Supplementary Document for: PlenOctrees for Real-time Rendering of Neural Radiance Fields

Alex Yu[1] Ruilong Li[1,2] Matthew Tancik[1]
Hao Li[1,3] Ren Ng[1] Angjoo Kanazawa[1]

[1]UC Berkeley [2]USC Institute for Creative Technologies [3]Pinscreen

In this supplementary document, we provide some additional results including more qualitative results and comparisons; detailed per-scene breakdown quantitative results; and an ablation study on different choices of spherical basis functions. Moreover, we also provide some technical details including a brief introduction of the spherical basis functions; the analytic derivatives of ORF rendering; ORF compression technology we used for in-browser experience; NeRF-SH training details and ORF optimization details.

1. Additional Results

1.1. Detailed comparisons

Here we provide further qualitative comparisons with baselines: SRN [10], Neural Volumes [7], NSVF [6] in Figure 1. We show more qualitative results of our method in Figure 2 and Figure 3. We also report a per-scene breakdown of the quantitative metrics against all approaches in Table 3, 4, 5, 6.

1.2. Spherical Basis Function Ablation

Here we provide ablation studies on the choice of spherical basis functions. We first ablate the effect on the number of spherical harmonics basis, then we explore the use of a learnable spherical basis functions. All experiments are conducted on NeRF-synthetic dataset and we report the average metric directly after training NeRF with spherical basis functions and after converting it to ORF with fine-tuning.

Number of SH basis functions First, we ablate the number of basis functions used for spherical harmonics. Average metrics across the NeRF-synthetic dataset are reported both for the modified NeRF model and the corresponding ORF. We found that switching between $\ell_{\max} = 3$ (*SH-16*) and 4 (*SH-25*) makes very little difference in terms of metrics or visual quality.

Spherical Gaussians Furthermore, we also experimented with spherical Gaussians (SGs) [2], which is another form of spherical basis functions similar to spherical harmonics, but with learnable Gaussian kernels. Please see §2.1 for a brief introduction of SHs and SGs. *SG-25* denotes our model using 25 SG components instead of SH, all with learnable lobe axis $\mathbf{p}$ and bandwidth λ. While this model has marginally better PSNR, the advantage disappears following ORF conversion and fine-tuning.

	NeRF-SH/SG			Converted PlenOctree				
Basis	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	GB ↓	FPS↑
SH-9	31.44	0.951	0.065	31.45	0.956	0.056	1.00	262
SH-16	31.57	0.952	0.063	31.71	0.958	0.053	1.93	168
SH-25	31.56	0.951	0.063	31.69	0.958	0.052	2.68	128
SG-25	31.74	0.953	0.062	31.63	0.958	0.052	2.26	151

Table 1: **Spherical Basis Function Ablation.** We experiment with various versions of spherical basis functions, including *SH-16*, *SH-25* and *SG-25*.

1.3. Clarification on Table 1.

In the main paper Table 1, we report our quantitative results comparing to various baselines on the NeRF-synthetic test scenes. A few baseline numbers were reported incorrectly/inappropriately, which we clarify and correct here. Table 2 shows the corrected version of Table 1 in the main paper, with all corrected number marked as red. Note this correction does not affect our claims or conclusions in the main paper. The details of the errata/clarifications are below:

- **Neural Volumes.** The PSNR, SSIM, LPIPS numbers we reported are taken from NeRF [9] paper Table 1. However, we mistakenly referred the numbers from the "Diffuse Synthetic 360°" section instead of the "Realistic Synthetic 36o°". section. This is updated.

- **SRN SSIM.** The PSNR, SSIM, LPIPS numbers we reported in Table 1 were taken from a excel sheet shared by the NSVF [6] authors. When we further examined these numbers with what's reported in the NeRF paper, we found conflicts on the SSIM score for SRN. We de-

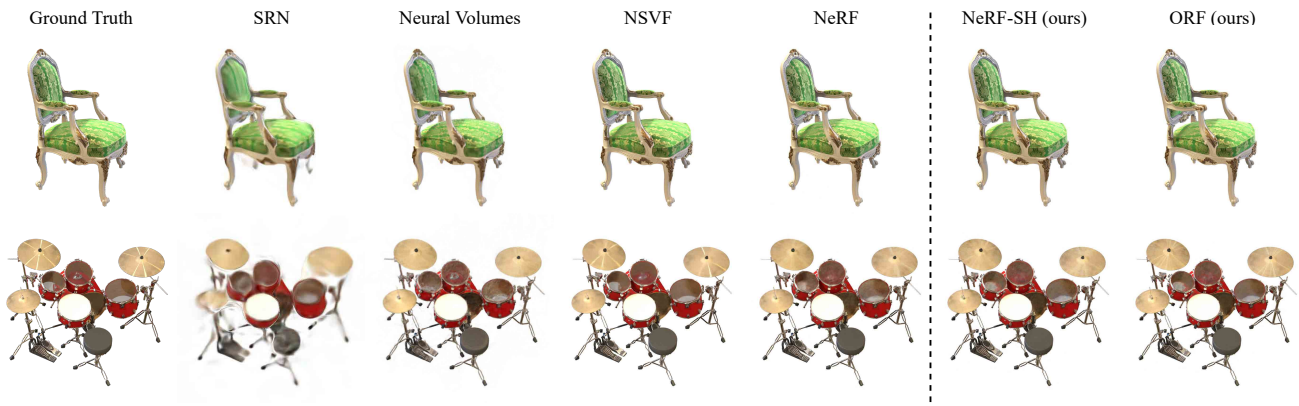

Figure 1: **Qualitative comparisons on NeRF-synthetic.**

Synthetic NeRF Dataset	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
NeRF (original)	31.01	0.947	0.081	0.023
NeRF	31.69	0.953	0.068	0.045
SRN	22.26	0.846	0.170	0.909
Neural Volumes	26.05	0.893	0.160	3.330
NSVF	31.75	0.953	0.047	0.815
AutoInt (8 sections)	25.55	0.911	0.170	0.380
NeRF-SH	31.57	0.952	0.063	0.051
ORF from NeRF-SH	31.02	0.951	0.066	167.68
ORF after fine-tuning	31.71	0.958	0.053	167.68

Table 2: **Corrected Table 1. in the main paper.** Corrected numbers are marked as red

cided to follow the number reported on the NeRF paper for this score.

- **NSVF FPS.** The FPS we reported in the main paper for NSVF is calculated by first averaging the run-time then calculating its reciprocal. However our FPS is calculated by directly averaging all per-scene FPS. Thus we update the NSVF FPS with the way we compute our FPS in order for this number to be comparable.

2. Technical Details

2.1. Spherical Basis Functions: SH and SG

In the main paper, we used the SH functions without defining their exact form. Here, we provide a brief technical discussion of both spherical harmonics (SH) and spherical Gaussians (SG) for completeness.

Spherical Harmonics. The Spherical Harmonics (SH) form a complete basis of functions $\mathbb{S}^2 \to \mathbb{C}$. For $\ell \in \mathbb{N} \cup \{0\}$ and $m \in \{-\ell, \dots, \ell\}$, the SH function of degree ℓ and order m is defined as:

$$Y_\ell^m(\theta, \phi) = \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} P_\ell^m(\cos\theta) e^{im\phi} \tag{1}$$

where $P_\ell^m(cos\theta)e^{im\phi}$ are the associated Legendre polynomials. A real basis of SH $Y_\ell^m : \mathbb{S}^2 \mapsto \mathbb{R}$ can be defined in terms of its complex analogue $Y_\ell^m : \mathbb{S}^2 \mapsto \mathbb{C}$ by setting

$$Y_\ell^m(\theta, \phi) = \begin{cases} \sqrt{2}(-1)^m \operatorname{Im}[Y_\ell^{|m|}] & \text{if } m < 0 \\ Y_\ell^0 & \text{if } m = 0 \\ \sqrt{2}(-1)^m \operatorname{Re}[Y_\ell^m] & \text{if } m > 0 \end{cases} \tag{2}$$

Any real spherical function $L : \mathbb{S}^2 \to \mathbb{R}$ may then be expressed in the SH basis:

$$L(\mathbf{d}) = L(\theta, \phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} k_\ell^m Y_\ell^m(\theta, \phi) \tag{3}$$

Spherical Gaussians. Spherical Gaussians (SGs), also known as the von Mises-Fisher distribution [2], is another form of spherical basis functions that have been widely adopted to approximate spherical functions. Unlike SHs, SGs are a learnable basis. A normalized SG is defined as:

$$G(\mathbf{d}; \mathbf{p}, \lambda) = e^{\lambda(\mathbf{d}\cdot\mathbf{p}-1)} \tag{4}$$

Where $\mathbf{p} \in \mathbb{R}^2$ is the lobe axis, and $\lambda \in \mathbb{R}$ is the bandwidth (sharpness) of the Gaussian kernel. Due to the varying bandwidths supported by SGs, they are suitable for representing all-frequency signals such as lighting [12, 11, 5]. A spherical function represented using n SGs is formulated as:

$$L(\mathbf{d}) \approx \sum_{\ell=0}^{n} k_\ell G_\ell(\mathbf{d}; \mathbf{p}, \lambda) \tag{5}$$

Where $k_\ell \in \mathbb{R}^3$ is the RGB coefficients for each SG.

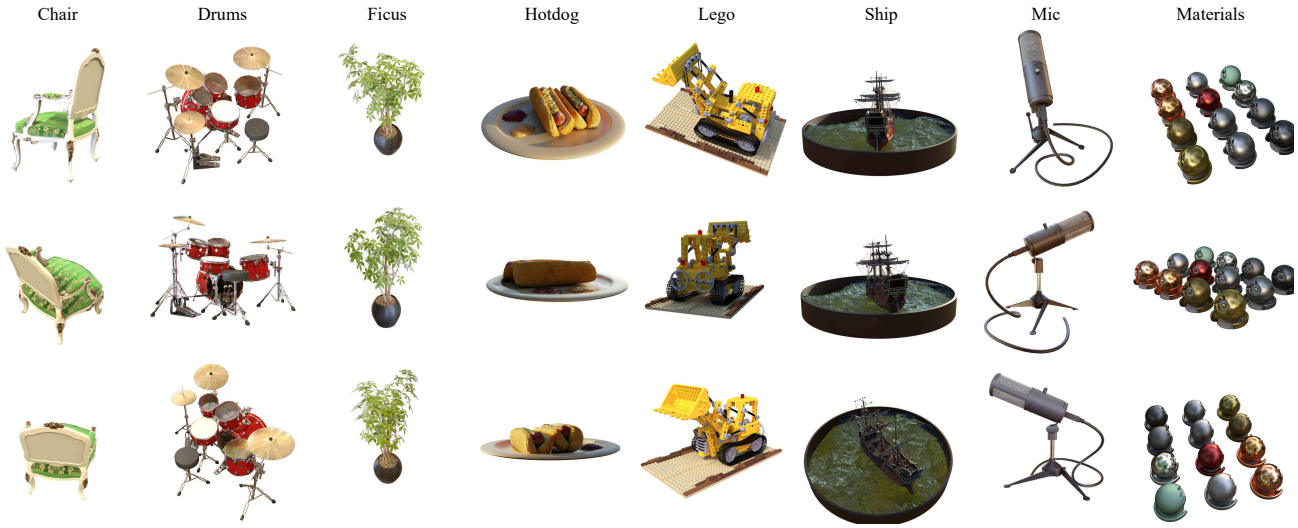

Figure 2: **More qualitative results of ORF on NeRF-synthetic.**

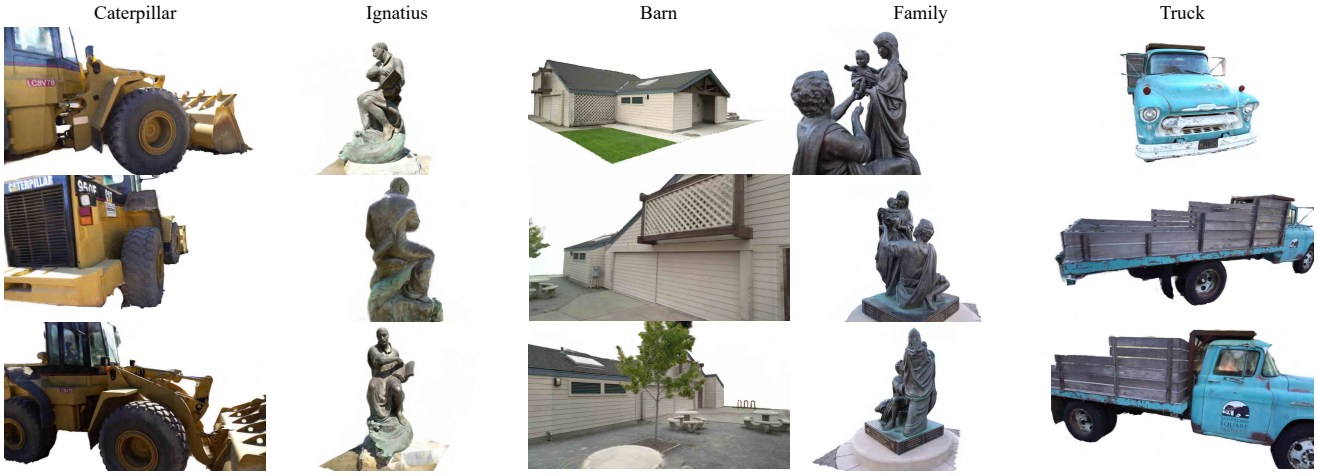

Figure 3: **More qualitative results of ORF on Tanks&Temples.**

2.2. ORF Compression for In-browser Experience

The uncompressed ORF file would be rather time-consuming for users to download for in-browser rendering. Thus, to minimize the size of ORF for viewing in the browser, we use SH-9 instead of SH-16 or SH-25 and apply a looser bounding box, which reduces the number of occupied voxels. On top of this, we compress the ORF directly in the following ways:

1. We quantize the SH coefficients in the tree using the popular median-cut algorithm [3]. More specifically, the σ values are kept as is; for each SH basis function, we quantize the RGB coefficients $k_\ell^m \in \mathbb{R}^3$ into 2^{16} colors. Afterwards, separately for each SH basis function, we store a $2^{16} \times 3$ codebook (as `float16`) along with pointers from each tree leaf to a position in the codebook (as `int16`).

2. We compress the entire tree, including pointers, using the standard DEFLATE algorithm from ZLIB [8].

This process reduces the file size by as much as 20-30 times. The tree is fully decompressed before it is displayed in the web renderer. We will also release this code.

2.3. Analytic Derivatives of ORF Rendering

In this section, we derive the analytic derivatives of the NeRF piecewise constant volume rendering model for optimizing ORF directly. Throughout this section we will consider a fixed ray with a given origin and direction.

2.3.1 Definitions

For preciseness, we provide definitions of quantities used in NeRF volume rendering. The NeRF rendering model considers a ray divided into N consecutive segments with endpoints $\{t_i\}_{i=0}^N$, where t_0 and t_N are the near and far bounds. The segments have constant densities $\boldsymbol{\sigma} = (\sigma_0, \dots, \sigma_{N-1})$ where each $\sigma_i \geq 0$. If we shine a light of intensity 1 at t_i, then at the camera position t_0, the light intensity is given by

$$T_i(\boldsymbol{\sigma}) := \prod_{j=0}^{i-1} \exp(-\delta_j \sigma_j), \tag{6}$$

Where $\delta_i := t_{i+1} - t_i$ are segment lengths as in the main paper. Note that T_i is also known as the accumulated transmittance from t_0 to t_i. It can be shown that this precisely models the absorption within each segment in the piecewise-constant setting.

Let $\mathbf{c} = (c_0, \dots c_{N-1})$ be the color associated with segments $0, \dots, N-1$, and c_N be the background light intensity; each $c_0, \dots, c_N \in [0,1]^3$ is an RGB color. We are interested in the derivative of the rendered color $\hat{C}(\boldsymbol{\sigma}, \mathbf{c})$ with respect to $\boldsymbol{\sigma}$ and $\mathbf{c}$. Note c_N (background) is typically considered a hyperparameter.

2.3.2 Derivation of the Derivatives

From the original NeRF rendering equation from the main paper (1), we can express the rendered ray color $\hat{C}(\boldsymbol{\sigma}, \mathbf{c})$ as:

$$\hat{C}(\boldsymbol{\sigma}, \mathbf{c}) = T_N(\boldsymbol{\sigma})\, c_N + \sum_{i=0}^{N-1} T_i(\boldsymbol{\sigma}) \left(1 - e^{-\sigma_i \delta_i}\right) c_i \tag{7}$$

$$= \sum_{i=0}^{N} w_i(\boldsymbol{\sigma})\, c_i \tag{8}$$

Where $w_i(\boldsymbol{\sigma}) = T_i(\boldsymbol{\sigma})\,(1 - \exp(-\sigma_i \delta_i)) = T_i(\boldsymbol{\sigma}) - T_{i+1}(\boldsymbol{\sigma})$ are segment weights, and $w_N(\boldsymbol{\sigma}) = T_N(\boldsymbol{\sigma})$.[1]

Color derivative. Since the rendered color are a convex combination of the segment colors, it's immediately clear that

$$\frac{\partial \hat{C}}{\partial c_i}(\boldsymbol{\sigma}, \mathbf{c}) = w_i(\boldsymbol{\sigma}) \tag{9}$$

Handling spherical harmonics colors is straightforward by applying the chain rule, noting that the SH basis function values are constant across the ray.

[1] Note that the background color c_N was omitted in equation (1) of the main paper for simplicity, and the indices are off by one. We use non-boldface c, C to distinguish from $\mathbf{c}, \boldsymbol{\sigma}$.

Density derivative. This is slightly more tricky. We can write the derivative wrt. σ_i,

$$\frac{\partial \hat{C}}{\partial \sigma_i}(\boldsymbol{\sigma}, \mathbf{c}) = c_N \frac{\partial T_N}{\partial \sigma_i} + \sum_{k=0}^{N-1} c_k \left(\frac{\partial T_k}{\partial \sigma_i} - \frac{\partial T_{k+1}}{\partial \sigma_i}\right) \tag{10}$$

Where the derivative of the intensity T_k, is

$$\frac{\partial T_k}{\partial \sigma_i}(\boldsymbol{\sigma}) = \frac{\partial}{\partial \sigma_i}\left[\prod_{j=0}^{k-1} e^{-\delta_j \sigma_j}\right] \tag{11}$$

$$= -\delta_i \left[\prod_{j=0}^{k-1} e^{-\delta_j \sigma_j}\right] 1_{k>i} \tag{12}$$

$$= -\delta_i T_k(\boldsymbol{\sigma})\, 1_{k>i} \tag{13}$$

$1_{k>i}$ denotes an indicator function whose value is 1 if $k > i$ or 0 else. Basically, we can delete any T_k for $k \leq i$ from the original expression, then multiply by $-\delta_i$. Therefore we can simplify (10) as follows

$$\frac{\partial \hat{C}}{\partial \sigma_i}(\boldsymbol{\sigma}, \mathbf{c}) = \delta_i \left[c_i T_{i+1}(\boldsymbol{\sigma}) - \sum_{k=i+1}^{N} c_k\, w_k(\boldsymbol{\sigma})\right] \tag{14}$$

Remark. Within the ORF renderer, this gradient can be computed in two rendering passes; the second pass is needed due to dependency on "future" weights and colors not seen by the ray marching process. The first pass store $\sum_{k=0}^{N} c_k\ w_k(\boldsymbol{\sigma})$, then subtracting a prefix from it. The overhead is still relatively small, and auxiliary memory use is constant.

If there are multiple colors, we simply add the density derivatives over all of them. In practice, usually the network outputs $\tilde{\sigma}_i \in \mathbb{R}$ and we set $\sigma_i = (\tilde{\sigma}_i)_+$, so we also need to take care of setting the gradient to 0 if $\tilde{\sigma}_i \leq 0$.

2.4. NeRF-SH Training Details

Our NeRF-SH model is built upon a Jax reimplementation of NeRF [1]. In our experiments, we use a batch size of 1024 rays, each with 64 sampled points in the coarse volume and 128 additional sampled points in the fine volume. The model is optimized with the Adam optimizer [4] using a learning rate that starts at 5×10^{-4} and decays exponentially to 5×10^{-6} over the training process. All of our models are trained for 2M iterations under the same protocol. Training takes around 50 hours to converge for each model on a single NVIDIA V100 GPU.

2.5. ORF Optimization Details

After converting the NeRF-SH model into an ORF, we further optimize the ORF on the training set with SGD. For NeRF-synthetic dataset, we use a constant 1×10^7

learning rate and optimize for maximum 80 epochs. For Tanks&Temples dataset, we set the learning rate to 1.5×10^6 and the maximum epochs to 40. We applied early stopping for the optimization process by monitoring the PSNR on the validation set[2]. On average it takes around 10 minutes to finish the ORF optimization for each scene on a single NVIDIA V100 GPU. The entire optimization process is done in `float32` but after it we storage the ORF with `float16` to reduce the model size.

Acknowledgements

We thank Vickie Ye and Ben Recht for help discussions, Zejian Wang of Pinscreen for helping with video capture, and BAIR commons for an allocation of GCP credits.

References

[1] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. 4

[2] Ronald Aylmer Fisher. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 217(1130):295–305, 1953. 1, 2

[3] Paul Heckbert. Color image quantization for frame buffer display. *SIGGRAPH*, 1982. 3

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[5] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *CVPR*, pages 2475–2484, 2020. 2

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

[7] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):65:1–65:14, 2019. 1

[8] Jean loup Gailly and Mark Adler. zlib, 2017. 3

[9] 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. *ECCV*, 2020. 1

[10] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. 1

[11] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH*, pages 527–536, 2002. 2

[12] Yu-Ting Tsai and Zen-Chung Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on graphics (TOG)*, 25(3):967–976, 2006. 2

[2]For Tanks&Temples dataset, we hold out 10% of the training set as validation set.

PSNR ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF (original)	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65	31.01
NeRF	34.08	25.03	30.43	36.92	33.28	29.91	34.53	29.36	31.69
SRN	26.96	17.18	20.73	26.81	20.85	18.09	26.85	20.60	22.26
Neural Volumes	28.33	22.58	24.79	30.71	26.08	24.22	27.78	23.93	26.05
NSVF	33.19	25.18	31.23	37.14	32.29	32.68	34.27	27.93	31.75
NeRF-SH	33.98	25.17	30.72	36.75	32.77	29.95	34.04	29.21	31.57
PlenOctree from NeRF-SH	33.19	25.01	30.56	36.15	32.12	29.56	33.01	28.58	31.02
PlenOctree after fine-tuning	34.66	25.31	30.79	36.79	32.95	29.76	33.97	29.42	31.71

SSIM ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF (original)	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856	0.947
NeRF	0.975	0.925	0.967	0.979	0.968	0.952	0.987	0.868	0.953
SRN	0.910	0.766	0.849	0.923	0.809	0.808	0.947	0.757	0.846
Neural Volumes	0.916	0.873	0.910	0.944	0.880	0.888	0.946	0.784	0.893
NSVF	0.968	0.931	0.960	0.987	0.973	0.854	0.980	0.973	0.953
NeRF-SH	0.974	0.927	0.968	0.978	0.966	0.951	0.985	0.866	0.952
PlenOctree from NeRF-SH	0.970	0.927	0.968	0.977	0.965	0.953	0.983	0.863	0.951
PlenOctree after fine-tuning	0.981	0.933	0.970	0.982	0.971	0.955	0.987	0.884	0.958

LPIPS ↓

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF (original)	0.046	0.091	0.044	0.121	0.050	0.063	0.028	0.206	0.081
NeRF	0.035	0.085	0.038	0.079	0.040	0.060	0.019	0.185	0.068
SRN	0.106	0.267	0.149	0.100	0.200	0.174	0.063	0.299	0.170
Neural Volumes	0.109	0.214	0.162	0.109	0.175	0.130	0.107	0.276	0.160
NSVF	0.043	0.069	0.017	0.025	0.029	0.021	0.010	0.162	0.047
NeRF-SH	0.037	0.087	0.039	0.041	0.041	0.060	0.021	0.177	0.063
PlenOctree from NeRF-SH	0.039	0.088	0.038	0.044	0.046	0.063	0.023	0.189	0.066
PlenOctree after fine-tuning	0.022	0.076	0.038	0.032	0.034	0.059	0.017	0.144	0.053

FPS ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF (original)	0.023	0.023	0.023	0.023	0.023	0.023	0.023	0.023	0.023
NeRF	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045
SRN	0.909	0.909	0.909	0.909	0.909	0.909	0.909	0.909	0.909
Neural Volumes	3.330	3.330	3.330	3.330	3.330	3.330	3.330	3.330	3.330
NSVF	1.044	0.735	0.597	0.660	0.633	0.517	1.972	0.362	0.815
NeRF-SH	0.051	0.051	0.051	0.051	0.051	0.051	0.051	0.051	0.051
PlenOctree	352.4	175.9	85.6	95.5	186.8	64.2	324.9	56.0	167.7

Table 3: **Per-scene quantitive results on NeRF-synthetic dataset.**

PSNR ↑

	Barn	Caterpillar	Family	Ignatius	Truck	Mean
NeRF (original)	24.05	23.75	30.29	25.43	25.36	25.78
NeRF	27.39	25.24	32.47	27.95	26.66	27.94
SRN	22.44	21.14	27.57	26.70	22.62	24.09
Neural Volumes	20.82	20.71	28.72	26.54	21.71	23.70
NSVF	27.16	26.44	33.58	27.91	26.92	28.40
NeRF-SH	27.05	25.06	32.28	28.06	26.66	27.82
PlenOctree from NeRF-SH	25.78	24.80	32.04	27.92	26.15	27.34
PlenOctree after fine-tuning	26.80	25.29	32.85	28.19	26.83	27.99

SSIM ↑

	Barn	Caterpillar	Family	Ignatius	Truck	Mean
NeRF (original)	0.750	0.860	0.932	0.920	0.860	0.864
NeRF	0.842	0.892	0.951	0.940	0.896	0.904
SRN	0.741	0.834	0.908	0.920	0.832	0.847
Neural Volumes	0.721	0.819	0.916	0.922	0.793	0.834
NSVF	0.823	0.900	0.954	0.930	0.895	0.900
NeRF-SH	0.838	0.891	0.949	0.940	0.895	0.902
PlenOctree from NeRF-SH	0.820	0.889	0.948	0.940	0.889	0.897
PlenOctree after fine-tuning	0.856	0.907	0.962	0.948	0.914	0.917

LPIPS ↓

	Barn	Caterpillar	Family	Ignatius	Truck	Mean
NeRF (original)	0.395	0.196	0.098	0.111	0.192	0.198
NeRF	0.286	0.189	0.092	0.102	0.173	0.168
SRN	0.448	0.278	0.134	0.128	0.266	0.251
Neural Volumes	0.479	0.280	0.111	0.117	0.312	0.260
NSVF	0.307	0.141	0.063	0.106	0.148	0.153
NeRF-SH	0.291	0.185	0.091	0.091	0.175	0.167
PlenOctree from NeRF-SH	0.296	0.188	0.094	0.092	0.180	0.170
PlenOctree after fine-tuning	0.226	0.148	0.069	0.080	0.130	0.131

FPS ↑

	Barn	Caterpillar	Family	Ignatius	Truck	Mean
NeRF (original)	0.007	0.007	0.007	0.007	0.007	0.007
NeRF	0.013	0.013	0.013	0.013	0.013	0.013
SRN	0.250	0.250	0.250	0.250	0.250	0.250
Neural Volumes	1.000	1.000	1.000	1.000	1.000	1.000
NSVF	10.74	5.415	2.625	6.062	5.886	6.146
NeRF-SH	0.015	0.015	0.015	0.015	0.015	0.015
PlenOctree (ours)	46.94	54.00	32.33	15.67	62.16	42.22

Table 4: **Per-scene quantitive results on Tanks&Temples dataset.**

PSNR ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Ours-1.9G	34.66	25.31	30.79	36.79	32.95	29.76	33.97	29.42	31.71
Ours-1.4G	34.66	25.30	30.82	36.36	32.96	29.75	33.98	29.29	31.64
Ours-0.4G	32.92	24.82	30.07	36.06	31.61	28.89	32.19	29.04	30.70
Ours-0.3G	32.03	24.10	29.42	34.46	30.25	28.44	30.78	27.36	29.60

GB ↓

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Ours-1.9G	0.830	1.240	1.791	2.674	2.067	3.682	0.442	2.689	1.93
Ours-1.4G	0.671	0.852	0.943	1.495	1.421	3.060	0.569	1.881	1.36
Ours-0.4G	0.176	0.350	0.287	0.419	0.499	0.295	0.327	1.195	0.44
Ours-0.3G	0.131	0.183	0.286	0.403	0.340	0.503	0.159	0.381	0.30

FPS ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Ours-1.9G	352.4	175.9	85.6	95.5	186.8	64.2	324.9	56.0	167.7
Ours-1.4G	399.7	222.2	147.3	163.5	247.9	68.0	393.8	75.4	214.7
Ours-0.4G	639.6	290.0	208.7	273.5	339.0	268.0	522.6	86.7	328.5
Ours-0.3G	767.6	424.1	203.8	271.7	443.6	189.1	796.4	181.1	409.7

Table 5: **Per-scene quantitive results on ORF conversion ablations.**

PSNR ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF-SH9	33.88	25.24	30.69	36.68	32.73	29.53	33.68	29.11	31.44
NeRF-SH16	33.98	25.17	30.72	36.75	32.77	29.95	34.04	29.21	31.57
NeRF-SH25	34.01	25.10	30.52	36.83	32.76	30.06	34.08	29.11	31.56
NeRF-SG25	34.08	25.40	31.21	36.92	32.93	29.77	34.31	29.28	31.74
PlenOctree-SH9	34.38	25.34	30.72	36.68	32.79	29.16	33.23	29.28	31.45
PlenOctree-SH16	34.66	25.31	30.79	36.79	32.95	29.76	33.97	29.42	31.71
PlenOctree-SH25	34.72	25.32	30.68	36.96	32.85	29.79	33.90	29.29	31.69
PlenOctree-SG25	34.37	25.52	31.16	36.67	32.98	29.41	33.63	29.32	31.63

SSIM ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF-SH9	0.973	0.928	0.968	0.978	0.966	0.948	0.984	0.864	0.951
NeRF-SH16	0.974	0.927	0.968	0.978	0.966	0.951	0.985	0.866	0.952
NeRF-SH25	0.973	0.926	0.967	0.978	0.966	0.952	0.985	0.864	0.951
NeRF-SG25	0.974	0.930	0.971	0.978	0.967	0.951	0.986	0.867	0.953
PlenOctree-SH9	0.980	0.934	0.970	0.982	0.970	0.950	0.984	0.881	0.956
PlenOctree-SH16	0.981	0.933	0.970	0.982	0.971	0.955	0.987	0.884	0.958
PlenOctree-SH25	0.981	0.935	0.971	0.983	0.971	0.955	0.987	0.883	0.958
PlenOctree-SG25	0.980	0.937	0.973	0.982	0.972	0.953	0.986	0.883	0.958

LPIPS ↓

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF-SH9	0.037	0.086	0.043	0.044	0.042	0.063	0.023	0.180	0.065
NeRF-SH16	0.037	0.087	0.039	0.041	0.041	0.060	0.021	0.177	0.063
NeRF-SH25	0.038	0.087	0.039	0.040	0.041	0.061	0.021	0.179	0.063
NeRF-SG25	0.036	0.083	0.034	0.042	0.041	0.060	0.020	0.176	0.062
PlenOctree-SH9	0.023	0.075	0.041	0.034	0.036	0.068	0.025	0.146	0.056
PlenOctree-SH16	0.022	0.076	0.038	0.032	0.034	0.059	0.017	0.144	0.053
PlenOctree-SH25	0.023	0.072	0.036	0.031	0.034	0.060	0.017	0.145	0.052
PlenOctree-SG25	0.023	0.069	0.034	0.033	0.033	0.064	0.019	0.144	0.052

GB ↓

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
PlenOctree-SH9	0.45	0.67	1.15	1.27	1.16	1.48	0.16	1.67	1.00
PlenOctree-SH16	0.83	1.24	1.79	2.67	2.07	3.68	0.44	2.69	1.93
PlenOctree-SH25	1.30	1.97	2.57	3.80	3.61	4.04	0.55	3.61	2.68
PlenOctree-SG25	1.03	1.68	2.43	2.66	2.66	4.44	0.49	2.71	2.26

FPS ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
PlenOctree-SH9	521.1	255.6	116.7	183.0	275.1	132.3	519.4	90.6	261.7
PlenOctree-SH16	352.4	175.9	85.6	95.5	186.8	64.2	324.9	56.0	167.7
PlenOctree-SH25	269.2	126.7	67.0	66.4	127.1	48.9	279.2	41.3	128.2
PlenOctree-SG25	306.6	151.9	74.1	104.3	153.3	51.0	294.2	69.6	150.6

Table 6: **Per-scene quantitive results on spherical basis function ablations.**