Research Analysis: NeRF

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Thomas Cedeno

Spring, 2021

Nada Basit, Department of Computer Science

Vicente Ordonez-Roman, Department of Computer Science

Research Analysis: NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis

Thomas Cedeno
Department Name
University of Virginia
Charlottesville, VA United States
tjc5tp@virginia.edu

ABSTRACT

I will summarize, analyze and re-implement the paper *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,* which can be found at this URL: https://arxiv.org/pdf/2003.08934.pdf. As the paper involves creating a representation of physical or virtual "scenes", I define any scene as a physical object or collection of objects captured in the context of a static environment. Scenes can be complex with various objects and environments, but can also consist of just a single object with no environment. The NeRF paper goes over a method of constructing scene representation through a continuous volumetric scene representation, using camera images as input to train and optimize this underlying function. It uses this to construct novel views of the scene, generating new images from different angles not displayed during training.

With increasingly accurate scene representations, techniques such as NeRF can enable sophisticated augmented reality applications, or improve the accuracy of depth-estimation through computer vision alone for fields such as robotic perception. NeRF incorporates material from the Machine Learning and Computer Vision courses at UVA. The technique is the current state of the art for novel view synthesis, with stark contrast from traditional computer vision algorithms. Understanding where NeRF improves on traditional approaches, as well as its current limitations will allow us to survey the future of computer vision.

INTRODUCTION

The long standing problem of view synthesis is a variation on the generalized problem of geometry reconstruction from computer vision. Novel view synthesis, as a whole, benefits from more accurate 3D reconstruction. The problem the NeRF paper attempts to solve is, given a set of input images, creating and optimizing a representation of the scene. After the model internalizes the geometry and lighting of a particular scene, querying the network with a desired viewpoint allows for the rendering of new views. Computer vision researchers have been exploring a new direction by using a multi-layer perceptron to directly map the features of a 3D scene to weights of a model. However, before the NeRF paper, this branch of techniques failed to perform better than the more classical geometry-based representations such as voxel grids or triangle meshes.

Early volumetric approaches used to directly color voxel grids, or sampled regions of the entire 3D space. This voxel grid approach is naturally differentiable, which allows models that render with voxel grids to optimize using gradient descent directly off a loss function from the generated view. But by using voxel grids and explicitly discretizing the space of the scene, these methods incurred excessive storage costs. Other methods have investigated optimizing deep networks to map (x,y,z) coordinates to signed distance functions, but have failed to empirically represent complex geometry. [1] Previous work in this field has been limited to lower-resolution shapes, and would often result in over smoothed renderings. NeRF has notable improvements over existing methods in both storage space and ability to represent complex geometry.

Previous techniques such as *Neural Volumes* (NV), *Scene Representation Networks* (SRN), and *Local Light Field Fusion* (LLFF) all have unique implementations for novel view synthesis. NV's synthesize novel views by optimizing a deep convolutional neural net to predict discrete RGBα samples within a voxel grid. The NV algorithm then renders new views by marching camera rays through these samples. SRN's represent scenes as a series of opaque surfaces, and optimizes a Multilayer Perceptron to map each (x,y,z) coordinate to a feature vector. SRN marches a ray through the scene, using the feature vector to direct where the ray will travel through and composite a final color for any given surface. LLFF's algorithm is designed to required a dense sampling, training a 3D convolutional neural net to predict RGBα samples, but unlike NV it renders new views through alpha compositing and blending nearby views to produce a novel viewpoint.

All the existing implementations to solve novel view synthesis have their own unique tradeoffs, with previous approaches all struggling with one aspect or multiple. For example, by using a voxel grid based approach, NV's and

LLFF's prohibitive storage costs come with the direct tradeoff of significantly faster training times when compared to NeRF. [1] Beyond these unique tradeoffs, all previous works struggle to represent high-resolution textures, transparency, complex occlusion effects, or view dependent artifacts such as radiance on a surface.

BACKGROUND

One of the central technologies that enabled NeRF is the Multilayer Perceptron (MLP). By using a MLP, the researchers leveraged machine learning to have the machine "discover" the correct function to represent the scene. As input, the model takes in a series 3D coordinates that together sample the scene throughout, in the format of a fixed-width vector. By taking in these coordinates and producing an RGB and density value along a ray, the model can composite the ray to generate a final pixel value for an image prediction. It then compares images it generates to those that already exist, or "ground truth". It then uses the difference between its prediction and the ground truth, on a per pixel basis to compute a loss value, or a rough estimate on how well this iteration of the model performed. Through back-propagation and stochastic gradient descent, the model can directly use this loss function to adjust its parameters and create a new iteration of the model that results in a lower loss. In the case of the NeRF paper, a lower loss means producing images closer to "ground truth", and finding an optimal representation of the scene through its weights. A simplified diagram of the model is provided in the figure below.

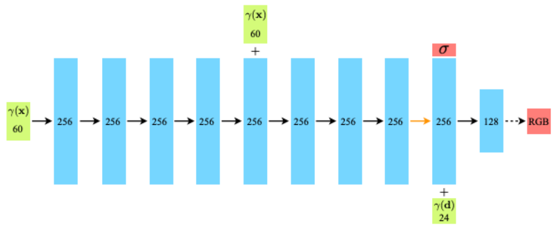


Figure 1: Flow diagram for the NeRF MLP (Source: *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis)*

The process of compositing the ray to render a final image is accomplished through volume rendering. The model itself only produces a density for every specific 3D coordinate and a RGB color for every specific 2D viewing direction. So for the model to actually render an image, it needs to sample points along these rays in the scene and compile these points to produce a final color value for every pixel in a final image. To integrate the expected color $C(r)$ of a camera ray $r(t) = o + td$, with near and far bounds t_n and t_f,

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right).$$

Figure 2: Expected color $C(r)$ of camera ray r

The function $T(t)$ can be interpreted as the probability that the ray travels from t_n to t uninterrupted and unscattered. The integral accomplishes ray compositing by integrating colors of points along the ray, proportional to density of the ray $\sigma(r(t))$ and $T(t)$. As the network only provides discrete points for density and color, the researchers needed to numerically estimate the integral, accomplished through quadrature. The function to estimate $C(r)$ from a set of (c_i, σ_i) values is also differentiable. Once a color is produced, a pixel for the predicted image is generated and the rendering process is finished.

RELATED WORK

In the field of 3D shape representation, early attempts to create implicit representations used deep networks that map coordinates to a signed distance function. Early models such as *DeepSDF*, however, were limited by their requirement to ground truth 3D geometry, which limited these functions to only approximating synthetic scenes [1]. Work with real scenes contained difficult to capture "ground truth" geometry, a fundamental barrier. This requirement was rectified, allowing later papers including NeRF to work by differentiable rendering functions that can effectively use 2D images to optimize their representation.

Some specific examples of previous work include *Niemeyer et al.* [2], which represented surfaces as 3D occupancy fields, and *Sitzman et al.* [3], which used a representation that outputs a feature vector and a RGB color at each continuous 3D coordinate. As explained previously, these previous approaches struggled to deal with complex geometry and view-dependent lighting effects.

With a voxel based rendering approach, another set of models attempted to get over the storage and compute costs for attending each discrete voxel. The papers optimized Convolution Neural Nets (CNN), and sampled the voxel grids for each scene. [1] The researchers had the CNN also account for discretization artifacts from low resolution voxel grids. While achieving impressive results, by using discrete sampling they are fundamentally limited in the resolution they can process in a given scene. NeRF gets around this by encoding a continuous volume for the parameters of their neural net.

1 System Design

Through the NeRF paper, researchers at UC Berkeley, Google Research and UC San Diego approach view synthesis in a new way. Their work comes as 3 major contributions. First, their overall architecture and design choice to use “5D” coordinates as input - composed of a 3D coordinate *(x,y,z)* and a 2D coordinate for the camera viewpoint (θ, Φ). Second, their positional encoding scheme, where 3D coordinates for their scene are “encoded” into higher dimensional frequencies. Third, their hierarchical volume sampling, which allows them to bias the sampling of the ray to encode the more information on the objects relevant surfaces with the same amount samples

1.1 Architecture The architecture for the NeRF paper revolves around the constructed their neural net, and how they generated their outputs to produce a useful loss function. Their neural net takes in this 5D coordinate (x, y, z, θ, Φ), and produces a color and density for every point. These colors and densities form rays that intersect the object. Volume rendering then composites these values into an actual image. After this image is produced, its rendering loss is computed on a per pixel basis. Establishing viewing direction as an input NeRF has an easier time generating lighting artifacts such as specular reflection. Encouraging the network to be accurate with its image predictions from various directions, optimizing the network through a 2D image loss forces the network to converge on the true shape representation.

1.2 Positional Encoding With previous attempts to encode a 3D space with a neural net, researchers would directly use the x,y and z values as inputs for their net. Neural nets in general have a long standing problem with representing high-frequency functions [1], and the field of view synthesis is no different. By mapping their 3D coordinate to a higher dimensional space, they can enable better fitting of data with high frequency variation. In practical terms this reduces the over smoothing effect seen with previous attempts. The exact encoding is shown in the figure below, and the researchers empirically found L=10 to be effective for the 3D coordinates and L=6 for the viewing direction.

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

Figure 3: The Fourier feature mapping

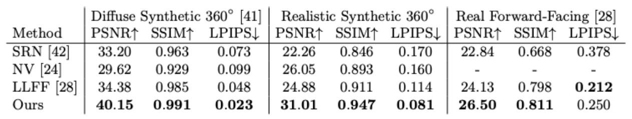
By encoding the coordinates before feeding them into the network, they are able to optimize for higher frequency functions, like those found in the domain of view synthesis. The authors of the NeRF paper later generalized their findings in a later paper *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains* [7], generalizing this encoding to a simple Fourier feature mapping.

1.3 Hierarchical Sampling Evaluating and compositing density and color along evenly spaced points for each camera ray is inefficient because the model is sampling free space and occluded regions that do not contribute to the final result. This sampling of empty regions represents wasted computation. The key intuition to fix this is to sample around regions that would have a larger effect on the final rendering; i.e. the number of samples should be proportional to the expected effect on color. The procedure is “biases” samples to cluster along surfaces and regions of particular interest.

The NeRF paper accomplishes this by creating two separate networks, a “coarse” and a “fine” network. The coarse network uses stratified sampling to evaluate a first pass for the ray. The fine network then uses the output of this coarse network to converge samples of the ray along more relevant points in the volume. This procedure biases samples to regions with visible content. By evaluating the fine network they can compute a final rendered color for the ray. This allows greater fidelity for scenes while keeping the number of samples relatively low.

RESULTS

NeRF quantitatively outperforms SRN, NV, and LLFF in the Diffuse Synthetic 360, Realistic Synthetic 360, and Real Forward-Facing datasets. Final numbers are shown in the figure below.



	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	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

Figure 4: Table measuring model performance with various accuracy metrics (Source: *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis)*

Beyond achieving quantitatively better results for synthetic and real scenes, NeRF has qualitatively more accurate results than other competing approaches, as show in the figure below.

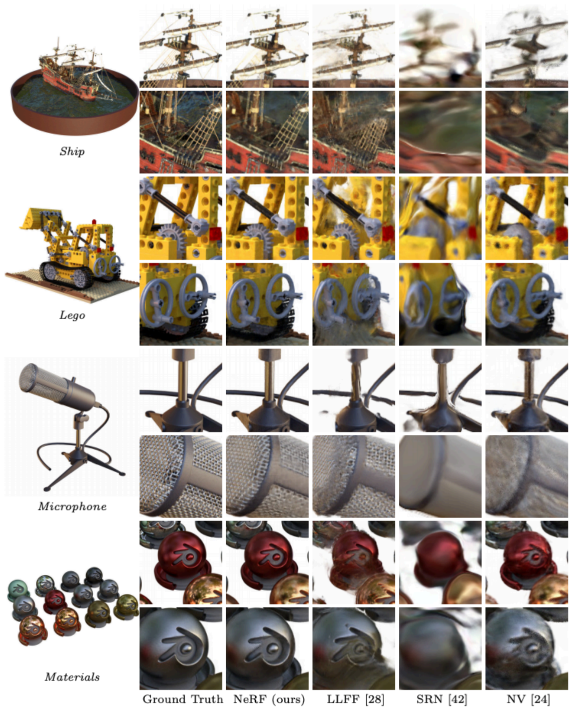


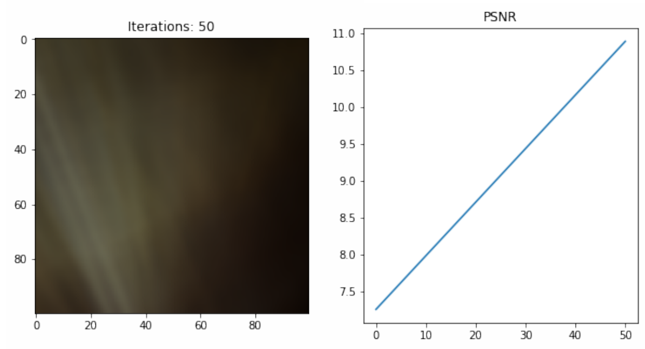
Figure 5: Qualitative comparison grid with other view synthesis methods (Source: *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis)*

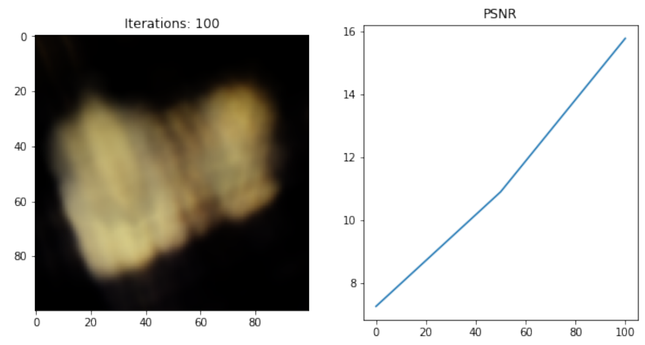
There are some quantitative points of interest to highlight when comparing NeRF with its alternatives. As seen in the texture of the Microphone, NeRF is the most accurate when evaluating texture fidelity. And in the Materials test, NeRF is the only one to accurately recreate the diffuse highlights present in the scene.

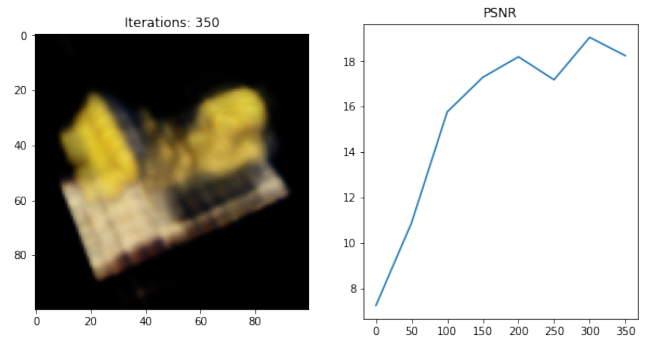
For the purpose of analysis and verification, I re-implemented the NeRF paper. Some challenges when initially writing the new implementation cropped up. For example, understanding how the volume rendering process works and utilizes the neural net on a conceptual level was difficult. The step from taking the outputs of the neural net to final image involved some complex matrix math and code to follow. Understanding this step meant understanding the rendering process of estimating a ray's integral to produce a final image. As another example, my first complete write had the model training off of a CPU, and it became clear early on that this would not be fast enough to produce results, even with my simplified model. I had to go back and refactor my code to place all tensors into GPU memory and have the model train with GPU's.

One interesting aspect I found while recreating the NeRF algorithm was the simplicity in parts of its approach. By comparing two images to produce loss function, it optimizes the whole representation by using a simple mean squared error to calculate the difference between the predicted and actual image. No requirement to ground truth geometry expands the scenarios where NeRF can be utilized, as the process for capturing such data can be cumbersome and imprecise. In code, the entire loss is calculate in one line after a final image is composited.

My output trained over the Lego dataset for 350 iterations, with printouts every 50. The code is hosted at: https://colab.research.google.com/drive/1gcoHyO0LgYgiz3MQKugybbzORvTYJG3W?usp=sharing. My implementation was written with the PyTorch framework, and trained on Google Colab. My personal results are shown below.







CONCLUSION

By representing scenes as 5D neural radiance fields, NeRF addresses the deficiencies of previous works using MLP's to represent objects and scenes. Their results are compressed and produce better renders than the previous approaches with discrete voxel grids. By explicitly incorporating viewing direction into their model, view-dependent effects such as lighting and occlusion are much more accurately represented. The NeRF model can approximate finer details than prior work, and can do so with relatively compressed file sizes. For example, one "Realistic Synthetic" scene produced with LLFF and its voxel grid-based approach required around 15GB, while NeRF reduced this to only 5MB. This file size for NeRF is actually smaller than the input images alone that were used to feed the model.[1] This is while Neural Radiance Fields produce better renderings than the previously dominant approach of discrete voxel grids.

FUTURE WORK

The NeRF method is not without flaws of its own. This paper lays a foundation for the technique and paves the way for multiple paths of questioning. For instance, the model faces a central drawback of only being able to represent a single scene. This limits any potential for the technique to generalize to new scenes, as any possible learned priors by the model are discarded for each new scene. An interesting work that follows up by using monocular depth predictions as a prior is the Neural Scene Flow Fields [4] paper. The training requirements for NeRF are currently prohibitive. While NeRf has significant file size advantages over LLFF, it takes at least 12 hours to train a scene. Compare this time with LLFF, which can process a small dataset in under 10 minutes. As NeRF requires longer training times than its alternatives, weight interpretability is crucial for a model to encode basic shape priors of the physical world. Encoding such priors could drastically lower the rather slow training times of the original paper. The Nerf++ [5] and Neural Sparse Voxel Fields [6] papers aim to improve general performance through optimizations with unbounded scenes and using a sparse voxel octree as an organizing data structure.

Another limitation for the paper is its lackluster performance with dynamic objects and conditions. This includes moving, dynamic physical objects, backgrounds, and lighting conditions. A variety of papers handle dynamic scenes, most notably the Neural Scene Flow Fields paper. Lighting has been augmented and baked into the model by using latent codes that can be used to re-light a scene.

REFERENCES

[1] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020, August). Nerf: Representing scenes as neural radiance fields for view synthesis. In European Conference on Computer Vision (pp. 405-421). Springer, Cham.

[2] Niemeyer, M., Mescheder, L., Oechsle, M., & Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3504-3515).

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

[4] Li, Z., Niklaus, S., Snavely, N., & Wang, O. (2020). Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. arXiv preprint arXiv:2011.13084.

[5] Zhang, K., Riegler, G., Snavely, N., & Koltun, V. (2020). Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492.

[6] Liu, L., Gu, J., Lin, K. Z., Chua, T. S., & Theobalt, C. (2020). Neural sparse voxel fields. arXiv preprint arXiv:2007.11571.

[7] Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., ... & Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*.