Project 2 : SfM and NeRF

Abhijeet Sanjay Rathi
M.S. Robotics
Worcester Polytechnic Institute
Email: asrathi@wpi.edu

Anuj Jagetia
M.S. Robotics
Worcester Polytechnic Institute
Email: ajagetia@wpi.edu

***Abstract*—This projects showcases the application of computer vision techniques for the Structure from Motion (SfM) method, which allows for the simultaneous estimate of camera postures and the reconstruction of 3D scenes. SfM can create point cloud-based 3D models that are similar to those made by LiDAR technology by examining a number of 2D photos. To determine the relative 3D poses of objects using stereo pairs, the method depends on the concepts of stereoscopic photography and triangulation, PnPRANSAC, Epipolar Geometry, and Bundle adjustment. This work showcases the use of SfM in 3D reconstruction and shows how it may be used with other deep learning techniques like Neural Radiance Fields (NeRF).**

I. Phase I : Classical Approach to the SfM

A. Feature Matching

The initial stage of our pipeline is to obtain feature matches between every pair of monocular camera images. For this, we used SIFT algorithm, this process begins after the camera intrinsic matrix is determined through a calibration procedure, and then distortion is removed from the images. We have these stored in a text filed named matching, but it has some repeated points. Therefore we removed those points and updated texxt files with no repetition.

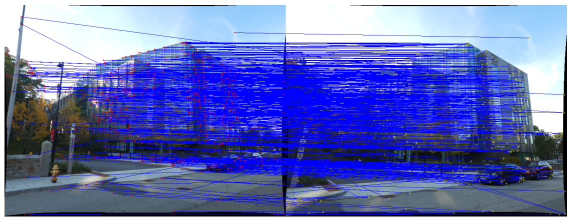


Fig. 1. *Feature Matching before RANSAC*

B. Estimating Fundamental Matrix

The fundamental matrix is denoted by F, a 3x3 matrix with rank of 2 which corresponds to set of points of same image with different views This is achieved by using the Epipolar constraint ($x_i^{'T} F x_i = 0$).

Singular value decomposition (SVD) is used to solve a system of linear equations represented by the Fundamental matrix that has nine unknowns. Once the system has been solved, the last singular value is set to zero, and the Fundamental matrix is recalculated in order to enforce the rank constraint we have with us.

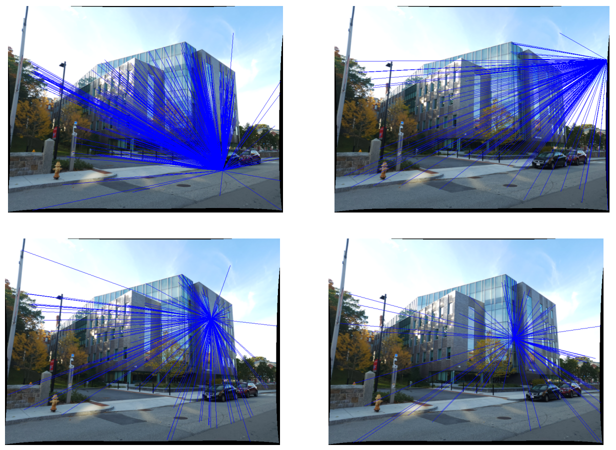


Fig. 2. Epipolar Lines on Images

C. RANSAC

As the the point correspondences are calculated through feature descriptors, there is some noise in the data and contain one to multiple outliers. To solve this issue of incorrect macthing, we employed the RANSAC algorithm to get a more accurate estimation of the matrix.This process is repeated until we get the best inliers. Thus, the F matrix with the greatest number of inliers is selected out of all the options.

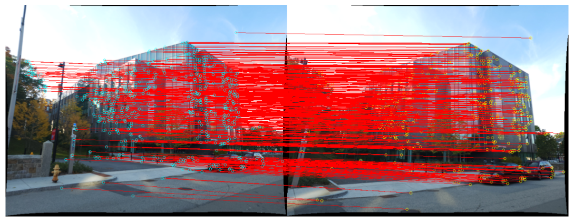


Fig. 3. *After applying RANSAC*

D. Estimating Camera Poses

Camera pose has 6 degrees of freedom 3 for rotation and 3 for translation. With the help of Essential Matrix we obtain the camera poses by decomposing the essential matrix.The camera pose can be expressed as $P = KR[I_{3\times 3} - C]$.

$$E = UDV_T$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This gives us four geometric poses which is represented as:
$C_1 = U(:,3) and R_1 = UWV^T$
$C_2 = -U(:,3) and R_2 = UWV^T$
$C_3 = U(:,3) and R_3 = UW^TV^T$
$C_4 = -U(:,3) and R_4 = UW^TV^T$

E. Triangulation Check for Cheirality Condition

We have two camera poses, (C_1, R_1) and (C_2, R_2), and correspondences, $x_1 \leftrightarrow x_2$. With the help of SVD, we triangulate the 2D points into 3D points. For that, we require one pose. Though all 4 poses are theoretically correct, we need one which is practically correct. To obtain this pose, we use the Cheirality constraint, to check the sign of the depth Z in the camera coordinate system with respect to the camera center. A 3D point X is considered to be in front of the camera if the following constraint holds: $r_3(X - C) > 0$, where r_3 is the third row of the rotation matrix. This process provides the best camera pose in the configuration (C, R, X).

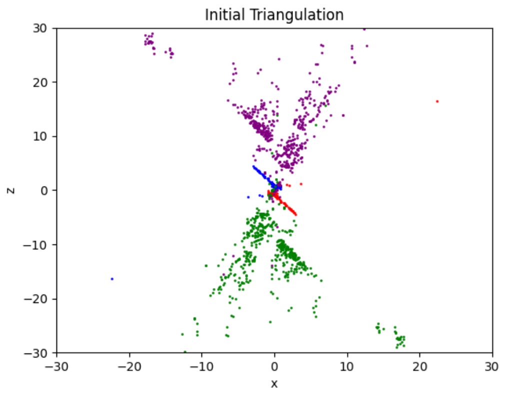


Fig. 4. *Linear Triangulation*

F. Non - Linear triangulation

We get the projection of 2D points in 3D which has low algebraic errors but it has some re-projection errors. For this we used Non-linear triangulation. To reduce the re-projection error, we therefore modify the locations of 3D points which were estimated by the linear triangulation using Scipy.optimize function. The error between measurement and re-projection error is given by:

$$\min_{x} \sum_{j=1,2} \left(u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}\tilde{X}} \right)^2 + \left(v^j - \frac{P_2^j T\tilde{X}}{P_3^{jT}\tilde{X}} \right)^2$$

Where, j is the index of each camera, $\tilde{X}$ is the homogeneous representation of X. P_i^T is each row of camera projection matrix.

G. Perspective-n-Points (PnP)

Now, since we have a set of n 3D points in the world, their 2D projections in the image, the intrinsic parameter and the 6 DOF camera pose. we can perform linear PnP on the features obtained from non-linear triangulation. The 2D points are normalized using K^{-1} x. For this we need 6 corresponding 2D and 3D points of the images and with that we can calculate the camera pose.

However, this camera pose is prone to error as there are outliers in the given set of point correspondences. To overcome this error, we again use RANSAC to make our camera pose more robust to outliers.

The problem after applying RANSAC is the same as in linear triangulation which did not account for geometric errors. Therefore we use Non-Linear PnP i.e., we refine the camera pose by minimizing the re-projection error which is calculated by :

$$\min_{C,q} \sum_{i=1,j} \left(u^j - \frac{P_1^{jT}\tilde{X}_j}{P_3^{jT}\tilde{X}_j} \right)^2 + \left(v^j - \frac{P_2^j T\tilde{X}_j}{P_3^{jT}\tilde{X}_j} \right)^2$$

where $\tilde{X}$ is the homogeneous representation of X. P_i^T is each row of camera projection matrix, P which is given by $P = KR[I_{3\times3} - C]$. This form used quarternions to optimize error and can thus be classified as non-linear PnP.

H. Bundle Adjustment and Visibility Matrix

For bundle adjustment we need a visibility matrix which is denoted by V, a I×J binary matrix which represents relationship between a camera and point, where V_{ij} is one if the j^{th} point is visible from the i^{th} camera.

Given initialized camera poses and 3D points, we need refine them by minimizing reprojection error, which is achieved by the bundle adjustment, it refines camera poses and 3D points simultaneously by minimizing the reprojection error over

$$C_{i_{i=1}}^I, q_{i_{i=1}}^I and X_{j_{j=1}}^J$$

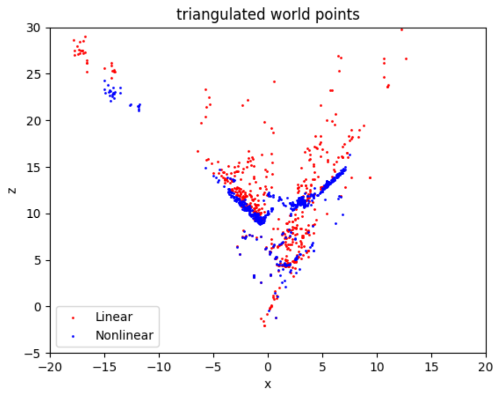
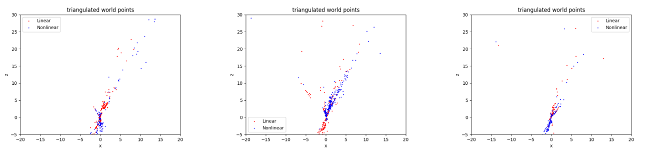


Fig. 5. *Linear and Non-Linear Triangulation Between Image 1 and 2*



(a) Between 1 and 3 (b) Between 1 and 4 (c) Between 1 and 5

Fig. 6. Linear and Non-Linear Triangulation

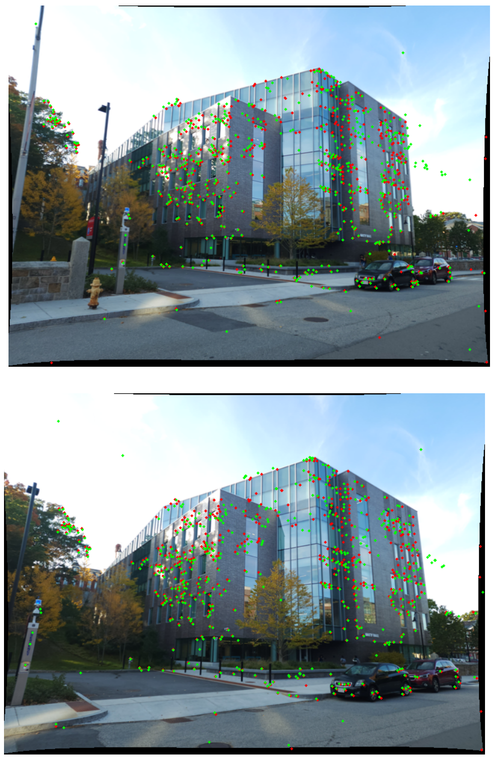


Fig. 7. Linear Re-Projection

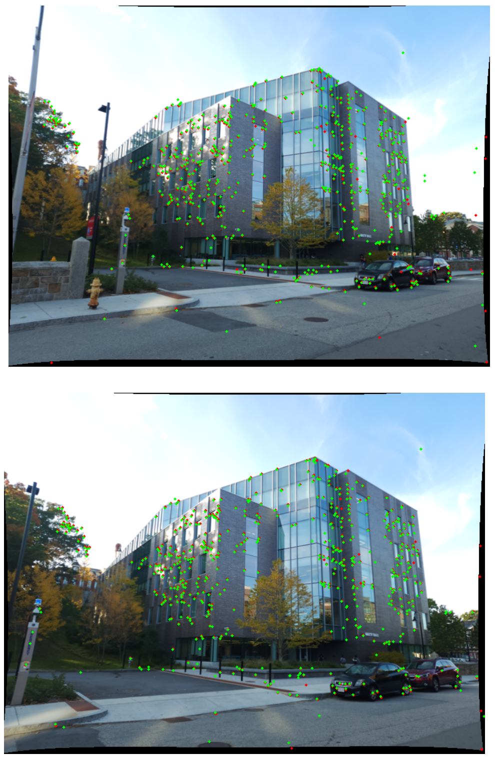


Fig. 8. Non-Linear Re-Projection

This minimization can be solved using a nonlinear optimization function *scipy.optimize.leastsq* it will be slow as the number of parameters are more. This method enhances accuracy, consistency, and reliability of the final 3D models by removing outliers from initial reconstructions through iterative optimization.

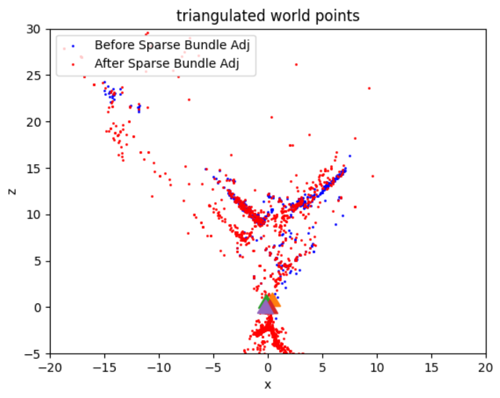


Fig. 9. *Before and After Bundle Adjustment on Images 1, 2, 3*

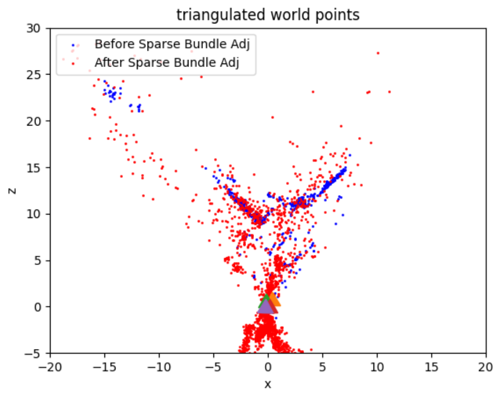


Fig. 10. *Before and After Bundle Adjustment*

I. Results and Conclusion

The Fundamental matrix, we got is:

$$F = \begin{bmatrix} -3.040358e-08 & 3.04345118e-05 & -1.283866e-02 \\ -3.292413e-05 & -2.843426e-06 & 3.441946e-02 \\ 1.471827e-02 & -3.275501e-02 & -9.986796e-01 \end{bmatrix}$$

The Essential matrix, we got is:

$$E = \begin{bmatrix} -0.0030538 & 0.59831704 & -0.11984659 \\ -0.64856261 & -0.04978099 & 0.74501285 \\ 0.16599481 & -0.78821197 & -0.02561337 \end{bmatrix}$$

Images	1-2	1-3	1-4	1-5
LT	862938705.88	64596.83	10319.07	747575.29
N-LT	286.31	7530.89	2796.20	4703.14
LPnP	NaN	6186.84	86960.44	29669.30
N-LPnP	NaN	11655.89	35346.66	90847.69

TABLE I
LINEAR AND NON-LINEAR ERRORS BETWEEN IMAGE 1 AND ALL THE REMAINING IMAGES

II. PHASE II : NEURAL RADIANCE FIELD (NERF)

A. Overview

In the second phase of our project, we implemented Neural Radiance Fields (NeRF), a technique designed to generate new, distinct views of intricate scenes by optimizing a continuous volumetric scene function with a sparse set of input views. NeRF takes a single continuous 5D coordinate as input, encompassing the viewing direction (θ, ϕ) and spatial position (x, y, z), and produces output comprising volume density and emitted radiance, tailored to the specific view. Furthermore, synthetic views were generated by projecting the output emitted color $(c = r, g, b)$ and volume density (σ) onto images using volume rendering methods, achieved through querying 5D coordinates along camera rays.

B. Method and Procedure

Obtaining the Ray: The image coordinates and each image's projection matrix are in the data we provided. First, we use a projection matrix to convert those image points to world points. Next, we make a ray that goes through both points. We will produce a specific number of rays from each image pixel by doing this. The number of rays is a hyperparameter that needs to be adjusted; it is correlated with both processing time and output quality.

Sampling the Rays: Using the direction and origin of the earlier-obtained rays, we are attempting to sample the ray using uniform sampling 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.

Encoding the Ray: The obtained sample points are simply encoded at higher frequencies in the sin and cos term by positional encoding. We obtain a better result when we encode with a bigger number of frequencies than when we don't. 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.

Volumetric Rendering : Volume density and RGB color values for a specific point in the 3D environment make up the output after the input has been processed via the network. After that, the volume rendering equation is employed to create the scene using these predictions. The final color for a given place in the scene is calculated using an equation that takes into consideration the expected color and density values. By comparing the projected color values with the actual image values, photometric loss was computed during 3D volume rendering and the acquisition of RGB color values.

$$\hat{C}_c(r) = \sum_{i=1}^{N_c} w_i c_i, \quad w_i = T_i(1 - \exp(-\sigma_i \delta_i)). \tag{1}$$

III. NETWORK

In our NeRF model all layers are normal fully-connected layers (Multi Layer Perceptron). Eight fully-connected ReLU layers, each with 256 channels, are used to process the positional encoding of the layer as the input location. we concatenate this input to the activation of the fifth layer via a skip link. And similarly an additional layer outputs the volume density and 256 dimensional feature vector, which is concatenated with positional encoding of input viewing direction.

Then it is processed by an additional fully-connected ReLU layer with 128 channels and finally , a layer with a sigmoid activation outputs the emitted RGB radiance at position x, as viewed by a ray with direction d.

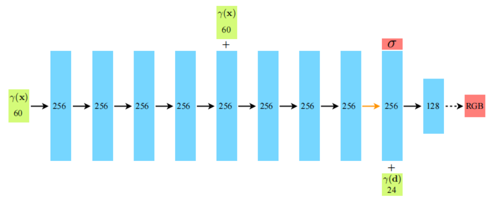


Fig. 11. *Network Architecture*

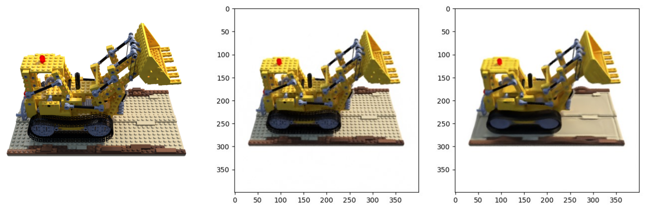
An architecture of our completely interconnected network is provided. Here, Green represents input vectors, blue represents intermediate hidden layers, red represents output vectors, and the number within each block denotes the vector's dimension. Black arrows show layers with ReLU activations, orange arrows show layers without activation, dashed black arrows show layers with sigmoid activation, and "+" indicates vector concatenation.

IV. RESULTS AND CONCLUSION

The significant increase in MSE loss and decrease in PSNR value observed during testing can be attributed to a notable change in the background. It's worth noting that the original test set featured a black background, whereas the predicted test set showcases a white background. However, despite this change, we can still rely on the SSIM value for comparison purposes. SSIM evaluates the similarity between grayscale images, and since both the original and predicted test sets are grayscale, SSIM remains a valid metric for assessment.

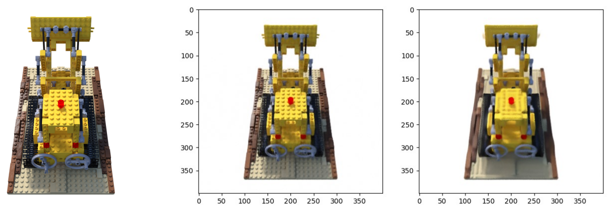
Furthermore, when examining the MSE loss and PSNR value for the test set, we notice that the model with positional encoding outperforms the other. This conclusion is drawn from the lower MSE loss and higher PSNR value achieved by the model with positional encoding. Hence, while the change in background may affect certain metrics, the superiority of the positional encoding model is evident from these specific evaluation criteria. From the results above we found that: A model is better when the PSNR is greater or the SSIM values are more On the other hand the model is not so good when the MSE loss is higher.

So, from all above values, we can say that NERF with positional encoding is far better than NERF without positional encoding.



(a) Ground Truth (b) With Positional Encoding (c) Without Positional Encoding

Fig. 12. *Lego Side View*



(a) Ground Truth (b) With Positional Encoding (c) Without Positional Encoding

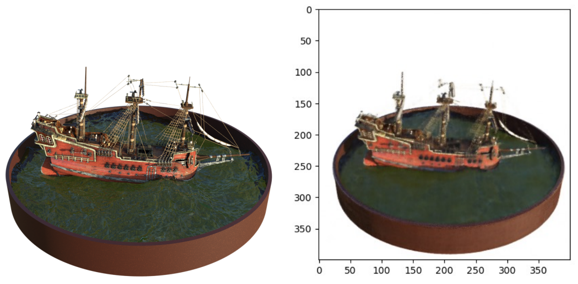
Fig. 13. *Lego Top View*

	Train	Validation	Test
MSE loss	2.505383089	0.000981321	23730053.39
PSNR	-3.908194267	30.17339134	-73.73959361
SSIM	0.855519765	0.85044057	0.167242077
Parameters	20		

TABLE II
Loss for Lego dataset with Positional Encoding

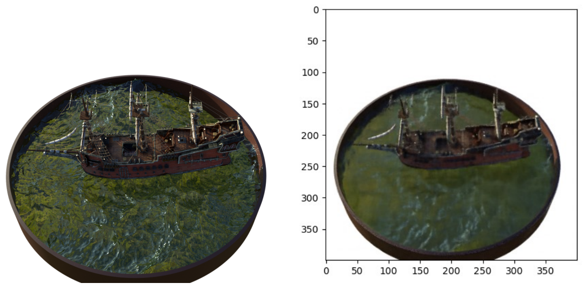
	Train	Validation	Test
MSE loss	6.499699064	0.002248945	22620379.96
PSNR	-8.071401348	26.5373485	-73.52949472
SSIM	0.843946934	0.838985419	0.196640152
Parameters	20		

TABLE III
Loss for Lego dataset without Positional Encoding



(a) Ground Truth (b) With Positional Encoding

Fig. 14. *Ship Side View*



(a) Ground Truth (b) With Positional Encoding

Fig. 15. *Ship Top View*

	Train	Validation	Test
MSE loss	3.706381693	0.001533915	28748085.43
PSNR	-5.636066691	28.20505308	-74.56816147
SSIM	0.938094233	0.936965059	0.171713094
Parameters	20		

TABLE IV
Loss for Ship Dataset with Positional Encoding

	Train	Validation	Test
MSE loss	911.3188056	0.290329762	28949009.26
PSNR	-29.59414334	5.373767514	-74.58214034
SSIM	0.002978119	0.003018735	0.178910808
Parameters	20		

TABLE V
Loss for Ship Dataset without Positional Encoding

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* 2020.
[2] https://github.com/yenchenlin/nerf-pytorch/tree/master
[3] https://www.udemy.com/course/neural-radiance-fields-nerf/?referralCode=DD33817D57404AF048DF