KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs

Christian Reiser[1,2] Songyou Peng[1,3] Yiyi Liao[1,2] Andreas Geiger[1,2]
[1]Max Planck Institute for Intelligent Systems, Tübingen [2]University of Tübingen [3]ETH Zurich
{firstname.lastname}@tue.mpg.de

Abstract

NeRF synthesizes novel views of a scene with unprecedented quality by fitting a neural radiance field to RGB images. However, NeRF requires querying a deep Multi-Layer Perceptron (MLP) millions of times, leading to slow rendering times, even on modern GPUs. In this paper, we demonstrate that real-time rendering is possible by utilizing thousands of tiny MLPs instead of one single large MLP. In our setting, each individual MLP only needs to represent parts of the scene, thus smaller and faster-to-evaluate MLPs can be used. By combining this divide-and-conquer strategy with further optimizations, rendering is accelerated by three orders of magnitude compared to the original NeRF model without incurring high storage costs. Further, using teacher-student distillation for training, we show that this speed-up can be achieved without sacrificing visual quality.

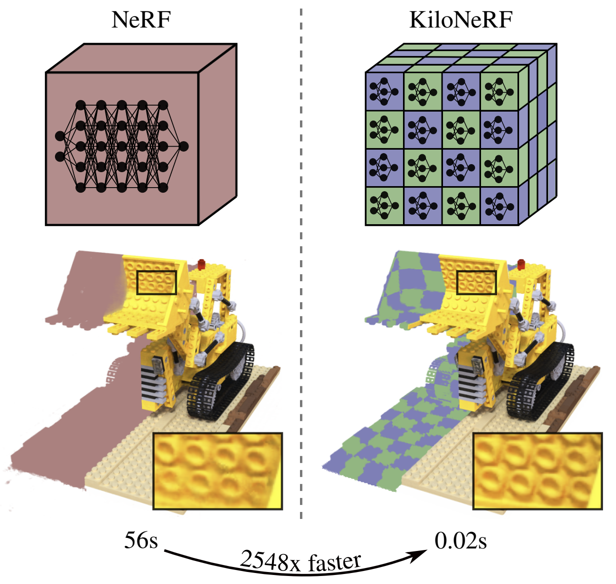


Figure 1: **KiloNeRF.** Instead of representing the entire scene by a single, high-capacity MLP, we represent the scene by thousands of small MLPs. This allows us to render the scene above 2548x faster without loss in visual quality.

1. Introduction

Novel View Synthesis (NVS) addresses the problem of rendering a scene from unobserved viewpoints, given a number of RGB images and camera poses as input, e.g., for interactive exploration. Recently, NeRF [29] demonstrated state-of-the-art results on this problem using a neural radiance field representation for representing 3D scenes. NeRF produces geometrically consistent, high-quality novel views even when faced with challenges like thin structures, semi-transparent objects and reflections. Additionally, NeRF's underlying representation requires only very little storage and thus can be easily streamed to users.

The biggest remaining drawbacks of NeRF are its long training and rendering times. While training can be sped up with a multi-GPU cluster, rendering must happen in real-time on the consumer's device for interactive applications like virtual reality. This motivates us to focus on increasing NeRF's rendering speed in this paper.

NeRF represents the scene's geometry and appearance with a Multi-Layer Perceptron (MLP). During volumetric rendering, this network is sampled hundreds of times for millions of pixels. As the MLP used in NeRF is relatively deep and wide, this process is very slow. A natural idea is to decrease the depth and number of hidden units per layer in order to speed up the rendering process. However, without any further measures, a reduction in network size leads to an immediate loss in image quality due to the limited capacity for fitting complex scenes. We counteract this by using a large number of independent and small networks, and by letting each network represent only a fraction of the scene.

We find that training our KiloNeRF with thousands of networks, naïvely from scratch leads to noticeable artifacts. To overcome this problem, we first train a regular NeRF as teacher model. KiloNeRF is then trained such that its outputs (density and color) match those of the teacher model for any position and view direction. Finally, KiloNeRF is fine-tuned on the original training images. Thanks to this three-stage training strategy, our model reaches the same visual fidelity as the original NeRF model, while being able to synthesize novel views three orders of magnitude faster as illustrated in Fig. 1. Crucial for achieving our speedup is an adequate implementation of the concurrent evaluation of many MLPs. Towards this goal, we published our efficient implementation using PyTorch, MAGMA, Thrust and custom CUDA kernels at https://github.com/creiser/kilonerf.

2. Related Work

Novel View Synthesis: NVS methods can be categorized according to the representations they use to model the underlying scene geometry. Mesh-based methods, including classical [5, 9, 65, 69] and learning-based [45, 46, 61] ones, typically require a preprocessing step, e.g., Structure from Motion (SfM) [49], to reconstruct the scene. Similarly, point cloud-based methods rely on SfM or RGB-D sensors to recover geometry [2]. In contrast, approaches using multi-plane images (MPIs) [11, 28, 55, 56, 64, 78] represent the scene as a stack of images or feature maps. While MPI approaches demonstrate photorealistic image synthesis, they only allow for small viewpoint changes during inference. Another line of methods considers voxel grids as scene representation [19, 25, 40, 51, 52, 58] which, however, are typically restricted in terms of their resolution.

The limitations of voxel grids can be alleviated by adopting neural function representations [29, 32, 53, 72]. Differentiable Volumetric Rendering (DVR) [32] and Implicit Differentiable Renderer (IDR) [72] adopt surface rendering and therefore rely on pixel-accurate object masks as input during training. Furthermore, they assume solid scenes and cannot handle semi-transparent objects or thin structures well. In contrast, NeRF [29] uses volumetric rendering [3] which enables training without masks and allows for recovering fine structures using alpha blending.

One reason for NeRF's success lies in its representation, which utilizes a parametric function to map 3D coordinates and viewing directions to volumetric densities and color values. Such function representations have a long tradition [47, 48] and have recently resurfaced in geometric computer vision [7, 27, 35]. These recent methods use deep neural networks as function class and have advanced the state-of-the-art in tasks like 3D reconstruction, shape generation, point cloud completion or relighting [7, 27, 33, 35, 38].

Faster NeRF Rendering: Neural Sparse Voxel Fields (NSVF) [24] speed up NeRF's rendering using classical techniques like empty space skipping and early ray termination. Additionally, NSVF's network is conditioned on feature vectors located on a uniform 3D grid to increase the model's capacity without increasing network depth or width. By applying empty space skipping already during training, NSVF can sample more densely around surfaces at the same computational budget. We also make use of empty space skipping during training and early ray termination during rendering. The key difference is that we use thousands of small networks, while NSVF uses a single feature-conditioned network. Since our networks are only responsible for representing a small region, they require lower capacity compared to NSVF's single network that must represent the entire scene. As a consequence, KiloNeRF renders views two orders of magnitude faster than NSVF.

Concurrently to us, numerous works [14, 16, 23, 30, 43, 74] were developed with the purpose of speeding up NeRF. DeRF [43] also represents the scene by a number of independent networks. However, in DeRF the scene is decomposed into sixteen irregular Voronoi cells. In this paper, we demonstrate that a much simpler strategy of decomposing the scene into thousands of MLPs arranged on a regular 3D grid leads to significantly higher speedups. AutoInt replaces the need for numerical integration in NeRF by learning a closed form solution of the antiderivative [23]. By making use of the fundamental theorem of calculus, a pixel is rendered with a significantly smaller amount of network queries. DONeRF [30] demonstrates the feasibility to render a pixel using only 4 samples on the ray. This is achieved by placing samples more closely around the first surface that the ray intersects. Towards this goal, for each ray an additional network is queried, that directly predicts suitable sample locations. We note that the latter two approaches are orthogonal to KiloNeRF and the combination of KiloNeRF with either of these new techniques is promising. Other works show that real-time rendering can be achieved by converting the neural representation into a discrete one after training [14, 16, 74]. In comparison to KiloNeRF, these approaches consume significantly more GPU memory which might make them less suitable for larger scenes.

NeRF Follow-ups: NeRF++ extends NeRF to unbounded scenes [76] while NeRF-W tackles unstructured photo collections [26]. GRAF [50], pi-GAN [6] and GIRAFFE [31] propose generative models of radiance fields. A series of works improve generalization from limited number of training views [13, 42, 44, 59, 63, 66, 75] while others [57, 68, 73] remove the requirement for pose estimation. [4, 15, 54] enable learning from images with varying light conditions. Finally, [10, 12, 21, 22, 34, 36, 39, 41, 62, 67, 70] extend NeRF to videos. Many of these methods would benefit from faster rendering and are compatible with KiloNeRF.

3. Method

Our method builds upon NeRF [29], which represents a single scene as a collection of volumetric density and color values. Crucially, in NeRF those densities and colors are not stored in a discrete voxel representation. Instead, NeRF encodes densities and colors at any continuous 3D position in the scene using an MLP. While NeRF uses a single network to represent the entire scene, we are inspired by [8] and represent the scene with a large number of independent and small MLPs. More specifically, we subdivide the scene into a 3D grid. Each MLP is tasked to represent the part of the scene that falls within a particular 3D cell. Since each network only represents a small portion of the scene, a low network capacity is sufficient for photo-realistic results as demonstrated by our experiments. Additionally, we employ empty space skipping and early ray termination to speed up rendering further. We start with a brief review of the original NeRF model which forms the basis for KiloNeRF.

3.1. Background

In NeRF [29], the scene is represented by a neural network f_θ with parameters θ. f_θ takes as input a 3D position $\mathbf{x}$ and viewing direction $\mathbf{d}$ and maps them to a color $\mathbf{c}$ and density σ. The architecture of f_θ is chosen such that only the color $\mathbf{c}$ depends on the viewing direction $\mathbf{d}$. This allows modeling of view-dependent effects like specularities and reflections while also encouraging a consistent geometry to be learned. In a deterministic pre-processing step, $\mathbf{x}$ and $\mathbf{d}$ are transformed by a positional encoding γ which promotes learning of high-frequency details, see [29, 60] for details.

A pixel is rendered by shooting a ray from the camera's eye through the pixel's center and evaluating the network f_θ for K sample points $\mathbf{x}_1, \ldots, \mathbf{x}_K$ along the ray. For each sample $\mathbf{x}_i$, the network outputs a color value $\mathbf{c}_i$ and a density value σ_i. In other words, we compute a list of tuples:

$$(\mathbf{c}_i, \sigma_i) = f_\theta(\mathbf{x}_i, \mathbf{d}) \quad \text{with} \quad i = 1, 2, \ldots, K \tag{1}$$

Here, the viewing direction $\mathbf{d}$ is computed by normalizing the vector from the camera center to the pixel. Subsequently, the final pixel color $\hat{\mathbf{c}}$ is calculated by α-blending the color values $\mathbf{c}_1, \ldots, \mathbf{c}_K$

$$\hat{\mathbf{c}} = \sum_{i=1}^{K} T_i \alpha_i \mathbf{c}_i \tag{2}$$

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{3}$$

$$\alpha_i = 1 - \exp(\sigma_i \delta_i) \tag{4}$$

where α_i determines the alpha value used for blending the color values and the calculation of α_i depends on the distance between adjacent sample points $\delta_j = \|\mathbf{x}_{j+1} - \mathbf{x}_j\|$. Finally, T_i corresponds to the transmittance which accounts for occlusion along the ray. This procedure needs to be repeated for every pixel in the image. Consequently, $W \times H \times K$ network evaluations are required for rendering an image of width W and height H.

NeRF is trained by minimizing a photometric loss between rendered images and training images. More specifically, for each parameter update step, we randomly sample a training image and B pixels $\mathbf{c}_1, \ldots, \mathbf{c}_B$ inside that image. Subsequently, the corresponding pixels $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_B$ are rendered according to Eq. 2. The model parameters θ are optimized by minimizing an L_2 reconstruction loss between the rendered pixels and their ground truth:

$$\mathcal{L} = \frac{1}{B} \sum_{b}^{B} \|\mathbf{c}_b - \hat{\mathbf{c}}_b\|_2^2 \tag{5}$$

3.2. KiloNeRF

KiloNeRF assumes knowledge of an axis aligned bounding box (AABB) enclosing the scene. Let $\mathbf{b}_{min}$ and $\mathbf{b}_{max}$ be the minimum and maximum bounds of this AABB. We subdivide the scene into a uniform grid of resolution $\mathbf{r} = (r_x, r_y, r_z)$. Each grid cell with 3D index $\mathbf{i} = (i_x, i_y, i_z)$ corresponds to a tiny MLP network with an independent set of parameters $\theta(\mathbf{i})$. The mapping g from position $\mathbf{x}$ to index $\mathbf{i}$ of the tiny network assigned to position $\mathbf{x}$ is defined through spatial binning

$$g(\mathbf{x}) = \lfloor (\mathbf{x} - \mathbf{b}_{min}) / ((\mathbf{b}_{max} - \mathbf{b}_{min}) / \mathbf{r}) \rfloor \tag{6}$$

where all operations are element-wise. Querying KiloNeRF at a point $\mathbf{x}$ and direction $\mathbf{d}$ involves determining the network $g(\mathbf{x})$ responsible for the respective grid cell:

$$(\mathbf{c}, \sigma) = f_{\theta(g(\mathbf{x}))}(\mathbf{x}, \mathbf{d}) \tag{7}$$

Network Architecture: As illustrated in Fig. 2, we use a downscaled version of the fully-connected architecture of NeRF which similarly to the NeRF architecture enforces that the predicted density is independent of the view direction. However, NeRF uses a total of 10 hidden layers where each of the first 9 hidden layers outputs a 256-dimensional feature vector and the last hidden layer outputs a 128-dimensional feature vector. In contrast, we use a much smaller MLP with only 4 hidden layers and 32 hidden units each. Due to the low depth of our network, a skip connection as in the original NeRF architecture is not required. Like in NeRF, we provide the viewing direction as an additional input to the last hidden layer. All affine layers are followed by a ReLU activation with two exceptions: The output layer that computes the RGB color $\mathbf{c}$ uses a sigmoid activation and no activation is applied to the feature vector that is given as input to the penultimate hidden layer.

3.3. Training with Distillation

Training a KiloNeRF from scratch can lead to artifacts in free space as visualized in Fig. 3. We observed that better results can be obtained by first training an ordinary NeRF model and distilling [17] the knowledge of this teacher into the KiloNeRF model. Here, the KiloNeRF model is trained such that its outputs match the outputs of the teacher model

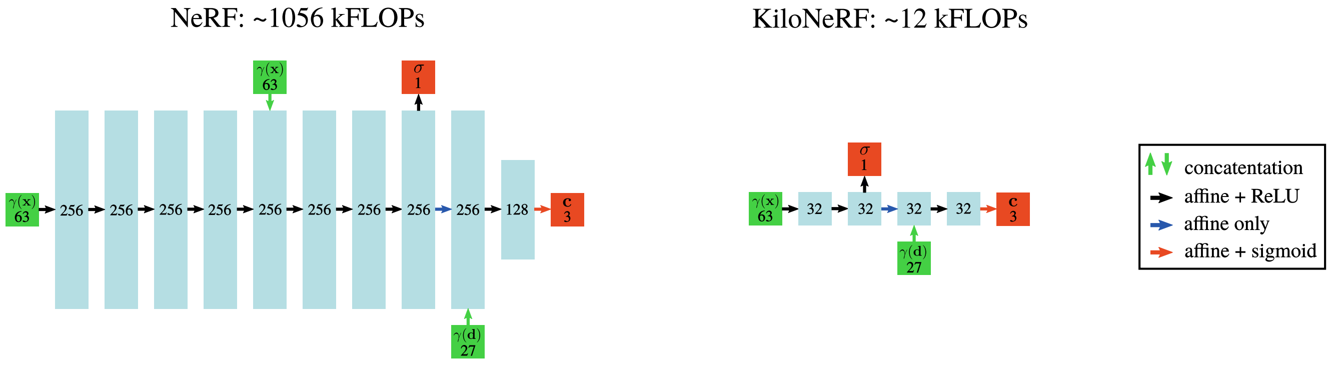
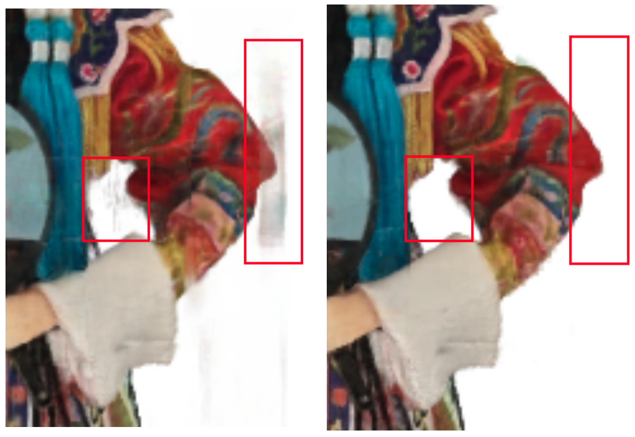


Figure 2: **Model Architecture.** KiloNeRF's MLP architecture is a downscaled version of NeRF's architecture. A forward pass through KiloNeRF's network only requires 1/87th of the floating point operations (FLOPs) of the original architecture.



(a) Without Distillation (b) With Distillation

Figure 3: **Distillation.** (a) Training KiloNeRF from scratch can lead to artifacts in free space. (b) Distillation by imitating a pre-trained standard NeRF model mitigates this issue.

for all possible inputs. More specifically, for each of our networks, we randomly sample a batch of 3D points inside the 3D grid cell that corresponds to the respective network. These batches are augmented with viewing directions which are drawn randomly from the unit sphere. We query both the student and the teacher in order to obtain the respective densities σ and color values $\mathbf{c}$. The obtained σ-values are converted to α-values according to Eq. 4, to put less emphasis on small differences between big density values.

We optimize the student's parameters using an L_2 loss between the α-values and color values predicted by the student and those obtained from the teacher. Note that for the distillation step we are neither performing volumetric rendering nor are we utilizing the training images. Therefore, distillation alone would imply that KiloNeRF's rendering quality is upper bounded by the teacher method's rendering quality. Hence, in a final step KiloNeRF is fine-tuned on the training images with the photometric loss from Eq. 5. From this perspective, distillation can be seen as a means to provide powerful weight initialization to KiloNeRF.

Regularization: The original NeRF architecture uses a small fraction of its total capacity to model the dependency of the color on the viewing direction as can be observed in Fig. 2 (left). Zhang et al. [76] postulate that this choice encourages that color is a simple function of the viewing direction. Further, they experimentally demonstrate how this inductive bias is crucial for avoiding artifacts in empty space [76]. Our first attempt was to replicate this strategy in KiloNeRF, i.e., we reduced the number of output features of the last hidden layer to get a similar regulatory effect. However, this lead to an overall loss in visual quality due to our small network width. We therefore instead apply L_2 regularization to the weights and biases of the last two layers of the network which are responsible for view-dependent modeling of the color. This strategy allows us to impose the same inductive bias as in NeRF without sacrificing visual quality. We refer to Fig. 6 in our experiments for an illustration.

3.4. Sampling

A maximum of K equidistant points are sampled along the ray. To reduce the number of network queries, we make use of empty space skipping (ESS) and early ray termination (ERT). Consequently, the number of sampled points per ray is variable and the hyperparameter K effectively controls the distance δ between sampled points. Thanks to the combination of ESS and ERT, only points in the vicinity of the first surface that a ray intersects are evaluated, given that the intersection is with an opaque object. We do not make use of the hierarchical sampling scheme introduced in NeRF, which also enables dense sampling in occupied space, but is costlier than ESS. Similar to NeRF [29], we use stratified sampling to avoid bias.

Empty Space Skipping: For empty space skipping, we instantiate a second uniform grid with a higher grid resolution in the given AABB. Each cell in this grid is populated with a binary value which indicates whether the scene has any content within that cell. This occupancy grid can be used to avoid querying the network in empty space [3]. Only if the cell, in which a sample point $\mathbf{x}$ lies, is occupied, the network gets evaluated at $\mathbf{x}$. We extract the occupancy grid from the trained teacher and use it for more efficient fine-tuning and inference. To populate the occupancy grid, we sample densities on a $3 \times 3 \times 3$ subgrid from the teacher model for each cell of the occupancy grid. A cell is marked as occupied, if any of the evaluated densities is above a threshold τ. No dataset-specific tuning of τ was necessary.

Early Ray Termination: If the transmittance value T_i becomes close to 0 during volumetric rendering, evaluation of subsequent sample points $\mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \dots$ is unnecessary, since the contribution of corresponding color values $\mathbf{c}_{i+1}, \mathbf{c}_{i+2}, \dots$ to the final pixel value becomes vanishingly small. Therefore, once transmittance T_i falls below a threshold ϵ, sampling further points along the ray can be avoided. ERT is only used during inference and is implemented by grouping network queries according to their distance from the camera, i.e., points close to the camera are evaluated first and then successively more distant points are queried for those rays that are not terminated yet.

3.5. Implementation

Our prototype implementation is based on PyTorch [37], MAGMA [1] and Thrust. Additionally, we developed custom CUDA kernels for sampling, empty space skipping, early ray termination, positional encoding, network evaluation and alpha blending. Crucial for high performance is the proper handling of the simultaneous query of thousands of networks. To illustrate this, let us consider a single linear layer of a fully connected network which requires the computation of a single matrix multiplication. More specifically, a $B \times I$ matrix is multiplied with an $I \times O$ matrix, where B is the batch size and I/O is the number of input/output features. In KiloNeRF, for each network a separate matrix multiplication is required. Assuming N networks in total, we need to multiply a $B_i \times I$ matrix with an $I \times O$ matrix for each $i \in 1, \dots, N$. Since B_i varies per network, traditional batched matrix multiplications cannot be used. To tackle this problem, we exploit the HPC library MAGMA which provides a CUDA routine for this situation [1]. For inference only we use a self-developed routine that fuses the entire network evaluation into a single CUDA kernel. Further, it is necessary to order the input batch such that subsequent inputs are processed by the same network. This step benefits largely from the prior reduction in number of samples caused by the interplay of ESS and ERT. More details can be found in the supplementary.

4. Experimental Evaluation

4.1. Setup

Datasets: Both NSVF [24] and KiloNeRF assume scenes to be bounded. We can therefore directly use the four datasets provided by NSVF, together with the given bounding boxes. Consequently, KiloNeRF is tested on both synthetic datasets (Synthetic-NeRF, Synthetic-NSVF) as well as datasets comprising real scenes (NSVF's variants of BlendedMVS [71] and Tanks&Temples [18]). In total, we evaluate on 25 scenes.

Baselines: We compare KiloNeRF to NeRF [29] and NSVF [24]. Note that we compare to the original NeRF model which adopts the hierarchical sampling strategy despite it is not used for our teacher. NSVF is particularly interesting as a baseline, because NSVF also makes use of ESS and ERT. Hence speedups over NSVF can be largely attributed to the smaller network size used in KiloNeRF. For comparisons to other NVS techniques [25, 53] we refer the reader to [24].

Hyperparameters: For each scene, we choose the network grid resolution such that the largest dimension equals 16 and the other dimensions are chosen such that the resulting cells are cubic, hence the maximally possible resolution is $16 \times 16 \times 16 = 4096$. To arrive at the occupancy grid's resolution we multiply the network grid resolution by 16, which implies that the occupancy grid resolution is upper bounded by $256 \times 256 \times 256$ cells. We find that setting the threshold τ for occupancy grid extraction to 10 works well on all datasets. We use the Adam optimizer with a learning rate of $5\mathrm{e}{-4}$, a batch size of 8192 pixels and the same learning rate decay schedule as NeRF. For L_2 regularization we use a weight of $1\mathrm{e}{-6}$. For early ray termination, we set the threshold to $\epsilon = 0.01$. As found in NSVF, early ray termination with this choice of ϵ does not lead to any loss of quality [24]. Further, using the same value for ϵ as NSVF increases comparability. The hyperparameter K, which indirectly controls the distance between sample points, is set to 384. We use the same number of frequencies for the positional encoding as NeRF.

Training Schedule: We train both the NeRF baseline and KiloNeRF for a high number of iterations to find the limits of the representation capabilities of the respective architectures. In fact, we found that on some scenes NeRF takes up to 600k iterations to converge. When comparing our qualitative and quantitative results to those reported by Liu et al. [24], one might notice that we obtained much better results for the NeRF baseline, which might be explained by our longer training schedule. Training the NeRF baseline on an NVIDIA GTX 1080 Ti takes approximately 2 days. For our teacher, we train another NeRF model but without hierarchical sampling for 600k iterations as NeRF's coarse model

Resolution		BlendedMVS 768 × 576	Synthetic-NeRF 800 × 800	Synthetic-NSVF 800 × 800	Tanks & Temples 1920 × 1080
PSNR ↑	NeRF	27.29	31.01	31.55	28.32
	NSVF	26.90	**31.74**	**35.13**	28.40
	KiloNeRF	**27.39**	31.00	33.37	**28.41**
SSIM ↑	NeRF	0.91	**0.95**	0.95	0.90
	NSVF	0.90	**0.95**	**0.98**	0.90
	KiloNeRF	**0.92**	**0.95**	0.97	**0.91**
LPIPS ↓	NeRF	0.07	0.08	0.04	0.11
	NSVF	0.11	0.05	**0.01**	0.15
	KiloNeRF	**0.06**	**0.03**	0.02	**0.09**
Render time (milliseconds) ↓	NeRF	37266	56185	56185	182671
	NSVF	4398	4344	10497	15697
	KiloNeRF	**30**	**26**	**26**	**91**
Speedup over NeRF ↑	NSVF	8	13	5	12
	KiloNeRF	**1258**	**2165**	**2167**	**2002**

Table 1: **Quantitative Results.** KiloNeRF achieves similar quality scores as the baselines while being significantly faster.

is too inaccurate for distillation and NeRF's fine model is biased towards surfaces. This model is distilled for 150k iterations into a KiloNeRF model, which takes up to 8 hours. Finally, the KiloNeRF model is fine-tuned for 1000k iterations, which requires only 17 hours thanks to ESS and the reduced size of our MLPs.

Measurements: Our test system consists of an NVIDIA GTX 1080 Ti consumer GPU, an Intel i7-3770k CPU and 32GB of RAM. For each scene, inference time is measured by computing the average time over all test images.

4.2. Results

Table 1 shows our main results. KiloNeRF attains similar visual quality scores as the baselines NeRF and NSVF. In terms of the perceptual metric LPIPS [77], KiloNeRF even slightly outperforms the other two methods on three out of four datasets. However, on all scenes, KiloNeRF renders three orders of magnitude faster than NeRF and two orders of magnitude faster than NSVF. In Fig. 4, we show novel view synthesis results for all three methods. More qualitative results can be found in the supplementary.

4.3. Ablations

In Fig. 5, we conduct an ablation study on KiloNeRF using the Lego bulldozer scene as an example.

Single Tiny MLP: For this ablation, KiloNeRF's network hyperparameters (number of layers/hidden units) are used, but only a single network gets instantiated. Unsurprisingly, quality suffers dramatically since a single MLP with only 6k parameters cannot accurately represent the entire scene.

Method	Render time ↓	Speedup ↑
NeRF	56185 ms	–
NeRF + ESS + ERT	788 ms	71
KiloNeRF	**22** ms	**2548**

Table 2: **Speedup Breakdown.** The original NeRF model combined with KiloNeRF's implementation of ESS and ERT is compared against the full KiloNeRF technique.

Half Network Resolution: We test KiloNeRF with half the network grid size ($5 \times 8 \times 5$ instead of $10 \times 16 \times 10$ for this particular scene) to verify whether the chosen resolution is not unnecessarily high. We found that reducing the grid resolution leads to a decline in quality which demonstrates that a fine grid of tiny networks is indeed beneficial.

Half #Units: We also tested whether it is possible to further reduce the number of hidden units. Again, we observe a decrease in image quality, although the loss of quality is not as pronounced as when decreasing grid resolution. A possible explanation might be that halving the network resolution decreases the number of parameters by a factor of 8, while halving the number of hidden units decreases the parameter count only by approximately a factor of 4.

No Fine-Tuning: For this experiment, the ultimate fine-tuning phase is omitted. Our results show that fine-tuning is crucial for achieving high quality and that the prior distillation phase only provides a good parameter initialization.

No Distillation: Fig. 3 compares training KiloNeRF from

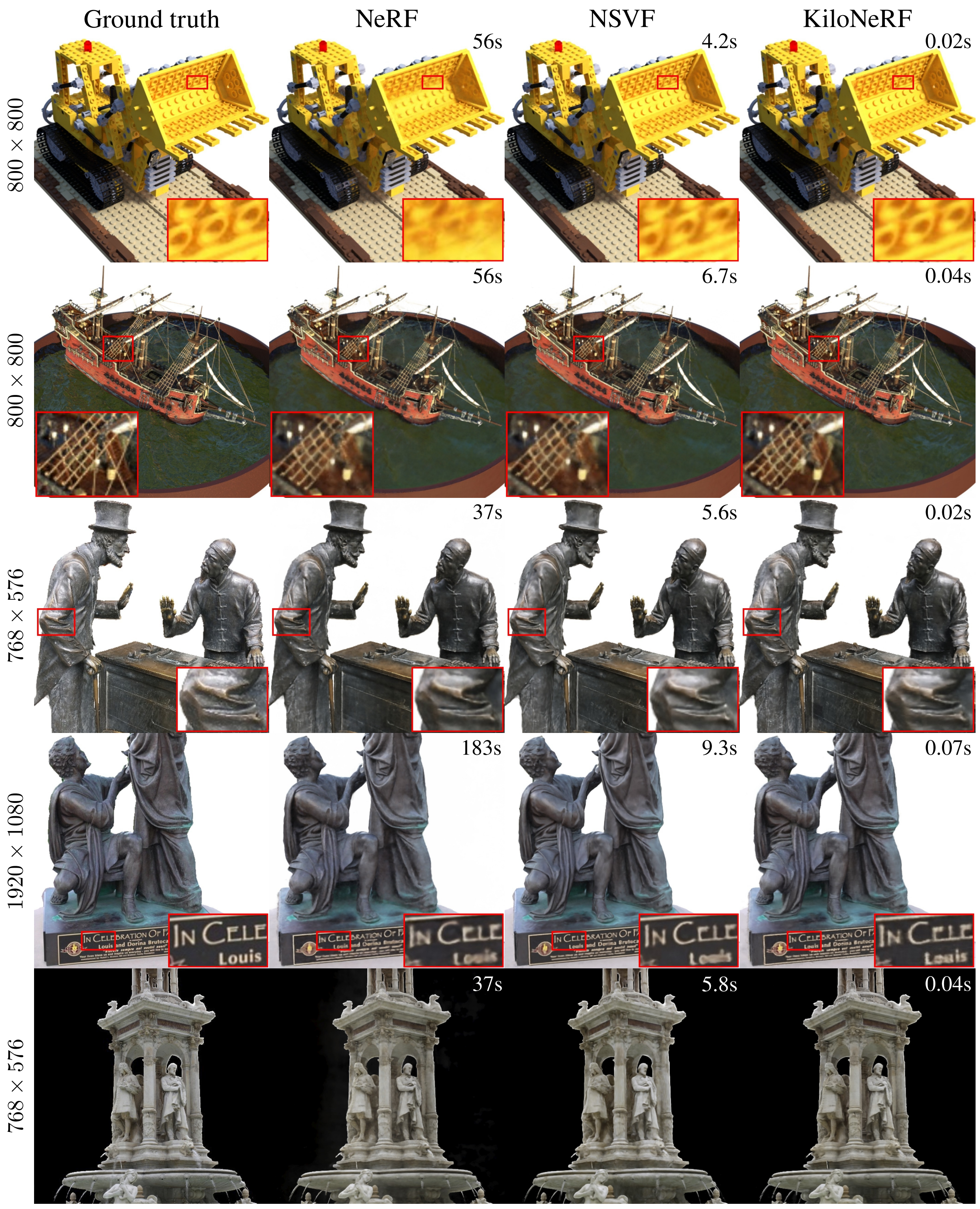


Figure 4: **Qualitative Comparison.** Novel views synthesized by NeRF, NSVF and KiloNeRF. Despite being significantly faster, KiloNeRF attains the visual quality of the baselines. The numbers in the top-right corner correspond to the average render time of the respective technique on that scene. The rendered image resolution (in pixels) is specified on the left.

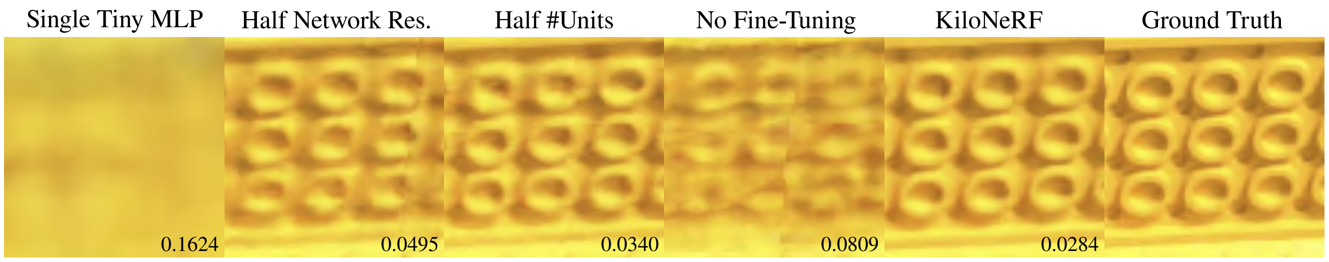
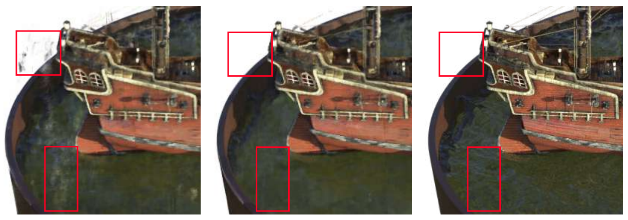


Figure 5: **Ablation Study.** Closeups of KiloNeRF on the Lego bulldozer scene, varying different parameters of the model. The numbers in the bottom-right corner correspond to perceptual similarity (LPIPS) wrt. the ground truth, lower is better.



(a) No Regularization (b) Regularization (c) Ground Truth

Figure 6: **Regularization.** Without weight regularization, visible artifacts in free space regions emerge in the rendered images as the view-dependent part of the MLP has too much capacity. Adding L_2 regularization alleviates this problem.

scratch against the proposed training pipeline. Both alternatives yield the same level of detail, but only when using distillation artifacts in empty space are avoided.

No Weight Regularization: Fig. 6 demonstrates that artifacts in free space can also emerge when L_2 regularization on the last two layers of the network is omitted. Consequently, both distillation and L_2 regularization play important roles in avoiding these artifacts.

ESS and ERT: Finally, we are interested in determining to which extent the combination of ESS and ERT and the reduction of network size contribute to the final speedup. Towards this goal, we run the original-size NeRF with our optimized ESS/ERT implementation and compare to the full KiloNeRF technique on the Lego bulldozer scene. As can be deduced from Table 2, the reduction in network size contributes significantly to KiloNeRF's overall speedup.

5. Discussion and Future Work

While KiloNeRF is able to render medium resolution images (800×800) at interactive frame rates, the speedups are not sufficient yet for real-time rendering of full HD images. Further speedups might be possible by scaling our approach to a higher number of smaller networks. Naïvely doing this, however, would lead to a higher storage impact. This might be mitigated by using memory-efficient data structures [8, 20] that allow networks to be exclusively instantiated in the vicinity of surfaces.

Furthermore, in this work, we mainly focused on making network queries faster by using many small MLPs. In contrast, the concurrently developed techniques AutoInt [23] and DONeRF [30] speed up inference by reducing the number of required samples along the ray. Therefore, the combination of KiloNeRF with either of these techniques constitutes a promising solution for achieving real-time rendering of high-resolution images in the near future.

Limitations: NSVF [24] and KiloNeRF share the assumption of a bounded scene which is a limitation that should be addressed in future work. Representing unbounded scenes requires a higher number of networks, leading to larger memory consumption. Again, efficient data structures could help with addressing this issue and therefore allow for scaling to larger (e.g., outdoor) scenes. On the other hand, the scenes considered in this paper are encoded with less than 100 MB of memory, respectively. This shows that – at least for medium-sized scenes – storage is not an issue.

6. Conclusion

In this paper, we demonstrated that real-time rendering of NeRFs can be achieved by spatially decomposing the scene into a regular grid and assigning a small-capacity network to each grid cell. Together with the other benefits inherited from NeRF – excellent render quality and a low storage impact – we believe that this constitutes an important step towards practical NVS. Moreover, the presented acceleration strategy might also apply more broadly to other methods relying on neural function representations including implicit surface models.

Acknowledgements: This work was supported by an NVIDIA research gift, the ERC Starting Grant LEGO-3D (850533) and the DFG EXC number 2064/1 - project number 390727645. Songyou Peng is supported by the Max Planck ETH Center for Learning Systems. Yiyi Liao is supported by the BMBF Tübingen AI Center, FKZ: 01IS18039A.

References

[1] Ahmad Abdelfattah, Azzam Haidar, Stanimire Tomov, and Jack Dongarra. Novel hpc techniques to batch execution of many variable size blas computations on gpus. In *International Conference on Supercomputing*, 2017. 5

[2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor S. Lempitsky. Neural point-based graphics. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2

[3] Johanna Beyer, Markus Hadwiger, and Hanspeter Pfister. State-of-the-art in gpu-based large-scale volume visualization. In *Computer Graphics Forum*, 2015. 2, 5

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P. A. Lensch. NeRD: Neural reflectance decomposition from image collections. *arXiv.org*, 2020. 2

[5] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In *ACM Trans. on Graphics*, 2001. 2

[6] Eric R. Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[8] Qiang Dai, Ying Song, and Yi Xin. Random-accessible volume data compression with regression function. In *International Conference on Computer-Aided Design and Computer Graphics*, 2015. 3, 8

[9] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *ACM Trans. on Graphics*, 1996. 2

[10] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. *arXiv.org*, 2020. 2

[11] John Flynn, Michael Broxton, Paul E. Debevec, Matthew DuVall, Graham Fyffe, Ryan S. Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[12] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. *arXiv.org*, 2020. 2

[13] Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. Portrait neural radiance fields from a single image. *arXiv.org*, 2020. 2

[14] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv.org*, 2021. 2

[15] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv.org*, 2020. 2

[16] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv.org*, 2021. 2

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv.org*, 2015. 3

[18] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. on Graphics*, 2017. 5

[19] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision (IJCV)*, 38(3):199–218, 2000. 2

[20] Samuli Laine and Tero Karras. Efficient sparse voxel octrees. *IEEE Trans. on Visualization and Computer Graphics (VCG)*, 2010. 8

[21] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *arXiv.org*, 2021. 2

[22] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[23] David B. Lindell, Julien N. P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. *arXiv.org*, 2020. 2, 8

[24] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 5, 8

[25] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. on Graphics*, 2019. 2, 5

[26] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[28] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. on Graphics*, 2019. 2

[29] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4, 5

[30] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Chakravarty R. Alla Chaitanya, Anton Kaplanyan,

and Markus Steinberger. Donerf: Towards real-time rendering of neural radiance fields using depth oracle networks. *arXiv.org*, 2021. 2, 8

[31] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[32] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[33] Michael Oechsle, Michael Niemeyer, Christian Reiser, Lars Mescheder, Thilo Strauss, and Andreas Geiger. Learning implicit surface light fields. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2020. 2

[34] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[36] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv.org*, 2020. 2

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 5

[38] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2

[39] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[40] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. on Graphics*, 2017. 2

[41] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv.org*, 2020. 2

[42] Amit Raj, Michael Zollhoefer, Tomas Simon, Jason Saragih, Shunsuke Saito, James Hays, and Stephen Lombardi. Pva: Pixel-aligned volumetric avatars. *arXiv.org*, 2021. 2

[43] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. *arXiv.org*, 2020. 2

[44] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned radiance fields from a single view. *arXiv.org*, 2021. 2

[45] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2

[46] Gernot Riegler and Vladlen Koltun. Stable view synthesis. *arXiv.org*, 2020. 2

[47] Vladimir V. Savchenko, Alexander A. Pasko, Oleg G. Okunev, and Tosiyasu L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 1995. 2

[48] B. Schölkopf, J. Giesen, and S. Spalinger. Kernel methods for implicit surface modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2005. 2

[49] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[50] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[51] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 1999. 2

[52] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[53] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 2, 5

[54] Pratul Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. *arXiv.org*, 2020. 2

[55] Pratul P Srinivasan, Ben Mildenhall, Matthew Tancik, Jonathan T Barron, Richard Tucker, and Noah Snavely. Lighthouse: Predicting lighting volumes for spatially-coherent illumination. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[56] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[57] Shih-Yang Su, Frank Yu, Michael Zollhoefer, and Helge Rhodin. A-nerf: Surface-free human 3d pose refinement via neural rendering. *arXiv.org*, 2021. 2

[58] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1998. 2

[59] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[60] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[61] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: image synthesis using neural textures. *ACM Trans. on Graphics*, 2019. 2

[62] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a deforming scene from monocular video. *arXiv.org*, 2020. 2

[63] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv.org*, 2020. 2

[64] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[65] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014. 2

[66] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[67] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhöfer. Learning compositional radiance fields of dynamic human heads. *arXiv.org*, 2020. 2

[68] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF–: Neural radiance fields without known camera parameters. *arXiv.org*, 2021. 2

[69] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *ACM Trans. on Graphics*, 2000. 2

[70] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[71] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5

[72] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NIPS)*, 2020. 2

[73] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. *arXiv.org*, 2020. 2

[74] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *arXiv.org*, 2021. 2

[75] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[76] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv.org*, 2020. 2, 4

[77] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[78] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. on Graphics*, 2018. 2