An End-to-End Neural Architecture for View Synthesis: Comparing NeRF + ColMap v. NeRF −−

Kimmy Chang
Stanford University
kchang08@stanford.edu

Jacky Lin
Stanford University
jackylin@stanford.edu

Xiaodan Zhu
Stanford University
xzhu08@stanford.edu

Abstract

Scene reconstruction, the process of creating a digital 3D representation of an object from a series of pictures, is important in many fields including motion capture, object detection, and augmented reality. Current scene reconstruction techniques, including voxel carving and NeRF, require known camera positions. NeRF−− introduces an end-to-end solution where the neural architecture is only given RGB images without precomputed camera parameters. In this paper, we evaluate our models on two datasets: a NeRF-provided lego dataset and self-generated Blender tank dataset. We demonstrate that NeRF−− achieves comparable results to NeRF, and surpasses the results of NeRF + pre-computed COLMAP camera parameters. Though we find the PSNR of NeRF to be much higher than that of NeRF−− for both datasets, we attribute this difference to the computational requirements for NeRF−− to reach convergence. Evaluating the PSNR graph for NeRF, we see early signs of convergence, whereas the value for NeRF−− continues to increase. In contrast, for most cases in NeRF + COLMAP in both datasets, we fail to reach convergence or a good reconstruction in the test datasets. We present a qualitative analysis in our paper of the reconstructed views to highlight the differences between the two methods. Hence, our paper demonstrates the successful implementation of NeRF from scratch and that NeRF−− produces comparable (and in many cases better) results than NeRF + COLMAP. Nevertheless, NeRF−−'s end-to-end solution comes at the cost of a much longer and expensive training time. This expense, however, is only incurred once during training, and for the purposes of scene reconstruction, NeRF−− achieves comparable results with only RGB images, a significant improvement over current NeRF-based approaches.

1. Introduction

In this paper, we investigate the representation of scenes as a neural radiance field (NeRF) for view synthesis, replicating the paper by Ng et al. [5]. The radiance field representation produces a volume dense and view dependent RGB color image that is photo realistic.

This problem is important because it allows neural scene representation of photo-realistic views of real objects to be captured in natural settings. With applications such as 3D reconstruction of images, NeRF presents itself as a novel approach to produce a volume dense and view dependent RGB color image from multiple views.

The fully connected deep network (i.e., MLP) will have an input that consists of a 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)), and output that is the view-dependent emitted radiance and volume density at the given spatial location. By querying 5D coordinates along a camera view ray, we can get a list of radiance and volume densities. Each pixel value can be evaluated by using classical volume rendering. The MLP is trained by using images taken from multiple views with known camera poses.

The authors of [5] provide an implementation of NeRF, but we will implement the method by ourselves. We will only use their code as a reference to examine our implementation. In addition to this, we optimize the neural radiance field with *hierarchical volume sampling* and *positioning encoding*[12]. Following this, we extend NeRF with COLMAP to process photos with unknown camera poses. Finally, we will explore the DNN solution of NeRF−−.

2. Background/Related Work

Scene reconstruction, also known as views synthesis, is the process of creating a digital 3D representation of an object from a series of pictures. Its applications are vast including in many important fields such as motion capture, object detection, and augmented reality. Views synthesis is a computer vision problem that goes back decades, and the vast literature can roughly be broken into two categories: explicit surface modeling and dense volume-based repre-

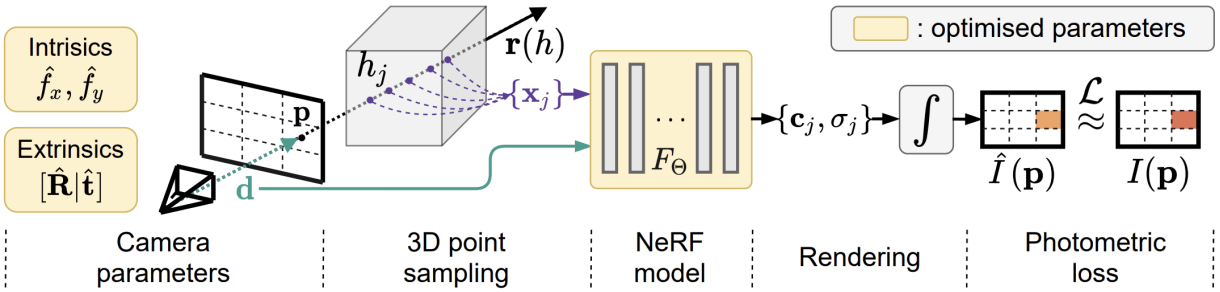


Figure 1: The NeRF −− model that jointly optimizes the NeRF model and the camera parameter. The objective function is to minimize the photometric reconstruction errors. The pipeline is trained with RGB images only and without the camera parameters. Diagram from [13]

.

sentation [13].

Explicit surface modeling (the older of the two approaches) seeks to directly reconstruct the surface geometry to generate the 3D geometric view from 2D images. Using techniques as structure from motion (SFM) and simultaneous localization and mapping (SLAM), one can solve the 3D geometry and the camera parameters by determining features correspondences, from which a system of linear equations arise and the solutions approximated. In practice, because these approaches assume diffuse surface textures, they do not recover view-dependent appearances and therefore do not generate realistic view rending [13]. To address this assumption, multi-view photometric stereo methods are used with complex bidirectional reflectance distribution function (BRDF) models to work with view-dependent appearances [14]. However, this approach suffers from the tradeoffs between quality and complexity. Though the geometric reconstruction estimates the camera parameters, explicit reconstruction of generated novel views is not photo realistic.

As an alternative to explicit reconstruction, volume-based reconstruction are used to directly model the appearances of 3D space [11]. In recent years, examples of volume-based reconstruction include Soft3D [7], Multi-plane Images (MPI) [1], and Neural Radiance Fields (NeRF) [5]. These dense volumetric representation allow for smooth gradients and whose view synthesis is more photo realistic even for highly complex shapes and appearances.

However, both categories of current scene reconstruction techniques, including voxel carving and NeRF, require known camera positions. These approaches require accurately estimated camera parameters, usually by traditional structure-from-motion technique or simultaneous localization and mapping (SLAM), as implemented in COLAMP [10]. Hence, these volumetric representation approaches involve a two-step process: camera parameter estimation and view estimation based on the estimated camera parameters.

Before the introduction of NeRF, existing techniques that sought to use a multi-layer perceptron to map from a 3D representation of the shape to a implicit representation did not achieve the same level of fidelity from techniques that used discrete representations (such as triangle meshes or voxel grids). The lack of fidelity in the implicit representation was especially problematic for complex shapes. NeRF was a significant improvement in generating photo-realistic views but still required camera parameters. NeRF−− produces a volumteric representation but does so without needing the camera parameters. The NeRF−− pipeline jointly optimizes the NeRF model and the camera parameters, minimizing the photometric reconstruction errors. The pipeline is trained with RGB images only and without the camera parameters.

3. Approach

3.1. Neural Radiance Field

NeRF expresses a complex scene with a continuous function as shown in Eq (1).

$$L_o, \sigma = f(x, y, z, \theta, \phi) \tag{1}$$

where (x, y, z) is a spatial location, (θ, ϕ) is the viewing direction, L_o and σ are the emitted radiance and volume density at (x, y, z).

From an arbitrary view point, by querying positions (x, y, z) along a certain camera view direction (θ, ϕ), we could estimate the corresponding pixel value and then synthesize the scene.

3.1.1 Volume Rendering

In the physical world, any radiance emitted from a point is partially absorbed by the medium before reaching the

eye/sensor [8] as shown in Figure 2.

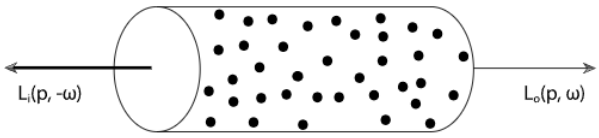


Figure 2: Volume Absorption

where $p = (x, y, z)$ is a spatial location and ω is a solid angle, L_i is the incident radiance, and L_o is the output radiance. Given the volume density is σ within a short distance dt, we have:

$$\begin{aligned} L_o - L_i &= -\sigma L_i dt \\ dL_o &= -\sigma L_i dt \end{aligned} \tag{2}$$

According to Eq (2), if a radiance travels a distance t, the remaining portion would be

$$T(t) = exp(-\int_0^t \sigma dt) \tag{3}$$

Therefore, the color of a pixel at o can be estimated by integrating the radiance at position $p(t) = o+dt$ through the view ray d from the near plane to far plane of the viewing frustum.

$$C(o, d) = \int_{t_{near}}^{t_{far}} T(t)\sigma(p(t))L_o(p(t))dt \tag{4}$$

The numerical expression of Eq (4) is shown below [4].

$$\begin{aligned} &\hat{C} = \textstyle\sum_{i=0}^{N} T_i \alpha_i l_i \\ \text{where} \\ &T_i = exp(-\textstyle\sum_{j=1}^{i-1} \sigma_i \delta_i) \\ &\alpha_i = 1 - exp(-\sigma_i \delta_i) \\ &\delta_i = t_{i+1} - t_i \end{aligned} \tag{5}$$

From Eq (5), we can see that l_i and σ_i can be gotten from querying $f(o + d(t_i), d)$. Therefore, we are able to synthesize the scene.

3.1.2 Multi-Layer Perceptron

Input: since camera poses are known, it is easy to generate a $W \times H \times N$ query points, where W and H are the width and height of the image and N is the number of samples along the view ray.

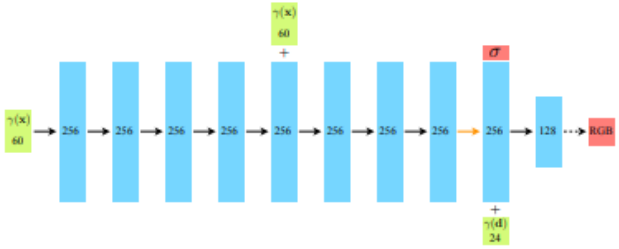


Figure 3: Structure of MLP

Structure: we follow the structure proposed in [5] which builds a fully connected neural network with 8 hidden layers. Each layer has 256 neurons followed by a ReLU activation layer. This is shown in Figure 3.

According to [6], inserting input in the middle layer could improve learning. Therefore, we concatenate the input with the fourth layer's output as the input of the fifth layer.

Output: we finally use ReLU to generate σ value and sigmoid to generate radiance.

Loss: the loss is the frobenius norm of the distortion between the estimated image and the ground truth, which is shown in Eq (6).

$$E = \sqrt{\sum_{i=1}^{W}\sum_{j=1}^{N} \|\hat{C_{ij}} - C_{ij}\|} \tag{6}$$

3.1.3 Optimizing NeRF

We build NeRF from the most basic version to optimized version. [5] performs two techniques to optimize NeRF which are position encoding and hierarchy sampling.

Position Encoding: [9] shows that the deep neural networks are biased towards learning low frequency functions, which means the basic NeRF may perform poorly on high-frequency color geometry. Actually, we also got this observation from our intermediate results shown in section 3.4. Therefore, before inputting into the neural network, we should first map the positions to higher frequency domain. Each dimension of the position is encoded with $\sin$ and $\cos$ function as shown in Eq (7).

$$\begin{aligned} \tilde{p} = [&\sin(2^0\pi p), \cos(2^0\pi p), \cdots, \\ &\sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)] \end{aligned} \tag{7}$$

Hierarchy Sampling: Uniformly sampling along every view direction is inefficient, because the empty and occluded regions do not contribute to the pixel color. Therefore, we borrow the idea from Monte Carlo integration. First, we train a coarse NeRF with small sampling number.

Then, we use the estimated volume density of the coarse NeRF to build a probability distribution. Finally, we use this probability distribution to generate samples for another fine NeRF to produce better estimations. Note that this is not used in our evaluation section, but is implemented in the code.

3.2. Extend NeRF with COLMAP

COLMAP is a general-purpose, open source structure from motion (SFM) and multi-view stereo (MVS) 3D reconstruction pipeline. Given a set of images, it recovers camera projection matrices and the observed 3d points. There are three stages to its process: (1) feature detection and extraction, (2) feature matching and geometric verification, and (3) structure and motion reconstruction. COLMAP for (1) finds sparse feature points in the images and attach it to a numerical descriptor; (2) does exhaustive matching to compare each image against every other image; (3) incrementally registers images and triangulates new points.

The output of COLMAP includes the intrinsic and extrinsic parameters as text files. The camera model that we chose to use is the simple radial camera. This gives the focal length f, translation vector $[c_x, c_y]^T$, and change of unit parameter in the two axes of the image plane k. Note that this model assumes that the camera has square pixels and does not account for skewness or distortion. The corresponding intrinsic camera matrix can be defined based on these parameters in Eq (8):

$$K = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \alpha & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where $\alpha = fk$. The extrinsic camera matrix is defined by the parameters R, T. The text file provides the translation parameters in the form t_x, t_y, t_z and rotation parameters in the form of a quaternion $q = q_r + q_x\mathbf{x} + q_j\mathbf{y} + q_k\mathbf{k}$. The rotation matrix is shown in Eq (9):

$$\mathbf{R} = \begin{bmatrix} 1-2s(q_j^2+q_k^2) & 2s(q_iq_j-q_kq_r) & 2s(q_iq_k+q_jq_r) \\ 2s(q_iq_j+q_kq_r) & 1-2s(q_i^2+q_k^2) & 2s(q_jq_k-q_iq_r) \\ 2s(q_iq_k-q_jq_r) & 2s(q_jq_k+q_iq_r) & 1-2s(q_i^2+q_j^2) \end{bmatrix} \tag{9}$$

where $s = ||q||^{-2}$ [2].

Finally, we can write the camera projection matrix in Eq (10):

$$M = K[RT] \tag{10}$$

The text output file parameters are converted into the camera projection matrices to match the input of NeRF as a json file.

Notably, COLMAP performance is expected to perform best when these image features a textured object, there are similar illumination conditions across the images, high visual overlap, and different viewpoints of the object of interest.

3.3. DNN solution: NeRF−−

Unlike NeRF, NeRF −− creates a novel view synthesis without the known camera parameters, instead opting for an end-to-end solution. Current methods of Neural Radiance Field assumes that the camera parameters are accessible during training or that they can be estimated from conventional techniques, such as structure from motion. The original NeRF paper proposed by Mildenhall et al. 2000 and the numerous variants use COLMAP to retrieve the intrinsic and extrinsic camera parameters. NeRF −− demonstrates that precomputed camera parameters are not necessary for a novel view synthesis and that an end-to-end solution can both optimize the 3D scene representation and the intrinsic and extrinsic parameters.

Since NeRF−− trains the poses with NeRF, we cannot get test pose in advance. Thus, we did the following:

1. Add the test image which is generated by the spin rotation of 0 degree into training set and retrieve the estimated pose at each iteration.
2. Multiply the estimated pose by 3.6 degree spin rotation matrix to get test pose.
3. Use the test pose to render a image.
4. Compare PSNR.

3.4. Experiment

We will use the original datasets used by the NeRF authors[5] to test our method.

- Link to NeRF trained weights: `https://github.com/bmild/nerf`
- Link to NeRF data: `https://drive.google.com/drive/folders/128yBriW1IG_3NJ5Rp7APSTZsJqdJdfc1`
- Link to model architecture: `https://arxiv.org/abs/2003.08934`

We will evaluate our results both qualitatively and quantitatively. Qualitatively, we expect figures and will compare the reconstructions to the originals. Quantitatively, we will use three performance metrics. The first metric is the Peak signal-to-noise ratio (PSNR) between the original and reconstruction. PSNR is defined as seen below:

$$10\log_{10}\left(\frac{(L-1)^2}{\text{MSE}}\right)$$

where L is the number of maximum possible intensity levels in an image, and MSE is the mean squared error.

After validating our methods, we plan to generate new scene data by using Blender.

3.4.1 Data Generation

We chose to locate our model at the origin of world frame. We took a photo of the model from a camera at spherical coordinate (r, θ, ϕ). This process is enumerated as follows: (1) move the camera along Z axis by r, (2) rotate around X axis by $-(\frac{\pi}{2} - \phi)$, (3) and rotate Y axis by θ. Equivalently, Equation (11),

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & -\cos\phi & \sin\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ r \\ 1 \end{bmatrix} = (r\sin\phi\sin\theta, r\cos\phi, r\sin\phi\cos\theta) \tag{11}$$

The results of data generation are shown in Figure 4.

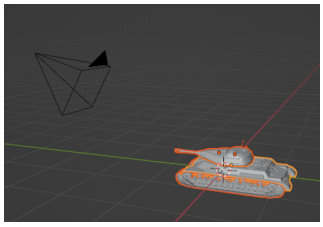


Figure 4: Data Generation

When generating the training data, we uniformly sampled the hemisphere to get the θ and ϕ.

$$\begin{aligned} \theta &= 2\pi\xi_1 \\ \phi &= \arccos\xi_2 \end{aligned} \tag{12}$$

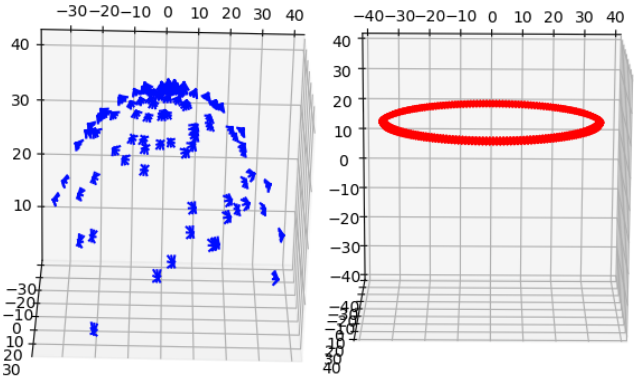
where ξ_1 and ξ_2 are random numbers.

When generating the test data, we let the camera rotate around Y axis by 360 degrees with fixed ϕ.

Finally, we save the photos and corresponding transform matrices. The synthetic training and test poses are shown in Figure 5.

3.4.2 NeRF Results

First, we present the results of the lego dataset. In Figure 6, the results of the test dataset are visualized. A test image at 15k iterations is also shown in Figure 7. From these, we can see that qualitatively, the replicated NeRF model is able to perform well on the test dataset. In Figure 7, the



(a) Synthetic Training Poses (b) Synthetic Test Poses

Figure 5: Visualize the Synthetic Poses

PSNR exemplifies that the PSNR steadily improves over the iterations. In addition to the images, we provide a video of our results at `https://youtu.be/YArYmPKbJKU`.

Figure 6: Lego NeRF Test Results (15k iterations)

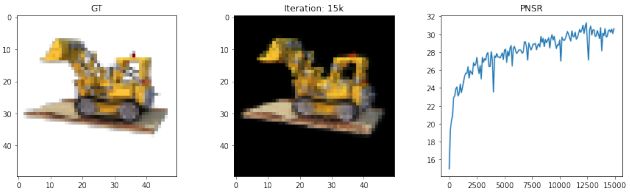


Figure 7: Lego NeRF Test Image

Next, we present the results of the tank dataset. In Figure

8, the results of the test dataset are visualized. A test image at 15k iterations is also shown in Figure 9. From these, we can see that qualitatively, the replicated NeRF model is able to perform well on the test dataset. In Figure 9, the PSNR over iterations graph exemplifies that the PSNR steadily improves over the iterations.

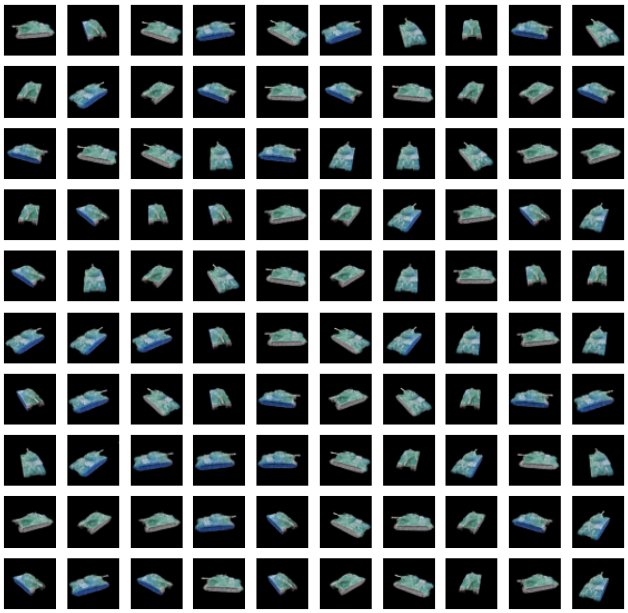


Figure 8: Tank NeRF Test Results (15k iterations)

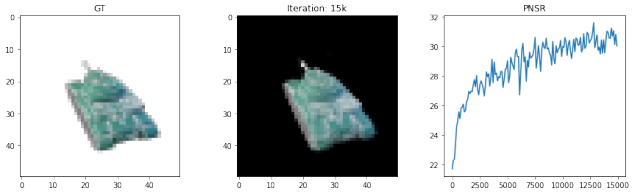


Figure 9: Tank NeRF Test Image

The NeRF PSNR results are shown below:

NeRF PSNR (15k iteration)	
	Test
Tank	25.130621
Lego	23.565414

The NeRF graph of the model structure is shown in Figure 10.

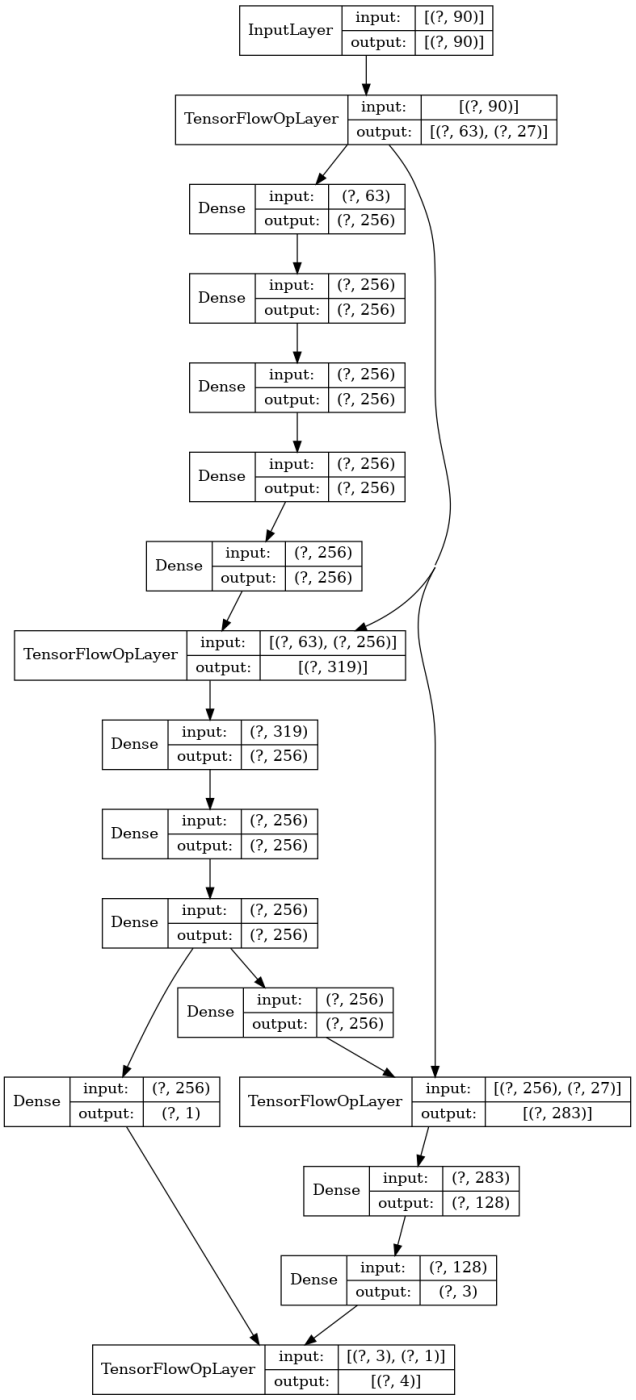


Figure 10: NeRF Graph

3.4.3 NeRF + COLMAP Results

First, we present the results of the lego dataset. In Figure 11, the results of the test dataset are visualized. From the results, we can see that qualitatively, the replicated NeRF model performs poorly on the test dataset. A few of the lego test images are able to reconstruct well on some iterations (see Figure 12). However, this is nowhere close to the majority.

Next, we present the results of the lego dataset. In Figure 13, the results of the test dataset are visualized. From the

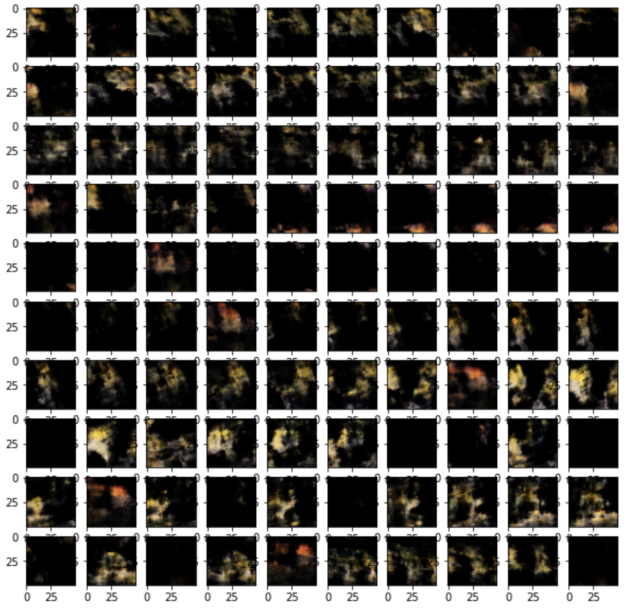


Figure 11: Lego NeRF + COLMAP Test Results (15k iterations)

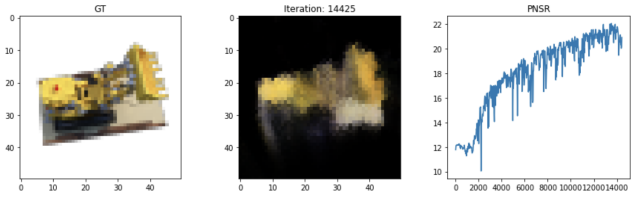


Figure 12: Lego NeRF + COLMAP Sample Test

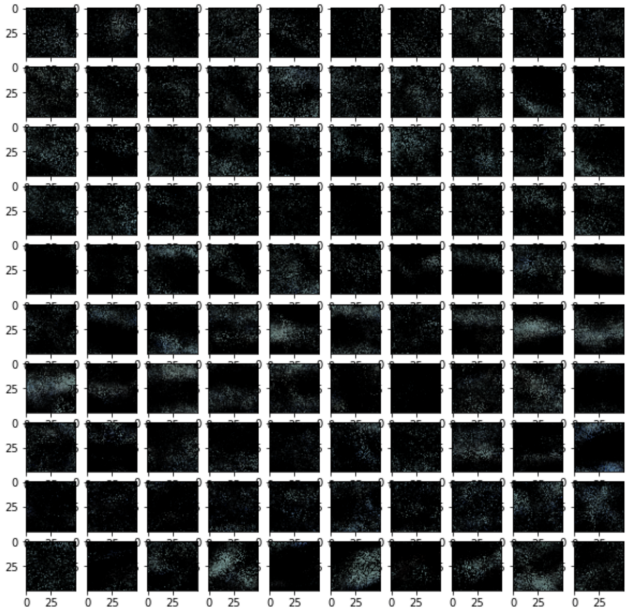


Figure 13: Tank NeRF + COLMAP Test Results (15k iterations)

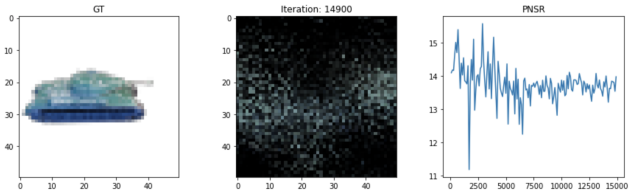


Figure 14: Tank NeRF + COLMAP Sample Test

results, we can see that qualitatively, the replicated NeRF model performs poorly on the test dataset. In Figure 14, like in the lego dataset, it can be seen that the PSNR for the tank dataset fails to improve over the 15k iterations.

The NeRF + COLMAP PSNR results are shown below:

NeRF + COLMAP PSNR (15k iteration)		
	Train	**Test**
Tank	22.843103	21.427067
Lego	25.447199	18.301394

The PSNR results show that for both datasets the train PSNR is greater than the test PSNR, which supports the good train visualizations and poor test visualizations. Both datasets have a test PSNR that is significantly lower than the test PSNR acheived in NeRF (with known camera poses).

Since the test datasets yielded poor results, we chose to investigate by looking at the train dataset visualizations (see Figure 15 for the lego dataset, see Figure 16 for the tank dataset). The train reconstructions are able to perform comparable to NeRF. However, for some tank train images, it is clear that the reconstruction is still significantly worse than that of NeRF (with known camera poses).

When delving further into the poor reconstruction of the test datasets, we contrasted the pose estimation of COLMAP against the true poses. In Figure 17 and Figure 18, it can be seen that the pose estimation of COLMAP against the true poses is poor, which is likely to explain the poor test reconstruction (and slightly lesser reconstruction of the train datasets).

From this, we conjecture that the poor results are likely from pose estimation not being close enough to the true poses. Nevertheless, we expect that given more iterations (note: the true NeRF undergoes around 200k iterations), the test results would be able to improve.

3.4.4 NeRF−− Results

Due to the computational expense of NeRF−−, in this section, we present the results of a single, random test image. Note that the results are for 1.5k iterations instead of the prior 15k iterations since NeRF runs through all images

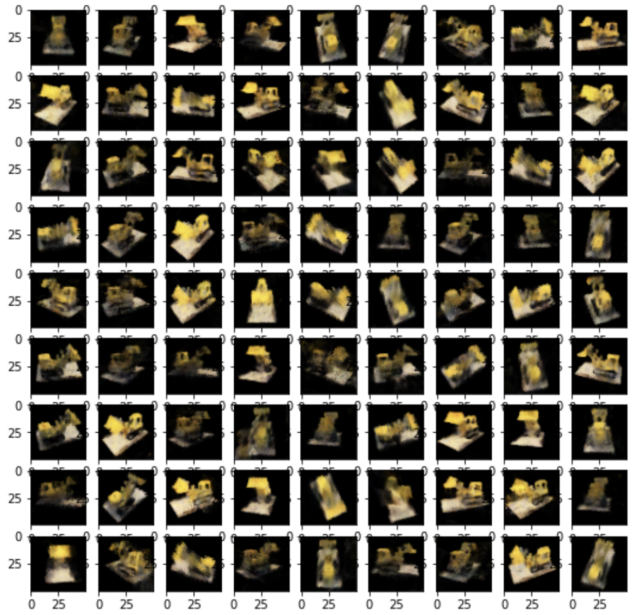


Figure 15: Tank NeRF + COLMAP Train Results (15k iterations)

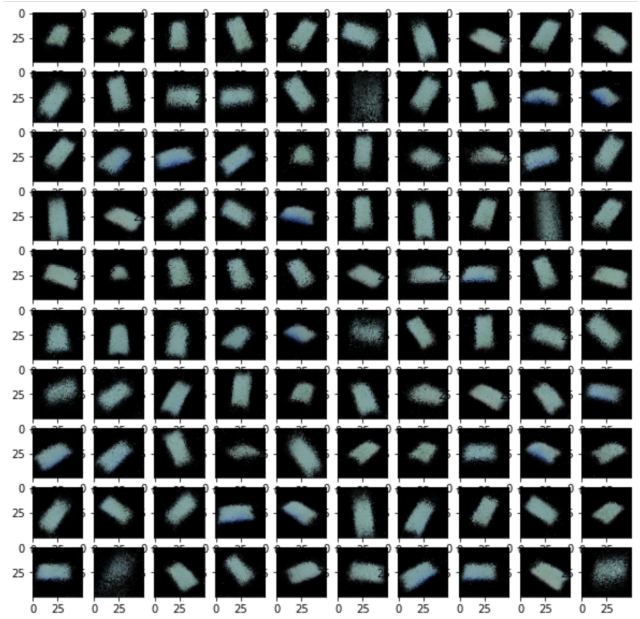
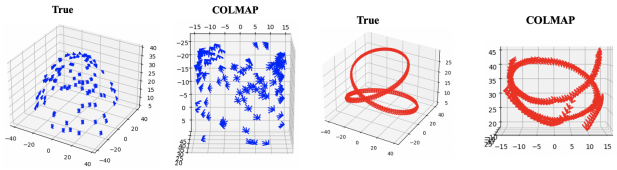


Figure 16: Lego NeRF + COLMAP Train Results (15k iterations)

each iteration.

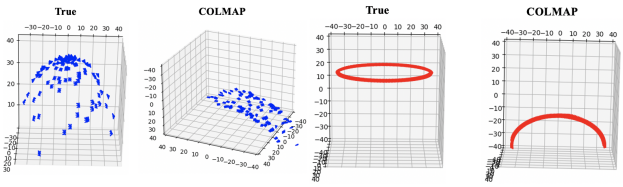
First, we present the results of the lego dataset. In Figure 19, it can be seen that we are able to render a reasonable reconstruction.

Next, we present the results of the tank dataset. In Figure 20, it can be seen that we are also able to render a reasonable



(a) Lego Train Poses (b) Lego Test Poses

Figure 17: Lego True v. COLMAP Poses



(a) Tank Train Poses (b) Tank Test Poses

Figure 18: Tank True v. COLMAP Poses

reconstruction.

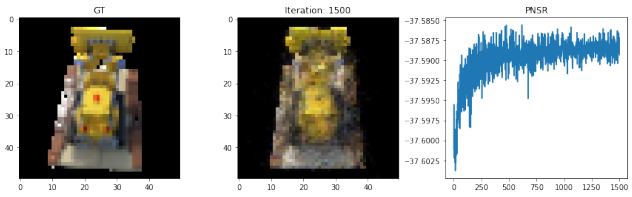


Figure 19: Lego NeRF−− Test Image (1.5k iterations)

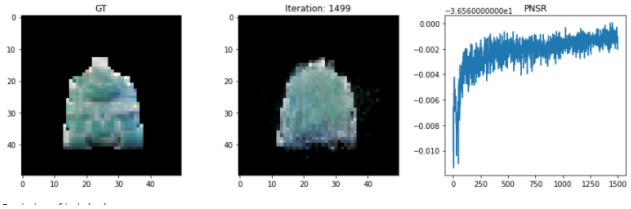


Figure 20: Tank NeRF−− Test Image (1.5k iterations)

The PSNR results are shown below:

NeRF−− PSNR (random image)		
	Train	**Test**
Tank	24.611	-36.5622
Lego	21.257	-37.5899

Notably, the train PSNRs are significantly higher than the test PSNRs for both datasets. Interestingly, the test PSNRs are negative, which indicates that the L2 loss is very large according to the PSNR equation. However, qualitatively, the test images do not seem to be poor.

4. Conclusion

We evaluated our models on two datasets: a NeRF-provided lego dataset and self-generated Blender tank dataset. We learned that NeRF−− achieves comparable results to NeRF qualitatively, and surpasses the results of NeRF + pre-computed COLMAP camera parameters. Though we found the test PSNR of our implemented NeRF to be much higher than that of NeRF−− for both datasets, we attribute this difference to the computational requirements for NeRF−− to reach convergence. Evaluating the PSNR graph for NeRF, we see early signs of convergence, whereas the value for NeRF−− continues to increase. In contrast, for most cases in NeRF + COLMAP in both datasets, we fail to reach convergence or a good reconstruction in the test datasets. Conclusively, we were able to demonstrate the successful implementation of NeRF from scratch and that NeRF−− produces comparable (and in many cases better) results than NeRF + COLMAP.

In terms of improving our work, we could first explore alternate metrics to evaluate our models. Specifically, [5] proposes two additional metrics. The first additional metric is Diffuse Loss. Since PSNR and Diffuse Loss cannot really tell how the output of 3D reconstruction improves, the second additional metric is Chamfer Distance. Defined in [3] as,

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2$$

We also propose running the model for more iterations in order to see if there are improvements in NeRF + COLMAP and NeRF−−. Finally, as an alternative to COLMAP to retrieve camera poses, we could explore the LLFF imgs2poses method provided by [5] in order to generate camera poses for the datasets in order to see if this produces improved pose estimations.

5. Acknowledgements

Much thanks to Professor Silvio Savarese and Professor Jeannette Bohg as well as the rest of the CS 231A staff for their support throughout the quarter. We are also immensely grateful to Yinan Zhang for her help and feedback throughout this quarter on the project.

6. GitHub Link

`https://github.com/xzhu08/cs231a_mynerf`

References

[1] I. Choi, O. Gallo, A. J. Troccoli, M. H. Kim, and J. Kautz. Extreme view synthesis. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7780–7789, 2019.

[2] J. A. et. al. Computer graphics algorithms. Faaqs.org, 2022.

[3] H. S. Haoqiang Fan and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[5] 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 *ECCV*, 2020.

[6] 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*, pages 165–174, 2019.

[7] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36:1 – 11, 2017.

[8] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[9] 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*, pages 5301–5310. PMLR, 2019.

[10] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[11] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35:151–173, 2004.

[12] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[13] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[14] Z. Zhou, Z. Wu, and P. Tan. Multi-view photometric stereo with spatially varying isotropic materials. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1482–1489, 2013.