DOI:10.1145/3503250

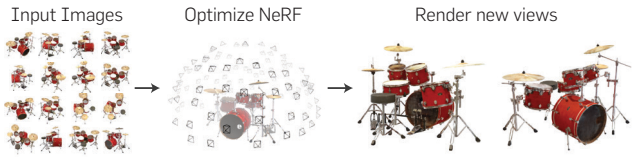
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

By Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng

Abstract

We present a method that achieves state-of-the-art results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views. Our algorithm represents a scene using a fully connected (nonconvolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) and whose output is the volume density and view-dependent emitted radiance at that spatial location. We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis.

Figure 1. We present a method that optimizes a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. We use techniques from volume rendering to accumulate samples of this scene representation along rays to render the scene from any viewpoint. Here, we visualize the set of 100 input views of the synthetic *Drums* scene randomly captured on a surrounding hemisphere, and we show two novel views rendered from our optimized NeRF representation.



1. INTRODUCTION

In this work, we address the long-standing problem of view synthesis in a new way. View synthesis is the problem of rendering new views of a scene from a given set of input images and their respective camera poses. Producing photorealistic outputs from new viewpoints requires correctly handling complex geometry and material reflectance properties. Many different scene representations and rendering methods have been proposed to attack this problem; however, so far none have been able to achieve photorealistic quality over a large camera baseline. We propose a new scene representation that can be optimized directly to reproduce a large number of high-resolution input views and is still extremely memory-efficient (see Figure 1).

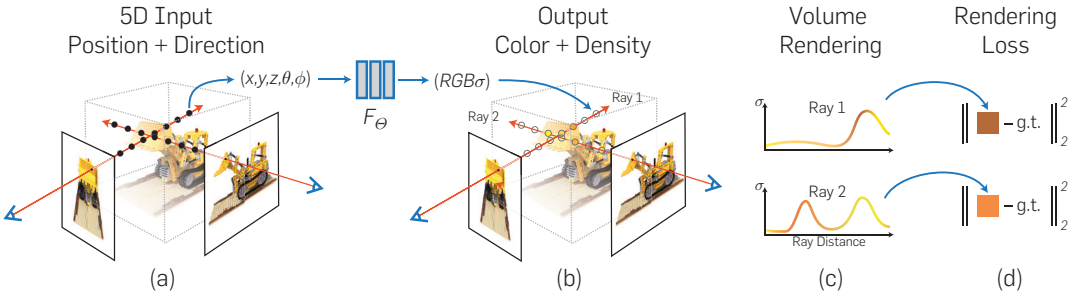
We represent a static scene as a continuous 5D function that outputs the radiance emitted in each direction (θ, ϕ) at each point (x, y, z) in space, and a density at each point which acts like a differential opacity controlling how much radiance is accumulated by a ray passing through (x, y, z). Our method optimizes a deep fully connected neural network without any convolutional layers (often referred to as a multilayer perceptron or MLP) to represent this function by regressing from a single 5D coordinate (x, y, z, θ, ϕ) to a single volume density and view-dependent RGB color. To render this *neural radiance field* (NeRF) from a particular viewpoint, we: 1) march camera rays through the scene to generate a sampled set of 3D points, 2) use those points and their corresponding 2D viewing directions as input to the neural network to produce an output set of colors and densities, and 3) use classical volume rendering techniques to accumulate those colors and densities into a 2D image. Because this process is naturally differentiable, we can use gradient descent to optimize this model by minimizing the error between each observed image and the corresponding views rendered from our representation. Minimizing this error across multiple views encourages the network to predict a coherent model of the scene by assigning high-volume densities and accurate colors to the locations that contain the true underlying scene content. Figure 2 visualizes this overall pipeline.

We find that the basic implementation of optimizing a neural radiance field representation for a complex scene does not converge to a sufficiently high-resolution representation. We address this issue by transforming input 5D coordinates with a positional encoding that enables the MLP to represent higher frequency functions.

Our approach can represent complex real-world geometry and appearance and is well suited for gradient-based optimization using projected images. By storing a scene in the parameters of a neural network, our method overcomes the prohibitive storage costs of *discretized* voxel grids when modeling complex scenes at high resolutions. We demonstrate that our resulting neural radiance field method quantitatively

The original version of this paper was published in *Proceedings of the 2020 European Conference on Computer Vision*.

Figure 2. An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).



and qualitatively outperforms state-of-the-art view synthesis methods, such as works that fit neural 3D representations to scenes as well as works that train deep convolutional networks (CNNs) to predict sampled volumetric representations. This paper presents the first continuous neural scene representation that is able to render high-resolution photorealistic novel views of real objects and scenes from RGB images captured in natural settings.

2. RELATED WORK

A promising recent direction in computer vision is encoding objects and scenes in the weights of an MLP that directly maps from a 3D spatial location to an implicit representation of the shape, such as the signed distance[3] at that location. However, these methods have so far been unable to reproduce realistic scenes with complex geometry with the same fidelity as techniques that represent scenes using discrete representations such as triangle meshes or voxel grids. In this section, we review these two lines of work and contrast them with our approach, which enhances the capabilities of neural scene representations to produce state-of-the-art results for rendering complex realistic scenes.

2.1. Neural 3D shape representations

Recent work has investigated the implicit representation of continuous 3D shapes as level sets by optimizing deep networks that map *xyz* coordinates to signed distance functions[15] or occupancy fields.[11] However, these models are limited by their requirement of access to ground truth 3D geometry, typically obtained from synthetic 3D shape datasets such as ShapeNet.[2] Subsequent work has relaxed this requirement of ground truth 3D shapes by formulating differentiable rendering functions that allow neural implicit shape representations to be optimized using only 2D images. Niemeyer et al.[14] represent surfaces as 3D occupancy fields and use a numerical method to find the surface intersection for each ray, then calculate an exact derivative using implicit differentiation. Each ray intersection location is provided as the input to a neural 3D texture field that predicts a diffuse color for that point. Sitzmann et al.[21] use a less direct neural 3D representation that simply outputs a feature vector and RGB color at each continuous 3D coordinate, and propose a differentiable rendering function consisting of a recurrent neural network that marches along each ray to decide where the surface is located.

Though these techniques can potentially represent complicated and high-resolution geometry, they have so far been limited to simple shapes with low geometric complexity, resulting in oversmoothed renderings. We show that an alternate strategy of optimizing networks to encode 5D radiance fields (3D volumes with 2D view-dependent appearance) can represent higher resolution geometry and appearance to render photorealistic novel views of complex scenes.

2.2. View synthesis and image-based rendering

The computer vision and graphics communities have made significant progress on the task of novel view synthesis by predicting traditional geometry and appearance representations from observed images. One popular class of approaches uses mesh-based scene representations.[1, 4, 23] Differentiable rasterizers[9] or pathtracers[7] can directly optimize mesh representations to reproduce a set of input images using gradient descent. However, gradient-based mesh optimization based on image reprojection is often difficult, likely because of local minima or poor conditioning of the loss landscape. Furthermore, this strategy requires a template mesh with fixed topology to be provided as an initialization before optimization,[7] which is typically unavailable for unconstrained real-world scenes.

Another class of methods use volumetric representations to address the task of high-quality photorealistic view synthesis from a set of input RGB images. Volumetric approaches are able to realistically represent complex shapes and materials, are well suited for gradient-based optimization, and tend to produce less visually distracting artifacts than mesh-based methods. Early volumetric approaches used observed images to directly color voxel grids.[19] More recently, several methods[12, 25] have used large datasets of multiple scenes to train deep networks that predict a sampled volumetric representation from a set of input images, and then use either alpha compositing[16] or learned compositing along rays to render novel views at test time. Other works have optimized a combination of CNNs and sampled voxel grids for each specific scene, such that the CNN can compensate for discretization artifacts

from low-resolution voxel grids[20] or allow the predicted voxel grids to vary based on input time or animation controls.[8] Although these volumetric techniques have achieved impressive results for novel view synthesis, their ability to scale to higher resolution imagery is fundamentally limited by poor time and space complexity due to their discrete sampling—rendering higher resolution images requires a finer sampling of 3D space. We circumvent this problem by instead encoding a *continuous* volume within the parameters of a deep fully connected neural network, which not only produces significantly higher quality renderings than prior volumetric approaches but also requires just a fraction of the storage cost of those *sampled* volumetric representations.

3. NEURAL RADIANCE FIELD SCENE REPRESENTATION

We represent a continuous scene as a 5D vector-valued function whose input is a 3D location $\mathbf{x} = (x, y, z)$ and 2D viewing direction (θ, ϕ), and whose output is an emitted color $\mathbf{c} = (r, g, b)$ and volume density σ. In practice, we express direction as a 3D Cartesian unit vector $\mathbf{d}$. We approximate this continuous 5D scene representation with an MLP network F_Θ: $(\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ and optimize its weights Θ to map from each input 5D coordinate to its corresponding volume density and directional emitted color.

We encourage the representation to be multiview consistent by restricting the network to predict the volume density σ as a function of only the location $\mathbf{x}$, while allowing the RGB color $\mathbf{c}$ to be predicted as a function of both location and viewing direction. To accomplish this, the MLP F_Θ first processes the input 3D coordinate $\mathbf{x}$ with 8 fully connected layers (using ReLU activations and 256 channels per layer), and outputs σ and a 256-dimensional feature vector. This feature vector is then concatenated with the camera ray's viewing direction and passed to one additional fully connected layer (using a ReLU activation and 128 channels) that output the view-dependent RGB color.

See Figure 3 for an example of how our method uses the input viewing direction to represent non-Lambertian effects. As shown in Figure 4, a model trained without view dependence (only $\mathbf{x}$ as input) has difficulty representing specularities.

Figure 3. A visualization of view-dependent emitted radiance. Our neural radiance field representation outputs RGB color as a 5D function of both spatial position x and viewing direction d. Here, we visualize example directional color distributions for two spatial locations in our neural representation of the *Ship* scene. In (a) and (b), we show the appearance of two fixed 3D points from two different camera positions: one on the side of the ship (orange insets) and one on the surface of the water (blue insets). Our method predicts the changing specular appearance of these two 3D points, and in (c) we show how this behavior generalizes continuously across the whole hemisphere of viewing directions.

(a) View 1 (b) View 2 (c) Radiance Distributions

4. VOLUME RENDERING WITH RADIANCE FIELDS

Our 5D neural radiance field represents a scene as the volume density and directional emitted radiance at any point in space. We render the color of any ray passing through the scene using principles from classical volume rendering.[5] The volume density $\sigma(\mathbf{x})$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location $\mathbf{x}$. The expected color $C(\mathbf{r})$ of camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with near and far bounds t_n and t_f is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \quad (1)$$

$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \quad (2)$$

The function $T(t)$ denotes the accumulated transmittance along the ray from t_n to t, that is, the probability that the ray travels from t_n to t without hitting any other particle. Rendering a view from our continuous neural radiance field requires estimating this integral $C(\mathbf{r})$ for a camera ray traced through each pixel of the desired virtual camera.

We numerically estimate this continuous integral using quadrature. Deterministic quadrature, which is typically used for rendering discretized voxel grids, would effectively limit our representation's resolution because the MLP would only be queried at a fixed discrete set of locations. Instead, we use a stratified sampling approach where we partition $[t_n, t_f]$ into N evenly spaced bins and then draw one sample uniformly at random from within each bin:

$$t_i \sim u\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right]. \quad (3)$$

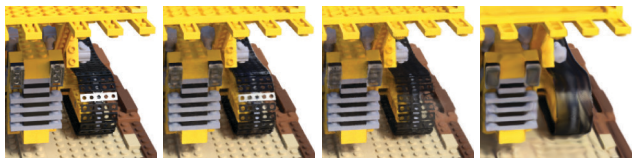
Although we use a discrete set of samples to estimate the integral, stratified sampling enables us to represent a continuous scene representation because it results in the MLP being evaluated at continuous positions over the course of optimization. We use these samples to estimate $C(\mathbf{r})$ with the quadrature rule discussed in the volume rendering review by Max[10]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \quad (4)$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right), \quad (5)$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples.

Figure 4. Here we visualize how our full model benefits from representing view-dependent emitted radiance and from passing our input coordinates through a high-frequency positional encoding. Removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the positional encoding drastically decreases the model's ability to represent high-frequency geometry and texture, resulting in an oversmoothed appearance.



Ground Truth Full Model No View Dep. No Pos. Enc.

This function for calculating $\hat{C}(\mathbf{r})$ from the set of $(\mathbf{c}_i, \sigma_i)$ values is trivially differentiable and reduces to traditional alpha compositing with alpha values $\sigma_i = 1 - \exp(-\sigma_i \delta_i)$.

5. OPTIMIZING A NEURAL RADIANCE FIELD

In the previous section, we have described the core components necessary for modeling a scene as a neural radiance field and rendering novel views from this representation. However, we observe that these components are not sufficient for achieving state-of-the-art quality. We introduce two improvements to enable representing high-resolution complex scenes. The first is a positional encoding of the input coordinates that assists the MLP in representing high-frequency functions. The second is a hierarchical sampling procedure that we do not describe here; for details, see the original paper.[13]

5.1. Positional encoding

Despite the fact that neural networks are universal function approximators, we found that having the network F_Θ directly operate on $xyz\theta\phi$ input coordinates results in renderings that perform poorly at representing high-frequency variation in color and geometry. This is consistent with recent work by Rahaman et al.,[17] which shows that deep networks are biased toward learning lower frequency functions. They additionally show that mapping the inputs to a higher dimensional space using high-frequency functions before passing them to the network enables better fitting of data that contains high-frequency variation.

We leverage these findings in the context of neural scene representations, and show that reformulating F_Θ as a composition of two functions $F_\Theta = F'_\Theta \circ \gamma$, one learned and one not, significantly improves performance (see Figure 4). Here γ is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$, and F'_Θ is still simply a regular MLP. Formally, the encoding function we use is:

$$\gamma(p) = \left(\sin(2^0 \pi p), \cos(2^0 \pi p), \cdots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)\right). \quad (6)$$

This function $\gamma(\cdot)$ is applied separately to each of the three coordinate values in $\mathbf{x}$ (which are normalized to lie in [−1, 1]) and to the three components of the Cartesian viewing direction unit vector $\mathbf{d}$ (which by construction lie in [−1, 1]). In our experiments, we set $L = 10$ for $\gamma(\mathbf{X})$ and $L = 4$ for $\gamma(\mathbf{d})$.

This mapping is studied in more depth in subsequent work[22] which shows how positional encoding enables a network to more rapidly represent higher frequency signals.

5.2. Implementation details

We optimize a separate neural continuous volume representation network for each scene. This requires only a dataset of captured RGB images of the scene, the corresponding camera poses and intrinsic parameters, and scene bounds (we use ground truth camera poses, intrinsics, and bounds for synthetic data, and use the COLMAP structure-from-motion package[18] to estimate these parameters for real data). At each optimization iteration, we randomly sample a batch of camera rays from the set of all pixels in the dataset. We query the network at N random points along each ray and then use the volume rendering procedure described in Section 4 to render the color of each ray using these samples. Our loss is simply the total squared error between the rendered and true pixel colors:

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left\| \hat{C}(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \quad (7)$$

where R is the set of rays in each batch, and $C(\mathbf{r})$, $\hat{C}(\mathbf{r})$ are the ground truth and predicted RGB colors for ray $\mathbf{r}$.

In our experiments, we use a batch size of 4096 rays, each sampled at $N = 192$ coordinates. (These are divided between two hierarchical "coarse" and "fine" networks; for details see the original paper.[13]) We use the Adam optimizer[6] with a learning rate that begins at 5×10^{-4} and decays exponentially to 5×10^{-5}. The optimization for a single scene typically takes about 1–2 days to converge on a single GPU.

6. RESULTS

We quantitatively (Table 1) and qualitatively (see Figures 5 and 6) show that our method outperforms prior work. We urge the reader to view our accompanying video to better appreciate our method's significant improvement over baseline methods when rendering smooth paths of novel views. Videos, code, and datasets can be found at https://www.matthew.

6.1. Datasets

Synthetic renderings of objects. We first show experimental results on two datasets of synthetic renderings of objects (Table 1, "Diffuse Synthetic 360°" and "Realistic Synthetic 360°"). The DeepVoxels[20] dataset contains four Lambertian

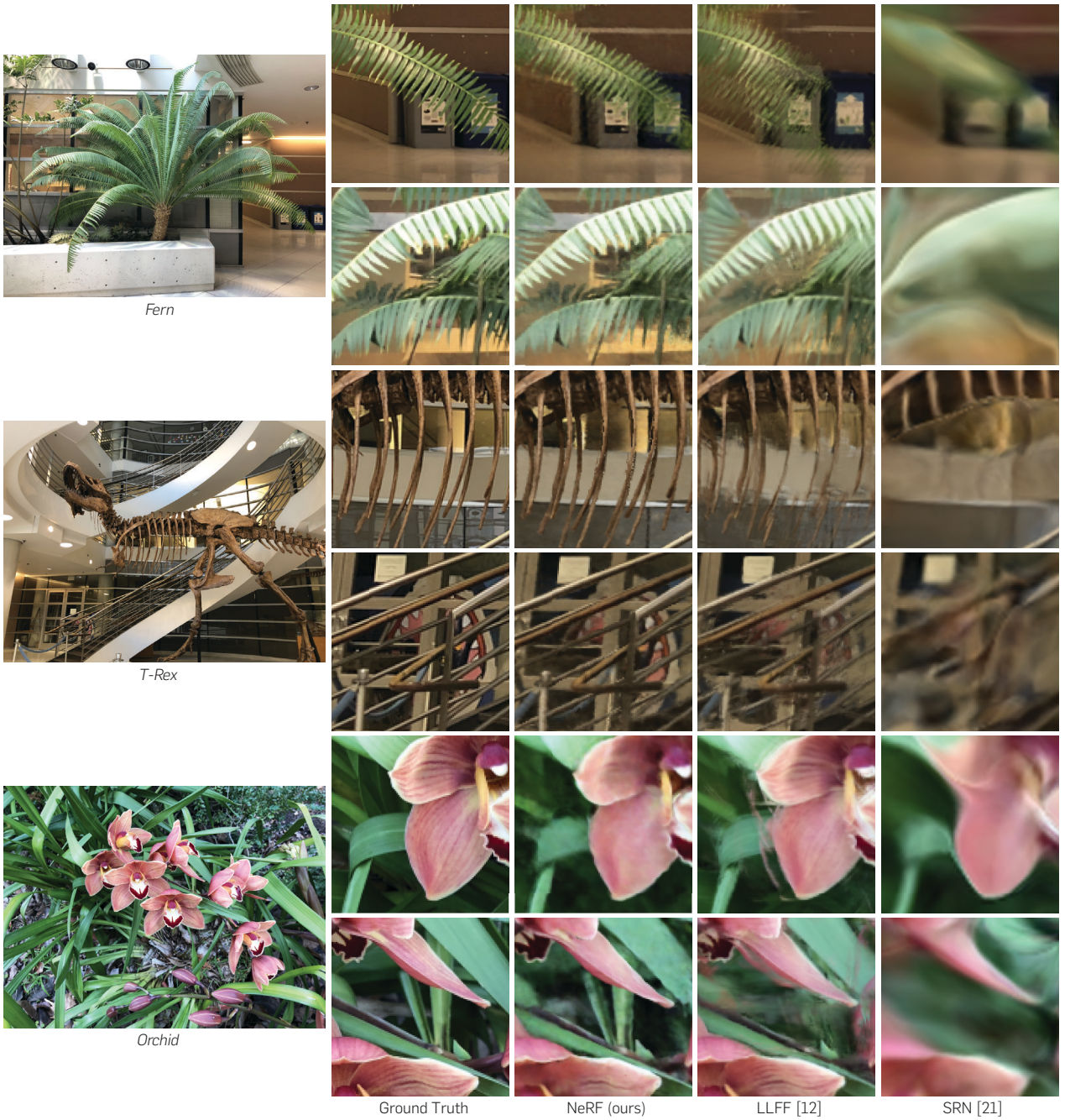
Table 1. Our method quantitatively outperforms prior work on datasets of both synthetic and real images.

	Diffuse Synthetic 360°[20]			Realistic Synthetic 360°			Real ForwardFacing[12]		
Method	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**
SRN[21]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV[8]	29.62	0.929	0.099	26.05	0.893	0.160	–	–	–
LLFF[12]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

We report PSNR/SSIM (higher is better) and LPIPS[24] (lower is better). The DeepVoxels[20] dataset consists of 4 diffuse objects with simple geometry. Our realistic synthetic dataset consists of pathtraced renderings of 8 geometrically complex objects with complex non-Lambertian materials. The real dataset consists of handheld forward-facing captures of 8 real-world scenes (NV cannot be evaluated on this data because it only reconstructs objects inside a bounded volume).
Bold values denote the top-performing algorithm for each of these metrics.

Figure 5. Comparisons on test-set views for scenes from our new synthetic dataset generated with a physically based renderer. Our method is able to recover fine details in both geometry and appearance, such as *Ship*'s rigging, *Lego*'s gear and treads, *Microphone*'s shiny stand and mesh grille, and *Material*'s non-Lambertian reflectance. LLFF exhibits banding artifacts on the *Microphone* stand and *Material*'s object edges and ghosting artifacts in *Ship*'s mast and inside the *Lego* object. SRN produces blurry and distorted renderings in every case. Neural Volumes cannot capture the details on the *Microphone*'s grille or *Lego*'s gears, and it completely fails to recover the geometry of *Ship*'s rigging.

Figure 6. Comparisons on test-set views of real-world scenes. LLFF is specifically designed for this use case (forward-facing captures of real scenes). Our method is able to represent fine geometry more consistently across rendered views than LLFF, as shown in *Fern*'s leaves and the skeleton ribs and railing in *T-rex*. Our method also correctly reconstructs partially occluded regions that LLFF struggles to render cleanly, such as the yellow shelves behind the leaves in the bottom *Fern* crop and green leaves in the background of the bottom *Orchid* crop. Blending between multiples renderings can also cause repeated edges in LLFF, as seen in the top *Orchid* crop. SRN captures the low-frequency geometry and color variation in each scene but is unable to reproduce any fine detail.



objects with simple geometry. Each object is rendered at 512 × 512 pixels from viewpoints sampled on the upper hemisphere (479 as input and 1000 for testing). We additionally generate our own dataset containing pathtraced images of eight objects that exhibit complicated geometry and realistic non-Lambertian materials. Six are rendered from viewpoints sampled on the upper hemisphere, and two are rendered from viewpoints sampled on a full sphere. We render 100 views of each scene as input and 200 for testing, all at 800 × 800 pixels.

Real images of complex scenes. We show results on complex real-world scenes captured with roughly forward-facing

images (Table 1, "Real ForwardFacing"). This dataset consists of eight scenes captured with a handheld cellphone (five taken from the local light field fusion (LLFF) paper and three that we capture), captured with 20 to 62 images, and hold out $^1/_8$ of these for the test set. All images are 1008×756 pixels.

6.2. Comparisons

To evaluate our model we compare against current top-performing techniques for view synthesis, detailed here. All methods use the same set of input views to train a separate network for each scene except LLFF,[12] which trains a single 3D CNN on a large dataset, then uses the same trained network to process input images of new scenes at test time.

Neural Volumes (NV)[8] synthesizes novel views of objects that lie entirely within a bounded volume in front of a distinct background (which must be separately captured without the object of interest). It optimizes a deep 3D CNN to predict a discretized RGBα voxel grid with 128^3 samples as well as a 3D warp grid with 32^3 samples. The algorithm renders novel views by marching camera rays through the warped voxel grid.

Scene Representation Networks (SRN)[21] represent a continuous scene as an opaque surface, implicitly defined by an MLP that maps each (x, y, z) coordinate to a feature vector. They train a recurrent neural network to march along a ray through the scene representation by using the feature vector at any 3D coordinate to predict the next step size along the ray. The feature vector from the final step is decoded into a single color for that point on the surface. Note that SRN is a better-performing follow-up to DeepVoxels[20] by the same authors, which is why we do not include comparisons to DeepVoxels.

LLFF[12] is designed for producing photorealistic novel views for well-sampled forward-facing scenes. It uses a trained 3D CNN to directly predict a discretized frustum-sampled RGBα grid (multiplane image or MPI[25]) for each input view, then renders novel views by alpha compositing and blending nearby MPIs into the novel viewpoint.

6.3. Discussion

We thoroughly outperform both baselines that also optimize a separate network per scene (NV and SRN) in all scenarios. Furthermore, we produce qualitatively and quantitatively superior renderings compared to LLFF (across all except one metric) while using only their input images as our entire training set.

The SRN method produces heavily smoothed geometry and texture, and its representational power for view synthesis is limited by selecting only a single depth and color per camera ray. The NV baseline is able to capture reasonably detailed volumetric geometry and appearance, but its use of an underlying explicit 128^3 voxel grid prevents it from scaling to represent fine details at high resolutions. LLFF specifically provides a "sampling guideline" to not exceed 64 pixels of disparity between input views, so it frequently fails to estimate correct geometry in the synthetic datasets which contain up to 400–500 pixels of disparity between views. Additionally, LLFF blends between different scene representations for rendering different views, resulting in perceptually distracting inconsistency as is apparent in our supplementary video.

The biggest practical trade-offs between these methods are time versus space. All compared single scene methods take at least 12 hours to train per scene. In contrast, LLFF can process a small input dataset in under 10 min. However, LLFF produces a large 3D voxel grid for every input image, resulting in enormous storage requirements (over 15GB for one "Realistic Synthetic" scene). Our method requires only 5MB for the network weights (a relative compression of 3000 $\times$ compared to LLFF), which is even less memory than the *input images alone* for a single scene from any of our datasets.

7. CONCLUSION

Our work directly addresses deficiencies of prior work that uses MLPs to represent objects and scenes as continuous functions. We demonstrate that representing scenes as 5D neural radiance fields (an MLP that outputs volume density and view-dependent emitted radiance as a function of 3D location and 2D viewing direction) produces better renderings than the previously dominant approach of training deep CNNs to output discretized voxel representations.

We believe that this work makes progress toward a graphics pipeline based on real-world imagery, where complex scenes could be composed of neural radiance fields optimized from images of actual objects and scenes. Indeed, many recent methods have already built upon the neural radiance field representation presented in this work and extended it to enable more functionality such as relighting, deformations, and animation.

Acknowledgments

We thank Kevin Cao, Guowei Frank Yang, and Nithin Raghavan for comments and discussions. RR acknowledges funding from ONR grants N000141712687, N000141912293 N000142012529, NSF Chase-CI and the Ronald L. Graham Chair. BM is funded by a Hertz Foundation Fellowship, and MT is funded by an NSF Graduate Fellowship. Google provided a generous donation of cloud compute credits through the BAIR Commons program. We thank the following Blend Swap users for the models used in our realistic synthetic dataset: gregzaal (ship), 1DInc (chair), bryanajones (drums), Herberhold (ficus), erickfree (hotdog), Heinzelnisse (lego), elbrujodelatribu (materials), and up3d.de (mic).

References

1. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M. Unstructured lumigraph rendering. In *SIGGRAPH* (2001).
2. Chang, A.X., Fhnkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. ShapeNet: An information-rich 3D model repository. arXiv:1512.03012 (2015).
3. Curless, B., Levoy, M. A volumetric method for building complex models from range images. In *SIGGRAPH* (1996).
4. Debevec, P., Taylor, C.J., Malik,J. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH* (1996).
5. Kajiya, J.T., Herzen, B.P.V. Ray tracing volume densities. *Comput. Graph. (SIGGRAPH)* (1984).
6. Kingma, D.P., Ba, J. Adam: A method for stochastic optimization. In *ICLR* (2015).
7. Li, T.-M., Aittala, M., Durand, F., Lehtinen, J. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (SIGGRAPH Asia)* (2018).

8. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph. (SIGGRAPH)* (2019).
9. Loper, M.M., Black, M.J. OpenDR: An approximate differentiable renderer. In *ECCV* (2014).
10. Max, N. Optical models for direct volume rendering. *IEEE Trans. Visual. Comput. Graph.* (1995).
11. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR* (2019).
12. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph. (SIGGRAPH)* (2019).
13. Mildenhall, B., Srinivasan, P.P, Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020).
14. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In *CVPR* (2019).
15. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR* (2019).
16. Porter, T., Duff, T. Compositing digital images. *Comput. Graph. (SIGGRAPH)* (1984).
17. Rahaman, N., Baratin, A., Arpit, D., Dräxler, F., Lin, M., Hamprecht, F.A., Bengio, Y., Courville, A.C. On the spectral bias of neural networks. In *ICML* (2018).
18. Schönberger, J.L., Frahm, J.-M. Structure-from-motion revisited. In *CVPR* (2016).
19. Seitz, S.M., Dyer, C.R. Photorealistic scene reconstruction by voxel coloring. *Int. J. Comput. Vision* (1999).
20. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhöfer, M. Deepvoxels: Learning persistent 3D feature embeddings. In *CVPR* (2019).
21. Sitzmann, V., Zollhoefer, M., Wetzstein, G. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS* (2019).
22. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS* (2020).
23. Wood, D.N., Azuma, D.I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D.H., Stuetzle, W. Surface light fields for 3D photography. In *SIGGRAPH* (2000).
24. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018).
25. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (SIGGRAPH)* (2018).

Ben Mildenhall ([bmild]@cs.berkeley.edu), UC Berkeley, Berkeley, CA, USA

Pratul P. Srinivasan, Matthew Tancik, and Ren Ng ([pratul, tancik, ren]@berkeley.edu), UC Berkeley, Berkeley, CA, USA.

Jonathan T. Barron ([barron]@google.com), Google Research Mountain View, CA, USA.

Ravi Ramamoorthi ([ravir]@cs.ucsd.edu), UC San Diego, La Jolla, CA, USA.

Ben Mildenhall, Pratul P. Srinivasan, and **Matthew Tancik** contributed equally to this work.