NGP-RT: Fusing Multi-Level Hash Features with Lightweight Attention for Real-Time Novel View Synthesis
Supplementary Material

Yubin Hu[1*†], Xiaoyang Guo[2*], Yang Xiao[3],
Jingwei Huang[4], and Yong-Jin Liu[1]

[1] BNRist, Department of Computer Science and Technology, Tsinghua University
[2] Horizon Robotics [3] Huawei Technologies [4] Game AI Center, Tencent Games

S1 Per-scene Results

Table S1: Per-scene comparison between 3D Gaussian Splattng [3], MERF [5] and our method on the Mip-NeRF 360 dataset.

	Method	Outdoor Scenes					Indoor Scenes				Mean
		bicycle	flower	garden	stump	tree	room	counter	kitchen	bonsai	
PSNR ↑	Gaussian-30K [3]	25.06	21.40	27.33	26.67	22.49	31.67	29.09	31.53	32.31	27.51
	Gaussian-7K [3]	23.39	20.32	26.05	25.57	22.06	29.53	27.26	29.23	29.80	25.91
	MERF [5]	**22.62**	**20.33**	25.58	**25.04**	**22.39**	29.28	25.82	27.42	28.68	25.24
	NGP-RT (Ours)	22.40	18.88	**26.22**	24.54	21.74	**29.75**	**27.27**	**28.95**	**31.04**	**25.64**
SSIM ↑	Gaussian-30K [3]	0.750	0.590	0.860	0.770	0.640	0.927	0.915	0.932	0.947	0.813
	Gaussian-7K [3]	0.640	0.510	0.810	0.720	0.590	0.903	0.887	0.910	0.928	0.766
	MERF [5]	0.595	**0.492**	0.763	0.677	**0.554**	0.874	0.819	0.842	0.884	0.722
	NGP-RT (Ours)	**0.610**	0.430	**0.820**	**0.690**	0.530	**0.894**	**0.852**	**0.889**	**0.930**	**0.737**
LPIPS ↓	Gaussian-30K [3]	0.244	0.360	0.122	0.243	0.347	0.197	0.183	0.116	0.180	0.211
	Gaussian-7K [3]	0.374	0.440	0.186	0.328	0.435	0.239	0.229	0.149	0.212	0.288
	MERF [5]	**0.371**	**0.406**	0.215	**0.309**	**0.414**	0.292	0.307	0.224	0.262	0.311
	NGP-RT (Ours)	0.400	0.485	**0.214**	0.366	0.449	**0.197**	**0.231**	**0.165**	**0.186**	**0.299**
Time(ms) ↓	Gaussian-30K [3]	19.65	10.02	16.67	11.54	11.21	9.65	9.12	10.99	7.30	11.79
	Gaussian-7K [3]	12.70	7.62	13.40	9.17	8.15	8.03	8.43	10.25	6.34	9.34
	MERF [5]	-	-	-	-	-	-	-	-	-	**8.40**
	NGP-RT (Ours)	9.00	11.70	8.72	10.23	10.29	7.97	10.15	7.37	7.85	9.25

We present a comparison of our method NGP-RT ($L = 2$, $L_C = 512$) with two recently prominent techniques, 3D Gaussian Splatting [3] and MERF [5], for real-time rendering of unbounded 360-degree scenes. The per-scene results are summarized in Table S1, including metrics such as PSNR, SSIM, LPIPS for rendering quality, and rendering time in *ms* at a resolution of 1080×1920.

Regarding the rendering time of MERF, we only provide the average time across all views calculated from the fps number reported in the original paper, as the released jax code[1] does not support real-time rendering of the baked

[1] https://github.com/google-research/google-research/tree/master/merf

MERF model. With the provided real-time rendering solution using web viewer, we cannot accurately measure the rendering time from the test views due to the upper bound of the rendering speed.

As for the rendering time of 3D Gaussian Splatting and NGP-RT, we measure them on the same Nvidia RTX 3090 GPU to ensure fairness. We evaluate 3D Gaussian Splatting with two configurations, 7K and 30K iterations, following the original paper. The Gaussian-30K version achieves better rendering quality with larger number of 3D gaussians, which leads to the slower rendering speed. As for the Gaussian-7K version, NGP-RT produces comparable rendering quality to it with similar rendering speed.

Different from our 1080×1920 resolution for evaluation, the original paper of 3D GS evaluates at the "reference resolution", which is specified as the resolution of *images_2* for indoor scenes (∼1000×1500) and the resolution of *images_4* for outdoor scenes (∼800×1200). To perform a more comprehensive comparison, we also evaluate NGP-RT at the same "reference resolution" and present the results in Table S2. The experimental results of 3D GS are directly copied from the original paper.

As demonstrated by the above comparisons, NGP-RT produces comparable rendering quality to Gaussian-7K with similar rendering speed, showing that our advancements allow NeRF-based methods to rival the performance of 3D Gaussian Splatting with similar model size (250MB for NGP-RT v.s. 523MB for Gaussian-7K).

S2 Implementation Details

S2.1 Network Design

Multi-level Hash Features. In NGP-RT, we set the resolution of the coarse-grained level to $L_C = 512$, and the resolutions of the fine-grained levels to 1024 and 2048 for $L = 2$. With each increase in L, we double the resolution of the finest level. As mentioned in Section 4.1, we employ an auxiliary NGP model to optimize the coarse-grained features and attention parameters. This auxiliary model comprises 6 coarse-grained resolution levels ranging from 16 to $L_C = 512$. Each coarse-grained level has a hash table with a maximum length of 2^{21} and a feature dimension of 4. After training and baking the features into feature grids, this auxiliary NGP model will be discarded. Regarding the L fine-grained resolution levels, we set the length of the hash tables to 2^{22} to accommodate diverse high-resolution features. The hash tables at the fine-grained levels consist of explicit 8-dimensional deferred NeRF features, as opposed to the implicit features used in the coarse-grained levels.

Forward Pass. For 3D points sampled along the casting ray of a training pixel, we extract coarse-grained hash features from different resolution levels. These features are then concatenated into a 24-dimensional vector. We feed this vector into a shallow MLP with 1 hidden layer containing 64 neurons.

Subsequently, the MLP decodes the concatenated features into a $(8 + 2L)$-dimensional coarse-grained feature. This feature includes a 1-dimensional density value $\tilde{\sigma}$, a 3-dimensional RGB color value $\tilde{\mathbf{c}}_\mathbf{d}$, a 4-dimensional feature vector $\tilde{\mathbf{f}}_\mathbf{s}$ for view-dependent colors, and a $2L$-dimensional set of attention parameters $\mathbf{a} = [\omega^1, \beta^1, \ldots, \omega^L, \beta^L]$ for the L fine-grained resolution levels. The attention parameters are used to aggregate the fine-grained hash features, while the density values are employed to accumulate the color features through volume rendering. After the alpha composition process, we utilize another MLP, denoted as MLPψ, to decode the view-dependent colors according to Eq. (5). MLPψ consists of two hidden layers, each with 64 neurons. In addition to the accumulated view-dependent features, MLPψ takes the embedding of the view direction as input. This view direction embedding is represented by a 16-dimensional vector that encodes a 4-degree spherical harmonics representation.

Table S2: Per-scene comparisons between 3D Gaussian Splattng [3] and our method using the resolution settings utilized in the 3D Gaussian Splatting paper on the Mip-NeRF 360 dataset.

	Method	Outdoor Scenes					Indoor Scenes				Mean
		bicycle	flower	garden	stump	tree	room	counter	kitchen	bonsai	
PSNR ↑	Gaussian-30K [3]	25.25	21.52	27.41	26.55	22.49	30.63	28.70	30.32	31.98	27.21
	Gaussian-7K [3]	23.60	20.52	26.25	25.71	22.09	28.14	26.71	28.55	28.85	25.60
	NGP-RT (Ours)	22.50	19.03	26.19	24.70	21.68	29.81	27.24	28.94	31.03	25.68
SSIM ↑	Gaussian-30K [3]	0.771	0.605	0.868	0.775	0.638	0.914	0.905	0.922	0.938	0.815
	Gaussian-7K [3]	0.675	0.525	0.836	0.728	0.598	0.884	0.873	0.900	0.910	0.770
	NGP-RT (Ours)	0.621	0.425	0.823	0.692	0.537	0.888	0.853	0.885	0.929	0.739
LPIPS ↓	Gaussian-30K [3]	0.205	0.336	0.103	0.210	0.317	0.220	0.204	0.129	0.205	0.214
	Gaussian-7K [3]	0.318	0.417	0.153	0.287	0.404	0.272	0.254	0.161	0.244	0.279
	NGP-RT (Ours)	0.391	0.475	0.215	0.360	0.437	0.201	0.231	0.162	0.187	0.295
FPS ↑	Gaussian-30K [3]	-	-	-	-	-	-	-	-	-	134
	Gaussian-7K [3]	-	-	-	-	-	-	-	-	-	160
	NGP-RT (Ours)	175	139	183	161	167	190	171	201	189	175

Activation Layers. In NGP-RT, we employ different activation functions and strategies for features of different modalities.

1) For the per-point density value σ, we apply the exponential function with truncated gradients for activation following [4].

2) For the per-point and per-level attention parameters $\{\omega^l, \beta^l\}$, we utilize a Sigmoid layer to normalize them to the range of $[0, 1]$.

3) As for features of the color modality, we first compute the RGB values for the training pixel using Eq. (5). Then, we apply a Sigmoid layer to normalize the color components to the range of $[0, 1]$.

Number of Feature Parameters. At the rendering stage, we discard the auxiliary NGP model and bake the coarse-grained features into a sparse feature grid $\tilde{\mathcal{F}}$. The dimensions of $\tilde{\mathcal{F}}$ are $L_C \times L_C \times L_C \times (8+2L)$, where L_C represents the coarse-grained resolution, and L denotes the number of fine-grained resolution levels. The sparsity ratio is around 2%, which is similar to that of MERF. Overall,

the number of feature parameters of the NGP-RT model in Table 1 is comparable to that in MERF, which contains a $512 \times 512 \times 512 \times 8$ coarse-grained feature grid with sparsity ratio of 2% and three $2048 \times 2048 \times 8$ feature planes. Thereby the comparisons between NGP-RT ($L = 2$, $L_C = 512$) and MERF in Table 1 and Table S1 are fair. We can calculate the number of parameters for both models as follows:

$$\begin{aligned} N_{\text{MERF}} &= 8 \times (0.02 \times 512^3 + 3 \times 2048^2) = 122.1\text{M}, \\ N_{\text{ours}} &= 12 \times 0.02 \times L_C^3 + 8 \times L \times 2^{22} = 99.32\text{M}. \end{aligned} \tag{S1}$$

S2.2 Loss Functions

We train NGP-RT with the Huber Loss [2] $\mathcal{L}_{color}$ of predicted color values and the regularizer $\mathcal{L}_{dist}$ proposed in Mip-NeRF 360 [1]. The overall loss function for each training pixel can be formulated as follows,

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{color} + \eta \mathcal{L}_{dist}, \\ \mathcal{L}_{color} &= \mathcal{L}_{Huber}(\mathbf{C}_{pred}, \mathbf{C}_{gt}), \\ \mathcal{L}_{dist} &= \sum_{i,j} \mathrm{w}_i \mathrm{w}_j \left| \frac{t_i + t_{i+1}}{2} - \frac{t_j + t_{j+1}}{2} \right| \\ &\quad + \frac{1}{3} \sum_i \mathrm{w}_i^2 (t_{i+1} - t_i), \end{aligned} \tag{S2}$$

where w_i denotes the rendering weight and t_i denotes the ray distance. In our experiments, we set $\eta = 0.01$ to adjust the strength of the regularization term and effectively regularize the underlying radiance field of NGP-RT.

We also observe that a high η leads to missing parts (e.g., petals and leaves) in the renderings of intricate small structures, resulting in degenerated rendering quality for outdoor scenes. As shown in Table S3, by tuning the weight η for each outdoor scene, we achieve comparable rendering quality to MERF while maintaining an acceptable average rendering speed (102 FPS) across all Mip-NeRF 360 scenes.

Table S3: Experimental results with tuned weights η for $\mathcal{L}_{dist}$.

	bicycle	flower	garden	stump	treehill	outdoor	Mip-NeRF 360
η	0.005	0.001	0.010	0.001	0.005	-	-
PSNR_{Ours}	**22.71**	19.86	**26.22**	24.93	**22.43**	**23.23**	**25.91**
PSNR_{MERF}	22.62	**20.33**	25.58	**25.04**	22.39	23.19	25.24

S2.3 Auto-tuning of Marching Step Size

During the training stage of NGP-RT, the regularization term $\mathcal{L}_{dist}$ gradually sparsifies the underlying radiance field of NGP-RT, which results in fewer sampled points along each casting ray. Such a reduction on the number of sampled

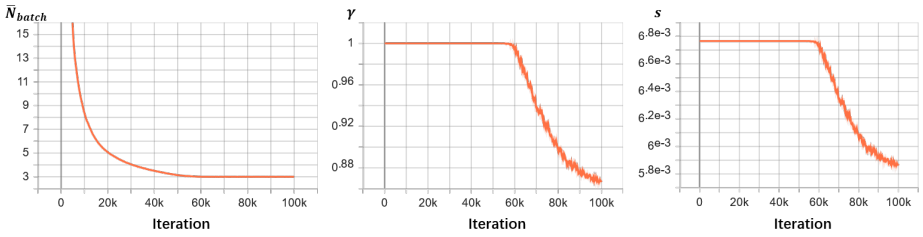

Fig. S1: Illustration of the optimization process of $\bar{N}_{batch}$, γ and s.

points alleviates the burden of frequent feature access and helps to achieve real-time rendering. However, the strength of regularization can make our model get stuck into suboptimals with only one even no sampled points on the casting rays. To prevent the collapse into an excessively small number of sample points, we introduce an auto-tuning strategy of ray marching step size when the number of sampled points hits a minimum value N_{min} at the training stage. Specifically, we count the average number of sampled points $\bar{N}_{batch}$ across all casting rays inside the training batch. According to $\bar{N}_{batch}$ of the last training batch, we design a strategy to automatically tune a scaler γ for the marching step size to avoid over sparsification. The auto-tuning strategy can be formulated as follows,

$$\begin{aligned} s &= \gamma \cdot s_0, \\ \gamma &= \begin{cases} \min\{1.0,\ (1+\kappa)\cdot\gamma\}, & \text{if } \bar{N}_{batch} \geq N_{min}\ , \\ \max\{\gamma_{min}, (1-\kappa)\cdot\gamma\}, & \text{otherwise,} \end{cases} \end{aligned} \tag{S3}$$

where s_0 denotes the base step size varying according to the specially designed contraction function, s denotes the scaled step size which is utilized in ray marching, and γ_{min} denotes the minimum value for the scaler γ. In our experiments, we set the base step size inside our $[-1,1]^3$ ROI grid to $\frac{2\sqrt{3}}{512}$, $N_{min} = 3$ to reserve a small number of marching points, $\kappa = 0.001$ for the progressive multiplier, $\gamma_{min} = 0.2217$ for the minimum value of scaler γ, and the initial value of γ to 1.0. We plot the optimization progress of $\bar{N}_{batch}$, γ and s in Figure S1, which reflects the effectiveness of our auto-tuning strategy.

S2.4 Optimization

We train NGP-RT for 100k iterations using an Adam optimizer with an linearly decaying learning rate. The learnig rate is warmed up during the first 1k iterations where it is increased from 0 to 0.01, and then decay to 0 in the following iterations. The weighting parameter η for $\mathcal{L}_{dist}$ is linearly warmed up in the first 50k iterations from 0 to 0.01, and then remains as a constant value. Adam's hyperparameters β_1, β_2 and ϵ are set to 0.9, 0.99 and 1e-8, respectively. When collecting the training batch of rays, we expect the total number of sample points to be less than 2^{21} and the total number of rays to be less than 8000.

S2.5 Real-time Rendering

At the rendering stage, we utilize several techniques to further improve the rendering speed.

1) We employ the visibility culling and occupancy dilation techniques similar to MERF [5] to generate the occupancy grid for rendering, which contains much less occupied positions compare to that utilized at the training stage. The occupancy grid is then down-sampled to 5 resolution levels including 512^3, 256^3, 128^3, 64^3, and 32^3 for fast occupancy check.

2) We stop the ray marching of a casting ray when the transmittance value of a sampled point is less than 2e-3, in order to avoiding redundant samplings in the empty space behind the visible surfaces.

3) We utilize the fully fused MLP implemented in tiny-cuda-cnn[2] to accelerate the execution of our view-dependent MLP_ψ.

4) We implement a CUDA-based C++ program to render NGP-RT at the fast speed. Our implementation utilizes aligned memory to increase the efficiency of memory fetching performed by CUDA kernels.

S3 Analysis of Hash Feature Attention

S3.1 Example of Hash Collision Reduction

To demonstrate the collision reduction effects of our attention mechanism, we examine two 3D points in the bonsai scene: Point P_A on the detailed flowers and Point P_B on the flat floor.

At the 2048 resolution, two corners of their enclosing voxels collide after the hash function. Therefore both points influence the learning of the same hash entry H. As shown in Table S4, P_A and P_B have similar influence weight ξ on H due to the similar point-to-corner distance. Similar ξ values result in hash collision and impede the optimization of H due to averaged gradient directions.

Differently, NGP-RT disambiguates P_A and P_B using different ω and β values. This offloads the influence of P_B to another resolution level and alleviates the hash collision. As a result, the improved allocation of hash features contributes to a notable enhancement in rendering the bonsai scene. Please refer to Figures S2 to S5 for more visualizations of the attention weights allocation.

Table S4: Pre-attention weights ξ and post-attention weights ξ'.

P_X	Resolution	ξ_X	$\omega(X)$	$\beta(X)$	$\xi'_{X,\omega} = \xi_X \cdot \omega(X)$	$\xi'_{X,\beta} = \xi_X \cdot \beta(X)$
P_A	2048	0.436	**0.257**	**0.636**	**0.112**	**0.277**
P_B	2048	**0.515**	0.033	0.164	0.017	0.084
P_B	1024	0.604	0.349	0.671	0.211	0.405

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

S3.2 Visualization of Attention Parameters

Besides the visualizations of multi-level color contributions in Figure 6, we further visualize the post-Sigmoid attention parameters of NGP-RT ($L = 4$) in Figures S2 to S5 to illustrate the functionality of the proposed lightweight attention mechanism. We generate the visualization results by applying volume rendering to the attention weights.

As shown in the visualizations, different fine-grained levels focus on different regions with the help of our lightweight attention mechanism, which sufficiently exploits the expressive power of multi-level hash features. We also notice that the regions-of-interest for attention parameters ω and β are different, indicating that it is necessary to employ separate attention parameters for the density values and color-related features.

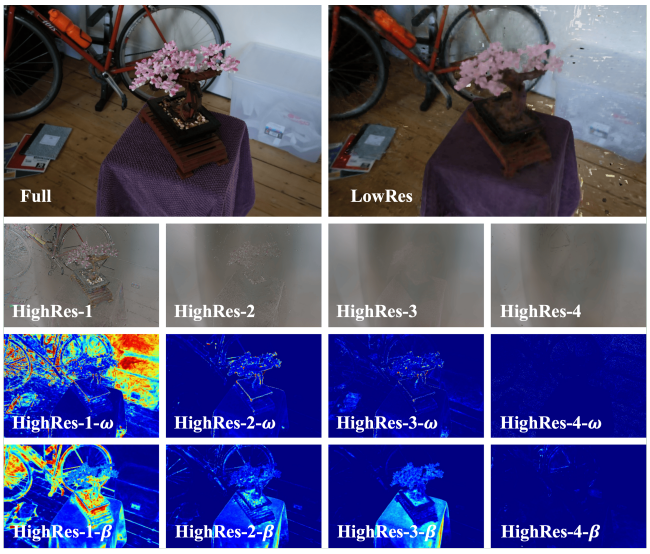

Fig. S2: Visualization of the color decomposition and multi-level attention parameters for the density values and color-related features.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 5460–5469. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.00539, https://doi.org/10.1109/CVPR52688.2022.00539 4

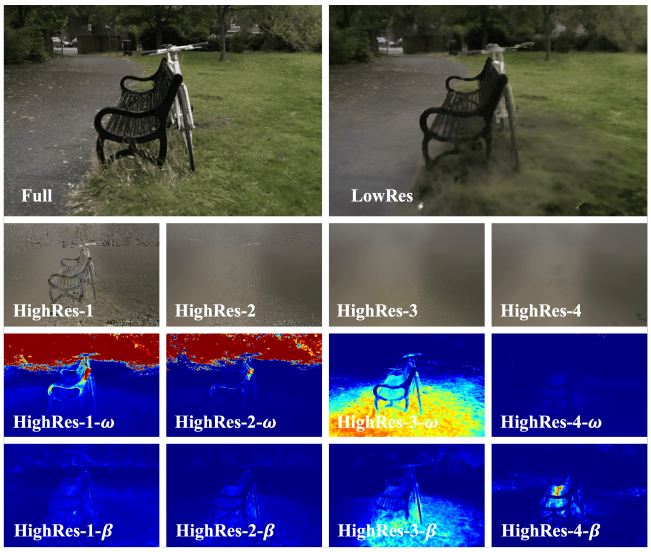

Fig. S3: Visualization of the color decomposition and multi-level attention parameters for the density values and color-related features.

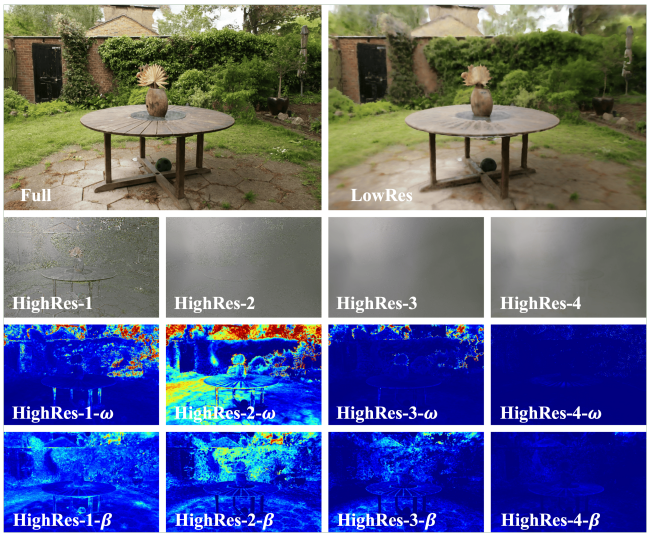

Fig. S4: Visualization of the color decomposition and multi-level attention parameters for the density values and color-related features.

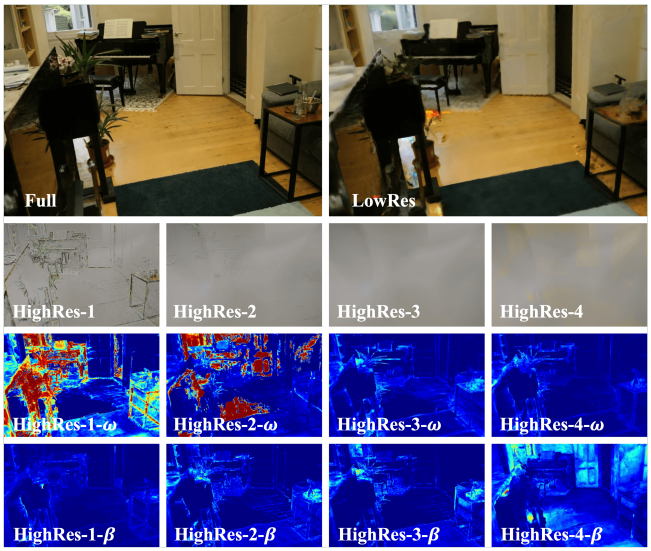

Fig. S5: Visualization of the color decomposition and multi-level attention parameters for the density values and color-related features.

2. Huber, P.J.: Robust Estimation of a Location Parameter. The Annals of Mathematical Statistics **35**(1), 73 – 101 (1964). https://doi.org/10.1214/aoms/1177703732, https://doi.org/10.1214/aoms/1177703732 4
3. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139:1–139:14 (2023). https://doi.org/10.1145/3592433, https://doi.org/10.1145/3592433 1, 3
4. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022) 3
5. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., Hedman, P.: Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. ACM Transactions on Graphics (TOG) **42**(4), 1–12 (2023) 1, 6