Project3: Buildings built in minutes - SfM and NeRF

*Note: 6 late days used

Thabsheer Jafer Machingal
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
tmachingal@wpi.edu

Krishna Madhurkar
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
ksmadhurkar@wpi.edu

I. Introduction

In this project, we move past 2D image scenes to work with a more 3D setting. For this we implement Structure from Motion (SfM) which is reconstructing a 3D scene and simultaneously obtaining the camera poses of a monocular camera with respect to the object of interest that is to be reconstructed. We create a sparse reconstruction from a set of images with different view points as though the camera is in motion. The pipeline to recreate this SfM algorithm can be condensed to Feature Matching and Outlier rejection using RANSAC, Estimating Fundamental Matrix, Estimating Essential Matrix from Fundamental Matrix, Estimate Camera Pose from Essential Matrix, Check for Cheirality Condition using Triangulation, Perspective-n-Point and Bundle Adjustment. We go about achieving this using a classical approach. For the second phase of this project, we implemented Deep learning based 3D constructional algorithm, which utilizes the idea of Neural Randiance fields.

II. Phase 1

For phase 1, the task is to reconstruct a 3D scene and simultaneously obtain the camera pose of a monocular camera. This procedure is known as Structure from Motion(SfM).

A. Dataset

We are given a set of five images of Unity Hall at WPI. The images were captured using Samsung S22 Ultra's primary camera at f/1.8 aperture, ISO 50 and 1/500 second shutter speed. The given images are already calibrated and Calibration matrix(K) is given.

$$K = \begin{bmatrix} 531.1221 & 0 & 407.1925 \\ 0 & 531.5417 & 313.30871 \\ 0 & 0 & 1 \end{bmatrix}$$

The feature matching on the images has already been performed and given to us as text files. All the possible matches between images have been identified and given in four different text files.

Fig. 1. Images on which sfm is performed

B. Traditional Method

The tradional method of sfm is performed to construct a 3D scene from the 2D images. The steps in brief are as follows.

- **Feature Matching**
- Outlier rejection using **RANSAC**
- Estimating **Fundamental Matrix**
- Estimating **Essential Matrix** from Fundamental matrix
- Estimate **Camera Pose** from Essential matrix
- check for **Cheirality Condition** using **Triangulation**
- **Perspective-n-point** projections
- **Bundle Adjustment**

1) Feature Matching: The matching coordinates were given in the dataset.Since there are are 5 images, there could be $5C_2$, 10, combinations of matches, four of which is demonstrated below in the figure.

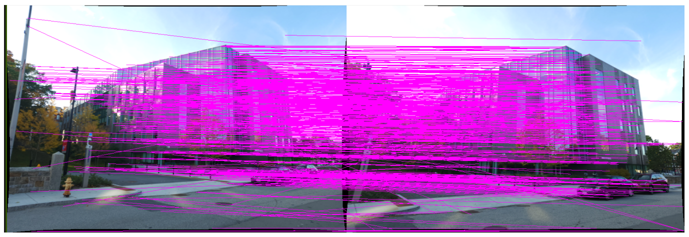

Fig. 2. Matches between image 1 and image 2

2) Outlier rejection using RANSAC: Outlier points were rejected using RANSAC and the Fundamental matrix with

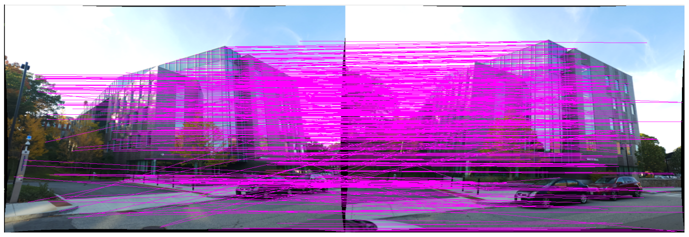
Fig. 3. Matches between image 2 and image 3

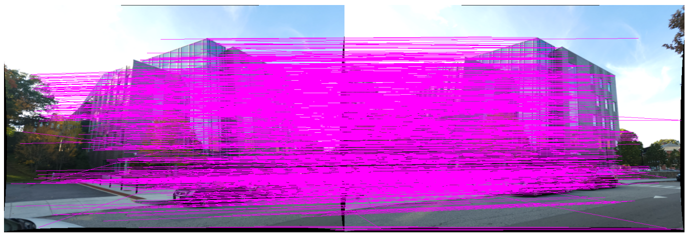
Fig. 4. Matches between image 3 and image 4

most number of inliers was chosen for the subsequent steps in the pipeline. A set of eight feature matches from a pair of images (same as in the above step,feature matching) was taken and if the value of $x_2^T F X_1$ is less than a threshold, then the match is classified as an inlier or else it is rejected as an outlier. The Fundamental matrix with maximum number of inlier is considered as the best and returned for the subsequent steps. The results from this step is given below.

3) Fundamental Matrix: Fundamental matrix is a 3X3 rank deficient matrix (rank =2), that relates corresponding set of points in images from different views. In this step eight point algorithm is implemented[1] The fundamental matrix, F obtained is:

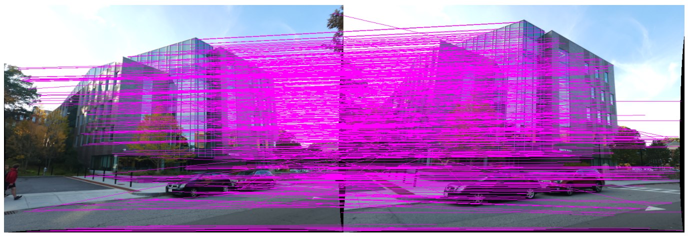
Fig. 5. Matches between image 4 and image 5

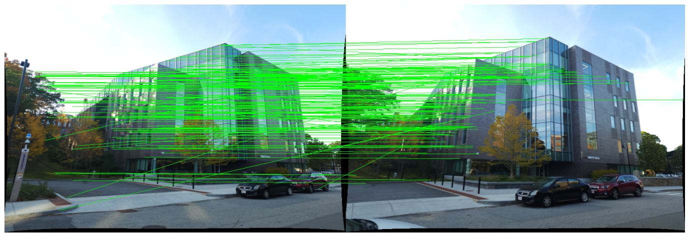
Fig. 7. Matches between image 2 and image 3 after outlier rejection

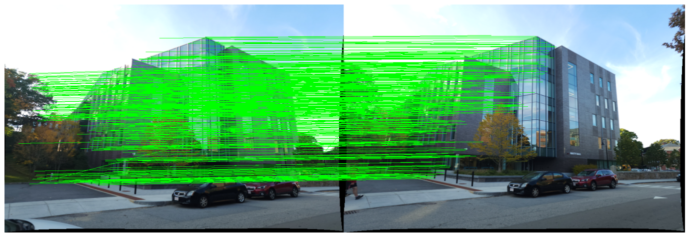
Fig. 8. Matches between image 3 and image 4 after outlier rejection

$$\text{F} = \begin{bmatrix} -0.27744998 & -0.22048552 & -0.06077586 \\ -0.09416008 & -0.06779006 & -0.07516563 \\ -0.06073737 & -0.17902026 & 1. \end{bmatrix}$$

4) Estimate essential matrix from fundamental matrix: Essential matrix can be used to find relative camera poses between two images, given that the fundemental matrix is along epipolar constraints. Essential matrix can be estimated using the following equation, E = $K^T F K$ Here, F is the fundamental matrix and K is the Calibration matrix. with the above F and K, we estimated E,

$$\text{E} = \begin{bmatrix} -0.13241201 & -0.65843784 & -0.4816113 \\ -0.66743431 & 0.60335075 & -0.17287408 \\ -0.47322142 & -0.17834297 & -0.47100571 \end{bmatrix}$$

C. Estimating Camera pose from Essential matrix

The camera pose consists of 6 DoF including rotation, R-P-Y and translation in x,y,z directions. The estimated set of Camera pose cofigurations (C_i, R_i), where i= 1,2,3,4 is given below.

$$R_1 = \begin{bmatrix} 0.35704605 & -0.59717274 & 0.71826376 \\ -0.58636767 & -0.74185514 & -0.32530585 \\ 0.72711145 & -0.30501748 & -0.61503924 \end{bmatrix}$$

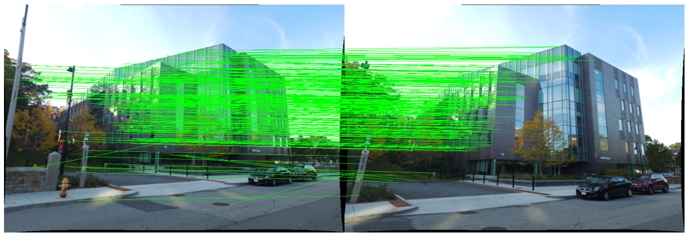
Fig. 6. Matches between image 1 and image 2 after outlier rejection

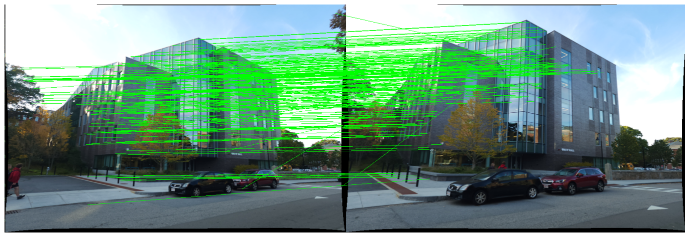
Fig. 9. Matches between image 4 and image 5 after outlier rejection

$$R_2 = \begin{bmatrix} 0.35704605 & -0.59717274 & 0.71826376 \\ -0.58636767 & -0.74185514 & -0.32530585 \\ 0.72711145 & -0.30501748 & -0.61503924 \end{bmatrix}$$

$$R_3 = \begin{bmatrix} -0.9870711 & 0.13205744 & 0.09083767 \\ 0.1378957 & 0.4107712 & 0.90125013 \\ 0.08170329 & 0.90212408 & -0.42367053 \end{bmatrix}$$

$$R_4 = \begin{bmatrix} -0.9870711 & 0.13205744 & 0.09083767 \\ 0.1378957 & 0.4107712 & 0.90125013 \\ 0.08170329 & 0.90212408 & -0.42367053 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 0.56300731 & 0.40076661 & -0.72277859 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} -0.56300731 & -0.40076661 & 0.72277859 \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 0.56300731 & 0.40076661 & -0.72277859 \end{bmatrix}$$

$$C_4 = \begin{bmatrix} -0.56300731 & -0.40076661 & 0.72277859 \end{bmatrix}$$

1) Triangulation check for Cheirality condition: The task is to refine the R and C (camera pose configurations) using linear triangulation using linear least squares. Given two camera poses, let's say, (C_1, R_1) and (C_2, R_2) and correspondence x1 and x2, we find linearly triangulated points in 3D X1 and X2 for all the correspondence of given two images. **Cheirality condition**, is that the reconstructed point must be in front of the camera. If triangulated point,X satisfies the condition R[3,:]X-C ¿ 0, then the point is infront ogf the camera. Remember that X is in 2D (x,y,1). This step return (C,R,X) configuration that produces maximum number of points satisfying cheirality condition.

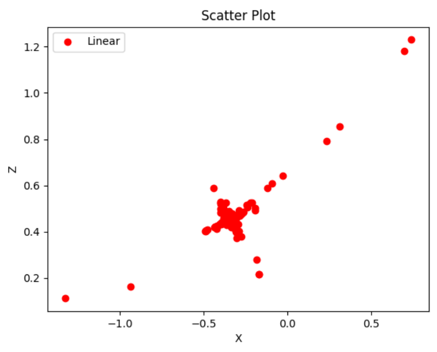

Fig. 10. 2D scatter plot of linearly triangulated points

2) Non-Linear Triangulation: Now that we have linearly triangulated points, we can find 3D points of the same, which minimizes the reprojection error.

3) Perspective-n-Points: With the estimated 3D points in the world coordinates and their 2D projections in the image and intrinsic parameters, we can find 6 DoF camera pose using linear least squares. This problem is known as PnP. The steps until now was performed on image 1 and image 2. From this point we perform the tasks on the next three images. Pnp algorithm require atleast three correspondence.

4) PnP RANSAC: The RANSAC algorithm is performed again to remove outliers in the reslut of PnP From linear PnP, we obtain the camera pose and find the perspective

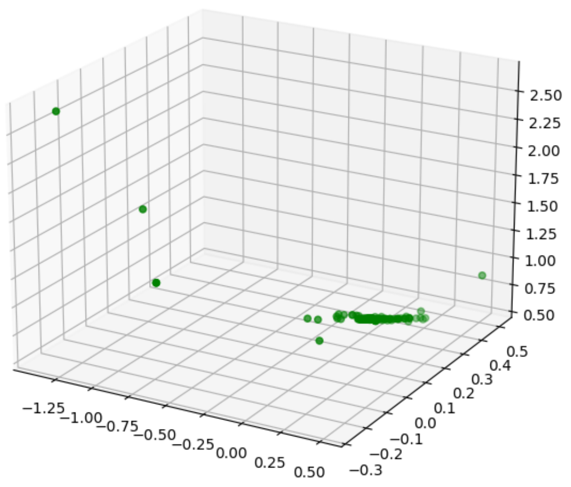

Fig. 11. 3D scatter plot of points after non-linear triangulation for image pairs 1 and 2

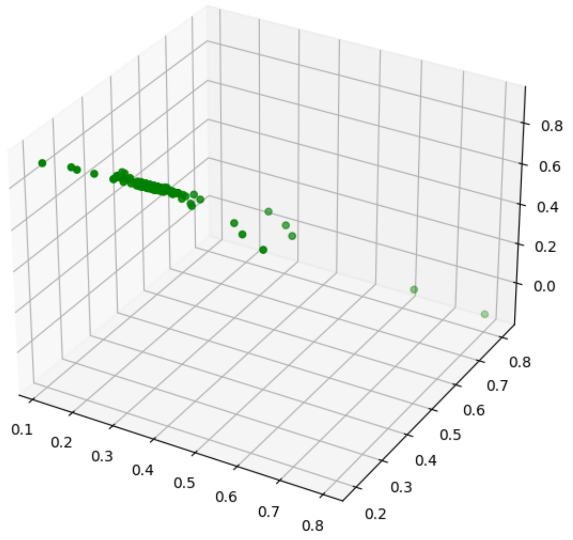

Fig. 12. 3D scatter plot of points after non-linear triangulation for image pairs 2 and 3

projection,P, which is used to find the reprojection error. We reject outlier points that has reprojection error greater than a threshold(epsilon).

Reprojection Error

$$error = \left(u - \frac{P_1^T X_{est}}{P_3^T X_{est}}\right)^2 + \left(v - \frac{P_2^T X_{est}}{P_3^T X_{est}}\right)^2 \tag{1}$$

if error is less that ϵ, the configuration is an inlier else an outlier.

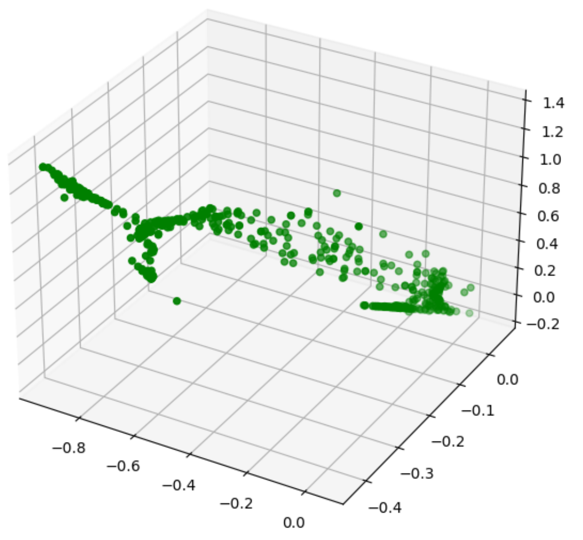

Fig. 13. 3D scatter plot of points after non-linear triangulation for image pairs 3 and 4

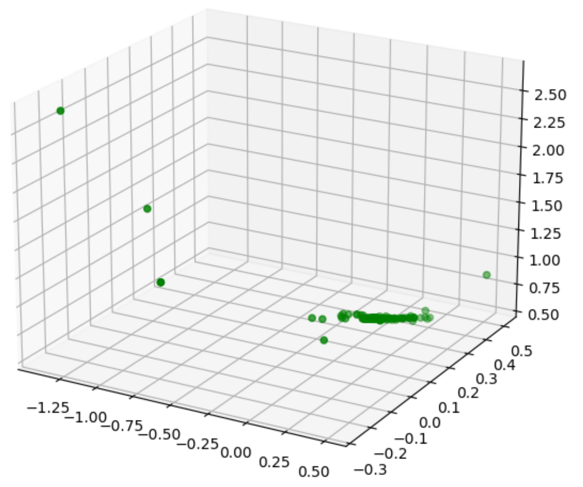

Fig. 14. 3D scatter plot of points after non-linear triangulation for image pairs 4 and 5

5) Non Linear PnP: We now refine the camera pose found using linear PnP. We do that by minimizing the reprojection error found in the above step for each Camera poses.

$$min_{C,R}(error) \tag{2}$$

We find the initial guess of the solution (C_0, R_0) (from Linear PnP), which is then used to minimize the reprojection error., this is implemented using the *optimize.leastsq* of **scipy**.

D. Bundle Adjustment

Now that we have Camera poses and points in 3D with respect to world coordinates, we refine the poses and 3D Points together. The bundle adjustment refines the camera poses and 3D points simulataneously by minimizing the reprojection error(equation 1) over R,R,X for all camera poses and 3D points.

Visibility matrix

Visibility matrix, V_{ij} is a binary matrix (with elements 0 and 1), which tells us if j^{th} point is visible from i^{th} camera. Given the visibility matrix and reprojection error(equation 1), the optimization problem is as follows.

$$min_{C,R,X} \sum_{i=1}^{I} \sum_{j=1}^{J} V_{ij}(error) \tag{3}$$

This minimization problem is solved using *optimize.leastsq* of **scipy**. The output form Bundle Adjustment function is the new camera poses R,C, and 3D points X. Using these results we can render the 3D view of the given images.

III. Phase 2

For Phase 2 we implemented NeRF: Neural Radiance Fields for view synthesis from this paper[2].

A. Dataset

The dataset has training, validation and testing images. 100 training images and annotations are given, which contains the focal length of the camera while the image being captured and poses of each image.

B. Pipeline

The input to the network is a 5D coordinates (spatial location (x,y,z) and viewing direction θ, ψ). This 5D coordinates are sampled along the ray for each pixels. The view direction are encoded for obtaining high frequency inputs. This has helped the network to learn better.The number of chnnels to the input netwrok is 3+3x2xf , where f is the number of positional encoding function.

1) Positional Encoding: We used a log scale(base = 2) alternating sine and cosine functions as the encoder. This function encodes the view directions and add the encoded channels to the input to the network. If the number of encoding function is f, then the number of inputs is 3 directions(x,y,z) and 2 view directions(θ, ψ) with 3 channels times f.

2) Neural network: A multi-layer perceptron (MLP) is used as the approximator function. There are multiple ways in which we can define our network. The model used is demonstrated in figure 1. The network we used has 5 layers and and 256 hidden layers with few changes from what is described in the original implementation. The changes to the network was made to better utilize the available computation resources and optimized for better results. We have tested for quite a few variants of the network and observed the changes in performance with respect to features like number of parameters and number of layers. The network has 5 layers

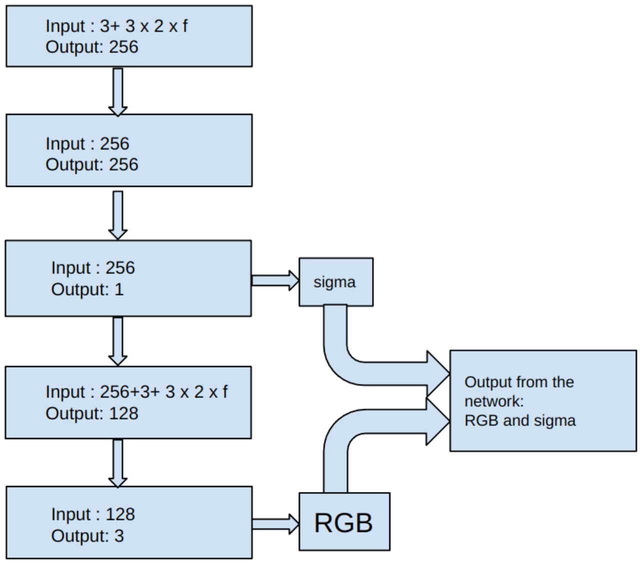

Fig. 15. flowchart of the neural network

and sigma value, which is the density is taken from the output of the third layer. And the RGB values are rendered fronm the output of the final layer. A flow chart of the network is given below(figure 15).

C. *Stratified sampling*

Each pixels project a ray and in this step, we sampled the ray between the nearest and farthest point. A number of samples is defined, which determine the number of points to be samples on the ray. Stratified sampling method samples the points randomly[3].

1) Volume rendering: Volumetric rendering is a classical rendering method where the idea is that intensity at a pixel is the collective dum of intensities at each sample posiions, this is done channel wise and the out put is rendered as a single RGB image.

2) Training and hyper-parameters used: We have tried and learned different parameters that works well for this particular network. IN the end the following hyperparameters gave the optimal results.
Learning rate = 0.002
Number of iterations = 100,000
The training time was around 10 hours with one Gpu, Nvidia GeForce RTX 3090 Ti, 24GB of vram
The network parameters was divided to speed up the training process. With the above model, parameters and configuration, GPU was optimized to run at 100% memory. The results from the network was observed at an interval of 1000 iterations and recorded on a single test image, which was tested on an image from validation set.

D. *Results*

The output from the NeRF algorithm (Figure 17) for the test set is recorded and converted to GIF of the collection and is presented with this document. The output we obtained in this project is of comparable standards to the current and original implementation of NeRF.

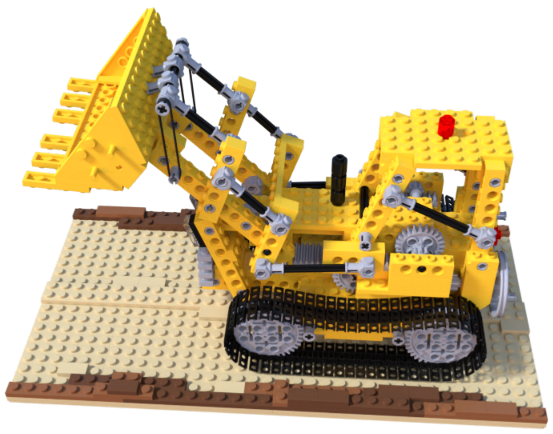

Fig. 16. Original test image

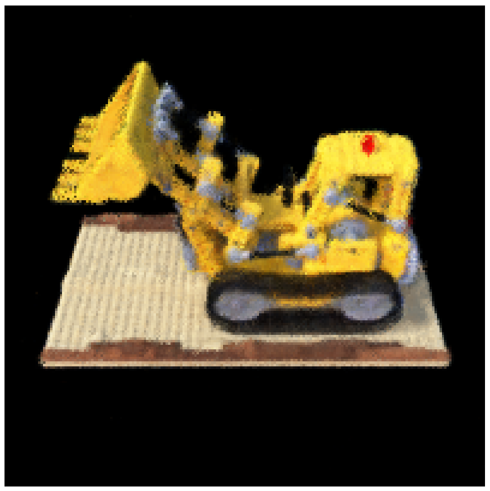

Fig. 17. output image from NeRF

References

[1] "Eight-point algorithm," Wikipedia. Apr. 10, 2022. Accessed: Nov. 12, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Eight-point$_algorithm&oldid = 1081924376$

[2] 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," Commun. ACM, vol. 65, no. 1, pp. 99–106, Jan. 2022, doi: 10.1145/3503250.

[3] [1] "Stratified sampling," Wikipedia. Nov. 06, 2022. Accessed: Nov. 12, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stratified$_samplingoldid = 1120420596$