Digging into Radiance Grid for Real-Time View Synthesis with Detail Preservation

Jian Zhang, Jinchi Huang, Bowen Cai, Huan Fu, Mingming Gong, Chaohui Wang, Jiaming Wang, Hongchen Luo, Rongfei Jia, Binqiang Zhao, et al.

Digging into Radiance Grid for Real-Time View Synthesis with Detail Preservation

Jian Zhang[1*] Jinchi Huang[1*] Bowen Cai[1*] Huan Fu[1†]
Mingming Gong[2] Chaohui Wang[3] Jiaming Wang[1] Hongchen Luo[1]
Rongfei Jia[1] Binqiang Zhao[1] Xing Tang[1]

[1] Tao Technology Department, Alibaba Group
[2] School of Mathematics and Statistics Melbourne Centre, University of Melbourne
[3] LIGM, Univ Gustave Eiffel, CNRS, ENPC, France

Abstract. Neural Radiance Fields (NeRF) [30] series are impressive in representing scenes and synthesizing high-quality novel views. However, most previous works fail to preserve texture details and suffer from slow training speed. A recent method SNeRG [10] demonstrates that baking a trained NeRF as a Sparse Neural Radiance Grid enables real-time view synthesis with slight scarification of rendering quality. In this paper, we dig into the Radiance Grid representation and present a set of improvements, which together result in significantly boosted performance in terms of both speed and quality. First, we propose an HieRarchical Sparse Radiance Grid (HrSRG) representation that has higher voxel resolution for informative spaces and fewer voxels for other spaces. HrSRG leverages a hierarchical voxel grid building process inspired by [29], and can describe a scene at high resolution without excessive memory footprint. Furthermore, we show that directly optimizing the voxel grid leads to surprisingly good texture details in rendered images. This direct optimization is memory-friendly and requires multiple orders of magnitude less time than conventional NeRFs as it only involves a tiny MLP. Finally, we find that a critical factor that prevents fine details restoration is the misaligned 2D pixels among images caused by camera pose errors. We propose to use the perceptual loss to add tolerance to misalignments, leading to the improved visual quality of rendered images.

Keywords: 3D representation, view synthesis, real-time rendering

1 Introduction

Neural rendering has emerged as a promising new avenue towards controllable image and video generation of photo-realistic virtual worlds. In the past two years, there has been an explosive interest in neural volumetric representations, such as Neural Radiance Fields (NeRF) [30], due to their superiority in synthesizing photo-realistic novel views of scenes. NeRF takes a deep multilayer

*: These authors contribute equally to this work.
†: Corresponding author.

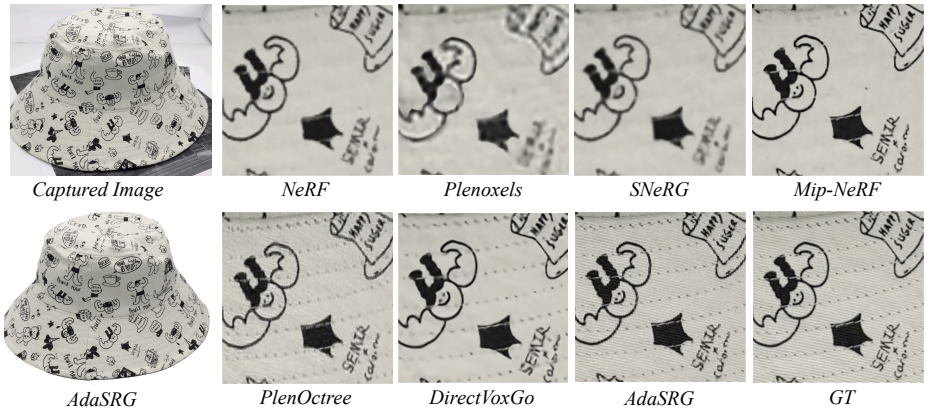

Fig. 1. The rendered image by our approach (HrSRG) is perceptually better than others while contains accurate fine details. Achieving such a quality, HrSRG brings the real-time rendering practice and requires less storage space compared to SNeRG [10] and PlenOctree [54] (See Fig. 2). Zoom in for more details.

perceptron (MLP) that can map from 3D points to their volume densities and view-dependent emitted colors to represent a scene.

The main obstacle to the practical deployment of NeRF is the slow inference and training speed. It takes roughly 2 minutes to render a 1920×1080 (1080p) image on a high-performing GPU. Reconstructing a scene from high-definition (HD) images would take more than 48 hours. One of the main reasons is that NeRF would query the MLP network hundreds of times even for rendering a single ray. Recent advances show that caching pre-computed values would bring real-time rendering experiences [10, 54]; however, their compacted representations require a large memory footprint and often impose a quality loss of 1∼2dB compared to NeRF. Towards the slow training speed issue, efforts have been made to learn image-based rendering by leveraging neural radiance fields [4, 49]. Still, their formulations are not compatible with published real-time rendering approaches yet since they rely on nearby source images when performing rendering. In addition, they are overly sensitive to the pose distances between source and target views. Several concurrent works [43, 53, 31] study direct voxel grid optimization for super-fast training.

Another relatively understudied open problem is that NeRF fails to recover texture characteristics or fine details. To the best of our knowledge, Mip-NeRF [2] have discussed and partially remedied blurring and aliasing issues. Still, we find it unable to preserve fine details of real scenes. In fact, NeRF and its variants commonly generate excessively blurred in close-up views when representing real scenes instead of synthetic scenes. One possible reason is there are inevitable misaligned 2D pixels among images caused by camera pose errors. Unfortunately, neither NeRF nor its variants can handle this issue well.

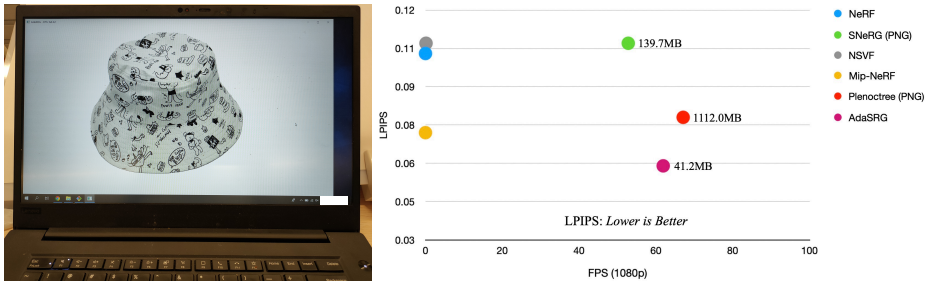

Fig. 2. LPIPS *vs.* FPS. We study the speed-quality tradeoff on Real 360° Objects (1080p images). HrSRG surpasses the two real-time methods [10, 54] on rendering speed, rendering quality, and storage space requirements. Obtaining an optimized HrSRG requires about 5 hours (vs. 1-2 days for NeRF) on a V100 GPU. We measure the rendering speed on a Lenovo ThinkPad P1 Gen 2 notebook. It has a 4GB NVIDIA Quadro T1000 mobile graphics card. Zoom in for the runtime texture details.

In the paper, we dig into a Radiance Grid representation raised by SNeRG [10], which enjoys real-time synthesis at the cost of rendering quality. We present a series of improvements that lead to significantly boosted performance in terms of both speed and quality. First, we argue that baking a uniformly high-resolution voxel grid to represent a scene is memory intensive and not flexible. We notice that, for a ray, only the points sampled around the surfaces contributes to its color rendering. Thus, we introduce an *HieRarchical Sparse Radiance Grid* (HrSRG) representation that has higher voxel resolution around informative spaces and shows lower voxel resolution for other 3D spaces. We produce HrSRG for a scene by exploiting an hierarchical voxel grid building process leveraging [29]. A voxel contains volume density, radiance, and a three-dimensional feature vector. Second, in light of 2D texture atlas [11], we treat voxel values as parameters and directly optimize them, leading to surprisingly good texture characteristics in the rendered images. Finally, NeRF often cannot produce images with excellent quality for real scenes since per-pixel losses such as ℓ_p-norm neither address the aforementioned misalignment issue nor consider the perceptual quality. Luckily, training HrSRG is memory-friendly, which allows us to learn more than 200K rays simultaneously in one single iteration. We thus incorporate a perceptual loss [14] to improve the clarity of rendered images further and preserve more fine details. Our experiments show fitting a scene through HrSRG requires takes about six hours, requiring multiple orders of magnitude less time than previous high-performing NeRFs.

In summary, our main contributions are as follows:

- We present an HieRarchical Sparse Radiance Grid (HrSRG) representation leveraging a hierarchical voxel grid building process. It enables representing a scene at high resolution without excessive memory footprint.
- We show that it is possible to recover a scene's texture characteristics by directly optimizing its radiance grid.

- We introduce a perceptual loss to address the misalignment issue caused by camera pose errors. It can further improves the clarity of rendered images and preserve more fine details.
- Experiments demonstrate our HrSRG supports real-time rendering and accelerated training while surpasses other methods on rendering quality and required memory footprint. See Fig. 1 and Fig. 2.

2 Related Work

The computer vision and graphics communities have put tremendous efforts into studying scene representation and novel view synthesis [5, 8, 21, 9, 45, 39, 18, 26, 15, 47, 42]. Recently, Neural Radiance Fields (NeRF) [30] have emerged as a promising technique for learning to represent 3D scenes and synthesizing novel views of the scene in impressive quality. Due to its superior performance, it has been extended to model deformable objects [6, 34], dynamic scenes [33, 22], and transparent objects [12]. Nevertheless, the vanilla NeRF fails to reconstruct fine details of scenes and suffers from slow training and inference speed.

Accelerated NeRF Training. Optimizing a NeRF to represent a scene typically takes around 1–2 days. To accelerate training, major efforts have been made to incorporate multi-view geometry into NeRF and utilize depths and pixel-wise correspondences as priors to guide the optimization process [46, 55, 52, 4, 44, 41, 23]. For example, GRF [46] and PixelNeRF [55] learn dense local features and retrieve feature vectors from each input image for an interested 3d point. NerfingMVS [52] predicts dense depth as priors to guide volume sampling in NeRF. It also exploits uncertainty based on multi-view consistency to determine the depth ranges of each ray. These image-based rendering approaches are not compatible with NeRF's real-time rendering techniques yet. Some other approaches [44, 41] study weights initialization by exploring meta-learning for neural representations, which result in faster convergence during optimization.

Real-time Volume Rendering. Due to the expensive sampling mechanism and costly neural network computations, rendering a NeRF is extremely slow. To speed up the rendering process, Decomposed Radiance Fields [35] and KiloNeRF [36] spatially decompose a scene into multiple cells to reduce the inference complexity. AutoInt [24] and DoNeRF [32] learns to reduce the samples for each queried ray. Other efforts have shown faster rendering even real-time rendering experiences can be reached by catching pre-computed values [10, 54, 7]. Especially, SNeRG [10] presents a deferred NeRF architecture so that it can store as many as NeRF values into a radiance grid data structure. PlenOctree [54] explores SH values to encode view-dependent effects and study octree representations for real-time view synthesis.

Preserving Details. A few works observe that NeRF, in its vanilla formulation, is limited in recovering fine details. The previous best practice is from Mip-NeRF

[2]. It rendered anti-aliased conical frustums instead of rays to provide a partial remedy to the aliasing and blurring issues. Other works such as NerfMM [51] and BARF [23] learn to optimize camera poses to reduce the adverse impacts of imperfect camera poses. NeRF-ID [1] focuses on improving the hierarchical volume sampling technique of NeRF. However, none of these methods can preserve fine details for real scenes.

Relation to Plenoxels [53], DirectVoxGO [43], and Instant-NGP [31]. These are some great concurrent works that study direct voxel grid optimization for super-fast NeRF training [53, 43]. Still, there are some difference betweens HrSRG and these works. Higher voxel resolution would better describe local details of scenes, but training a high-resolution voxel grid is memory intensive. For example, optimizing a 1024^3 voxel grid often leads to OMM errors on modern GPUs. Besides, high-resolution voxel grid requires large hardware storage spaces. [43, 53] provide a solution by utilizing the "Trilinear Interpolation (Tri. Interp.)" operation for low-resolution voxel grids. However, incorporating "Tri. Interp." would degrade the rendering speed (about 3x). Our HrSRG presents to adaptively consider voxel resolutions for different 3D spaces of a scene. It can describe a scene at high resolution without excessive memory footprint. Instant-NGP [31] introduces a great hash encoding approach that can train NeRF in 5 minutes while preserving the texture details. In contrast, HrSRG directly optimize the color values (or 3D texture atlas) inspired by the 2D texture atlas optimization success [11].

3 Method

Beyond NeRF, our primary goal is to design a representation or approach that (1) can recover scenes' texture characteristics for scenes, (2) enables real-time rendering of 1080p images on conventional devices, (3) accelerates the training procedure, and (4) reduces the hardware storage space requirement. This section presents our solution HrSRG and explains how it can open a potent avenue towards these goals.

3.1 Review of NeRF

NeRF [30] takes a continuous volumetric function parameterized by a deep MLP to represent a scene. The MLP inputs a single 5D coordinates (x, y, z, θ, ϕ), *i.e.*, a 3D location $\mathrm{x} = (x, y, z)$ and 2D viewing direction $\mathrm{d} = (\theta, \phi)$, and outputs a radiance $\mathrm{c} = (r, g, b)$ and volume density σ at this location. The formulation is expressed as:

$$\mathrm{c}, \sigma = \mathrm{MLP}_{\Theta}(\mathrm{x}, \mathrm{d}), \tag{1}$$

where Θ denotes the MLP's weights that to be learned.

To render a 2D pixel, NeRF casts a ray r from its corresponding camera center o along the direction d passing through the pixel's center. Then, it samples N

points along the ray, and approximate a volume rendering integral [27] to obtain the pixel's color $\hat{C}(\mathrm{r})$:

$$\hat{C}(\mathrm{r}) = \sum_{i=1}^{N} w_i \mathrm{c}_i, \tag{2}$$

$$w_i = T_i(1 - \exp(-\sigma_i \delta_i)), \tag{3}$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$ is the accumulated transmittance along r, and δ_i denotes the distance of two consecutive samples.

NeRF simply minimizes the mean squared errors (MSE) between the rendered and true pixel colors ($\hat{C}(\mathrm{r})$ and $C(\mathrm{r})$) to learn its MLP:

$$\mathcal{L}_r = \sum_{\mathrm{r} \in \mathcal{R}} \|\hat{C}(\mathrm{r}) - C(\mathrm{r})\|, \tag{4}$$

where $\mathcal{R}$ is the sampled rays in each batch. In practice, NeRF simultaneously optimizes a "coarse" network and a "fine" network that have the same architectures. The two networks (or MLPs) allow NeRF to perform hierarchical volume sampling to secure better rendering quality.

3.2 A Deferred NeRF Variant (Def-NeRF†)

As mentioned in Sec. 1, real-time rendering is possible by caching as many as precomputed values. The volume density can be easily stored, but it is non-trivial to catch the emitted colors because NeRF relies on (θ, ϕ) and the geometry feature mapping from (x, y, z) to encode view-dependent effects. A subsequent work SNeRG [10] provides a remedy by introducing a deferred NeRF architecture. It contains a deep MLP that maps from a 3D location to the diffuse color c, 4-dimension feature vector v, volume density σ at this location.

$$\mathrm{c}, \sigma, \mathrm{v} = \mathrm{MLP}_{\Theta}(\mathrm{x}, \mathrm{d}). \tag{5}$$

Then, for a ray r, it passes the accumulated feature vector $V(\mathrm{r})$ and viewing direction d to a tiny MLP with parameters Φ to produce a specular color (or view-dependent residual). The final color $\hat{C}(\mathrm{r})$ is obtained by an addition of $\mathrm{MLP}_{\Phi}(V(\mathrm{r}), \mathrm{d})$ and the accumulated diffuse color $\hat{C}_d(\mathrm{r})$:

$$\begin{aligned} \hat{C}(\mathrm{r}) &= \mathrm{MLP}_{\Phi}(V(\mathrm{r}), \mathrm{d}) + \hat{C}_d(\mathrm{r}), \\ \hat{C}_d(\mathrm{r}) &= \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathrm{c}_i. \end{aligned} \tag{6}$$

After training, this formulation enables baking diffuse colors, 4-dimension feature vectors, and opacities within a 3D voxel grid data structure.

Def-NeRF† in this paper has slight differences in two aspects compared to the vanilla Deferred NeRF. First, we initialize the opacity prediction MLP from a non-converged NeRF. We incorporate a background regularization term [26]

$$\mathcal{L}_{BG} = -\sum(\log(T_{BG}) + \log(1 - T_{BG})). \tag{7}$$

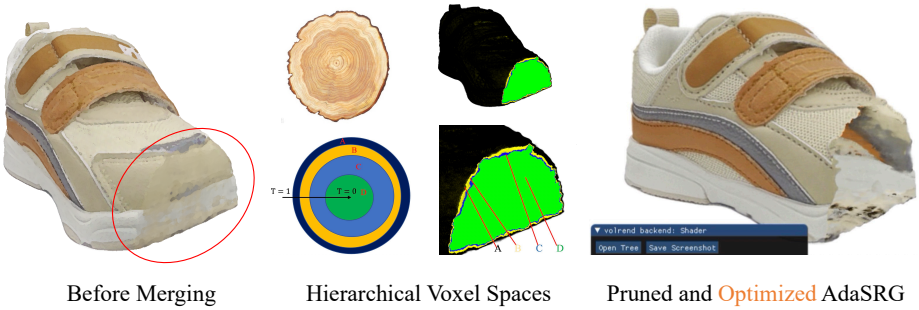

Fig. 3. Constructing HrSRG. We bake a non-converged Deferred NeRF Variant (Sec. 3.2) into a Adaptive Sparse Radiance Grid structure. We refer to Sec. 3.3 for how we produce the HrSRG representation by exploiting the hierarchical voxel spaces. A voxel in HrSRG contains volume density σ, diffuse color c, and a 3-dimension feature v. Optimizing HrSRG means we treat these values as parameters to be learned (See Sec. 3.4). Zoom in for a better view.

It ensures NeRF reconstructs the object of interest. We regard the pre-trained MLP as a backbone, and fixed its parameters while learning the diffuse color c, feature vector v, and volume density σ. Second, we find learn a 3-dimension feature vector is sufficient to preserve most of the high-frequency lighting details.

Our experiments show training Def-NeRF† several epochs is sufficient as the following HrSRG optimization stage is robust to an imperfect initialization. It is worth mentioning that we surprisingly find, for some cases, the specular feature v and the emitted color c can be learned from scratch in Sec. 3.4. The more important thing is to give a good geometry (or volume density σ) initialization.

3.3 Adaptive Sparse Radiance Grid (HrSRG)

We first bake Def-NeRF† into a N^3 voxle grid to represent a scene. A voxel contains opacity (or volume density), diffuse color, and a 3-dimensional specular feature. For a specific voxel, we randomly cast K rays from K origins passing through the voxel, and compute their accumulated transmittance values (See T_i in Eqn. 2). With this operation, we can assign each voxel a maximum transmittance value. Then, we define four transmittance intervals (or 3D spaces) A, B, C, and D as shown in Fig. 3 (*Middle*) based on the transmittance thresholds τ_1, τ_2, and τ_3, respectively. We find from Fig. 3 that space A contains most of the high-frequency texture details while other spaces contribute slightly to the base color in the rendering process. This observation motives us to utilize higher voxel resolution for space A, and lower voxel resolution for spaces B, C, and D. For the purpose, we introduce the merging and splitting operations leveraging the octree data structure [28, 54, 29].

Merging. By converting the N^3 voxle grid to a tree structure with maximum depth L, we have a transmittance value for each node and leaf. For each node

that is with depth L_i and has eight leaves, if the maximum transmittance value among itself and its leaves is below τ, we delete the eight leaves. We recursively consider depth L_1, L_2, and L_3 with thresholds τ_3, τ_2, and τ_1, respectively. As a results, we capture the approximate voxel resolutions $(N/2)^3$, $(N/4)^3$, $(N/8)^3$ for spaces B, C, and D, respectively.

Splitting. After the merging operation, if a leaf is with depth L, we take it as a parent of eight new leaves. A newly created leaf has the same voxel values (*i.e.*, $(\mathrm{c}, \sigma, \mathrm{v})$) as its parent. With this splitting operation, the space A would have a voxel resolution of $(2N)^3$.

In this paper, the hyperparameters are set to 512, 100, 0.3, 0.01, 0.001 for N, K, τ_1, τ_2, and τ_3, respectively. The HrSRG construction process takes about 30 minutes on average. One may capture more 3D space levels and tune τ for better performance in term of both memory footprint and rendering quality.

3.4 Optimizing HrSRG

We now have the HrSRG representation and a tiny MLP with weights Φ. We can treat $\{(\mathrm{c}, \sigma, \mathrm{v})\}$ as parameters and directly optimize them like Φ since the volume rendering process is differentiable with respect to these values. Note that, SNeRG [10] only fine-tunes its MLP_Φ after obtaining its sparse radiance grid.

Perceptual Loss. NeRF often yields blurry renderings for real scenes. A possible reason is that there are misalignment errors in estimated camera poses as discussed in [11]. Per-pixel losses such as ℓ_p-norm overlook this issue. Besides, ℓ_p-norm fails to encourage high-frequency crispness, thus in many cases would result in perceptually unsatisfying solutions with overly smooth textures, as analyzed in [20, 14, 13, 19]. [11] has demonstrated a PatchGAN loss [13] can well tackle these two problems. Here, we simply introduce a perceptual loss [14]. Experiments show it tolerates the misalignment issue and can largely improve the perceptual quality of rendered images. It is worth mentioning that it is non-trivial for conventional NeRFs to do so since training them is exquisitely memory-intensive. Fortunately, optimizing HrSRG is memory-friendly as it does not involve a large MLP. It allows us simultaneously learn more than 200K rays (*vs.* 6K for NeRF) in one single iteration. The perceptual loss encourages the rendered image $\hat{I}$ to be perceptually similar to its ground-truth image I by matching their semantic features:

$$\mathcal{L}_p = \frac{\lambda_p}{H_j * W_j} \sum_{h,w} \|\phi_j(\hat{I}) - \phi_j(I)\|, \tag{8}$$

where ϕ_j is the output of the jth convolution block of a pre-trained VGG-19 network [40], H_j and W_j are the spatial dimensions of the feature maps. In our experiments, we render a 384×384 image patch per-iteration during training. We consider the first three convolution blocks and set the hyperparameter λ_p

Table 1. Comparisons on Real Scenes. HrSRG achieves competitive or best performance. Especially on the Real 360° Objects dataset, it outperforms the compared methods by a large margin over the image quality metric LPIPS. It is worth mentioning that though HrSRG captures more details and produces perceptually better visual quality as shown in Fig. 1 and Fig. 4, it cannot yield a significant improvement on PSNR. As stated in [56, 14], PSNR might fail to account for many nuances of human perception. We find that LPIPS might measure the perceptual quality and texture details better. SNeRG [10] has the voxel resolution of 1000^3 (the default hyperparameter).

	Tanks and Temples [17]			Real 360° Objects		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [30]	27.94	0.904	0.168	26.42	0.911	0.103
NSVF [25]	28.40	0.901	0.155	28.48	0.908	0.107
SNeRG [10]	25.39	0.904	0.186	28.37	0.910	0.106
PlenOctree [54]	27.99	0.917	0.131	28.44	0.925	0.077
Mip-NeRF [2]	27.68	0.918	0.113	**29.44**	0.920	0.072
DirectVoxGo [43]	**28.41**	0.911	0.155	28.75	0.917	0.079
Plenoxels [53]	27.41	0.906	0.162	28.06	0.905	0.109
HrSRG	27.33	**0.919**	**0.099**	29.05	**0.927**	**0.059**

to 0.01. It takes about 80 seconds to optimize HrSRG for 100 iterations (or to optimize 2.0×10^7 rays) on a single V100 GPU.

3.5 Further Pruning and Real-Time Rendering

On the Synthetic-NeRF dataset [30], our uncompressed HrSRG require an average storage space of 210M (See Table 2). Note that, it has an approximate voxel resolution of 1,024 at 3D space A in Fig. 3. To minimize the file size, SNeRG [10] applies the PNG compression technique by flattening and reshaping the features and emitted colors into the image planes. The voxel values here have been pre-quantified to 8 bits. There is a uncompressing step that we need to reshape the PNG image to its original data structure when performing rendering. It is OK for computers but is not friendly to mobile devices as the uncompressing step might require many computation resources. We utilize the median cut algorithm to further prune our tree structure [54, 3].

Finally, we obtain a compacted HrSRG of about 40M, an acceptable size for transmission over the internet. The compacted HrSRG can be directly loaded without any uncompressing computations. HrSRG enables rendering 1080p images in unprecedented quality at 61 FPS (*vs.* 67 FPS of PlenOctree) on a laptop GPU. Our main bottleneck is the tiny MLP which targets encoding view-dependent effects. Removing it can bring an improved real-time rendering experience (up to ∼75 FPS) with a slight quality loss.

4 Experiments

Implementation Details. As discussed before, our approach contains a Def-NeRF† pre-training stage and a HrSRG optimization stage. For Def-NeRF†, we

Table 2. Comparisons with Baselines. Opt. is the training time. Our approach can bring the real-time rendering practice while preserving fine details of scenes. Our unpruned HrSRG already requires small hardware storage space. Applying the median cut (Med. Cut) algorithm [54, 3] to HrSRG results in approximately 6x compression rates, *i.e.*. The PNG compression technique can reduce the file size by more than 10 times, but suffers from a decompressing operation [54, 10]. Our rendering speed is slightly slower than PlenOctree because HrSRG contains a tiny specular MLP.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	MB ↓	FPS ↑	Opt. ↓
Synthetic-NeRF@800^2 [30]						
NeRF [30]	**31.69**	0.953	0.068	-	0.03	>1 day
SNeRG (PNG) [10]	30.38	0.950	**0.071**	86.7	87.1	>1 day
PlenOctree-512 (PNG) [54]	31.67	**0.957**	0.053	340.2	**129.8**	>1 day
HrSRG (w/o Med. Cut)	31.06	0.954	0.051	210.4	110.2	6h
HrSRG	31.04	0.954	0.051	**35.8**	113.4	6h
Real 360° Objects@1080p						
NeRF [30]	26.42	0.911	0.103	-	0.01	>1 day
SNeRG (PNG) [10]	28.14	0.902	0.107	139.7	52.8	>1 day
PlenOctree-1024 (PNG) [54]	28.39	0.923	0.078	1112.0	**67.1**	>1 day
HrSRG (w/o Med. Cut)	**29.07**	**0.927**	**0.059**	250.7	60.3	5h
HrSRG	29.05	0.927	0.059	**41.2**	61.9	5h

pre-cache all the training rays. Then, we train the vanilla NeRF for 5 epochs with a batch size of 8,192. As stated in Sec. 3.2, we take the pre-trained MLP of NeRF as a backbone, and fixed its parameters to learn the tiny specular MLP for another 5 epochs. The learning rate begins at 5×10^{-3} and linearly decays to 5×10^{-5} over the training epochs. For real scenes, we resize the 1080p images to 960×540 and train Def-NeRF† with mixed precision. When optimizing HrSRG, we randomly render a 384×384 image patch per iteration. We train HrSRG in 45 (about 1.4 minutes a epoch) epochs and decay the learning rate from 5×10^{-2} to 5×10^{-5} over the course of optimization. We use the Adam optimizer [16] for both the two training stages with default hyperparameters. It takes about 5.5 hours (*vs.* 1-2 days for NeRF) on average to fitting a Synthetic-NeRF [30] scene on a single 16GB V100 GPU. For octree things and the web renderer, we borrow the PlenOctree codes [54]. We implement the view-dependence MLP_{Φ} in the WebGL shader.

Datasets. We conduct experiments on three datasets, including Synthetic-NeRF [30], a subset of Tanks and Temples [17], and our Real 360° Objects dataset. Synthetic-NeRF contains eight objects with rendered 800×800 images, and the ground truth camera poses. There are 300 images per object, 100 for training and 200 for testing. We use the same Tanks and Temples subset as NSVF [25]. It provides five real scenes with 1080p images, their manually labeled masks, and known camera poses. To better study fine details of real scenes, we capture

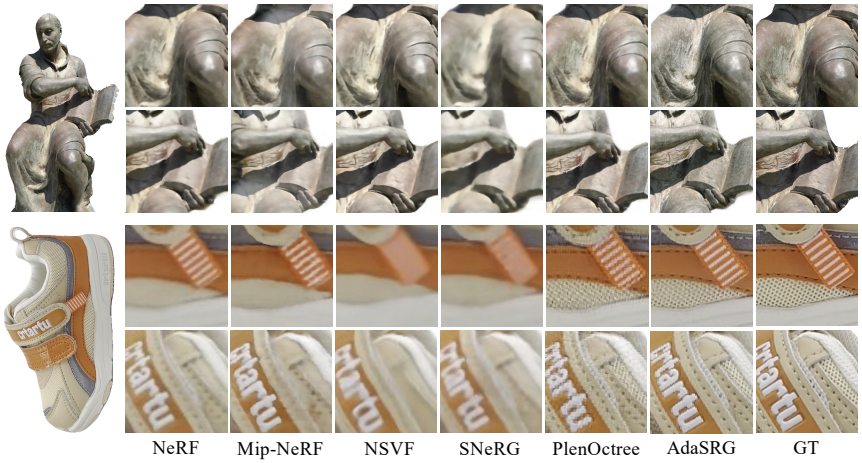

Fig. 4. Qualitative Comparisons. HrSRG can recover more fine details of scenes while improve the perceptual quality of the rendered images. See Table 1 and Table 2 for quantitative comparisons. Zoom in for a better view.

eight object-centric videos via a mobile phone to construct the Real 360° Objects dataset. Each object is with rich details in some regions. We select 300 1080p images per object, 100 for training and 200 for testing. We simply estimate the camera poses via COLMAP [37, 38] without any post-processing. As discussed in [25] and [11], the camera pose errors are often unavoidable. Real 360° Objects may also help to investigate weather our approach is tolerant to the misalignment issue caused by camera pose errors or not.

Evaluation. We measure the render-time performance, training speed, storage cost, and rendering quality in our experiments. The rendering speed is evaluated via a Lenovo ThinkPad P1 Gen 2 notebook with a 4GB NVIDIA Quadro T1000 mobile graphics card. We compute PSNR, SSIM [50], and LPIPS [56] as previously. The scores of the compared approaches are source (or computed) from their papers if provided. We make qualitative comparisons to showcase the texture details. We argue the LPIPS metric might measure the perceptual quality and fine details better.

4.1 Benchmark Comparisons

Compared Methods. We take SNeRG [10] and PlenOctree [54] as the baseline methods. Specifically, we adopt the "PlenOctree after fine-tuning" setting, which has been labeled as the complete model by the PlenOctree paper. On the Real 360° Objects dataset, we train PlenOctree at the voxel resolution of 1024^3 on a 32GB V100 GPU. Learning PlenOctree-1024 on a GPU card with less than 26GB capacity would lead to OMM errors. For SNeRG, we examine the further fine-tuned version. We also make comparisons with other represented

Table 3. Ablation Studies on the "Hat" case of Real 360° Objects. Def-NeRF†-Plus is the converged Def-NeRF†. HrSRG-minus is the diffuse version of HrSRG. HrSRG-N means we capture the HrSRG representation from a N^3 voxel grid (See Sec. 3.3).

Method	PSNR ↑	SSIM ↑	LPIPS ↓	MB ↓	FPS ↑
Def-NeRF†-Plus	25.02	0.838	0.128	-	0.01
HrSRG (w/o $\mathcal{L}_p$)	26.43	0.894	0.061	48.0	59.7
HrSRG	26.64	0.895	0.056	48.0	59.7
HrSRG-minus	25.65	0.876	0.069	27.4	68.5
HrSRG-256	24.72	0.847	0.107	8.7	132.4
HrSRG-512	26.64	0.895	0.056	48.0	59.7

NeRF variants such as NSVF [25], Mip-NeRF [2], DirectVoxGo [43], and [53]. For all the compared methods, we use their released codes (if not specified) and take care to follow their best configurations. On the synthetic and Tank and Temples datasets, the scores for NSVF are computed using the pre-trained models shared by KiloNeRF [36].

Performance. The scores are reported in Table 1 and Table 2. HrSRG achieves promising performance with respect to rendering quality and rendering speed, moreover requires less storage space than real-time NeRF baselines. Especially on the Real 360° Objects dataset, HrSRG outperforms the compared methods by a large margin over the image quality metric LPIPS. Some qualitative comparisons are presented in Fig. 1 and Fig. 4. HrSRG can produce high-quality rendering with rich fine details, while images rendered by other methods contain blurring artifacts. Furthermore, learning to represent a single scene takes around 5.5 hours and 4.5 hours on Synthetic-NeRF [30] and Real 360° Objects, respectively. Other methods take about 1–3 days to obtain a converged model. It is worth mentioning that our experiments show that fine-tuning a PlenOctree from a converged NeRF-SH cannot recover more fine details compared to NeRF-SH. We optimize HrSRG from a non-converged Def-NeRF†. Nevertheless, as shown in the supplementary, we find that fine-tuning PlenOctree from a non-converged NeRF-SH often produce unreasonable texture artifacts. Overall, HrSRG surpasses the compared methods, especially on the texture characteristic preservation ability.

4.2 Ablation Studies

In Table 3 (*Top*) and Figure 5, we discuss the core techniques of the HrSRG approach. We take Def-NeRF†-Plus as the baseline. "Plus" means we train Def-NeRF† until it has converged. Ablation "HrSRG (w/o $\mathcal{L}_p$)" shows (1) removing L_p has significant effect on the perceptual quality of rendered images; and (2) directly optimizing color values (*vs.* Def-NeRF†-Plus) can impressively recover the fine details. We also study HrSRG-minus which removes the specular effects. We find it can achieve competitive rendering performance while further boost the

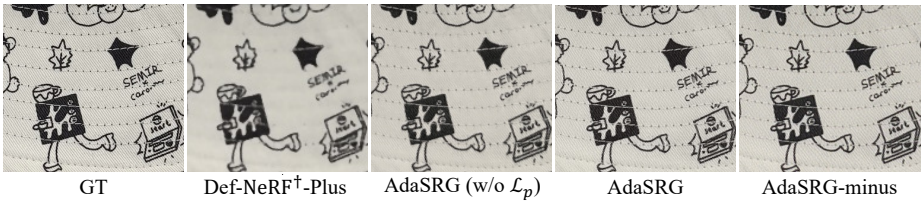

Fig. 5. A qualitative study of the impact of the key components. HrSRG-minus can produce reasonable renderings as some reflected content can be hided inside the objects' surfaces. Zoom in to see the fine details.

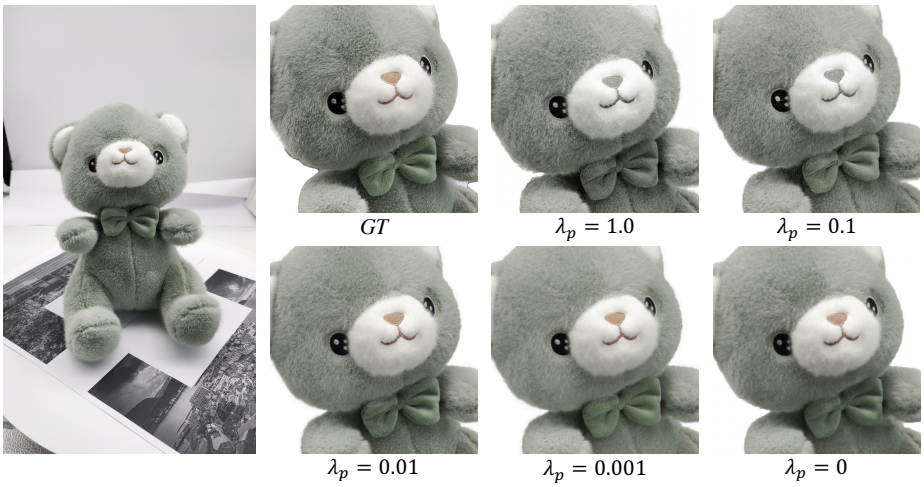

Fig. 6. Perceptual Loss. A very large λ_p yields incorrect colors (See nose and cravat of the "Bear") as the model would focus more on optimizing the perceptual quality. Zoom in to see the fine details.

running performance. As analyzed in [10], a diffuse model may reasonably fake view-dependent effects by hiding mirrored versions of reflected content inside the objects' surfaces. Table 3 (*Bottom*) studies the HrSRG construction approach presented in Sec. 3.3. HrSRG-512 is the default HrSRG version in this paper.

Fig. 6 explores the impact on rendering quality of different λ_p for the perceptual loss (See Eqn. 8). Our main target is to minimize Eqn. 4, thus a very large λ_p yields inaccurate colors. A small λ_p could also help to recover more fine details (*vs.* HrSRG (w/o λ_p)), but has a minimal effect in improving the clarity of rendered images. We find that the perceptual loss does not has a significant affect on the studied rendering quality metrics.

5 Discussion & Limitation

SNeRG [10] prunes invalid voxels through a sparsity regularization term $\mathcal{L}_s$ and an opacity pruning (OP) policy. The OP approach culls macroblocks that the voxel visibilities (or maximum transmittance) are low. PlenOctree [54] also applies OP to reduce the memory footprint. However, our experiments show that enlarging the loss weight of $\mathcal{L}_s$ would easily filters some valid voxels out. As a result, a rendered image would show different degrees of transparency in some regions. Besides, OP with a very low threshold τ could also delete many voxels by mistakes. Examples are shown in the supplementary. Luckily, our 6-DoF viewing applications show HrSRG is robust to these issues.

One of the limitations of HrSRG is the slow training speed. It takes about 5 hours to obtain a complete HrSRG model of a real scene on a V100 GPU. Though requiring several multiple orders of magnitude less time than conventional NeRFs, it is still not friendly for practical requirements. Taking E-commerce as an example, HrSRG can bring 6-DOF immersive viewing experiences of objects, but there are billions of goods to be reconstructed. Several concurrent works [53, 43, 31] study the voxel grid representations for super-fast training. We will investigate their techniques such as coarse-to-fine optimization to quickly produce a good geometry initialization for HrSRG. Another limitation is that HrSRG cannot well reconstruct the strong specular materials like other NeRFs. A recent formulation Ref-NeRF [48] may bring some inspirations.

6 Conclusion

We notice that previous NeRF works fail to preserve texture characteristics for real scenes. We dig into the radiance grid representation [10], a real-time paradigm of NeRF, and propose a set of improvements, which together produce surprising rendering quality and speed. In particular, we present a hierarchical voxel grid building approach that capture an Adaptive Sparse Radiance Grid (HrSRG) to represent a scene. It has higher voxel resolution for informative spaces and fewer voxels for other spaces. Furthermore, taking inspiration from the 2D texture atlas success [11], we show that directly optimizing the color and specular feature of HrSRG could surprisingly reconstruct the fine details of scenes. Moreover, benefiting from our formulation, we introduce a perceptual loss to tackle the misalignment issues caused by camera pose errors. It can bring more details to rendered images while largely improving the perceptual quality. HrSRG enables rendering 1080p images at about 62 FPS in unprecedented quality on a notebook. A pruned HrSRG requires about 41M storage space to represent a scene.

References

1. Arandjelović, R., Zisserman, A.: Nerf in detail: Learning to sample for view synthesis. arXiv preprint arXiv:2106.05264 (2021)

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. In: ICCV (2021)
3. Bentley, J.L.: K-d trees for semidynamic point sets. In: Proceedings of the sixth annual symposium on Computational geometry. pp. 187–197 (1990)
4. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: ICCV (2021)
5. Davis, A., Levoy, M., Durand, F.: Unstructured light fields. In: Computer Graphics Forum. vol. 31, pp. 305–314. Wiley Online Library (2012)
6. Gafni, G., Thies, J., Zollhofer, M., Nießner, M.: Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8649–8658 (2021)
7. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. arXiv preprint arXiv:2103.10380 (2021)
8. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 43–54 (1996)
9. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (TOG) **37**(6), 1–15 (2018)
10. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: ICCV (2021)
11. Huang, J., Thies, J., Dai, A., Kundu, A., Jiang, C., Guibas, L.J., Niessner, M., Funkhouser, T.: Adversarial texture optimization from rgb-d scans. In: CVPR. pp. 1559–1568 (2020)
12. Ichnowski, J., Avigal, Y., Kerr, J., Goldberg, K.: Dex-nerf: Using a neural radiance field to grasp transparent objects. arXiv preprint arXiv:2110.14217 (2021)
13. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR. pp. 1125–1134 (2017)
14. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. pp. 694–711. Springer (2016)
15. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. arXiv preprint arXiv:1708.05375 (2017)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
18. Kutulakos, K.N., Seitz, S.M.: A theory of shape by space carving. International journal of computer vision **38**(3), 199–218 (2000)
19. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. In: ICML. pp. 1558–1566. PMLR (2016)
20. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR. pp. 4681–4690 (2017)
21. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 31–42 (1996)
22. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)

23. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: IEEE International Conference on Computer Vision (ICCV) (2021)
24. Lindell, D.B., Martel, J.N., Wetzstein, G.: Autoint: Automatic integration for fast neural volume rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14556–14565 (2021)
25. Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. NeurIPS (2020)
26. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751 (2019)
27. Max, N.: Optical models for direct volume rendering. IEEE TVCG **1**(2), 99–108 (1995)
28. Meagher, D.: Geometric modeling using octree encoding. Computer graphics and image processing **19**(2), 129–147 (1982)
29. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
30. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421. Springer (2020)
31. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. arXiv preprint arXiv:2201.05989 (2022)
32. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. arXiv preprint arXiv:2103.03231 (2021)
33. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2856–2865 (2021)
34. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Deformable neural radiance fields. arXiv preprint arXiv:2011.12948 (2020)
35. Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K.M., Tagliasacchi, A.: Derf: Decomposed radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14153–14161 (2021)
36. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. arXiv preprint arXiv:2103.13744 (2021)
37. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
38. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016)
39. Seitz, S.M., Dyer, C.R.: Photorealistic scene reconstruction by voxel coloring. International Journal of Computer Vision **35**(2), 151–173 (1999)
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
41. Sitzmann, V., Chan, E.R., Tucker, R., Snavely, N., Wetzstein, G.: Metasdf: Meta-learning signed distance functions. arXiv preprint arXiv:2006.09662 (2020)

42. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2437–2446 (2019)
43. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. arXiv preprint arXiv:2111.11215 (2021)
44. Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P.P., Barron, J.T., Ng, R.: Learned initializations for optimizing coordinate-based neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2846–2855 (2021)
45. Thies, J., Zollhöfer, M., Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019)
46. Trevithick, A., Yang, B.: Grf: Learning a general radiance field for 3d scene representation and rendering. In: arXiv:2010.04595 (2020)
47. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2626–2634 (2017)
48. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. arXiv preprint arXiv:2112.03907 (2021)
49. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: CVPR. pp. 4690–4699 (2021)
50. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
51. Wang, Z., Wu, S., Xie, W., Chen, M., Prisacariu, V.A.: NeRF−−: Neural radiance fields without known camera parameters. arXiv preprint arXiv:2102.07064 (2021)
52. Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., Zhou, J.: Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5610–5619 (2021)
53. Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. arXiv preprint arXiv:2112.05131 (2021)
54. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: PlenOctrees for real-time rendering of neural radiance fields. In: ICCV (2021)
55. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images (2020)
56. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)