# Neural Radiance Fields (NERF) for Novel View Synthesis

Chris Fritz,
E-mail: fritz17236@hotmail.com

**Abstract**—Novel view synthesis seeks to reconstruct a visual scene from an arbitrary viewpoint given sparse snapshots (images) of the scene. Discrete methods such as voxel-grids can perform this task, but at the cost of prohibitively high 3D sample rates that quickly exceed hardware capability, or conversely perform on hardware but at substantial degredation in image quality. Here, we reproduce and study *Neural Radiance Fields*, an emerging approach that overcomes the classical limitations by leveraging a fully-connected neural network (multilayer perceptron, MLP) in order to learn a dense representation of the visual scene and allow for photorealistic view synthesis with more reasonable hardware requirements. We also briefly explore how the image synthesis quality varies as the synthetic view is captured at subsequently larger distances from the nearest known snapshot used to train the network.

## 1 Introduction

Novel view synthesis is a longstanding problem in computer vision that seeks to infer unseen perspectives of a visual scene from a set of existing viewpoints of that scene. Synthesis assumes that the underlying visual scene maintains constant geometric structure that is preserved under translation and rotation. The goal of synthesis is then to first learn a representation of the underlying visual scene, then generate views of the rotated object. Such learned representations are analogous to a person's "mental image" of an object, and the concept of view synthesis can be traced back to early psychophysical experiments requiring subjects to reason about unseen rigid transformations of a known object [1]

The specific problem here is sketched in figure (1). We seek to learn a dense representation from a sparse set of known views of a visual scene. By dense representation, we mean that the scene can be queried by placing a virtual camera at an arbitrary location in space, pointing it at an arbitrary direction, and expecting a photorealistic view of the original scene. By sparse known views, we mean that the only information we have concerning the visual scene is a finite set of photographs of some resolution, taken of the visual scene by a camera at known positions and directions. Once we have learned this dense representation, we can synthesize novel viewpoints by querying the learned representation from the viewpoint of a virtual camera placed at our desired synthesis point of view.

## 2 Related Work

View synthesis requires a geometric and color representation that preserves the intrinsic aspects of the visual scene. Prior work focusing on finding these representations can be roughly divided into two classes: 1) "classic" discrete techniques that sought to use input images to fill samples of a 3D grid [2] [3], and 2) "modern" discrete techniques that sought to leverage neural networks to learn the representation directly by minimizing a loss function [4] [5]. The former methods, while conceptually straightforward are inefficient due to the poor scaling properties of sampling a 3D volume with subsequently higher resolution. While the latter methods have in some cases circumvented the poor scaling by using e.g. convolutial networks (CNNs) to address low-resolution artifacts [6], both types of methods are fundamentally limited in the sense that they seek to discretely represent the visual scene at a fixed sampling rate. Conversely, NERF encodes a continuous volumetric representation, avoiding the disadvantages and limitations of discrete sampling altogether.

## 3 Proposed Method

Neural Radiance Fields (NERF) [7] generate image representations as shown in figure (2). Briefly, a virtual camera at a desired viewpoint is first specified. Next, rays are cast from that camera into the visual scene. Multiple samples along these rays form position and direction query points which are then passed through the neural network and integrated to form the pixel value of the output image. The output image is then compared to the target image via mean-square-error (MSE) in order to generate a loss function that trains the network. We describe this process in more detail in the following sections.

### 3.1 Scene Representation with a Neural Radiance Field

To represent the scene in NERF, we train a neural network to learn a mapping from a pair of position-direction vectors, written as $(x, d) \in \mathbf{R}^6$ to a 4-vector containing red, green, and blue color channels, and a sclar volumetric density $\sigma$, written as $(c, \sigma) \in \mathbf{R}^4$. This mapping is termed the Neural Radiance Field, since it defines a 4-vector field with a 6-dimensional coordinate system. Formally,

$$F_\Theta : \mathbf{R}^6 \to \mathbf{R}^4.$$

• *Chris Fritz is with the Department of Electrical Engineering, Stanford University Stanford, CA, 9403.*

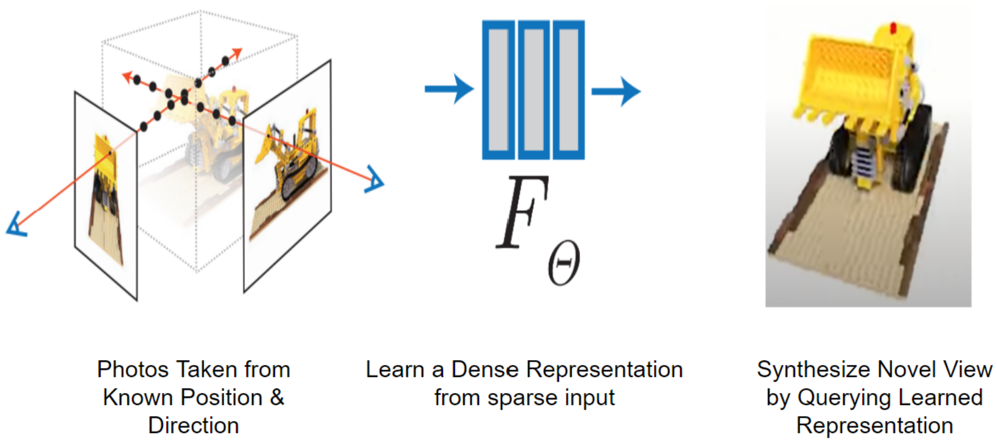

$F_\Theta$
Photos Taken from Known Position & Direction
Learn a Dense Representation from sparse input
Synthesize Novel View by Querying Learned Representation

Fig. 1. Sketch of the Visual Synthesis Problem. Left: A visual scene is sparsely sampled via snapshots (images) taken by a camera of known position and direction. Middle: The sparse input is used to learn an arbitrarily dense representation of the visual scene. Right: The learned representation can then be queried to synthesize a novel view of the scene.

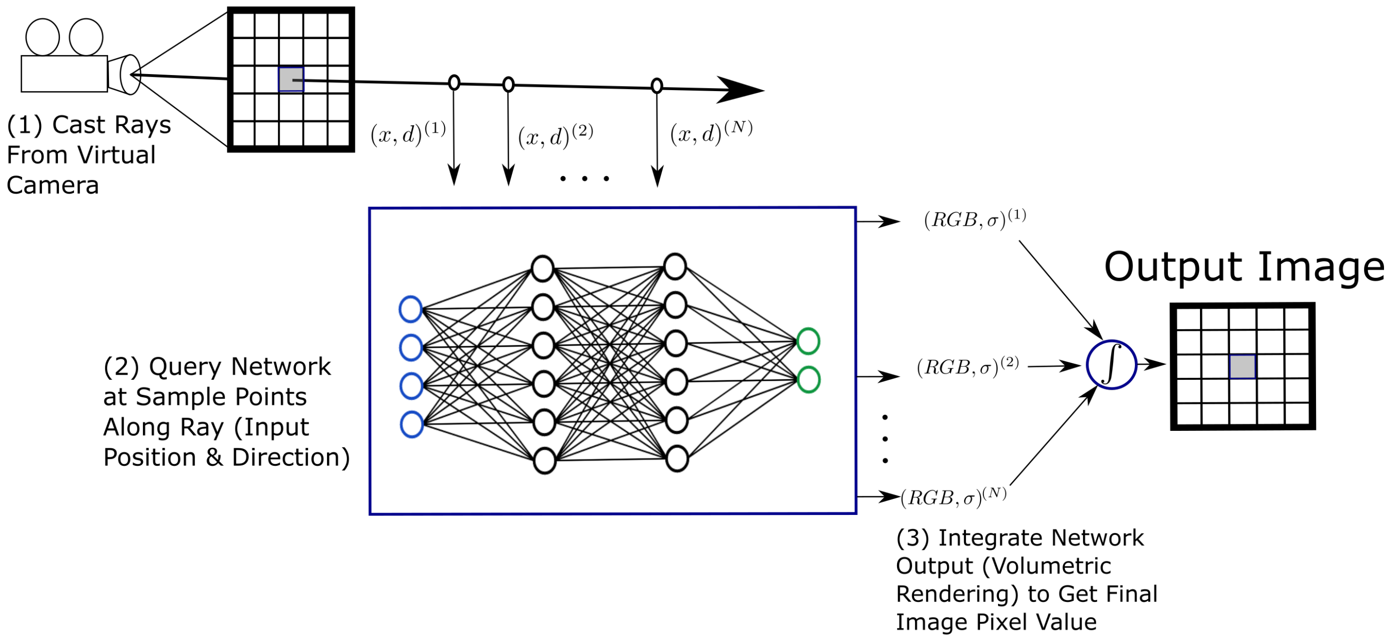

(1) Cast Rays From Virtual Camera
$(x,d)^{(1)}$
$(x,d)^{(2)}$
$(x,d)^{(N)}$
(2) Query Network at Sample Points Along Ray (Input Position & Direction)
$(RGB,\sigma)^{(1)}$
$(RGB,\sigma)^{(2)}$
$(RGB,\sigma)^{(N)}$
Output Image
(3) Integrate Network Output (Volumetric Rendering) to Get Final Image Pixel Value

Fig. 2. Overview of the NERF Rendering Pipeline. (1) To render an image, a virtual camera at the desired viewpoint is first specified, giving an image plane where the output image will be populated. Rays are case from the camera position through the center of each pixel. (2) Each cast ray is sampled repeatedly so that the resulting position and directions along the ray are passed to the network. The network accepts the queries as input, and returns a 4-vector containing color and volumetric density as output. (3) The resulting color and density are integrated along the ray to give afinal output color for a given pixel. The mean-square error (MSE) between the true image view and the output image form the loss used to train the neural network.(The volumetric rendering integration is differentiable).

Since this is the conceptual basis of NERF, we could represent the scene as written and proceed to train a neural network. However, we would find that the representation struggles to capture high-resolution details, leading to oversmoothed geometry and more generally loss of fine detail. This loss of high-frequency content appears to be caused by an intrinsic low-frequency bias inherent to deep neural networks [8]. To address this bias, NERF embeds its coordinate input to a higher-dimensional space using functions with high frequency content. Specifically, NERF specifies an embedding dimension $L$ and defines the mapping $\gamma : \mathbf{R} \to \mathbf{R}^{2L}$ where

$$\gamma(x) = \begin{bmatrix} \sin(2^0 \pi x) \\ \cos(2^0 \pi x) \\ \vdots \\ \sin(2^{L-1} \pi x) \\ \cos(2^{L-1} \pi x), \end{bmatrix}$$

is applied to each dimension of the network input. This high-dimensional mapping is referred to as *positional encoding*, and substantially improves scene reconstruction performance. All that remains to specifying the scene representation is to normalize the position direction values to the range $[-1, 1]$.

The neural network architecture underlying $F_\Theta$ is shown in figure (3). The network is a multilayer perceptron (fully connected feed-forward neural network) featuring 8 hidden layers. Each layer contains $256$ channels and all but one use a ReLU activation functions. To encourage the network to learn separate representations of position and direction (rather than learn both simultaneously), the position and direction input are first separated. The (positional-encoded) position is fed into top layer of the network, and runs through the network until the final hidden layer. Note that a skip connection re-injects the input at the 4th hidden layer. At the final hidden layer, the volumetric density $\sigma$ is peeled off as output and the positional-encoded direction vector is injected (concatenated) with the remainin goutput before being fed to an output layer with 128 channels. A final sigmoid is applied to the final layer's output to get the RGB channel values.

### 3.2 Volumetric Rendering of Network Output

Given the the network output, an image is rendered is rendered as previously described in figure (2). A ray

$$r(t) = o + td$$

is cast through the center of each camera pixel we wish to render. Computing the color of the ray is performed by estimating the path integral

$$C(r) = \int_{t_n}^{t_f} T(t) \cdot \sigma(r(t)) \cdot c(r(t)) dt$$

where $t_n$ and $t_f$ specify the near and far clipping planes of the view frustum respectively, $c(r(t))$ and $\sigma(r(t))$ are the network outputs, and

$$T(t) = \exp(-\int_{t_n}^{t} \sigma(r(s)) ds).$$

In our implementation, this is performed in three steps: 1) To ensure the network is queried along a continuous space (and not a fixed set of discrete locations), the ray along the view frustum ($t_n \leq t \leq t_f$) is divided into $N$ bins and a sample is drawn uniformly from each bin, a strategy termed *stratified sampling*. Each sample is given by

$$t_i \sim \text{Uniform}(t_n + \frac{i-1}{N}(t_f - t_n),\ \ t_n + \frac{i}{N}(t_f - t_n)).$$

The resulting positions (sample locations), and directions (ray direction) form a set of query points passed to the network model. 2) The network is evaluated at the given query points to get a volumetric density and RGB color. 3) The integral above is approximated by a numerical quadrature rule discussed in [9]. The integrals above become the sums

$$\hat{C}(r) = \sum_i T_i(1 - \exp(-\sigma_i \delta_i)) c_i,$$

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

where $\delta_i = t_{i+1} - t_i$. The preceding method can is used to render a pixel and consequently an entire image by querying the NERF. Because the rendering process is differentiable, the resulting image can be compared with a training image and used as a loss within a back-propagation algorithm to train the network.

To avoid overloading GPU memory, we split the images into sub-blocks, rendered each sub-block as described above, and updated network parameters for that sub-block. In addition to reducing the video memory footprint at of the program, this approach also served to regularize the network, analogously to batched gradient descent.

Performance of the network can be extended further by a *Hierarchical Sampling Scheme*. In this scheme, two NERF networks are simultaneously trained. One network - termed coarse - is used to sample the scene at a lower resolution as previously described. The coarse sampling produce weights that are normalized to form a probability distribution. The weights are given by

$$w_i = T_i(1 - \exp(-\sigma_i \delta_i)).$$

After normalization of the weights, the resulting PDF is sampled to produce a fine-resolution set of query points that are importance-sampled, i.e their distribution is fit to sample the visual scene preferentially at areas of higher density. These fine-query points are concatenated to the coarse query points before being passed through a fine-grained network to get an output image. Both networks are trained in tandem.

## 4 Experimental Results

We implement the architecture described in section 3 using PyTorch, a deep learning framework for the Python programming language. [10]. The implementation formed the bulk of this work. To compare NERF with other synthesis methods, we refer the reader to table 1 and figures 5 and 6 of the original NERF paper [7]. To quantify the impact of each model feature (positional encoding, hierarchical sampling, view-direction dependence), ablation studies for NERF are recapitulated in figure (5). Metrics are over 8 synthetic scenes. Due due to the extensive hardware overhead of an additional network, and the relatively small performance improvement (approximately $3\%$), we implemented hierarchical sampling, but elected not to use it in order to obtain a feasible training iteration speed on a personal GPU (Nvidia RTX 3080 Ti), whereas the original NERF was trained extensively on an industrial GPU (Nvidia V100).

A sample of our network output is shown in figure (6). In this case, we trained the network on the *hotdog* synthetic dataset, which consists of 100 images and poses of a hotdog, where each image is $800 \times 800$ pixels. The synthetic output is best understood by viewing the supplementary video material.

In addition to reproducing NERF results above, we also explore the performance of NERF inference as a function of distance from the nearest training point. Since each NERF is trained on a sparse set of images, we expect NERF performance to be highest when the network is evaluated at those training points. When inferring novel views, however the performance of the network is less clear. We evaluate the likelihood of two somewhat conflicting hypotheses: 1) Inference performance (PSNR) degrades as the euclidean

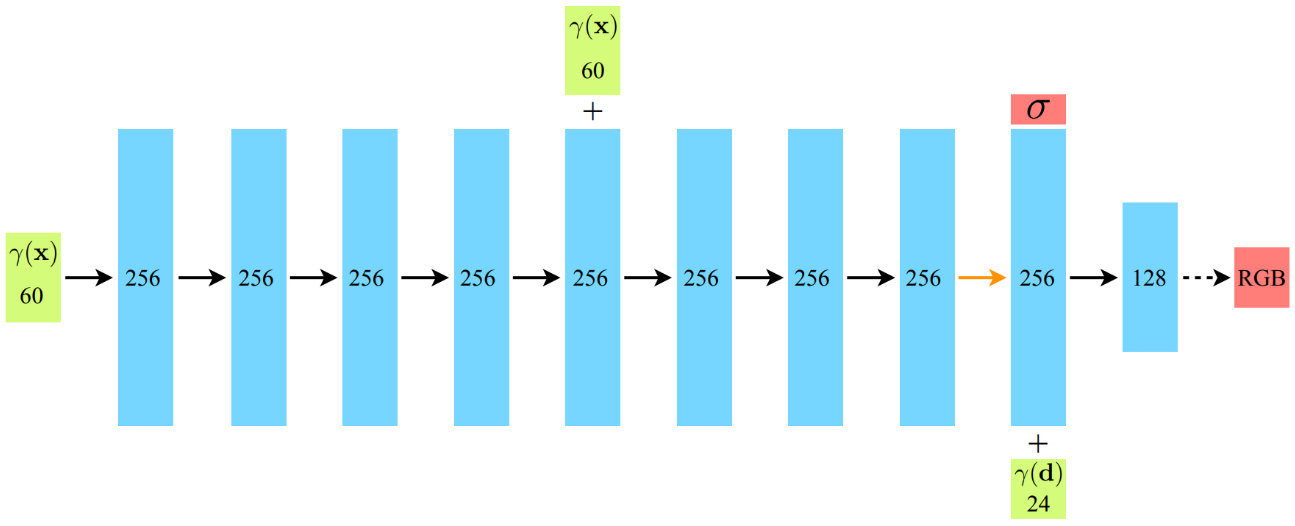


Fig. 3. Nerf Network Hierarchy. The network consists of 1 input and 8 hidden layers each having 256 channels per layer and fully connected. The last output layer consists of 128 channels. Position and directional are split then embedded into a higher-dimensional space via positional encoding. Position input are fed into the top of the network with a reinjection (skip connection) at the fourth hidden layer. Density output $\sigma$ is extracted from the last hidden layer before direction input and the result fed into the output layer to produce RGB color values. Black arrows denote ReLU Activation functions, the orange arrow denotes no activation function (identity), and the dotted arrow denotes the sigmoidal activation function.

distance to the nearest training image/pose increases 2) Inference performance initially degrades with euclidean distance, but quickly reaches a floor corresponding to an accurate dense representation of the visual scene. The former hypothesis is intuitive: the further we query the network from a known (ground-truth) location, the worse we expect the network perform. The latter hypothesis reflects the design choices meant to make the representation dense, i.e continuous between image sample points.

To evaluate these hypotheses we study the performance of a hold-out test set of images/poses. In particular, we 1) Generate the network output at the test pose location, 2) compute PSNR of the output image versus the ground truth test image. 3) Find the nearest training image/pose sample by Euclidean distance of the respective camera locations. We plot the resulting PSNR vs distance for the test-set in figure (4). Note that the correlation is negative, but weak ($\rho = -0.048$), suggesting a tenuous negative relationship between test-image prediction quality and Euclidean distance to the nearest training point. This observation is consistent with hypothesis 2. A limitation of this approach is that it does not factor direction when comparing the distance between two images/poses - e.g two images/poses might be next to each other but rotated 180 degree about the y-axis with respect to one another. To explore this possibility, we performed the same experiement, but with *angular* distance. Instead of selecting images by Euclidean distance, we computed their rotations about the xyz-axes, and formed a 3-vector which then we computed the Euclidean distance of. The results (not shown) were similar, giving ($\rho = -0.65$).

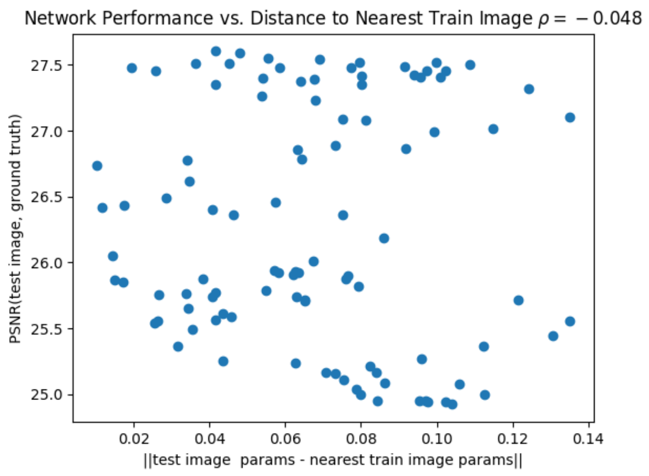
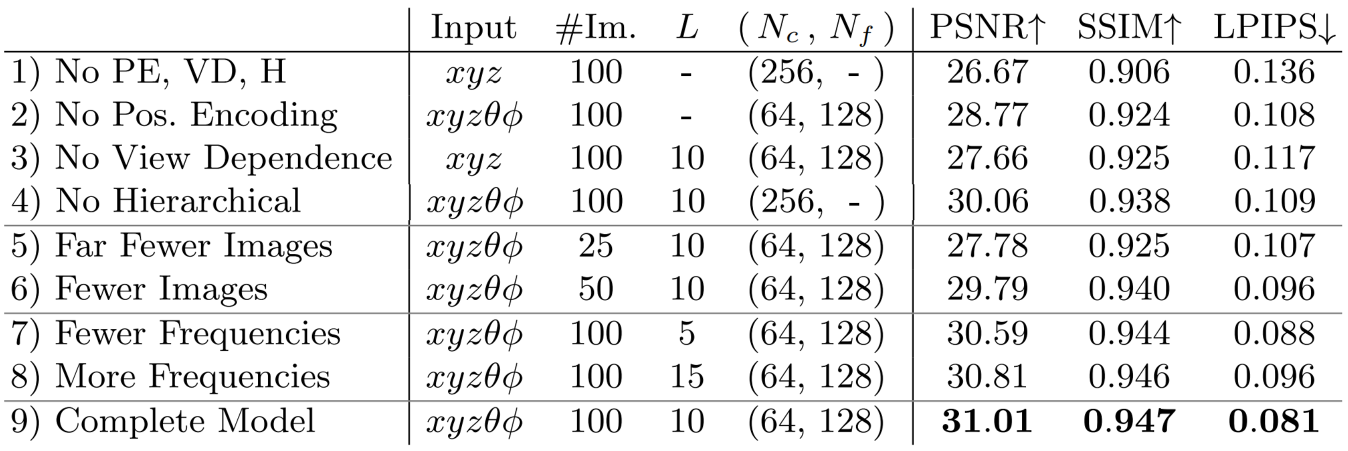


Fig. 4.

## 5 Conclusions & Further Work

At the time of release, Neural Radiance Fields provided state of the art performance for the view synthesis problem, representing a substantial advancement. NERF networks may one day be embedded within a graphics pipeline as a tool for rendering real-world object within a virtual environment for use in applications such as mixed reality. Although relatively recent, they have spawned numerous research directions, with many focused on addressing limitations of NERF. Here we briefly discuss two such limitations: rendering cost, and transient features.

Rendering Cost: In NERF, a single ray is sampled hundreds of times. This leads to prohibitively high computational costs to render an image on the order of Teraflops. On modern consumer graphics hardware (Nvidia RTX 2080 Ti), rendering a single $800 \times 800$ frames takes roughly 10 seconds. Performance extensions of NERF address e.g. via ”baking” storage of a NERF representation in more efficient data structures for supporting real-time rendering



| | Input | #Im. | $L$ | $(N_c, N_f)$ | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|---|---|---|---|
| 1) No PE, VD, H | $xyz$ | 100 | - | (256, - ) | 26.67 | 0.906 | 0.136 |
| 2) No Pos. Encoding | $xyz\theta\phi$ | 100 | - | (64, 128) | 28.77 | 0.924 | 0.108 |
| 3) No View Dependence | $xyz$ | 100 | 10 | (64, 128) | 27.66 | 0.925 | 0.117 |
| 4) No Hierarchical | $xyz\theta\phi$ | 100 | 10 | (256, - ) | 30.06 | 0.938 | 0.109 |
| 5) Far Fewer Images | $xyz\theta\phi$ | 25 | 10 | (64, 128) | 27.78 | 0.925 | 0.107 |
| 6) Fewer Images | $xyz\theta\phi$ | 50 | 10 | (64, 128) | 29.79 | 0.940 | 0.096 |
| 7) Fewer Frequencies | $xyz\theta\phi$ | 100 | 5 | (64, 128) | 30.59 | 0.944 | 0.088 |
| 8) More Frequencies | $xyz\theta\phi$ | 100 | 15 | (64, 128) | 30.81 | 0.946 | 0.096 |
| 9) Complete Model | $xyz\theta\phi$ | 100 | 10 | (64, 128) | **31.01** | **0.947** | **0.081** |

Fig. 5. Ablation Study for NERF Model Features. Reproduced from [7]. PE = Positional Encoding, whether or not to pass input to high-dimensional embedding before passing to network; VD = View-Dependence, whether or not to include directional input to the network at all; H/Hierarchical = Hierarchical Sampling, whether or not to train two networks, using one to draw a sample distribution for the other. PSNR = Peak-signal-to-noise ratio, SSIM = structural similarity index, LPIPS = Learned perceptual image patch similarity, $N_c$, $N_f$ = number of coarse and fine samples (fine only if using H sampling)., L = positional encoding dimension.

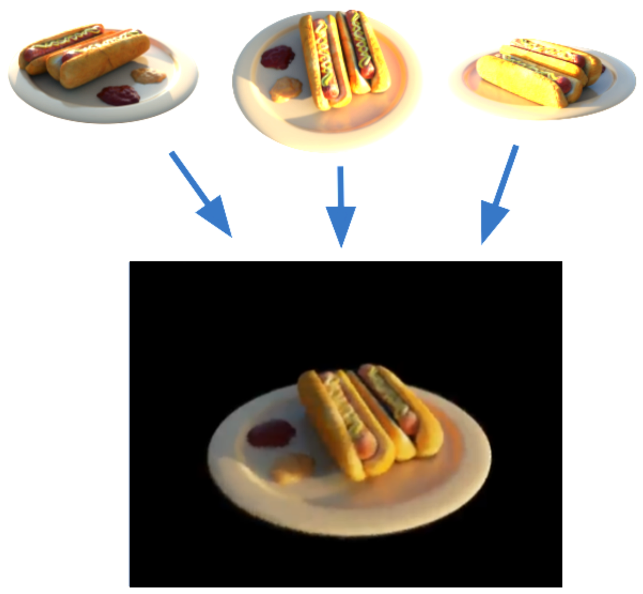


Fig. 6. Top: Input images and poses used to train NERF network. Bottom: Synthesized novel view. The network was trained on the *hotdog* synthetic dataset available at the original author's website [https://www.matthewtancik.com/nerf]. Readers are encouraged to view supplementary video.

on conventional hardware [11].

Transient Features: An implicit assumption within NERF is that all the image samples are taken under the same lighting and occlusion conditions. Many real-world samples violate these controlled conditions. For example, tourist photos of the *Fontana di Trevi* occur under a wide variety of lighting (day, night, overcast), and occlusion (person / object in frame) conditions. To address these transient features, NERF in the Wild (NERF-W) extends NERF to simultaneously learn a transient embedding that is superimposed on top of the constant features of the visual scene [12].

## References

[1] J. S. McGee, C. van der Zaag, J. G. Buckwalter, M. Thiébaux, A. Van Rooyen, U. Neumann, D. Sisemore, and A. A. Rizzo, "Issues for the assessment of visuospatial skills in older adults using virtual environment technology," *CyberPsychology & Behavior*, vol. 3, no. 3, pp. 469–482, 2000.

[2] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 307–314.

[3] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, vol. 35, no. 2, pp. 151–173, 1999.

[4] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[5] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. S. Overbeck, N. Snavely, and R. Tucker, "Deepview: High-quality view synthesis by learned gradient descent," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. [Online]. Available: https://arxiv.org/abs/1906.07316

[6] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "Deepvoxels: Learning persistent 3d feature embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2437–2446.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*. Springer, 2020, pp. 405–421.

[8] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5301–5310.

[9] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[11] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view

synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884.

[12] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.