Project 2 - Buildings built in minutes - NeRF RBE/CS549 Computer Vision

Shambhuraj Mane
MS Robotics Engineering
samane@wpi.edu

Swati Shirke
MS Robotics Engineering
svshirke@wpi.edu

***Abstract*—This paper provides a pipeline for generating novel views from sparsed input images of a scene captured from different camera positions. The technique used is Neural Field Radiance (NeRF). NeRF is a fully-connected neural network that can generate novel views of complex 3D scenes, based on a partial set of 2D images. It is trained to use a rendering loss to reproduce input views of a scene. It works by taking input images representing a scene and interpolating between them to render one complete scene. The pipeline includes ray generation, ray sampling, positional encoding, model training and volume rendering.**

Index Terms* — *ray generation, volume rendering, positional encoding

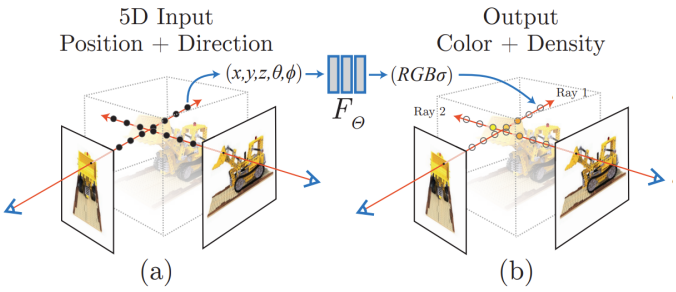

Fig. 1. Physical Interpretation of NeRF

I. 3D RECONSTRUCTION: SYNTHETIC DATA

The entire pipeline is as follows:

- Data pre-processing
- Pixel to ray generation
- Ray sampling
- Positional encoding
- RGB and depth value generation
- Volume rendering

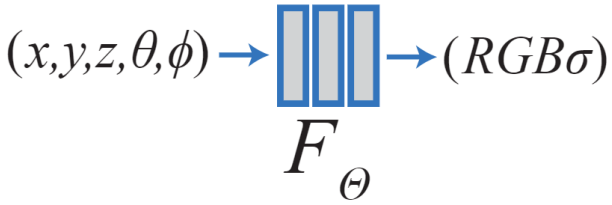

Fig. 2. NeRF input output

A. Data pre-processing

In the data set used, we have a projection matrix of each input image and angular field of view. Further, we calculated the focal length from the angular field of view and the height of the image.

B. Pixel to Ray generation and sampling

All image pixels are converted into 3D points using a projection matrix. Further, for each pixel point, a ray is passed in 3D space from the camera centre and passes through the pixel point. N number of rays are generated for each image which is a hyper-parameter. Each ray is further sampled uniformly between near and far points. Random noise is added in sampled points to avoid over-fitting of a model. For every sampled 3D point on each ray, we have (x, y, z, ϕ, θ), which is input to the neural network.

C. Positional Encoding

With (x, y, z, ϕ, θ) feeding as an input to the network, the resulting renderings struggle to accurately capture high-frequency variations in both colour and geometry. This problem is solved by using positional encoding. It can be mathematically expressed as below.

$$\gamma(p) = \left(\sin^2(0\pi p), \cos^2(0\pi p), \ldots, \sin^2((L-1)\pi p), \cos^2((L-1)\pi p)\right) \quad (1)$$

This function γ is applied separately to each of the three coordinate values in x (which are normalized between 0 and 1) and to the three components of the Cartesian viewing direction unit vector d. In our experiments, we set L = 10 for $\gamma(x)$ and L = 4 for $\gamma(d)$.

D. Network Architecture

We have used a fully connected neural network with architecture as shown below.

Fig. 3. Network Architecture

E. Volume Rendering

The 5D neural radiance field encapsulates a scene's essence by encoding both its volumetric density and directional emitted radiance at every spatial point. Leveraging classical volume rendering principles, we compute the color of any given ray traversing through the scene. The volume density $\sigma(x)$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location x. The expected color C(r) of camera ray r(t) = o + td with near and far bounds tn and tf is given by the below equation.

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

F. Results

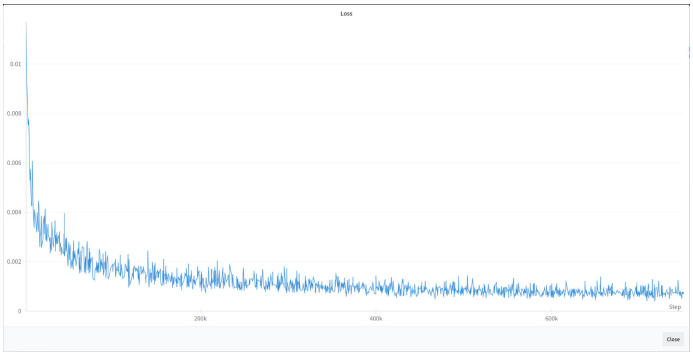

Fig. 4. Training Loss for Lego data

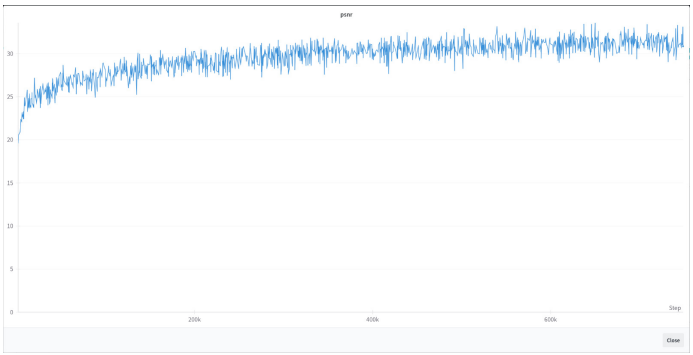

Fig. 5. Training PSNR for Lego data

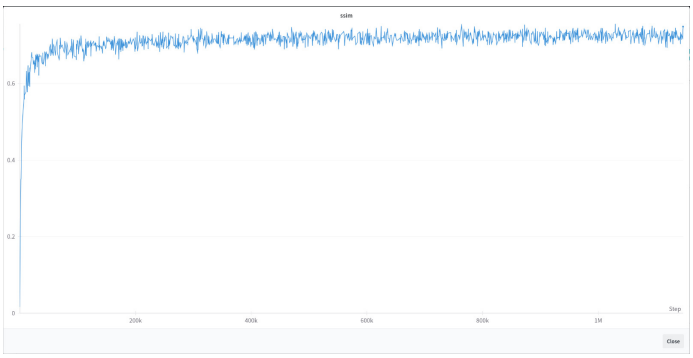

Fig. 6. Training SSIM Lego data

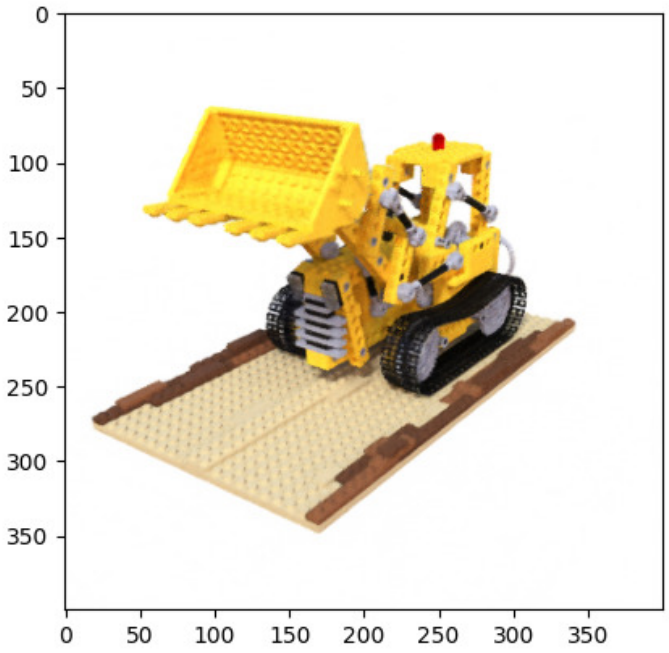

Fig. 7. NeRF Result for Lego data

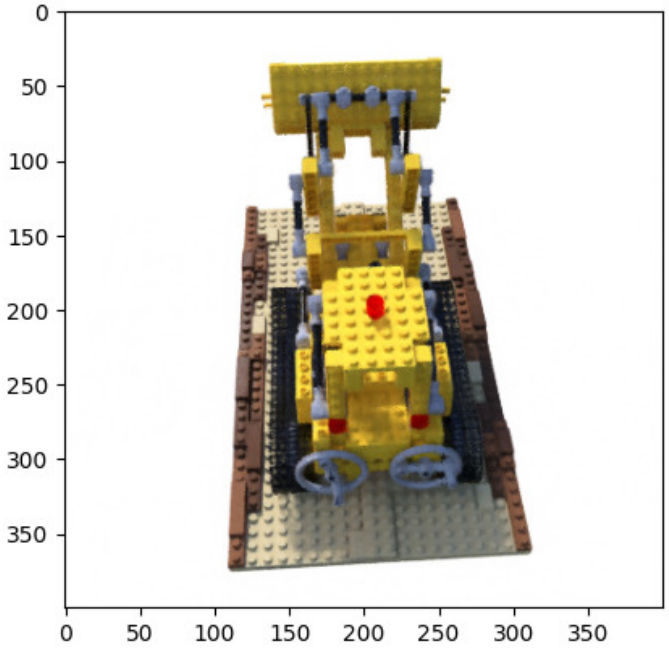

Fig. 8. NeRF Result for Lego data

II. 3D RECONSTRUCTION: CUSTOM MODEL

We created a custom data set and trained a NeRF model on the same to generate synthetic views for custom image data set. The entire process is as follows.

- Camera Calibration
- Data Generation
- Model training

A. Camera Calibration

We used Iphone 11 for capturing images of a randomly selected object to create custom data set. We performed camera calibration on the same using checkerboard images show in figure 9 and 10. We captured such 30 images for camera calibration. Further, we used functions CV2 library to find camera intrinsic matrix saved this data in .pnz file format.

Fig. 9. Checkerboard image used for Camera Calibration

Fig. 10. Checkerboard image used for Camera Calibration

B. Data generation

We chose a random object, a cardboard box to generate image data set. We captured around 200 images keeping all camera poses on spherical surface as shown in the figure. We performed this process deliberately, as we can choose near and far distances for spherical system, which are used further for ray sampling of NeRF. In order to obtain data of camera poses,

we used VisualSFM software which uses a computer vision technique called Structure from Motion (SfM). We performed all required steps finding matches, sparse reconstruction, dense reconstruction, bundle adjustment to get fine camera poses in .nvm file format. Further, we put camera intrinsic matrix and camera poses obtained together in required json format.

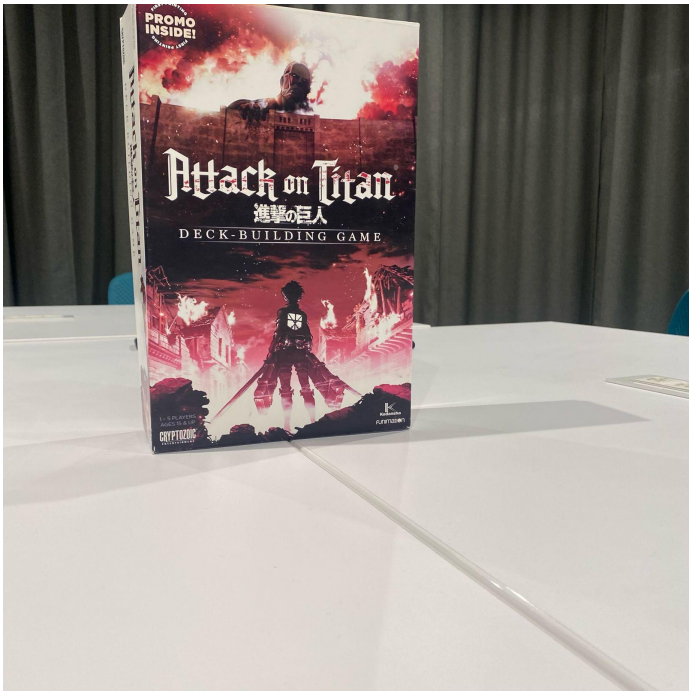

Fig. 11. Custom data image 1

Fig. 12. Custom data image 2

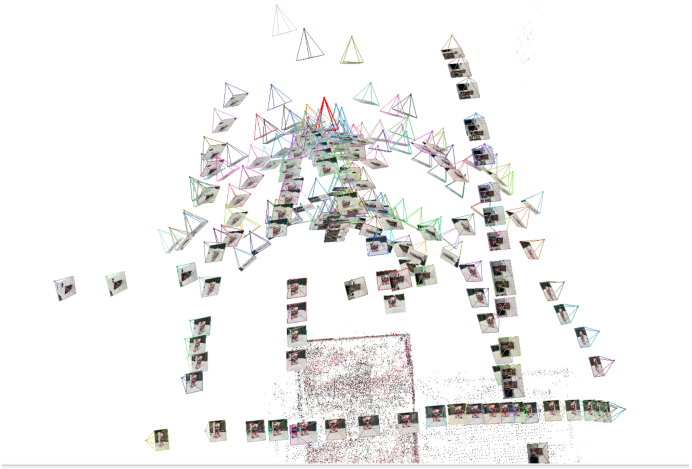

Fig. 13. Camera poses detected by VisualSFM software

C. Model training and results

We used the same model perform training on custom and satisfactory results are achieved which can be observed in figure 12, 13, 14.

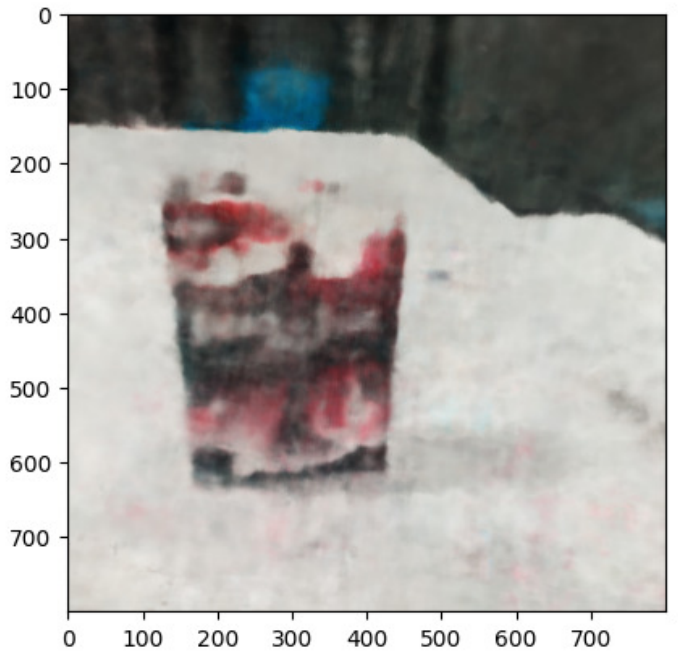

Fig. 14. Custom data result: NeRF

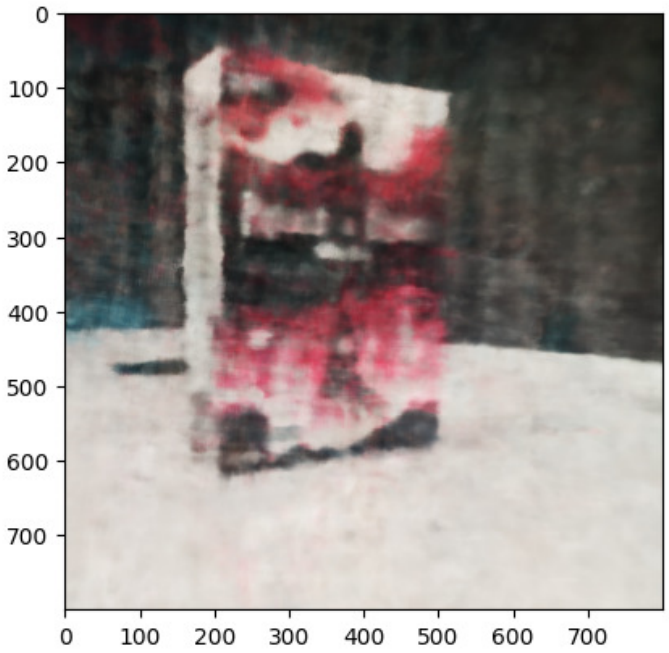

Fig. 15. Custom data results: NeRF

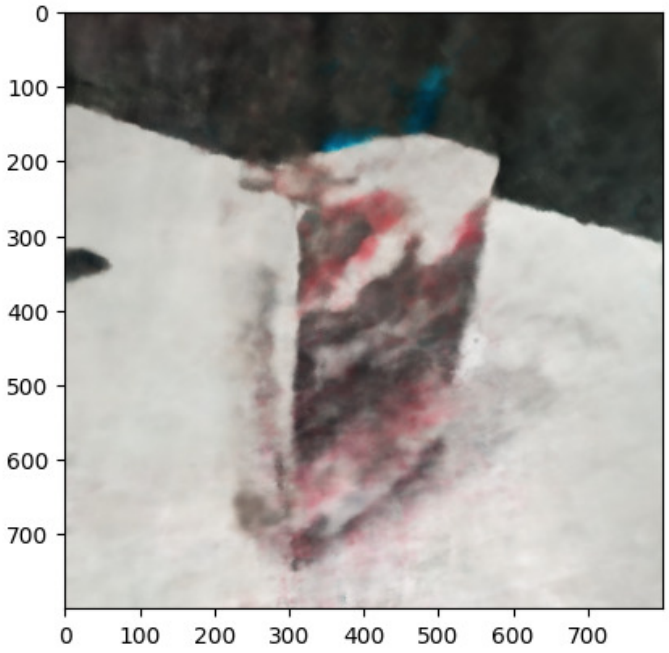

Fig. 16. Custom data results: NeRF

III. Conclusion

We performed synthetic view generation using NeRF with both synthetic and custom data. We found satisfactory results for both. This process includes key steps as pixel to ray generation, ray sampling, positional encoding and volume rendering.

References

[1] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ArXiv. /abs/2003.08934
[2] https://github.com/Prasannanatu
[3] ChatGPT. https://www.openai.com/chatgpt