RBE/CS 549 Computer Vision
Project 2 Phase 2
Structure from Motion

Puneet Shetty
MS in Robotics
Worcester Polytechnic Institute
Email: ppshetty@wpi.edu

Using 1 Late Day

Edwin Clement
MS in Robotics
Worcester Polytechnic Institute
Email: eclement@wpi.edu

Using 1 Late Day

***Abstract*—In this project, the Structure from Motion (SfM) method of computer vision techniques is implemented for simultaneous camera pose estimation and 3D scene reconstruction. SfM is able to construct point cloud-based 3D models that are similar to those made by LiDAR technology by analyzing a set of 2D photos. In order to determine the relative 3D poses of objects using stereo pairs, the method depends on the concepts of stereoscopic photogrammetry, triangulation, perspective-n-points, RANSAC, Epipolar Geometry, and Bundle adjustment. The application of SfM in 3D reconstruction is demonstrated in this study, along with its potential for use in conjunction with other deep learning techniques like Neural Radiance Fields (NeRF).**

Index Terms*— *RANSAC, Triangulation, Perspective-n-Points, Bundle Adjustment, Visibility Matrix, Structure from Motion

I. PHASE 2: DEEP LEARNING APPROACH NEURAL RADIANCE FIELD (NERF)

Using a sparse collection of input views, we will optimize a continuous volumetric scene function to synthesize unique views of complicated scenes by using Neural Radiance Fields (NeRF) in the Deep Learning section. A 5D continuous array serves as the NeRF's input. The first three components of the array describe the spatial location's 3D coordinates, while the latter two elements indicate the direction of the ray created by connecting a specific image pixel to the camera center. The RGB color (radiance field) of that particular pixel and the volume density at that spatial location are the NeRF's outputs. The steps for NeRF is as follows:

- Preprocessing Data
- Neural network
- Post Processing the Network output

A. Pre-Processing Data

1) Obtaining the Ray: The picture coordinates and each image's projection matrix are the given data. The first step involves utilizing a projection matrix to convert those image points to world points. Next, a ray is created that passes through both spots till. We will produce a specific quantity of rays from each image pixel by doing this.

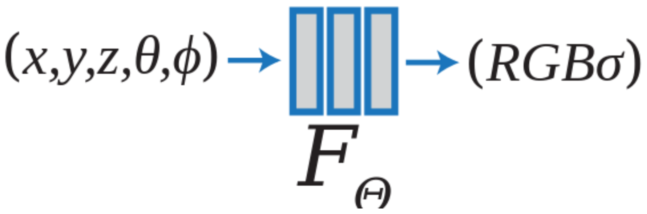


Fig. 1: Input and Output to nerf

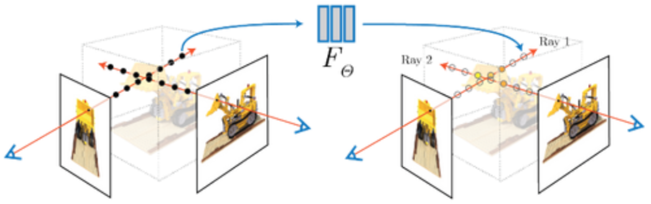


Fig. 2: Physical interpretation

The number of rays is a hyperparameter that can be adjusted; it is closely correlated with the output quality and computation time.

2) Sampling the Rays: Using the direction and origin of the earlier-obtained rays, we are attempting to sample the rays uniformly in this instance. Both uniform and non-uniform rays can be sampled. We sampled the rays linearly for the current circumstance, and the results are passable. To prevent overfitting and obtain a strong model for additional testing, we purposefully introduce noise during this procedure.
3) Encoding the Ray: The obtained sample points are simply encoded at higher frequencies in the sin and cos terms using positional encoding. When encoding with more frequencies than when not encoding, the result is superior. However, because the input multiplies as the number of frequencies increases, the calculating time will grow as the frequencies increase. In this case, the quantity of

higher dimnesion frequencies is a hyperparameter that can be adjusted to get better outcomes.

B. Neural Network

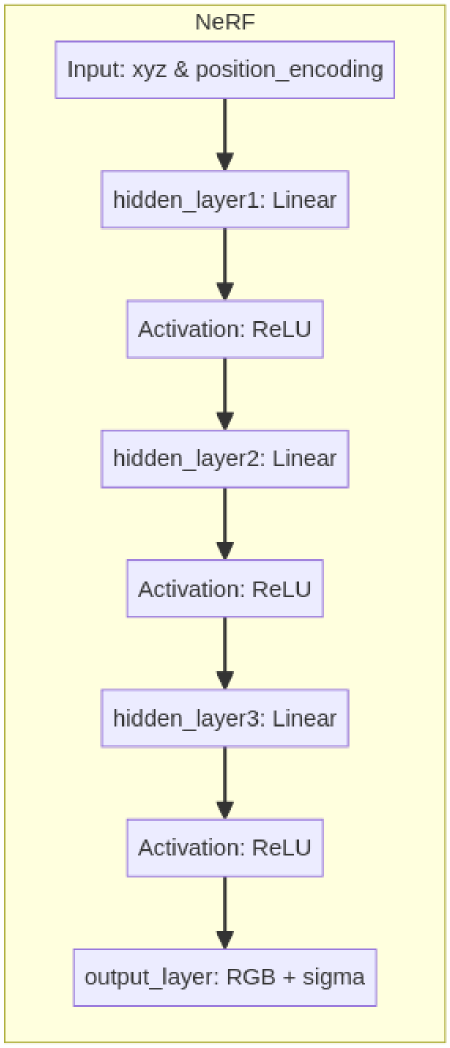


Fig. 3: Caption

C. Post-Processing

The network output consists of the volume density at a certain place and the RGB color value. Thus, to render the 3D scene, we employ these expected values. To determine the color of a specific location, the predictions made by the network are entered into the traditional volume rendering equation. To determine the final color in the image (radiance field), we first compute the transmittance up to a certain sample site using volume density. We then multiply that result by the predicted color at that location. For every pixel, we go through this process again.

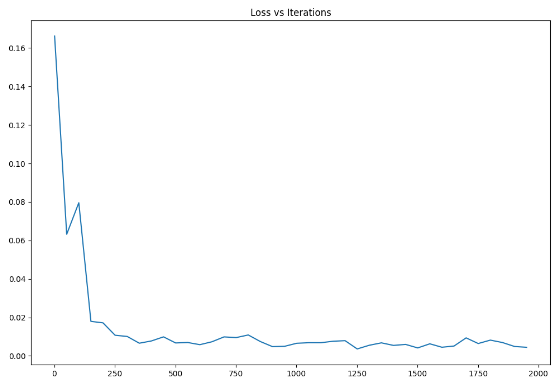
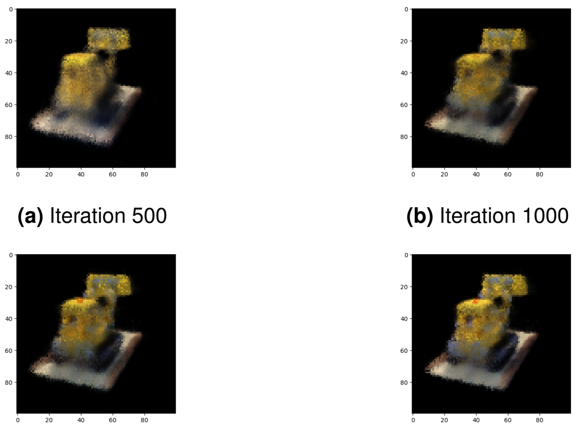


Fig. 4: Loss vs Iterations

D. Loss Function

After 3D volume rendering, we can simply compute the photometric loss between the expected color values and the actual image values after we have all the color (RGB) values.

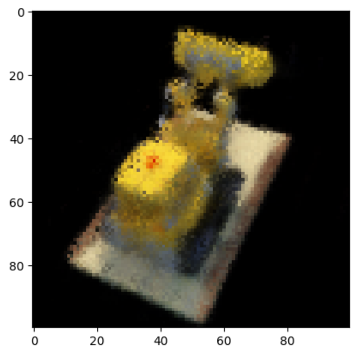


(a) Iteration 500 **(b)** Iteration 1000 **(c)** Iteration 1500 **(d)** Iteration 1900

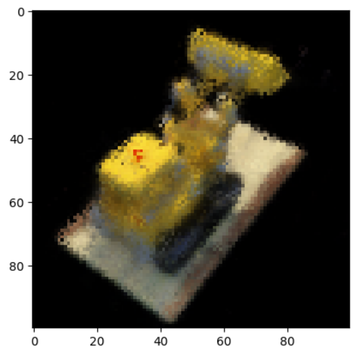
Fig. 5: Rendered view for same pose at various iterations

E. Conclusion

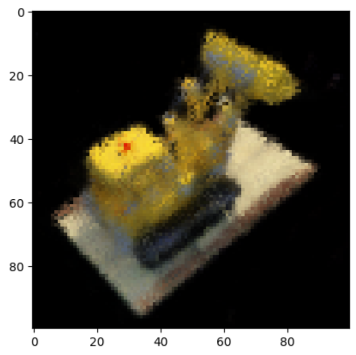
For the parameters mentioned above, there was no discernible change in the image output during the 500 epoch period. We will undoubtedly have better results if the model is trained for larger epochs and much better hyperparameters. If we try utilizing more samples per ray, the results will be considerably better because they will contain more features and data. Tensor forms are extremely important since one error might spell disaster.



(a) Test Pose 0



(b) Test Pose 3



(c) Test Pose 6

Fig. 6: Generated Images for test poses