such as mobile phones. Instead, our representation can render in generic graphics library such as WebGL, and decodes into reasonably small size of 32.7 MB per scene. Our scene representation can be seen as a special instance of recent Dictionary Fields [9], which unify various representations, including vanilla NeRF, NGP, and TensoRF, by encompassing permutations of coordinate, basis/coefficient, and activation representation. Unlike Dictionary Fields' broad unification, our work focuses on solving real-time rendering with minimal data transfer. We achieve this with a dual representation design choice: 1) encoding NeRF into a compact 1D/2D factorized tensor for efficient data transfer, and 2) decoding it into a GL-compatible 3D feature grid for real-time web rendering.

To summarize, Table 1 presents the capabilities of current NeRF-based approaches in terms of model size, training speed, rendering speed, and web compatibility. While each has its strengths, our approach meets all requirements for wide application in daily use.

3. Plenoptic Portable Neural Graphics

The goal of PPNG is to encode a set of images with known poses into a small model that can be efficiently transmitted and rendered on ubiquitous devices. To achieve this, we first present a novel, compact neural representation that uses a sinusoidal function encoded feature volume (Section 3.1). Unlike spatial coordinate indexed volumes, this approach allows for feature sharing by design, thus requiring fewer parameters to achieve the same capacity. We demonstrate that our Fourier-indexed volume can be factorized into low-rank tensor approximations to further reduce the model size (Section 3.2). We implement a fast training scheme in *tiny-cuda-nn*, utilizing a voxel-based density cache [37] (Sec-

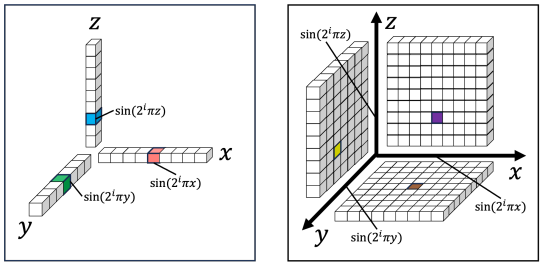


Figure 3. **Visualization of Two Factorized Plenoptic PNG Representations:** PPNG-1 (Equation 4) utilizes tensor-rank decomposition (left), while PPNG-2 (Equation 5) employs tri-plane decomposition (right).

tion 3.3), and design a new GLSL-based renderer to decode the parameters for real-time rendering (Section 3.4). Figure 2 shows an overview of the representation and rendering pipeline.

3.1. Plenoptic Portable Neural Graphics

Our core contribution lies in a novel volumetric neural feature representation referred to as plenoptic portable neural graphics. The goal of this learnable representation is to approximate the mapping from coordinates $\mathbf{p} \in \mathbb{R}^3$ to color and opacity values $(\mathbf{c}, \sigma) \in \mathbb{R}^{3+1}$. We develop an efficient and compact representation that leverages: (1) positional encoding to convert the spatial coordinate into a multi-scale, multi-dimensional Fourier embedding; and (2) volume-based feature queries to enable fast inference.

Given the input query point $\mathbf{p} = (x, y, z)$, positional encoding [35, 48] is applied to map the Euclidean coordinates input to sinusoidal activations across L different frequency levels for each axis:

$$\gamma(\mathbf{p}) = \text{concat}([\gamma(x), \gamma(y), \gamma(z)]) \in \mathbb{R}^{3\times L\times 2}, \quad (1)$$

where activation for each axis is

$$\gamma(w) = [(\sin(f_i \pi w), \cos(f_i \pi w))]_{i=0}^{L-1} \in \mathbb{R}^{L\times 2},$$

This resulting encoding is a continuous, multi-scale, periodic representation of $\mathbf{p}$ along each coordinate.

We maintain $L \times 2$ feature volume cubes, $\{\mathbf{V}_i^{\sin}, \mathbf{V}_i^{\cos} \in \mathbb{R}^{Q^3\times D}\}_i^{L\times 2}$, each with a resolution of Q^3 and D-dimensional features per entry. These features are indexed by a 3D slice of the positional encoding $\gamma_i^{\sin} = [\sin(f_i\pi x), \sin(f_i\pi y), \sin(f_i\pi z)]$ (or cosine embedding $\gamma_i^{\cos}$) at corresponding frequency.

We query the feature vector $\mathbf{z}_i^{\sin}, \mathbf{z}_i^{\cos}$ from each volume across each frequency as:

$$\forall i: \ \mathbf{z}_i^{\sin} = \pi_{\text{tri}}(\gamma_i^{\sin}, \mathbf{V}_i^{\sin}), \ \mathbf{z}_i^{\cos} = \pi_{\text{tri}}(\gamma_i^{\cos}, \mathbf{V}_i^{\cos}) \quad (2)$$

where π_{tri} is tri-linear interpolation. We then concat all the features $\mathbf{z}(\mathbf{p}) = \text{concat}[...\mathbf{z}_i^{\sin}, \mathbf{z}_i^{\cos}...]_{i=0}^{L} - 1$, which result

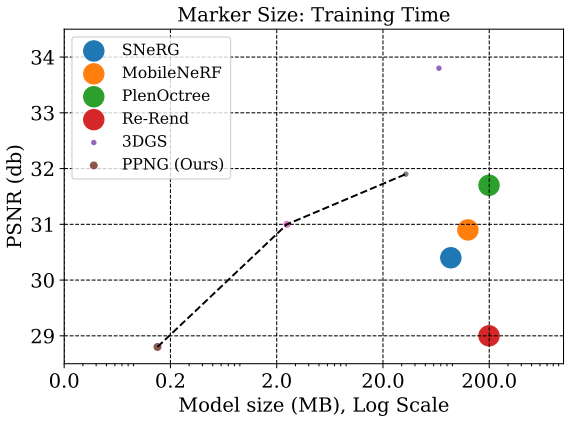


Figure 4. **Quantitative Comparison with Real-Time, Web-Compatible NeRF Models on NeRF Synthetic dataset**. Our approaches are **2-3 orders of magnitude** smaller than baselines in terms of model size (x-axis) and over **10x-100x** faster in training speed (marker size), while maintaining competitive PSNR (y-axis).

in a feature vector of length $F = 2 \times L \times D$. Finally, this feature vector is passed into a shallow MLP $g_{\boldsymbol{\theta}}$ with encoded viewing direction $\mathbf{d}$ to regress opacity and view-dependent color:

$$(\mathbf{c}, \sigma) = g_{\boldsymbol{\theta}}(\mathbf{z}(\mathbf{p}), \mathbf{d}) \quad (3)$$

Despite its simplicity, our design offers multiple benefits. *Efficiency:* Similar to volume-based neural fields, our approach is extremely efficient, enabling real-time rendering. *Feature sharing:* Features at each voxel location in the Fourier domain are shared and accessed simultaneously by multiple spatial locations. This design allows us to extract redundancies over space (compared to spatial volumes) while maintaining smoothness (unlike spatial hashing). *Compactness:* The total model size is $L \times 2 \times Q^3 \times D + |\boldsymbol{\theta}|$, where $|\boldsymbol{\theta}|$ represents the number of parameters for the shallow MLP. In practice, given the redundancies and the multi-scale nature of each volume, the size of each volume Q can be significantly smaller than the typical size for spatial volumes S (e.g. 80 vs 512), resulting in a smaller overall model size.

3.2. Factorized Plenoptic PNG

The vanilla Plenoptic PNG (denoted as PPNG-3) stores a dense 3D volume $\mathbf{V}_i^{\sin} \in \mathbb{R}^{Q^3\times D}$ for each frequency, with memory complexity being $Q^3 \times D$. We note that this Fourier-indexed feature representation is axis-aligned and smooth in its index coordinates (sinusoidal space). Inspired by the success of tensorial factorization in spatial fields [8], we propose leveraging low-rank tensor decomposition techniques to further compress Plenoptic PNG.

Our first approach, PPNG-1, inspired by CP-decomposition [8], decomposes 3D volumes into a set of triplets of 1D vectors $(\mathbf{v}_x^r, \mathbf{v}_z^r, \mathbf{v}_y^r)$. Specifically, for each $\mathbf{V}_i$ (omitting sin and cos superscripts for simplicity), we

approximate them using the following equation, instead of directly storing the 3D volume:

$$\mathbf{V}_i = \sum_{r}^{R} \mathbf{v}_{i,x}^r \otimes \mathbf{v}_{i,y}^r \otimes \mathbf{v}_{i,z}^r \text{ (PPNG-1)}, \tag{4}$$

where $\otimes$ is the outer product, R is the total number of triplet components. PPNG-1 has a memory complexity being $Q \times 3 \times R \times D$ for each volume and offers the most compressed representation among all variants with some trade-off in training speed and quality.

Our second approach, PPNG-2, incorporates tri-plane decomposition to approximate 3D volumes into a set of triplets of 2D feature planes $(\mathbf{v}_{xy}, \mathbf{v}_{xz}, \mathbf{v}_{yz})$:

$$\mathbf{V}_i = \sum_{r}^{R} \mathbf{v}_{i,xy}^r \otimes \mathbf{v}_{i,xz}^r \otimes \mathbf{v}_{i,yz}^r \text{ (PPNG-2)}, \tag{5}$$

PPNG-2's memory complexity is $Q^2 \times 3 \times R \times D$ for each volume. It offers a good balance between quality and model size, positioned between PPNG-1 and PPNG-3.

During inference, decoding PPNG-1 and PPNG-2 into the PPNG-3 tensor and loading them into the renderer takes $\mathbb{O}(Q^3 \times R)$ time and can be easily parallelized, making the decoding efficient. Figure 3 depicts the two representations for one single volume.

3.3. Encoding and Implementation Details

During the training/encoding stage, given a collection of posed images, we jointly train our Fourier feature volume $\mathbf{V}$ and our shallow MLP network g_θ with volume rendering. We minimize Huber-loss between the volume-rendered pixel colors and observed pixel colors for its robustness to outliers.

We set volume quantization size $Q = 80$, # of factorized components $R = 8$ for PPNG-1, # of factorized components $R = 2$ for PPNG-2, # of frequency levels $L = 4$ and feature dimension $D = 4$ throughout all implementations. We implement PPNG-1, 2 and 3 in *tiny-cuda-nn* [36], which can be integrated into Instant-NGP platform with voxel-based density caching [37] for accelerated ray integration.

The input to our MLP is a feature vector of size $F = 32 = 2 \times L \times D$. A single linear layer produces a density value and a 15-length feature vector. These values and the degree three spherical harmonics (length 16) of the viewing direction are input to a 2-layer MLP that outputs RGB color and contains a size 16 hidden layer. In total, our MLP contains 1,072 parameters. Our experiments ablate using different sized networks.

With the chosen parameters, we have parameter size of $125KB$ for PPNG-1, $2.45MB$ for PPNG-2 and $32.7MB$ for PPNG-3 using half-precision floating points (including shallow MLP weights). We encode voxel-based density cache using run-length encoding (RLE). This typically results in additional $50KB$ to $150KB$ depending on the complexity of the scene. We use CBOR [6] to aggregate the Fourier feature parameters, shallow MLP weights, and the voxel-based density caches into a single binary file. We emphasize that all three approaches are directly encoded end-to-end and there are no additional processes (a.k.a baking) in converting the optimized PPNG into a binary encoding. Both RLE and CBOR are very efficient to decode; for PPNG, we observe a decoding time of less than 20 ms for both RLE and CBOR decodings combined.

3.4. Real-time, Interactive and Portable Viewing

What sets PPNG apart from other spatial volume feature-based NeRF methods [8, 37] is its significantly more compact Fourier feature volume (80^3 vs 512^3). This enables us to store and render it directly as a GL 3D texture on low-cost GPUs with limited memory, such as those in mobile devices. Inspired by this, we implement PPNG representations in real-time by porting the volume rendering of PPNG in a traditional GL pipeline using WebGL2 with GLSL. Figure 2 illustrates this process. Given a binary PPNG file, our decoder first checks which PPNG type (among PPNG-1, 2, and 3) that the binary file contains. If the given binary is PPNG-1 or PPNG-2, we efficiently convert it to PPNG-3 using Eq 4 or Eq 5 respectively. The conversion is parallelized with GLSL-based code.

Given PPNG-3, we load each volumetric Fourier feature V as a 3D texture image. Since we set $D = 4$, we can use a texture format set as RGBA to load each volume into a single 3D texture image. We set the texture filtering parameter set to linear, which enables tri-linear interpolation for texture sampling the loaded volumes in GLSL. We then load RLE encoded voxel-based density cache for empty-space skipping, by decoding it as occupancy grid. Similar to the volumetric Fourier feature volumes, we load the occupancy grid as another single-channel 3D texture image. Finally, we load shallow MLP by chunking the MLP into set of 4x4 matrices (`Mat4`) for faster inference.

At render time, we use a fragment shader to perform volume rendering. From the camera origin, we cast a ray to each pixel in world space, and densely sample along the ray. For each sample, we check if it is occupied with the occupancy grid. If occupied, we query density and color at the sampled point using the method described in Section 3.1. The sampled color and density are integrated with volume rendering equation [35] and are terminated if accumulated transmittance falls below a threshold. We include a GLSL implementation in the supplementary material.

4. Experiments

We evaluate our model on multiple object-level datasets to validate the re-rendered models qualitatively and quantitatively. Then, we test our real-time renderer on various devices, including a desktop with a GPU, various laptops, and several mobile phones. We further conduct several analyses on how different designs contribute to the final performance

Table 2. **Quantitative Evaluation of Our Method on Various Datasets:** Implicit methods typically feature a smaller model size and better performance but require specialized renderers or hardware (e.g., CUDA GPUs). In contrast, web-ready methods are not memory-efficient. Our approach achieves a good balance, particularly excelling in model size. It tends to yield better rendering quality in object-centric scenes than in unbounded scenes. * indicates results from us.

Model Type	Real-time Web	Synthetic NeRF						Blended MVS			Tanks and Temples		
		PSNR ↑	SSIM ↑	LPIPS ↓	Size	FPS	Training Time	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [35]	✗	31.01	0.947	0.081	5 MB	-	35 hrs	24.15	0.828	0.192	25.78	0.864	0.198
NSVF [32]	✗	31.74	0.953	0.047	-	-	48 hrs (8 GPU)	26.90	0.898	0.113	28.40	0.900	0.153
Diver [53]	✗	32.32	0.960	0.032	67.8 MB	-	50 hrs	27.25	0.910	0.073	28.18	0.912	0.116
TensoRF-CP [8]	✗	31.56	0.949	0.041	3.9 MB	-	25.2 min	-	-	-	27.59	0.897	0.144
TensoRF-VM [8]	✗	33.14	0.963	0.027	71.8 MB	-	17.4 min	-	-	-	28.56	0.920	0.125
InstantNGP* [37]	✗	32.82	0.960	0.037	11.6 MB	-	5 min	28.70	0.943	0.037	28.36	0.930	0.099
Dictionary Field [9]	✗	33.14	0.961	-	5.1MB	-	12.2 min	-	-	-	29.00	0.938	-
WaveletNeRF [40]	✗	31.94	-	-	846 KB	-	24.0 min	-	-	-	27.77	-	-
WaveletNeRF* [40]	✗	25.90	0.891	0.142	199 KB	-	23.6 min	-	-	-	-	-	-
PlenOctree [56]	✓	31.71	0.958	0.049	1976 MB	168	10 min	-	-	-	27.99	0.917	0.131
SNeRG [21]	✓	30.4	0.950	0.050	87 MB	502	15 hrs	-	-	-	-	-	-
MobileNeRF [11]	✓	30.9	0.947	0.062	126 MB	762	20 hrs	-	-	-	-	-	-
Re-Rend [41]	✓	29.0	0.934	0.080	199 MB	1013	60 hrs	-	-	-	-	-	-
Gaussian Splatting* [26]	✓	33.80	0.970	0.030	67.3 MB	-	4.9 min	24.95	0.867	0.109	27.94	0.930	0.097
PPNG-3*	✓	31.90	0.949	0.044	32.8 MB	128	4.9 min	26.89	0.909	0.068	27.83	0.925	0.112
PPNG-2*	✓	30.99	0.944	0.053	2.49 MB	127	9.8 min	26.53	0.894	0.080	27.23	0.912	0.136
PPNG-1*	✓	28.89	0.926	0.080	151 KB	127	13.1 min	24.77	0.855	0.134	25.68	0.892	0.178

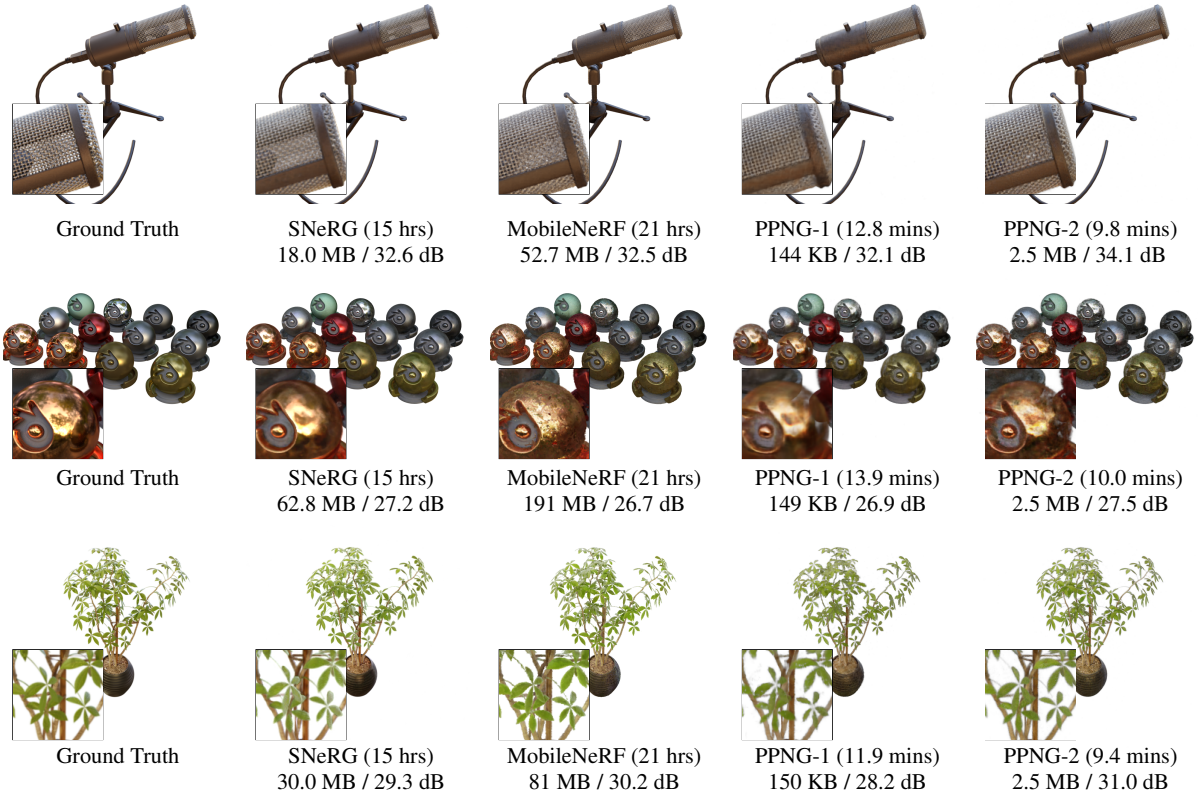


Figure 5. **Qualitative Comparison on the Synthetic NeRF Dataset:** We show qualitative results and compare real-time NeRF models (SNeRG and MobileNeRF) in terms of training time, model size, and quality. PPNG-1 delivers similar or superior visual quality compared to other web-friendly baselines while being at least **120x smaller in model size**. PPNG-2 offers enhanced quality with a model size more than 8x smaller.

and finally discuss the current limitations of our approach on unbounded scenes.

4.1. Experimental Details

Datasets: We evaluate on Synthetic NeRF [35], Blended MVS [54] and Tanks and Temples [27] datasets with a resolution of 800×800, 768×576, 1920×1080 respectively. We use the author-provided training/testing splits for Syn-

Table 3. **Ablation studies of PPNG on Synthetic NeRF dataset.** The reference implementation uses volume resolution $Q = 80$, Max Freq = 2^3, 1 Layer MLP, number of components (R) for PPNG-1 set as 8 and PPNG-2 set as 2. We mark significant improvements and losses with Green and Red respectively.

Ablations	PPNG-1			PPNG-2			PPNG-3		
	PSNR (+- 0.5)	Size (+- 20%)	Training Time (+- 20%)	PSNR (+- 0.5)	Size (+- 20%)	Training Time (+- 20%)	PSNR (+- 0.5)	Size (+- 20%)	Training Time (+- 20%)
Reference in Table 2	28.89	151 KB	13.1 min	30.99	2.49 MB	9.8 min	31.90	32.8 MB	4.9 min
Vol. Res. $Q = 60$	28.51	120 KB	14.9 min	30.63	1.42 MB	11.0 min	31.05	13.9 MB	4.0 min
Vol. Res. $Q = 100$	29.09	182 KB	13.3 min	31.12	3.72 MB	9.2 min	32.02	64.0 MB	5.2 min
Max Freq = 2^1	28.56	151 KB	15.1 min	30.12	2.49 MB	11.1 min	30.61	32.8 MB	4.1 min
Max Freq = 2^5	28.18	151 KB	12.3 min	30.48	2.49 MB	9.8 min	31.63	32.8 MB	4.9 min
2 Layer MLP	29.14	153 KB	14.7 min	31.04	2.49 MB	9.7 min	31.82	32.8 MB	4.5 min
# Comp $\times$ 0.5	27.50	89.2 KB	10.6 min	30.09	1.26 MB	9.31 min	-	-	-
# Comp $\times$ 2	29.87	274 KB	24.2 min	31.50	4.94 MB	17.0 min	-	-	-

thetic NeRF, and use processed scenes and train/test splits provided by NSVF [32] for Blended MVS and Tanks and Temples. We measure PSNR, LPIPS, and SSIM for visual quality, and report rendering speed in FPS, model size in MB, and optimization time for each scene.

Baselines: Our goal is to achieve compact, real-time, and web-compatible neural rendering. To the best of our knowledge, there is no preceding work that achieves our proposed level of compression (KB-level). To ensure a fair comparison, we primarily evaluate and compare our approach against the current best real-time, web-ready NeRF approaches, including SNeRG [21], MobileNeRF [11] and Re-render [41]. Additionally, we also reference state-of-the-art and classic NeRF approaches [8, 32, 35, 53, 57] for context, despite them not being in the same categories as our proposed approach.

Table 4. **Rendering Speed on Different Devices.** We report the rendering FPS for PPNG-2 on various devices using the Lego scene [35] at an 800 $\times$ 800 resolution on a web browser. For mobile devices, all measurements were conducted in battery mode, without external power connected.

Device	FPS
iPhone 10 Pro Max	20
iPhone 14	30
iPhone 15 Pro	50
M1 iPad	40
M1 Macbook Pro	45
M3 Macbook Air	50
M3 Max Macbook Pro	100
Desktop with Nvidia 3090 GPU	127

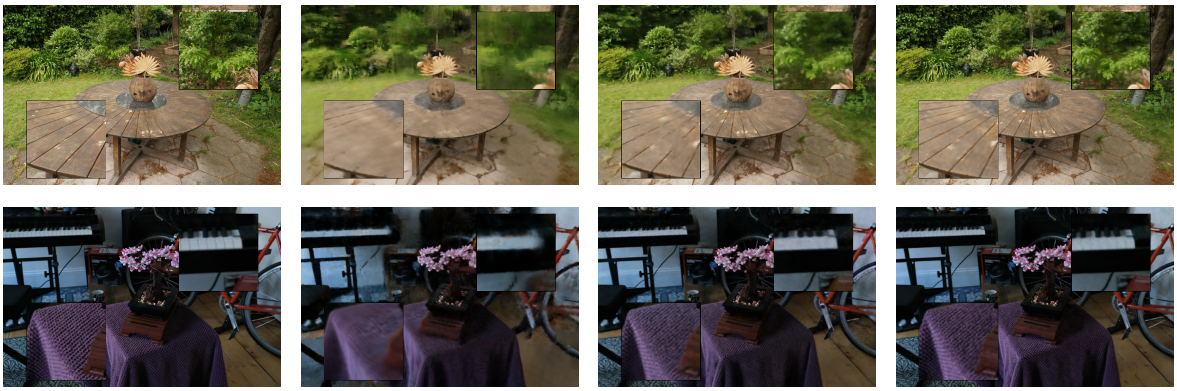
4.2. Experimental Results

Qualitative results: Figure 5 presents a comprehensive comparison of qualitative results. We note that the quality is reasonable for all competing algorithms, but our proposed approach achieves a significant reduction in model size (over 40x to 1500x) and over 50x reduction in training time. We would like to particularly highlight that our approach, despite its extreme compactness, effectively captures thin structures (such as the stems of plants), reflective materials (e.g., metal balls), as well as repetitive high-frequency patterns.

Quantitative results: We conduct two major quantitative evaluations. The first focuses on real-time, web-compatible approaches using the Synthetic NeRF dataset. Figure 4 displays the results. This study shows that our smallest model, PPNG1, significantly outperforms all other methods in terms of model size (**being 2-3 orders of magnitude smaller**) and training speed (20 times faster), while maintaining competitive rendering quality (PSNR = 28.5 dB). Our most robust model, PPNG3, has a model size over 5 times smaller and a training speed 40 times faster, and it achieves the best rendering quality among these real-time NeRF methods (PSNR = 31.9 db). Importantly, our GPU memory during rendering is only **47 MB**, making it particularly suitable for low-cost mobile devices (>10 times smaller than other competing methods). More details on vram consumption can be found in appendix.

Our second quantitative evaluation comprehensively compares the most representative and state-of-the-art novel view synthesis models across various datasets. We evaluate the rendering quality, speed, training time, and model size. We broadly categorize these models into two groups: mobile-friendly and non-friendly approaches, differentiated based on their GL compatibility. Table 2 presents the results. Our general observations are: 1) *Implicit approaches* offer the best quality and relatively compact size, but they are not mobile compatible, and their rendering speed tends to be slow; 2) *Real-time, mobile-compatible methods* tend to sacrifice some rendering quality for speed and typically have a larger model size; 3) *Our approaches* achieve the best trade-off in terms of model size, rendering quality, and training time. In particular, our model size significantly outperforms all competing baselines in model size and has one of the fastest training speeds.

To further provide a comparison for low-bit model sizes, we optimized the open source SotA low-bit model, Wavelet-NeRF [40] to a 200KB size to our best effort with an experimental combination (i.e., $4\times$ smaller feature length, $2\times$ smaller feature grid, and a larger ($1e-8$) mask loss). PPNG



Ground Truth | PPNG-1 (512 KB) | PPNG-2 (2.9 MB) | PPNG-3 (33.3 MB)

Figure 6. **Qualitative Results on Unbounded 360° Scenes:** We highlight the background region in the top right corner and the central region in the bottom left corner. PPNG-3 provides compelling results with detailed textures in both cases. Factorized representations reach their capacity limits in such scenes. PPNG-1, with only 128 KB parameters, fails to recreate fine details in both the central and background regions, and PPNG-2 also cannot recreate details in the background regions due to capacity with limited volume size.

outperforms optimized WaveletNeRF by a large margin at the Internet-friendly size, demonstrating the advanced compression capacity of periodic encoding over sparsity encoding. The rendering quality of PPNG-1 and PPNG-2 is comparable to other real-time methods, while the quality of PPNG-3 is comparable to implicit approaches.

Table 5. **Quantitative evaluation on unbounded 360° scenes.** We let * to denote author measured time; † to denote reported time on paper.

	PSNR	SSIM	LPIPS	Size	Training Time
MobileNeRF [11]	22.0	0.470	0.470	347 MB	21+ hours†
BakedSDF [55]	24.5	0.697	0.309	457 MB	7+ hours*
MERF [39]	25.2	0.722	0.311	162 MB	2 hours (8 GPUs)*
SMERF [15]	28.0	0.728	0.212	139 MB	17+hours†
PPNG-1	20.2	0.476	0.658	512 KB	20.7 min
PPNG-2	21.9	0.543	0.499	2.93 MB	15.0 min
PPNG-3	23.7	0.618	0.392	33.3 MB	7.8 min

Multi-platform analysis: Table 4 shows that PPNG can be efficiently rendered with various devices including mobile devices. On supplementary material, we demonstrate that mobile phones can load multiple scenes at once.

4.3. Ablation studies

We evaluate how each component of our model impacts the performance and report the results in Table 3.

Number of MLP layers: We demonstrate that adding one additional layer to the shallow MLP improves the PSNR by 0.25 dB in PPNG-1, yet does not alter the quality much for PPNG-2 and PPNG-3. While a deeper MLP is known to offer a stronger capacity for modeling complex appearances [37], they also increase the computation required per sample. Therefore, we use a one-layer MLP to ensure speed and compatibility in our final model.

Levels of quantization: A finer feature grid improves performance (with $Q = 100$) but increases file and memory size, which grows rapidly at a rate of $O(Q^3)$ for PPNG-3. Since PPNG-1 and PPNG-2 are converted into PPNG-3 at rendering time, we consider $Q = 80$ to be an appropriate level, effectively balancing quality and memory size.

Frequency Ranges: Choosing the frequencies for sinusoidal positional encoding is crucial. We show that using too low or high frequency can degrade performance. Additionally, optimal frequency may vary based on levels of quantization and scene scale.

Number of Components: The number of factorized components is crucial for balancing model size and performance in PPNG-1 and PPNG-2. Fewer components reduce the model size to under 100KB, while more components improve rendering quality.

4.4. Limitations

Plenoptic PNG is designed for scenes with a limited range. Although it can model unbounded scenes with reasonable quality at a tiny size (as shown in Figure 6), it may not perform as effectively as large real-time models (see Table 5). In the future, we plan to extend PPNG to include contracted space modeling and blocks to address this limitation.

5. Conclusion

We present Plenoptic PNG (PPNG), a highly compact representation for real-time, web-compatible free-viewpoint rendering. PPNG leverages Fourier feature modeling and volume factorization to achieve a small model size and fast training time. Compared to other real-time NeRF models, PPNG offers the smallest size and quickest training with minimal quality loss. Additionally, PPNG can be efficiently loaded on lightweight devices like mobile phones using GL-texturing. We believe PPNG will enable new applications requiring easy sharing of 3D immersive visual content.

References

[1] Balsa, M., Gobbetti, E., Iglesias-Guitian, J.A., Makhinya, M., Marton, F., Pajarola, R., Suter, S.K.: State-of-the-art in compressed gpu-based direct volume rendering. Computer Graphics Forum **33** (2014) 3

[2] Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 5835–5844 (2021) 2

[3] Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. 2023 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 19640–19648 (2023) 2

[4] Bird, T., Ball'e, J., Singh, S., Chou, P.A.: 3d scene compression through entropy penalized neural representation functions. 2021 Picture Coding Symposium (PCS) pp. 1–5 (2021) 3

[5] Biswas, S., Liu, J., Wong, K., Wang, S., Urtasun, R.: Muscle: Multi sweep compression of lidar using deep entropy models. Advances in Neural Information Processing Systems **33**, 22170–22181 (2020) 3

[6] Bormann, C., Hoffman, P.E.: Concise binary object representation (cbor). RFC **7049**, 1–54 (2013) 5

[7] Cao, J., Wang, H., Chemerys, P., Shakhrai, V., Hu, J., Fu, Y., Makoviichuk, D., Tulyakov, S., Ren, J.: Real-time neural light field on mobile devices. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 8328–8337 (2022) 3

[8] Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. ArXiv **abs/2203.09517** (2022) 2, 3, 4, 5, 6, 7

[9] Chen, A., Xu, Z., Wei, X., Tang, S., Su, H., Geiger, A.: Dictionary fields: Learning a neural basis decomposition. ACM Transactions on Graphics (TOG) **42**, 1 – 12 (2023) 3, 6

[10] Chen, Y., Wu, Q., Cai, J., Harandi, M., Lin, W.: Hac: Hash-grid assisted context for 3d gaussian splatting compression. arXiv preprint arXiv:2403.14530 (2024) 3

[11] Chen, Z., Funkhouser, T.A., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 16569–16578 (2022) 2, 6, 7, 8

[12] Crassin, C., Neyret, F., Lefebvre, S., Eisemann, E.: Gigavoxels: ray-guided streaming for efficient and detailed voxel rendering. In: ACM Symposium on Interactive 3D Graphics and Games (2009) 3

[13] Datta, S., Marshall, C., Dong, Z., Li, Z., Nowrouzezahrai, D.: Efficient graphics representation with differentiable indirection. SIGGRAPH Asia 2023 Conference Papers (2023) 3

[14] De Queiroz, R.L., Chou, P.A.: Compression of 3d point clouds using a region-adaptive hierarchical transform. IEEE Transactions on Image Processing **25**(8), 3947–3956 (2016) 3

[15] Duckworth, D., Hedman, P., Reiser, C., Zhizhin, P., Thibert, J.F., Lucic, M., Szeliski, R., Barron, J.T.: Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration. ArXiv **abs/2312.07541** (2023) 8

[16] Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of gaussians. arXiv preprint arXiv:2403.14166 (2024) 3

[17] Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.P.C.: Fastnerf: High-fidelity neural rendering at 200fps. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 14326–14335 (2021) 2

[18] Garcia, D.C., de Queiroz, R.L.: Intra-frame context-based octree coding for point-cloud geometry. 2018 25th IEEE International Conference on Image Processing (ICIP) pp. 1807–1811 (2018) 2

[19] Gordon, C., Ch'ng, S.F., MacDonald, L.E., Lucey, S.: On quantizing implicit neural representations. 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) pp. 341–350 (2022) 3

[20] Gupta, A., Cao, J., Wang, C., Hu, J., Tulyakov, S., Ren, J., Jeni, L.A.: Lightspeed: Lighter and faster neural light field on mobile devices. In: Thirty-seventh Conference on Neural Information Processing Systems (2023) 3

[21] Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.E.: Baking neural radiance fields for real-time view synthesis. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 5855–5864 (2021) 2, 6, 7

[22] Heitz, E., Dupuy, J., Crassin, C., Dachsbacher, C.: The sggx microflake distribution. ACM Transactions on Graphics (TOG) **34**, 1 – 11 (2015) 3

[23] Heitz, E., Neyret, F.: Representing appearance and pre-filtering subpixel data in sparse voxel octrees. In: EGGH-HPG'12 (2012) 3

[24] Huang, L., Wang, S., Wong, K., Liu, J., Urtasun, R.: Octsqueeze: Octree-structured entropy model for lidar compression. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1310–1320 (2020) 2

[25] Huang, Y., Peng, J., Kuo, C.C.J., Gopi, M.: A generic scheme for progressive point cloud coding. IEEE Transactions on Visualization and Computer Graphics **14**, 440–453 (2008) 2

[26] Kerbl, B., Kopanas, G., Leimkuehler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (TOG) **42**, 1 – 14 (2023) 2, 3, 6

[27] Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples. ACM Transactions on Graphics (TOG) **36**, 1 – 13 (2017) 6

[28] Kurz, A., Neff, T., Lv, Z., Zollhofer, M., Steinberger, M.: Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In: European Conference on Computer Vision (2022) 2

[29] Levoy, M., Hanrahan, P.: Light field rendering. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996) 3

[30] Li, L., Shen, Z., Wang, Z., Shen, L., Bo, L.: Compressing volumetric radiance fields to 1 mb. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4222–4231 (2022) 3

[31] Li, S., Li, H., Liao, Y., Yu, L.: Nerfcodec: Neural feature compression meets neural radiance fields for memory-efficient scene representation. arXiv preprint arXiv:2404.02185 (2024) 3

[32] Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. ArXiv **abs/2007.11571** (2020) 2, 6, 7

[33] Lu, Y., Jiang, K., Levine, J.A., Berger, M.: Compressive neural representations of volumetric scalar fields. Computer Graphics Forum **40** (2021) 3

[34] Material, S.: Plenvdb: Memory efficient vdb-based radiance fields for fast training and rendering (2023) 2

[35] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**, 99–106 (2020) 2, 3, 4, 5, 6, 7

[36] Müller, T.: tiny-cuda-nn (4 2021) 5

[37] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (TOG) **41**, 1 – 15 (2022) 2, 3, 5, 6, 8

[38] Park, J.J., Florence, P.R., Straub, J., Newcombe, R.A., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 165–174 (2019) 3

[39] Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P.P., Mildenhall, B., Geiger, A., Barron, J.T., Hedman, P.: Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. ACM Transactions on Graphics (TOG) **42**, 1 – 12 (2023) 2, 8

[40] Rho, D., Lee, B., Nam, S., Lee, J.C., Ko, J.H., Park, E.: Masked wavelet representation for compact neural radiance fields. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 20680–20690 (2022) 3, 6, 7

[41] Rojas, S., Zarzar, J., Perez, J.C., Sanakoyeu, A., Thabet, A.K., Pumarola, A., Ghanem, B.: Re-rend: Real-time rendering of nerfs across devices. 2023 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 3609–3618 (2023) 2, 6, 7

[42] Schnabel, R., Klein, R.: Octree-based point-cloud compression. PBG@ SIGGRAPH **3** (2006) 2

[43] Shin, S., Park, J.: Binary radiance fields. In: Advances in Neural Information Processing Systems (2023) 3

[44] Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5449–5459 (2021) 2

[45] Takikawa, T., Evans, A., Tremblay, J., Müller, T., McGuire, M., Jacobson, A., Fidler, S.: Variable bitrate neural fields. ACM SIGGRAPH 2022 Conference Proceedings (2022) 3

[46] Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C.T., Nowrouzezahrai, D., Jacobson, A., McGuire, M., Fidler, S.: Neural geometric level of detail: Real-time rendering with implicit 3d shapes. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 11353–11362 (2021) 3

[47] Takikawa, T., Müller, T., Nimier-David, M., Evans, A., Fidler, S., Jacobson, A., Keller, A.: Compact neural graphics primitives with learned hash probing. SIGGRAPH Asia 2023 Conference Papers (2023) 3

[48] Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. In: NeurIPS (2020) 4

[49] Tang, D., Dou, M., Lincoln, P., Davidson, P.L., Guo, K., Taylor, J., Fanello, S., Keskin, C., Kowdle, A., Bouaziz, S., Izadi, S., Tagliasacchi, A.: Real-time compression and streaming of 4d performances. ACM Transactions on Graphics (TOG) **37**, 1 – 11 (2018) 3

[50] Tang, D., Singh, S., Chou, P.A., Haene, C., Dou, M., Fanello, S., Taylor, J., Davidson, P.L., Guleryuz, O.G., Zhang, Y., Izadi, S., Tagliasacchi, A., Bouaziz, S., Keskin, C.: Deep implicit volume compression. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1290–1300 (2020) 3

[51] Wang, J., Zhu, H., Ma, Z., Chen, T., Liu, H., Shen, Q.: Learned point cloud geometry compression. arXiv preprint arXiv:1909.12037 (2019) 3

[52] Wang, Z., Shen, T., Nimier-David, M., Sharp, N., Gao, J., Keller, A., Fidler, S., Muller, T., Gojcic, Z.: Adaptive shells for efficient neural radiance field rendering. ACM Transactions on Graphics (TOG) **42**, 1 – 15 (2023) 2

[53] Wu, L., Lee, J.Y., Bhattad, A., Wang, Y., Forsyth, D.A.: Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 16179–16188 (2021) 2, 6, 7

[54] Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1787–1796 (2019) 6

[55] Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Bakedsdf: Meshing neural sdfs for real-time view synthesis. ACM SIGGRAPH 2023 Conference Proceedings (2023) 2, 8

[56] Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5491–5500 (2021) 2, 6

[57] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 5732–5741 (2021) 2, 7

[58] Yuan, S., Zhao, H.: Slimmerf: Slimmable radiance fields. arXiv preprint arXiv:2312.10034 (2024) 3