RBE 549 Computer Vision Project 2 - NeRF

Kaushik Kavuri Subrahmanya
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: ksubrahmanya@wpi.edu

Butchi Adari Venkatesh
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: badari@wpi.edu

***Abstract*—In this project, we implement the original NeRF method to render scenes of objects with complicated geometries and appearance.**

I. Dataset

This approach takes 5D coordinates viz. spatial location (x, y, z) and viewing direction (θ, ϕ)) as input and the output is the volume density and view-dependent emitted radiance at that spatial location. To train the model, we use the same 100 images of a LEGO piece, and a pirate ship piece provided by the original authors of the paper. These images capture the object from various poses all around the object. The corresponding poses for each of the images are also provided in the same source.

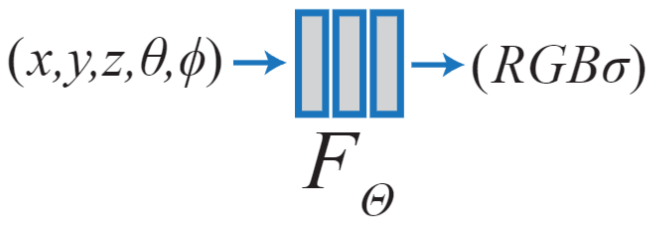

Fig. 1: Input and Output of NeRF

A. Network Architecture and Pipeline

The architecture used in NeRF is a multi-layer perceptron (MLP). The MLP uses ReLU activation for each layer and Adam optimizer is used. There are 8 fully connected layers with 256 channels in each layer. In NeRF, positional encoding plays a crucial role in capturing spatial information about the scene's geometry and viewpoint directions.

1. **Input Representation**: NeRF takes as input the 3D points sampled along camera rays in the scene and corresponding viewing directions or viewpoints. Each point's 3D coordinates (x, y, z) and viewing directions (θ, ϕ) are encoded using positional encoding to capture spatial information effectively.

2. **Positonal Encoding**: Positional encoding involves the use of sinusoidal functions to encode the spatial coordinates (x, y, z) and viewing directions (θ, ϕ). These sinusoidal encodings enable the neural network to understand the geometric relationships between different points in the scene and viewing directions. Stored as $\gamma(63)$ and $\gamma(27)$.

3. **Neural Network Architecture**: The positional encoded inputs are then fed into a neural network architecture, typically consisting of multiple fully connected layers. These layers process the encoded inputs to predict radiance values (RGB color and opacity) for each input point in the scene.

4. **Training Objective**: NeRF is trained to minimize the discrepancy between the predicted radiance values and ground truth values observed from known viewpoints in the scene. This training objective allows the network to learn an accurate representation of the scene's geometry and appearance.

5. **Rendering Novel Views**: Once trained, the NeRF model can synthesize novel views of the scene by evaluating the volumetric scene function at different viewpoints. By leveraging the learned spatial representations encoded through positional encoding, NeRF can generate high-quality renderings of scenes from arbitrary viewpoints.

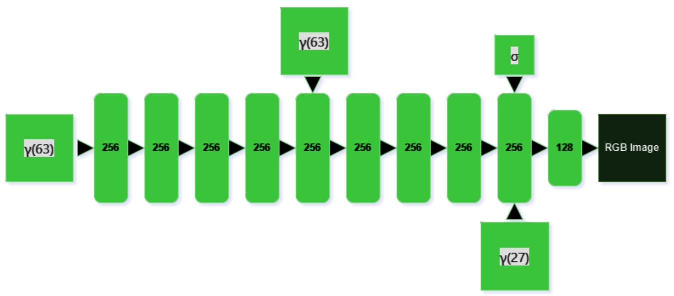

Fig. 2: NeRF Architecture

B. Challenges and Workarounds

1) Out of Memory: While training the model, we ran into CUDA out-of-memory issue. To solve this, we only loaded images into the GPU in batches, and only when necessary, instead of storing all of them throughout the process. In addition, we also stopped loading unnecessary variables into the GPU.

2) Training Time: As the training time required to train was very high, we resized all of the input images to 400*400px sized images. This significantly reduced the time to train the model at the cost of reduced quality of output. We initially resized the images to 100*100px but the reduction in quality was extremely high and the results were quite unsatisfactory.

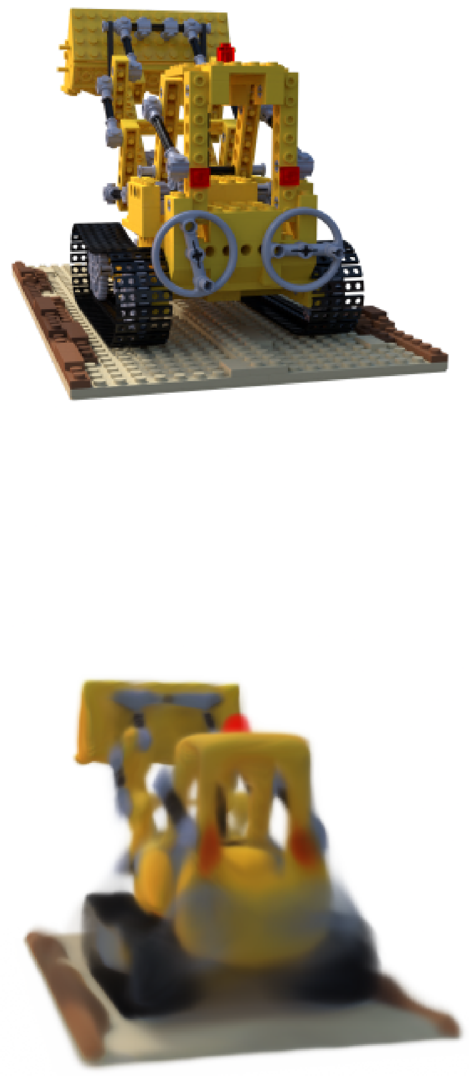
Fig. 3: LEGO - Ground Truth and NeRF prediction without positional encoding at 20k iterations

C. Results

Ground truth vs NeRF outputs with and without positional encoding are given in Figs. 3 - 5.

Mode	Loss	PSNR Loss	SSIM
Train	7e-4	31.2	-
Test	.5e-03	28.4	.95

TABLE I: NeRF Loss at 100k iterations

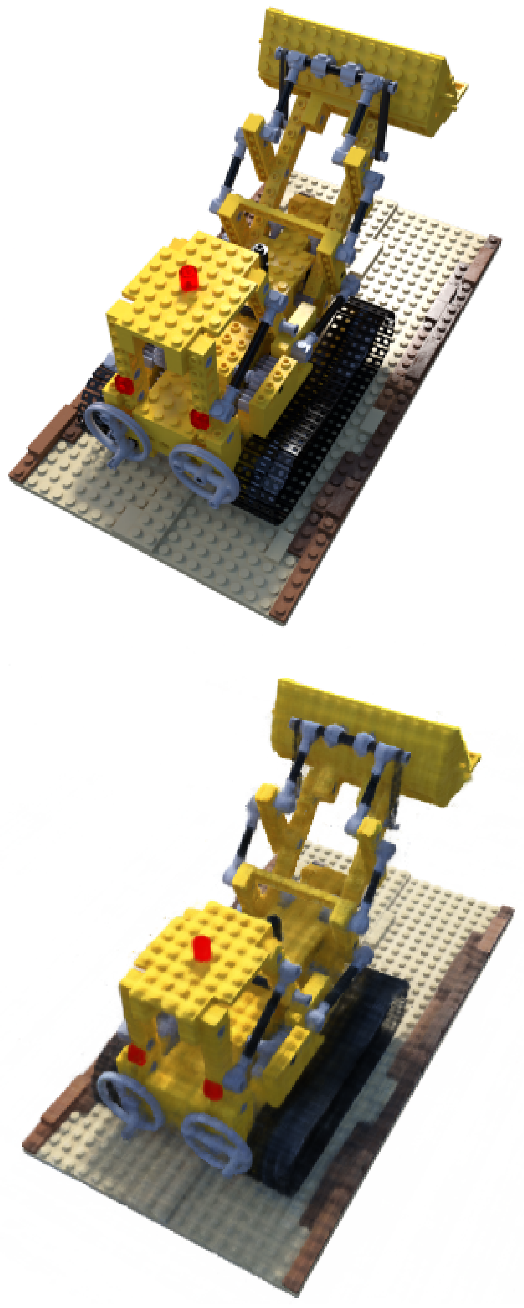
Fig. 4: LEGO - Ground Truth and NeRF prediction using positional encoding at 100k iterations

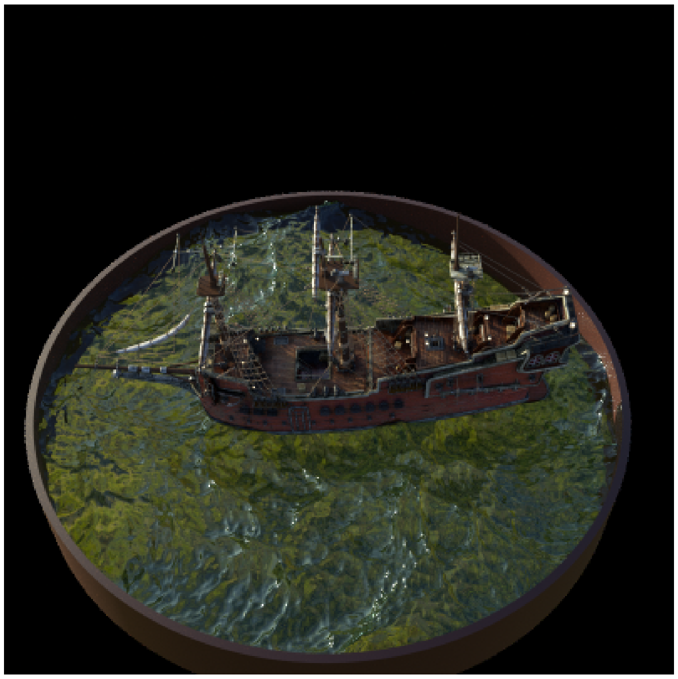
Fig. 5: Ground Truth

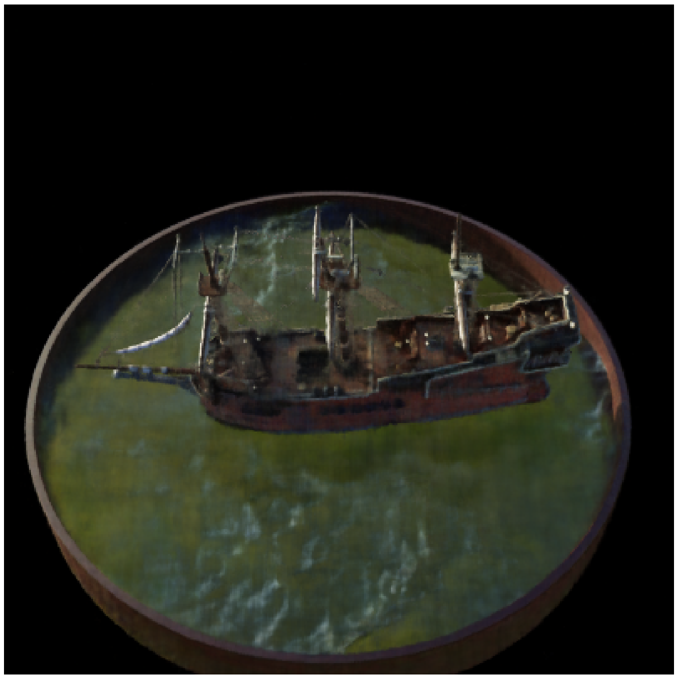
Fig. 6: NeRF Prediction

Fig. 7: Ship - Ground Truth and NeRF prediction using positional encoding at 100k iterations

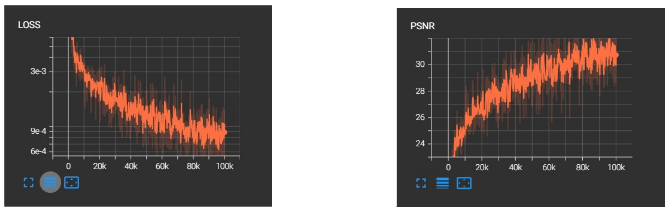

Fig. 8: Loss and PSNR per iteration for Train set

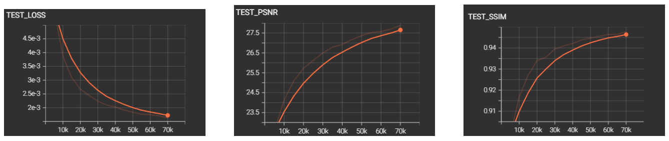

Fig. 9: Loss, PSNR and SSIM per iteration for Test set

II. Conclusion

As we can see, the NeRF outputs are close to the input images. A better output would require us not to resize the images, but the time to train the model would also greatly increase.

Acknowledgment

The authors would like to thank the professor for the instructions and knowledge imparted via the course.