NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

DAVID SCHNEIDER, Technical University of Munich

Neural Radiance Fields (NeRF) have revolutionized novel view synthesis by representing a 3D scene as a continuous 5D function that maps a 3D location and a 2D viewing direction to an emitted color and volume density. This implicit scene representation, modeled by a multilayer perceptron (MLP), is optimized by minimizing the rendering loss between predicted and ground-truth 2D images. By leveraging classical volume rendering principles, NeRF can synthesize photorealistic novel views of complex scenes with intricate geometry and view-dependent appearance. This report details the fundamental concepts, network architecture, and training methods of NeRF. It further examines its impressive quantitative and qualitative results and discusses some interesting follow-up works and extensions.

1 INTRODUCTION

Synthesizing photorealistic images of a 3D scene from novel viewpoints is a well established challenge in computer graphics and vision. Prior approaches often relyed on explicit 3D representations like meshes or point clouds, which can struggle to capture complex geometries and view-dependent lighting effects correctly.

Neural Radiance Fields (NeRF), introduced by Mildenhall et al. [Mildenhall et al. 2020], is regarded as a great breakthrough in view synthesis. NeRF proposes a new approach to implicitly represent a 3D scene as a continuous function, preventing the need for explicit geometry. This function is modeled by a fully connected neural network (Multi-Layer Perceptron), which learns to map 5D coordinates (3D position and 2D viewing direction) to color and volume density (i.e., how much light accumulates as it passes through that position).

The main contributions of the NeRF work can be summarized as follows:

- A novel implicit scene representation that leverages a neural network to learn a continuous volumetric function of color and density.
- A differentiable rendering pipeline that allows for optimization of the neural network parameters directly from a set of input 2D images and their corresponding camera poses.
- The ability to synthesize photorealistic novel views with intricate details and view-dependent effects (e.g., reflections, translucency).
- A compact scene representation stored entirely within the weights of a small neural network.

2 METHOD

NeRF implicitly represents a static 3D scene as a continuous 5D function $F_\Theta : (\mathbf{x}, \mathbf{d}) \to (\mathbf{c}, \sigma)$, where $\mathbf{x} = (x, y, z)$ is a 3D location, $\mathbf{d} = (\theta, \phi)$ is a 2D viewing direction (in practice represented as a 3D cartesian unit vector), $\mathbf{c} = (R, G, B)$ is the emitted color, and σ is the volume density. This function is parameterized by the weights Θ of a Multi-Layer Perceptron (MLP).

This report is a part of the Seminar - Deep Learning in Computer Graphics, Technical University of Munich.

The original work is introduced by [Mildenhall et al. 2020].

2.1 Ray Generation

To render an image from a given camera viewpoint, the first step is to generate rays that emanate from the camera's optical center and pass through each pixel of the desired image plane. Knowledge of the camera's parameters is required for the method to work.

Inputs:

- Image Height (H) and Width (W) in pixels.
- **Camera Intrinsic Matrix (K):** A 3×3 matrix defining the camera's internal optical properties (e.g., focal length, principal point). The Camera Intrinsic Matrix looks like this:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

  where f_x, f_y are the focal lengths in x and y pixel units, and c_x, c_y are the coordinates of the optical center.
- **Camera-to-World Extrinsic Matrix (c2w):** A 4×4 transformation matrix specifying the camera's 3D position and orientation in the global world coordinate system. This matrix is derived from camera pose angles (e.g., Roll, Pitch, Yaw) and translation.

$$\mathbf{c2w} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

  where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector representing the camera's position in world coordinates. For further information on the camera matrix and its role in projection, refer to these lecture slides [Kitani 2017].

Process: For each pixel (i, j) in the image grid, a ray is defined by its origin $\mathbf{v_o}$ and direction $\mathbf{v_d}$. The ray origin $\mathbf{v_o}$ is consistently the camera's 3D position (extracted from **c2w**). The ray direction $\mathbf{v_d}$ is computed by first transforming the pixel coordinates (i, j) into directions in the camera's local frame (using **K**) and then rotating these directions into the world coordinate system (using the rotation component of **c2w**). The resulting $\mathbf{v_d}$ is a normalized unit vector.

A point $\mathbf{P}$ along a ray can then be parameterized as $\mathbf{P} = \mathbf{v_o} + t \cdot \mathbf{v_d}$, where t is the distance from the ray origin.

2.2 Point Sampling along Rays

After generating rays, the next step is to select a discrete set of 3D query points along each ray's path. These points will be used as input for the NeRF MLP.

Input: A bundle of rays $\{(\mathbf{v_o}, \mathbf{v_d})\}_{H \times W \times n}$, where n is the number of camera poses.

Challenge of Sampling: A ray passes through an infinite number of points. Uniform sampling can be inefficient (sampling many empty regions).

Stratified Sampling: NeRF uses a stratified sampling approach to sample points along each ray more effectively.

- The ray in between near (t_n) and far (t_f) bounds is partitioned into N evenly spaced bins.
- One sample is drawn uniformly from within each bin.

This strategy ensures broad coverage along the ray while also infusing randomness to capture high-frequency details. For each sampled point, its 3D Cartesian coordinates (x, y, z) and the ray's 2D viewing direction, are obtained.

2.3 Neural Radiance Fields Network Architecture

The core of NeRF is a fully connected MLP that maps the 5D input (3D position and 2D viewing direction) to color and volume density.

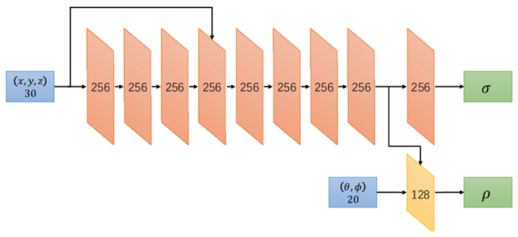


Fig. 1. NeRF Network Architecture. The 3D position (x, y, z) is fed into an 8-layer MLP, outputting density (σ) and a feature vector. This feature vector is concatenated with the 2D viewing direction (θ, ϕ) and fed into a 1-layer MLP to output the RGB color (ρ). (Adapted from [Shen et al. 2021])

Architecture Details:

- **Positional Input Branch:**
  - The 3D position (x, y, z) is fed into an 8-layer MLP.
  - Each layer utilizes ReLU activations.
  - The MLP processes the input with 256 channels.
  - The outputs are a predicted volume density (σ) and a feature vector.
- **View-Dependent Color Branch:**
  - The 2D viewing direction (θ, ϕ) is concatenated with the feature vector (output from the positional branch).
  - This combined vector is then fed into a 1-layer MLP.
  - This MLP also uses ReLU activations, processing with 128 channels.
  - The final output of this part is the view-dependent RGB color.

This two-branch design reflects an important assumption in NeRF: while volume density σ depends only on the spatial location (i.e., it is view-independent), the emitted color c can vary with the viewing direction d due to effects like specular reflection. Separating the network into position and view-dependent parts allows NeRF to model this behavior better. The feature vector acts as an internal representation that carries information from the position branch to inform the view-dependent color branch.

2.4 Positional Encoding

Before being fed into the MLP, the 3D coordinates (x, y, z) and 2D viewing directions (θ, ϕ) are preprocessed in a step called positional encoding. Positional Encoding maps the input coordinates from $\mathbb{R}$ to a higher-dimensional space $\mathbb{R}^{2L}$. The mapping for an input coordinate p:

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \ldots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p))$$

This encoding helps the MLP to represent high-frequency variations in the scene, like fine textures and sharp edges, which a normal MLP might struggle with.

2.5 Volume Rendering

Once the NeRF MLP predicts color and volume density for a set of sampled points along a ray, these values are composited into a final 2D pixel color using classical volume rendering techniques.

Volume rendering visualises 3D data by accumulating contributions from all points along a ray as it traverses a volume. Unlike traditional surface rendering, which focuses on explicit geometry, volume rendering is best at representing content with varying opacities and small transitions, such as smoke, clouds, or, in NeRF's case, the continuous density and color fields that implicitly define a scene. This approach allows for the visualization of semi-transparent materials, which is crucial for NeRF's to render photorealistic scenes. The core idea is to integrate the emitted light and the reduction of light intensity along the ray's path.

The continuous integral equation for the color $C(\mathbf{r})$ of a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ from near (t_n) to far (t_f) bounds is:

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

Where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$ is the transmittance, representing the probability that the ray travels from t_n to t without being occluded.

Numerical Quadrature: Since the MLP provides discrete samples, the continuous integral is approximated using numerical quadrature. For N sampled points:

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

Here, σ_i and $\mathbf{c}_i$ are the predicted density and color for sample i, and δ_i is the distance between adjacent samples. T_i is the discrete transmittance, representing the probability that the ray reaches sample i without being occluded by prior samples. This process effectively accumulates color and opacity along the ray to determine the final pixel color.

Importantly, the entire volume rendering pipeline used by NeRF is differentiable. This differentiability allows gradient-based optimization of the network parameters using backpropagation. It allows NeRF to learn a scene's geometry and appearance directly from 2D image supervision by minimizing pixel-wise rendering errors. It also makes it possible to train the model end-to-end without requiring ground-truth 3D geometry or intermediate supervision.

3 TRAINING AND IMPLEMENTATION

3.1 Optimization Details

NeRF is trained to minimize the difference between its rendered pixel colors and the ground-truth pixel colors from the input images.

Loss Function:

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

where $\hat{C}(\mathbf{r})$ is the predicted color, $C(\mathbf{r})$ is the ground-truth color for a ray $\mathbf{r}$, and $\mathcal{R}$ is a batch of rays sampled during training.

Optimizer: The Adam optimizer is employed to update the network weights. It dynamically adjusts learning rates for each parameter based on previous gradients.

Learning Rate Scheduling: To improve convergence and stability, the global learning rate (e.g., ‘args.lrate‘) is decayed exponentially over training iterations. This allows for larger steps initially for faster exploration and smaller steps later for fine-tuning. For example:

$$\text{new\_lrate} = \text{initial\_lrate} \cdot (\text{decay\_rate})^{(\text{global\_step}/\text{decay\_steps})}$$

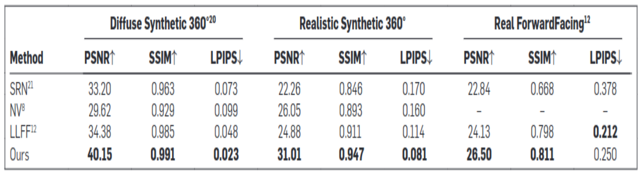
This scheduler is applied external to the Adam algorithm's internal parameter-wise adaptations. For further information on the implementation, please refer to [Yen-Chen 2020].

3.2 Efficiency and Storage

Despite its photorealistic output, a NeRF model for one scene is relatively compact (5MB). This is significantly smaller than explicit volumetric representations (e.g., LLFF voxel grids, 15GB per scene). However, training a NeRF model for a single scene can be computationally intensive, often requiring 1-2 days on a powerful GPU, and a new model needs to be trained from scratch for each new scene.

4 QUANTITATIVE RESULTS

The original NeRF paper demonstrated superior quantitative performance compared to prior state-of-the-art methods in novel view synthesis. Metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) are used to evaluate the fidelity of the rendered images against ground truth. Higher PSNR and SSIM indicate better quality, while lower LPIPS indicates higher perceptual similarity.



	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

Fig. 2. Quantitative results comparing NeRF with previous methods (SRN, NV, LLFF) across various synthetic and real-world datasets. NeRF consistently achieves significantly higher PSNR and SSIM, and lower LPIPS, demonstrating its superior rendering quality for novel views. (Adapted from [Mildenhall et al. 2020])

5 QUALITATIVE RESULTS

NeRF demonstrates a remarkable ability to capture intricate details, fine-grained textures, and complex view-dependent effects that were challenging for previous methods (see Figure 3).

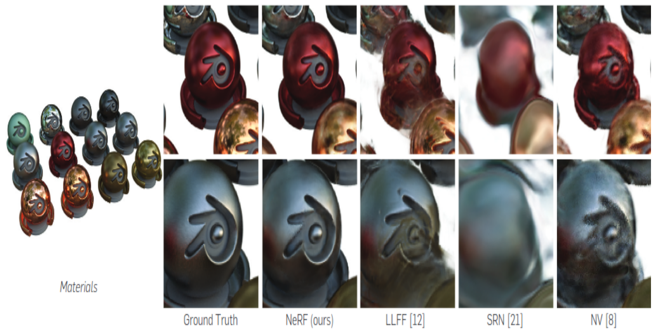


Fig. 3. NeRF produces sharper details and more accurate appearance compared to alternative methods (LLFF, SRN, NV). (Adapted from [Mildenhall et al. 2020])

6 FOLLOW-UP WORKS AND EXTENSIONS

The success of the original NeRF paper led to many follow-up research aimed at addressing its limitations and expanding its capabilities. These extensions have focused on improving speed, robustness, handling dynamic scenes, and enabling more complex rendering applications.

6.1 Faster More Robust NeRF Variants

Many works have focused on accelerating NeRF training and rendering, and improving its robustness to challenging real-world data.

- **Instant-NGP** ([Müller et al. 2022]): Introduces multi-resolution hash encoding and a tiny MLP, enabling NeRF-like quality with training times reduced from days to minutes, and real-time rendering.

6.2 Relighting Inverse Rendering

Extensions in this area aim to decompose the scene into intrinsic properties (e.g., albedo, normal, roughness) and lighting, allowing for novel illumination conditions.

- **NeRF-W** ([Martin-Brualla et al. 2021]): Extends NeRF to handle "in-the-wild" photo collections, which often contain varying illumination, transient objects, and camera parameters. It learns latent codes for appearance and illumination to account for these variations.

6.3 Complex Scene Dynamics

Original NeRF models static scenes. This line of work focuses on representing and rendering dynamic environments.

- **D-NeRF** ([Pumarola et al. 2020]): Extends NeRF to model dynamic 3D scenes. It introduces a deformation network that maps a point from a static space to its deformed position in a specific time step, allowing for reconstruction of moving objects.

7 CONCLUSIONS

NeRF introduced a powerful method for novel view synthesis by representing 3D scenes as implicit neural radiance fields. Its ability to capture complex geometry and view-dependent appearance with a compact neural network has made it a foundational work in implicit neural representations. While computational cost during training and the need for accurate camera poses remain challenges, NeRF has opened the door for future research in 3D reconstruction, rendering, and scene understanding.

8 ACKNOWLEDGEMENTS

AI tools (e.g., ChatGPT) were used during the preparation of this report to assist with language refinement and clarification of technical concepts related to NeRF.

REFERENCES

Kris Kitani. 2017. Camera Matrix and Projection. https://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf. Lecture slides, Carnegie Mellon University.

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv:2003.08934 [cs.CV]

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–15. https://doi.org/10.1145/3528223.3530127

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

Siyuan Shen, Zi Wang, Ping Liu, Zhengqing Pan, Ruiqian Li, Tian Gao, Shiying Li, and Jingyi Yu. 2021. Non-line-of-Sight Imaging via Neural Transient Fields. https://doi.org/10.48550/arXiv.2101.00373

Lin Yen-Chen. 2020. NeRF-pytorch. https://github.com/yenchenlin/nerf-pytorch/.