Fast Training of Neural Lumigraph Representations using Meta Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Novel view synthesis is a long-standing problem in machine learning and computer vision. Significant progress has recently been made in developing neural scene representations and rendering techniques that synthesize photorealistic images from arbitrary views. These representations, however, are extremely slow to train and often also slow to render. Inspired by neural variants of image-based rendering, we develop a new neural rendering approach with the goal of quickly learning a high-quality representation which can also be rendered in real-time. Our approach, MetaNLR++, accomplishes this by using a unique combination of a neural shape representation and 2D CNN-based image feature extraction, aggregation, and re-projection. To push representation convergence times down to minutes, we leverage meta learning to learn neural shape and image feature priors which accelerate training. The optimized shape and image features can then be extracted using traditional graphics techniques and rendered in real time. We show that MetaNLR++ achieves similar or better novel view synthesis results in a fraction of the time that competing methods require.

1 Introduction

Learning 3D scene representations from partial observations captured by a sparse set of 2D images is a fundamental problem in machine learning, computer vision, and computer graphics. Such a representation can be used to reason about the scene or to render novel views. Indeed, the latter application has recently received a lot of attention (e.g., [1]). For this problem setting, the key questions are: (1) How do we parameterize the scene, and (2) how do we infer the parameters from our observations efficiently? With our work, we offer new solutions to answer these questions.

Several classes of scene representation learning approaches have recently been proposed. One popular approach consists of coordinate-based neural networks combined with volume rendering, like NeRF [2]. Although these representations offer photorealistic quality for synthesized images, they are slow to train and render. Coordinate-based networks that implicitly model surfaces combined with sphere tracing–based rendering are another popular approach [3–6]. One benefit of an implict surface is that, once trained, it can be extracted and rendered in real time [5]. However, training these representations is equally slow as training volumetric representations. Finally, approaches that use a proxy geometry with on-surface feature aggregation are also fast to render [7, 8], but the quality and runtime of these methods is limited by the traditional 3D computer vision algorithms that pre-compute the proxy shape, such as structure from motion (SfM) or multiview stereo (MVS).

Here, we develop a new framework for neural scene representation and rendering with the goal of enabling fast training and rendering times. To optimize the training time of our framework, we do not learn a representation network for the view-dependent radiance, as other neural volume or surface methods do, but directly aggregate the features extracted from the source views on the surface of our

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

learned proxy shape. To cut down on pre-processing times required by SfM and MVS, we optimize a coordinate-based network representing the proxy shape end-to-end with our CNN-based feature encoder and decoder, and learned aggregation function. A key contribution of our work is to combine this unique surface-based neural rendering framework with meta learning, which enables us to learn efficient initializations for all of the trainable parts of our framework. We demonstrate high-quality training of neural scene representations in minutes or tens of minutes, rather than hours or days, which can then be rendered at real-time framerates.

2 Related Work

Image-based rendering (IBR). Classic IBR approaches synthesize novel views of a scene by blending the pixel values of a set of 2D images [9–18]. Recent IBR techniques leverage neural networks to learn the required blending weights [19–24]. These neural IBR methods either use proxy geometry, for example obtained by SfM or MVS [25, 26], together with on-surface feature aggregation [7, 8] or use learned pixel aggregation functions [27, 28] for geometry-free image-based view synthesis. Our approach is closely related to the geometry-assisted and feature-interpolating view synthesis techniques. Existing approaches, however, require the proxy geometry to be estimated as a pre-processing step, which can take minutes to hours for a single scene, preventing fast processing of novel views. Instead, our approach estimates a coordinate-based neural shape representation directly from the input images which we accelerate using meta learning.

Neural scene representations and rendering. Emerging neural rendering techniques [1] use explicit, implicit, or hybrid scene representations. Explicit representations, for example those using proxy geometry (see above), object-specific shape templates [29], multi-plane [2, 30–33] or multi-sphere [34, 35] images, or volumes [36, 37], are fast to evaluate but memory inefficient. Implicit representations, or coordinate-based networks, can be more memory efficient but are typically slow to evaluate [3–5, 38–54]. Hybrid representations make a trade-off between computational and memory efficiency [55–58]. Among these, NeRF [59] and related methods (e.g., [60–65]) use coordinate-based network representations and volume rendering, which requires many samples to be processed along a ray, each requiring a full forward pass through the network. Although recent work has proposed enhanced data structures [66–68], pruning [51], importance sampling [65], fast integration [69], and other strategies to accelerate the rendering speed, training times of all of these methods are extremely slow, on the order of hours or days for a single scene. DVR [3], IDR [4], NLR [5] and UNISURF [6], on the other hand, leverage implicitly defined surface representations, which are faster to render than volumes but are equally slow to train. Our approach builds on generalization approaches for neural scene representations to accelerate the training time of 2D-supervised neural surface representations.

Generalization with neural scene representations. Being able to generalize across neural representations of different scenes is crucial for learning priors and for 3D generative-adversarial networks with coordinate-based network backbones [70–74]. A variety of different generalization approaches have been explored for neural scene representations, including conditioning by concatenation [38, 39], hypernetworks [46], modulation or feature-wise linear transforms [73, 75, 76], and meta learning [77, 78]. Inspired by these works, we propose a meta-learning strategy that allows us to quickly learn a high-quality neural shape representation for a given set of multi-view images. As opposed to the 3D point cloud supervision proposed by MetaSDF [77], we meta-learn signed distance functions (SDFs) using 2D images, and as opposed to meta-learned initializations for NeRF volumes [78], we operate with SDFs and features. Our approach is unique in enabling fast, high-quality shape representations to be learned from multi-view images that, once optimized, can be rendered in real time.

3 Method

3.1 Image formation model

In this section, we outline the NLR++ novel view synthesis image formation model, which is presented in Figure 1. NLR++ takes as inputs a set of images $\{I_i\}_{i=1}^N$ each with corresponding binary foreground object masks $\{M_i\}_{i=1}^N$ and known camera intrinsic parameters $C_i \in \mathbb{R}^{3\times 3}$ and extrinsic

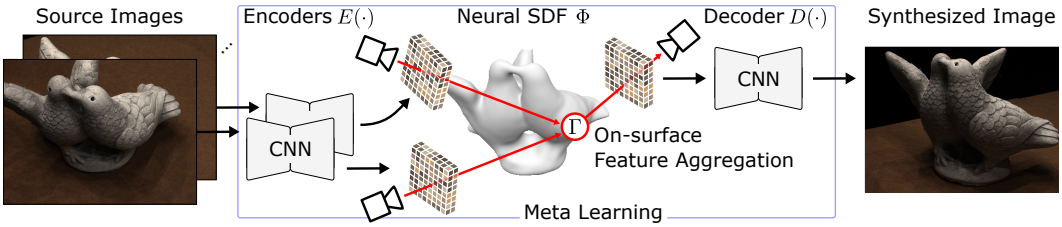

Figure 1: Overview of MetaNLR++.

parameters $R_i|t_i \in \mathbb{R}^{3\times 4}$. At output, NLR++ synthesizes an image $\hat{I}_t$ of the scene from the target viewpoint $\{C_t, R_t|t_t\}$.

Drawing inspiration from classic image-based rendering methods, we define the image formation model as a learned pixel-wise aggregation $\Gamma_\zeta(\cdot)$ of input image features $\{E_\xi(I_i)\}_{i=1}^N$ and their target viewing direction $V_t \in \mathbb{R}^{H\times W\times 3}$ on the surface of the object represented by the neural surface Φ_θ. To project the non-occluded, visible input features of $E_\xi(I_i)$ into the target view before aggregation, we define the function $\mathcal{P}_{i\to t}(E_\xi(I_i); \{C_i, R_i|t_i\}, \{C_t, R_t|t_t\})$. These neurally aggregated features are then decoded into an image by decoder $D_\psi(\cdot)$:

$$\hat{I}_t = D_\psi(\Gamma_\zeta(\{\mathcal{P}_{i\to t}(E_\xi(I_i))\}_{i=1}^N, V_t)). \tag{1}$$

The feature encoder E and decoder D are implemented as resolution-preserving convolutional neural network (CNN) architectures [79, 80] with learned parameters ξ, ψ:

$$E_\xi : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{H\times W\times d}, \quad D_\psi : \mathbb{R}^{H\times W\times d} \to \mathbb{R}^{H\times W\times 3} \tag{2}$$

To aggregate the input image features from E into a target feature map to be decoded by D, we use a learned feature aggregation (or blending) function Γ_ζ, which operates on the surface of our shape representation Φ. To define the surface of our shape, we choose to use a SIREN [50] which represents the signed-distance function (SDF) in 3D space. This encodes the surface of the object as the zero-level set, L_0, of the network:

$$L_0(\Phi_\theta) = \{x \in \mathbb{R}^3 | \Phi_\theta(x) = 0\}, \quad \Phi_\theta : \mathbb{R}^3 \to \mathbb{R}. \tag{3}$$

The aggregation is performed on surface for each pixel of the target image $\hat{I}_t$ with camera $\{C_t, R_t|t_t\}$. To find the point in $L_0(\Phi_\theta)$ corresponding to each pixel ray, we perform sphere tracing on the neural SDF model $\mathcal{R}(\Phi_\theta, C_t, R_t|t_t)$, retaining gradients for the final step of evaluation [4, 5, 52, 53]. These individual rendered surface points are projected into the image plane of each of the N input image views and used to sample interpolated features from the source feature maps for each pixel, which can be arranged into N re-sampled feature maps corresponding to each input image $\{F_i\}_{i=1}^N \in \mathbb{R}^{N\times H\times W\times d}$. To check whether or not a feature is occluded, we use sphere tracing for each pixel from the input views $\{\mathcal{R}(\Phi_\theta, C_i, R_i|t_i)\}_{i=1}^N$, and ensure that the target view surface position projected into each of these surfaces is at the same depth as the source view surface position. Occluded features are then discarded. These three steps of sphere tracing, feature sampling, and occlusion checking make up the function $\mathcal{P}_{i\to t}(E_\xi(I_i))$ which outputs each of the re-sampled feature maps $\{F_i\}_{i=1}^N$.

Once the re-sampled feature maps $\{F_i\}_{i=1}^N$ have been generated, the aggregation function Γ_ζ aggregates them into a single target feature $F_t \in \mathbb{R}^{H\times W\times d}$ which can be processed by the decoder into an image. This is done by performing a weighted averaging operation on the input features using the relative feature weights $L \in \mathbb{R}^{N\times H\times W}$:

$$\Gamma_\zeta : \mathbb{R}^{N\times H\times W\times d} \times \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{H\times W\times d},$$
$$\{F_i\}_{i=1}^N, V_t \mapsto \Gamma_\zeta(\{F_i\}_{i=1}^N, V_t) = \sum_{i=1}^N (L_i / \sum_{j=1}^N L_j) \circ F_i = F_t, \tag{4}$$

where $\circ$ is the Hadamard product between the feature and weight maps, broadcasted over the feature dimension d. The weight map L used in the feature aggregation function Γ_ζ is obtained from an

MLP γ_ζ which is applied pixel-wise to each of the N re-sampled feature maps and each pixels target viewing direction:

$$[\{F_i\}_{i=1}^N, V_t] \mapsto \gamma_\zeta([\{F_i\}_{i=1}^N, V_t]) = L. \tag{5}$$

Here, the dependence upon viewing direction allows for the modeling of view-dependent image properties. These pixel-wise operations making up Γ result in a $H \times W \times d$ feature map which can be input into the decoder D.

The usage of features instead of pixel values directly allows D some opportunity to inpaint and correct artifacts from imperfect geometry to create a photorealistic novel view, unlike methods which render pixels individually [4, 5]. Additionally, the use of the CNN encoder and decoder increases the receptive field of the image loss applied, allowing for more meaningful gradients to be propagated back into Φ_θ.

3.2 Supervision and training

Since NLR++ is end-to-end differentiable, we can optimize the parameters ξ, ψ, θ, ζ end-to-end to reconstruct target views. For each iteration of training we sample a set of $k \leq N$ images $\{I_n\}_{n=1}^k$, and designate one of these images to be the ground-truth target image used for supervision I_t, and sample its corresponding binary object mask M_t and parameters $\{C_t, R_t|t_t\}$.

Using the NLR++ image formation model, we generate a synthesized target image $\hat{I}_t$ from viewpoint $\{C_t, R_t|t_t\}$. The loss evaluated on the synthesized image consists of three terms:

$$\mathcal{L}(\{\hat{I}_t, \hat{M}_t\}, \{I_t, M_t\}, \Phi_\theta) = \mathcal{L}_R(\hat{I}_t, \{I_t, M_t\}) + \lambda_1 \mathcal{L}_M(\Phi_\theta, M_t) + \lambda_2 \mathcal{L}_E(\Phi_\theta). \tag{6}$$

The image reconstruction loss $\mathcal{L}_R$ is computed as a masked L1 loss on rendered images:

$$\mathcal{L}_R(\hat{I}_t, \{I_t, M_t\}) = \frac{1}{\sum_p M_t[p]} \sum_{p|M_t[p]=1} |I_t[p] - \hat{I}_t[p]|. \tag{7}$$

To quickly bootstrap the neural shape learning from the object masks, we apply a soft mask loss on the rendered image masks [4, 5]:

$$\mathcal{L}_M(\Phi_\theta, M_t) = \frac{1}{\alpha \sum_p M_t[p]} \sum_{p|M_t[p]=0 \vee \Phi_{min}[p]<\tau} \text{BCE}(\text{sigmoid}(-\alpha \Phi_{min}[p], M_t[p])), \tag{8}$$

where the notation $\Phi_{min}[p]$ denotes the minimum value of the SDF Φ_θ along the ray traced from pixel p, τ is a threshold for whether the zero level set $L_0(\Phi_\theta)$ has been intersected, and α is a softness hyperparameter. Finally, we regularize the shape representation to model a valid SDF by enforcing the eikonal constraint on randomly sampled points $p_i \in \mathbb{R}^3$ in a unit cube containing the scene:

$$\mathcal{L}_E(\Phi_\theta) = \frac{1}{P} \sum_{i=0}^{P} ||(||\nabla_p \Phi_\theta(p_i)||_2 - 1)||_2^2 \tag{9}$$

However, to make our training more efficient, we augment the loss supervision schedule and batching strategy for our model. Specifically, for each sampled batch of k images, instead of computing gradients for a single selected target image, we treat $l < k$ images as target images, and reconstruct them from the other images. Since all views must be sphere traced and passed through E for a single target view, this additional batching only adds additional forward passes to Γ and D, which are fast to evaluate. This batching strategy gives more accurate gradients for our model at each iteration. Additionally, since $\mathcal{L}_M$ requires us to find a minimum of Φ_θ along a ray, it requires dense sampling of this network and accounts for most of the compute time of each forward pass. Thus, while optimizing NLR++, we propose to only enforce shape-related losses $\mathcal{L}_M$ and $\mathcal{L}_E$ for the first t_1 iterations, and then every t_2 iterations thereafter. This allows NLR++ to learn a shape approximation in the first t_1 iterations, and then further refine it as the feature encoding scheme with E, D, Γ get significantly better. This is only possible since, unlike prior Neural Lumigraph work [4, 5], the appearance modeling is outsourced to the feature extraction from input images and is more independent from the current shape than a dense appearance representation in 3D space.

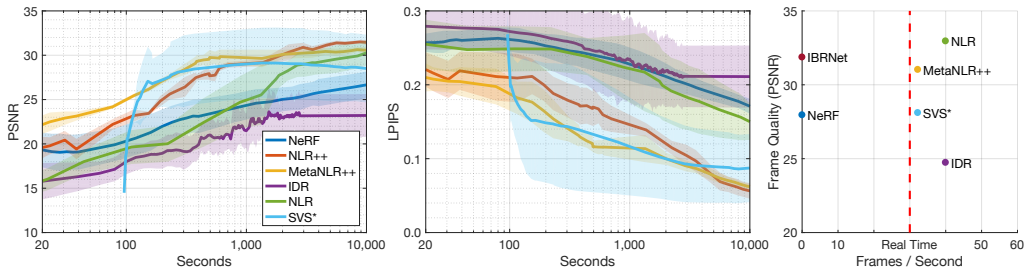

Figure 2: We demonstrate that at all training-times, MetaNLR++ is comparable to or outperforms all competitive representation learning methods, including both neural volumetric and surface representations in PSNR↑ (left) and LPIPS↓ [83] (center). We also plot the render time versus converged image quality, showing that MetaNLR++ can generate high-quality frames at real-time rates (right). The shaded area around each line represents the standard deviation of the method across three DTU scenes.

3.3 Generalization using meta learning

As our goal is to learn scene representations quickly, we use meta learning to learn a prior over feature encoding, decoding, aggregation, and shape representation using datasets of multi-view images. This prior is realized via the initialization of the networks $E_\xi, D_\psi, \Gamma_\zeta$ and Φ_θ which dictates the network convergence properties during gradient descent optimization. For simplicity of notation and since we are meta-learning the initializations for all networks in NLR++, we define all NLR++ parameters as $\Theta = [\xi, \psi, \zeta, \theta]$.

Let Θ_0 denote the NLR++ parameters at initialization, and Θ_i denote the parameters after i iterations of optimization. For a fixed number of steps m of optimization, Θ_m will depend significantly on the initialization Θ_0, resulting in possibly significantly different NLR++ losses. We adopt the notation from [78], and will emphasize the dependence of parameters on initialization by writing $\Theta_m(\Theta_0, T)$, where T is the particular scene which we would like to represent. We aim to optimize the initial weights Θ_0 that will result in the lowest possible expected loss after m iterations when optimizing NLR++ for an unseen object T, sampled from a distribution of objects $\mathcal{T}$. This expectation over objects is denoted as $E_{T\sim\mathcal{T}}$, resulting in the meta learning objective of:

$$\Theta_0^* = \arg\min_{\Theta_0} E_{T\sim\mathcal{T}}[\mathcal{L}(\Theta_m(\Theta_0, T))]. \tag{10}$$

To learn this initialization for Θ_0, we use the Reptile [81] algorithm, which computes the weight values $\Theta_m(\Theta_0, T)$ for a fixed inner loop step size of m. Each optimization of NLR++ for m steps for a different task sampled from $T_j \sim \mathcal{T}$ is referred to as an outer loop, and is indexed by j. To avoid having to compute second-order gradients through the NLR++ model, Reptile updates the initialization Θ_0 in the direction of the optimized task weights with the following equation:

$$\Theta_0^{j+1} = \Theta_0^j + \beta(\Theta_m(\Theta_0^j, T_j) - \Theta_0^j), \tag{11}$$

where β is the meta-learning rate hyperparameter. We label NLR++ with the meta-learning initializations applied MetaNLR++.

3.4 Implementation details

We plan to release the full code for our method, and the full set of implementation details including hyperparameters, training schedules, and architectures are described in our supplement for each of the various datasets evaluated on. We implement MetaNLR++ in PyTorch and use the Adam [82] optimizer for all optimization steps of NLR++, including for the inner-loop in meta learning, with a starting learning rate of 1×10^{-4} for Φ and 5×10^{-4} for E, D, Γ. We use $\alpha = 50$, $\tau = 1 \times 10^{-3}$, and $\beta = 1 \times 10^{-1}$ as starting hyperparameter values, which are progressively decayed (or increased in the case of α) through training (full schedules are described in the supplement). We use shape loss training hyperparameter values of $t_1 = 50$ and $t_2 = 7$, and loss weight parameters of $\lambda_1 = 1 \times 10^2/\alpha$, $\lambda_2 = 3$. In the case of NLR++, we initialize Φ as a unit sphere of radius 0.5 by pre-training our SIREN to represent this shape. We train each of our models on a single Nvidia Quadro RTX8000 GPU.

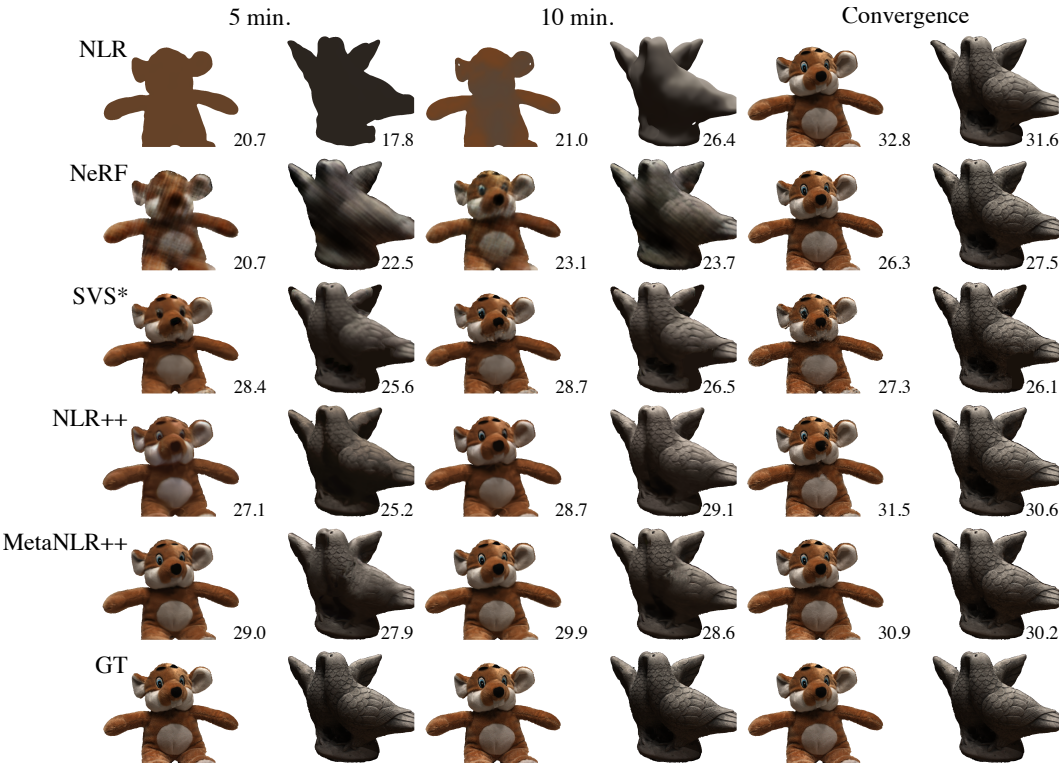

Figure 3: Novel views synthesized using various methods after a set training time. MetaNLR++ outperforms other surface and volume representation methods, especially for a training time budget on the order of minutes, and does not sacrifice quality of the final converged result.

We also use an Nvidia Quadro RTX6000 GPU for rendering and training iteration time computation. In total, we have an internal server system with four Nvidia Quadro RTX8000 GPUs and six Nvidia Quadro RTX6000 GPUs, of which we used a subset of three RTX8000s and one RTX6000.

4 Experiments

Baselines. Our main contribution is the rapid learning of a representation which can be used to render high-quality novel views of a scene in real time. We demonstrate this by comparison to several state-of-the-art methods. Specifically, we evaluate the volumetric representation of NeRF [59], a mesh-based representation similar to SVS [8], the neural signed distance function-based representations of IDR [4] and NLR [5], and the image-based rendering of IBRNet [28]. For SVS [8] we use our own simplified implementation and denote it SVS*. Our implementation trains the same E, D, Γ as in MetaNLR++ but we replace the learnable shape by a surface reconstruction from COLMAP [25, 26].

Training time vs. quality trade-off. For the following comparisons, we use the DTU dataset [4, 84], which has been made public by its creators, and contains multi-view images of various inanimate objects, none of which are offensive or personally identifiable. In Figure 2, we plot the average PSNR and LPIPS score of three held-out test views on three test DTU scenes as a function of training time as measured on a Nvidia Quadro RTX6000. Each of these representations are trained using only seven ground-truth views from the DTU scene. The meta-learned initializations Θ_0^* are optimized using complete view sets from another 15 DTU scenes, distinct from the testing scenes. In these plots, we see that MetaNLR++ maintains maximum reconstruction quality throughout the training process which results in predictable quality progression for time-constrained applications. Beyond PSNR we showcase results of the perceptual LPIPS metric [83] as we observe that PSNR is not robust to small inaccuracies in the object masks and prefers low-frequency images.

This trade-off is exemplified in Table 1, where we show that MetaNLR++ is able to reach the 25dB and 30dB PSNR milestones faster than any other method. The difference is particularly large for the volumetric method NeRF that aggregates many radiance samples along each rendered ray. SVS* is initially offset by the runtime of COLMAP. Afterwards, it quickly fits E, D, Γ, but its maximum performance is limited by the initial geometry. Consequently, D must learn to inpaint the resulting holes which results in over-fitting to training views. This is notable in Figures 2, 3 and Table 1, where the quality of novel views quickly saturates or even degrades. Additionally, the meshing step time scales with the number of input views, and it can take up to 2 hours for scenes with dense view coverage as reported in [8]. We emphesize that learning a representation from only seven views is a difficult task. NeRF in particular has difficulty avoiding over-fitting to training views in this scenario. We show qualitative results in Figure 3, which highlight that MetaNLR++ performs progressively higher quality view synthesis using the learned features and geometry as the training advances.

Table 1: Time to reach specified PSNR level. The best times and PSNR values are bolded.

	25dB PSNR	30dB PSNR	Maximum PSNR
NeRF	33.3 min.	-	27.95dB
IDR	-	-	24.73dB
NLR	14.7 min.	191.4 min.	**32.95dB**
SVS	2.1 min.	-	28.19dB
NLR++	3.2 min.	37.4 min.	31.02dB
MetaNLR++	**1.9 min.**	**22.5 min.**	30.57dB

Training to convergence. In Table 1 and Figure 3, we show that MetaNLR++ does not sacrifice on final converged quality for the sake of speed. Given unconstrained learning time, MetaNLR++ is still able to produce images which are competitive with converged results of state-of-the-art representations. IBRNet [28] is able to also produce high quality images quickly, but as the rendering time is based off of neural volume rendering, it cannot generate frames at nearly a real-time rate. Additionally, since IBRNet cannot be turned into a pre-computed mesh-based or volume representation and requires input images for each rendered frame, we consider it to be an image-blending method instead of a representation. We omit it from Table 1, but note that by pre-training on the 15 held-out DTU scenes it can render images with 29.20dB PSNR, and be fine tuned until convergence to render images at 31.86dB PSNR. Additional comparisons are provided in the supplementary document. A quantitative evaluation of image quality and runtime at rendering time is shown in Figure 2, where we see that our surface-based method is able to render in real time, unlike neural volume rendering methods.

Table 2: Meta-learning ablation study. We report average PSNR of synthesized views after 10 minutes for each method.

	Meta init. applied to:		
	No net.	Φ only	All net.
RGB pixels & fixed Γ	27.85dB	28.13dB	28.13dB
RGB pixels & learned Γ	26.82dB	26.90dB	26.91dB
CNN E/D & fixed Γ	28.62dB	29.16dB	29.88dB
CNN E/D & learned Γ	27.32dB	28.48dB	**29.91dB**

Real-time rendering. Because MetaNLR++ extracts the appearance directly from the source images and transforms them to the novel view using a compact shape model, we can greatly accelerate the rendering of our trained representation by pre-computing these components. First, we use the marching cubes algorithm [85] to extract a mesh as an iso-surface of the SDF. Second, we store the features computed from the set of input images by our encoder E as well as their associated depth maps. At render time, we simply need to aggregate these features based on our target viewing direction, and evaluate the decoder on the aggregated features for each frame. For the DTU images and network architecture, the decoder takes 8.7ms, and the aggregation takes 22.3ms, so we are able to render frames at real-time rates as shown in Figure 2.

Meta-learning ablation. We investigate the efficacy of meta learning in speeding up learning of individual components of our representation. Specifically, we ablate the contribution of meta learning on the shape representation parameters (θ_0), and feature encoding network parameters (ξ_0, ψ_0, ζ_0). Additionally, we also compare MetaNLR++ to a variant with the encoder and decoder replaced by a direct extraction of image RGB pixel values and with and without a learned aggregation function. The results of this ablation study are shown in Table 2.

We see that meta-learning of each component contributes to fast learning of a high-quality representation (last row). The learned Γ improves the performance for the learned E/D but it decreases

Figure 5: We compute additional results using the NLR dataset. The frames highlighted are supervised, and the intermediate frames are interpolated viewpoints for this MetaNLR++ model. Please see the supplement for additional results.

the performance for the directly extracted RGB values. This implies that the classical unstructured lumigraph blending [14] works well for direct pixel values, but the additional flexibility of the learned Γ can be advantageous with deep features. The learned Γ performs well when meta-learned but it has difficulty to accurately learn the angular dependencies in only 10 minutes of training otherwise. More details of this experiment and additional results are available in the supplementary document.

Input size ablation. We illustrate the robustness of MetaNLR++ to low numbers of available training views in Figure 4. Here we see that our view synthesis quality decreases very gracefully with decreasing number of input views and it produces meaningful results even in the extreme case of three views. This is unlike COLMAP, which produces geometry with significant holes in occluded regions. These holes are responsible for the highly variable performance of SVS* shown in Figure 2. Additionally, while SVS* is able to quantitatively perform well when evaluated within the ground-truth mask, the holes in the geometry result in inaccurate rendering masks and thus severely limit novel view synthesis in practice. Additional results showing this phenomenon are included in the supplementary material. Our learned shape models are overly smooth relative to other methods and thus not particularly quantitatively accurate, but provide sufficient accuracy for the image-based feature blending method to model appearance information observed in synthesized views.

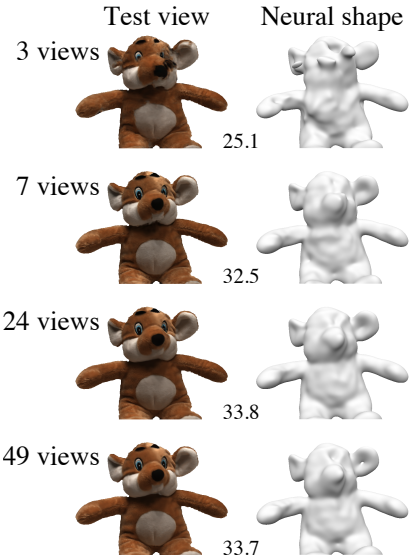

Figure 4: MetaNLR++ is robust to the number of views captured, which is essential in many applications where capturing a dense dataset is infeasible. In all cases, the learned Φ provides an adequate support for projection of our encoded appearance features.

Additional results. To show that MetaNLR++ is robust to datasets beyond DTU, we evaluate using the multi-view dataset in NLR [5]. This dataset has been publicly released, and while the faces of the subjects in the dataset are personally identifiable, the subjects are the authors of NLR and have provided their consent of making this dataset public. The scenes in this dataset each have 22 multi-view images, taken with various cameras. We opt to use the final 6 views taken with high-resolution cameras to train our representation. We use 5 scenes in this dataset to learn the meta initialization Θ_0^*, and test on a withheld sixth scene. The meta learning in this case leads to significantly improved performance, despite only being able to learn a prior from a small number of scenes.

In Table 3, we report our fit on the training frames after 30 minutes of training as benchmarked on our system. Additionally, we show interpolated frame results in Figure 5, demonstrating that MetaNLR++ is capable of generalizing to this scene and producing convincing novel-view synthesis results. Additional implementation information and results are included in the supplementary document.

	30 min. PSNR (LPIPS)	Convergence PSNR (LPIPS)
NLR++	28.22 (0.145)	35.54 (0.046)
MetaNLR++	**31.12 (0.083)**	**37.55 (0.034)**

Table 3: Table comparing PSNR score on the supervised views of the NLR dataset. Since there are only 6 training views, we use all for training and report PSNR averaged on the three views shown in Figure 5, which shows that we are able to interpolate between these views.

5 Discussion and Conclusion

The fundamental problem of learning 3D scene representations from sparse sets of 2D images is rooted in machine learning, computer vision, and computer graphics. We provide an answer to this problem by proposing a novel parameterization of a 3D scene, and an efficient method for inferring these parameters from observations using meta learning. We demonstrate that our representation and training method are able to reduce representation training time consistently and render at real-time rates without sacrificing on image synthesis quality. This opens several exciting directions for future work in efficient training and rendering of representations, including using more advanced generalization methods to learn representations in real time. With this work, we make important contributions to the field of neural rendering.

Limitations and future work. While our method is able to produce compelling novel view synthesis results in a fraction of the time of other methods, we note that there are a few shortcomings. Specifically, in order to bootstrap the learning of a neural shape quickly, object masks are required to supervise the ray-mask loss. While these can be computed automatically for some data, this poses a challenge in cluttered scenes, or applications which could generalize to arbitrary scenes. Additionally, all of our experiments have used known camera poses to reconstruct the shape. Future work on jointly optimizing camera pose with our representation is certainly possible, and a step in the direction for general view synthesis. Our method is also limited by memory consumption, since the CNN feature encoder/decoders process the entire image at a time. This method could likely be improved by shifting to training and evaluating on image patches for higher resolution rendering. Finally, our method tends to produce overly-smoothed shape models, which, while beneficial for feature aggregation, are not always representative of high-frequency scene geometry. This highlights one fundamental trade-off: the capacity of the feature generation method versus the quality of the shape. With feature or color generation methods which are sufficiently regularized [4, 5], the model has no choice but to explain observed details in the neural shape model. We opt to utilize the full capacity of the CNN feature processing, as learning a detailed neural shape is slower than modeling fine details with features.

Conclusion. The question of how to trade-off image synthesis quality with representation training and rendering time is of paramount importance to engineers, producers, or any other users of neural rendering technology. In the space of neural rendering methods, this work takes steps towards making representation learning and rendering more practical by optimizing this trade-off. Our novel scene parameterization and generalization method may provide a starting point for future work in optimizing this trade-off: speeding up representation training and rendering time and bringing modern neural rendering to the forefront of industry standard techniques.

6 Broader Impact

Methods such as MetaNLR++ for learning 3D scene representations from 2D representations allow for photorealistic image synthesis using only collections of other images. We have shown that MetaNLR++ improves upon the axes of training and rendering time of these representations, and as such may make it less computationally restrictive to use for individuals who want to learn and use 3D models from only collections of easily acquirable images. While this proliferation of neural rendering technology may be extremely helpful for many, it has the potential for misuse. As with any synthesis method, the technology could enable approaches to synthesis of deliberately misleading or offensive images, posing challenges similar to those posed by generative-adversarial models.

References

[1] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. *Eurographics*, 2020.

[2] 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. Graph. (SIGGRAPH)*, 38(4), 2019.

[3] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020.

[4] 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 *NeurIPS*, 2020.

[5] Petr Kellnhofer, Lars Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *CVPR*, 2021.

[6] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *arXiv preprint arXiv:2104.10078*, 2021.

[7] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020.

[8] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021.

[9] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH*, pages 279–288, 1993.

[10] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996.

[11] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[12] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[13] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH*, pages 231–242, 1998.

[14] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432, 2001.

[15] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing*, volume 4067, pages 2–13, 2000.

[16] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3):1–12, 2013.

[17] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6):1–11, 2017.

[18] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[19] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph. (SIGGRAPH Asia)*, 37(6), 2018.

[20] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *ICCV*, pages 7781–7790, 2019.

[21] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *CVPR*, pages 4090–4100, 2019.

[22] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, pages 6878–6887, 2019.

[23] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4):1–12, 2019.

[24] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: implicit neural view-, light-and time-image interpolation. *ACM Trans. Graph.*, 39(6):1–15, 2020.

[25] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[27] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021.

[28] 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. *arXiv preprint arXiv:2102.13090*, 2021.

[29] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, pages 371–386, 2018.

[30] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (SIGGRAPH)*, 2018.

[31] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proc. CVPR*, pages 2367–2376, 2019.

[32] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, pages 551–560, 2020.

[33] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021.

[34] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Trans. Graph. (SIGGRAPH)*, 39(4), 2020.

[35] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. MatryODShka: Real-time 6DoF video view synthesis using multi-sphere images. In *Proc. ECCV*, August 2020. URL `https://visual.cs.brown.edu/matryodshka`.

[36] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. CVPR*, 2019.

[37] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph. (SIGGRAPH)*, 38(4), 2019.

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR*, 2019.

[39] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

[40] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, pages 7154–7164, 2019.

[41] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Deep structured implicit functions. *arXiv preprint arXiv:1912.06126*, 2019.

[42] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, pages 5939–5948, 2019.

[43] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *ICCV*, pages 4743–4752, 2019.

[44] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *CVPR*, 2020.

[45] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, pages 2304–2314, 2019.

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

[47] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019.

[48] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[49] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation, 2020.

[50] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

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

[52] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *CVPR*, 2020.

[53] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, 2020.

[54] Amit Kohli, Vincent Sitzmann, and Gordon Wetzstein. Inferring semantic information with 3d neural scene representations. *arXiv preprint arXiv:2003.12673*, 2020.

[55] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

[56] Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[57] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision (ECCV)*, 2020.

[58] Julien N.P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural representation. *ACM Trans. Graph. (SIGGRAPH)*, 2021.

[59] 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 *ECCV*, 2020.

[60] 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 *CVPR*, 2021.

[61] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021.

[62] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[63] Pratul P 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 preprint arXiv:2012.03927*, 2020.

[64] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[65] 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 preprint arXiv:2103.03231*, 2021.

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

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

[68] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

[69] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*, 2021.

[70] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, 2019.

[71] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. In *NeurIPS*, 2020.

[72] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020.

[73] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, 2021.

[74] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. *arXiv preprint arXiv:2103.17269*, 2021.

[75] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *CVPR*, 2021.

[76] Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations. *arXiv preprint arXiv:2104.03960*, 2021.

[77] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. In *NeurIPS*, 2020.

[78] 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 *CVPR*, 2021.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[80] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

[81] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[83] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018.

[84] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[85] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH*, page 163–169, 1987.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We plan to release the code completely but have not yet with submission.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary document and Section 3.4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] As models take a significant amount of time to train, averaging over many seeds would be computationally impractical. We report standard deviation error bars with respect to multiple testing scenarios, which demonstrate the robustness of our method.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We were unable to find the license of these datasets, but the authors have explicitly published their datasets to the public. We also discuss the nature of the datasets including their content and consent in Section 4
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Section 4

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 4, and supplementary document for additional discussion on the datasets.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]