RBE/CS549: Project 2
Buildings built in minutes - SfM and NeRF

Using 4 late days

Prasanna Natu
M.S. Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pvnatu@wpi.edu

Peter Dentch
M.S. Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pdentch@wpi.edu

***Abstract*—This project presents an implementation of computer vision techniques for the reconstruction of 3D scenes and simultaneous estimation of camera poses, utilizing the method known as Structure from Motion (SfM). By analyzing a series of 2D images, SfM can generate point cloud-based 3D models that are comparable to those produced by LiDAR technology. The technique relies on the principles of stereoscopic photogrammetry and triangulation, PnPRANSAC, Epipolar Geometry and Bundle adjustement to calculate the relative 3D poses of objects from stereo pairs. This project highlights the application of SfM in 3D reconstruction and its potential for use in conjunction with other deep learning approaches, such as Neural Radiance Fields (NeRF).**

***Index Terms*—SfM, NeRF, RANSAC**

I. Phase I: Classical Approach

In this method, we implemented the classical methods to reconstruct a 3-Dimensional scene from only images and the intrinsic parameters of the camera which captured them. The algorithm employed for this task is SfM or Structure from Motion, the steps of which are as follows:

- Filter matched image feature points using the fundamental matrix
- Estimate the essential matrix
- Triangulate 3D points
- Perform perspective-n-points
- Perform bundle adjustment

An overview of this method which was implemented is outlined in the sections below:

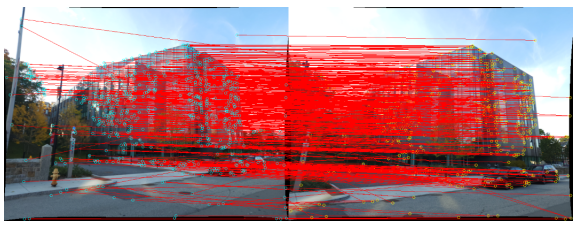

Fig. 1. Unfiltered matches

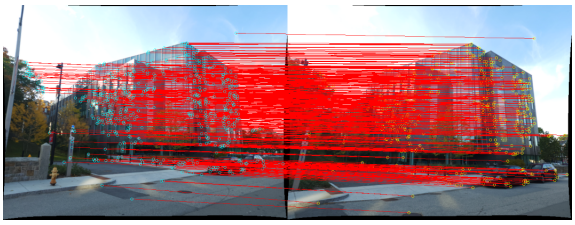

Fig. 2. Filtered matches using RANSAC

A. Feature filtering with fundamental matrix

A corner detection algorithm is used to identify unique features on the images and the pixel coordinates are provided in a text file. However, some matched points between images are incorrect and must be filtered out. The method employed is using the Random Sample Consensus algorithm or RANSAC, which samples 8 random point pairs used to compute the fundamental matrix. The fundamental matrix correspondence condition is checked with all homogenized point pairs and the fundamental matrix which produces the least residual error is returned as the best estimate by RANSAC.

$$F = \begin{bmatrix} 8.77155311e^{-08} & -2.86590912e^{-05} & 1.21254387e^{-02} \\ 3.10681286e^{-05} & 2.99652600e^{-06} & -3.37076509e^{-02} \\ -1.40218315e^{-02} & 3.19921809e^{-02} & 9.98747543e^{-01} \end{bmatrix}$$

B. Estimation of essential matrix

Once the fundamental matrix has been computed, the essential matrix can be calculated from the fundamental matrix and camera intrinsic parameters. This will allow for the second image view camera pose to be found with respect to the first camera which is set to the origin. The resulting four computed poses are all valid mathematically but the real physical pose must be distinguished between them which is done through cheirality checking.

$$E = \begin{bmatrix} 0.00437685 & -0.57506889 & 0.11698349 \\ 0.62750143 & 0.05251681 & -0.76334246 \\ -0.16528565 & 0.80519329 & 0.0269838 \end{bmatrix}$$

C. Triangulation and cheirality checking

The matched points can be triangulated using each of the four camera poses to get four sets of 3D world points. The pose with most points in front of the camera is selected as the proper orientation and used as the final camera position and orientation. The final selected 3D points are then used as initial conditions to a nonlinear optimizer with treats the error of the reprojected 3D points from the image as noise. The final estimation of the world points are much less sparse than the original estimate and clearly show the corner of the building in the images.

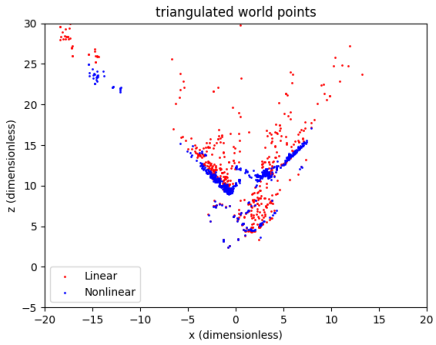

Fig. 3. Linear and nonlinear triangulation

D. Perspective-n-points

The perspective-n-points or PnP problem is to register the poses of new cameras from additional images of the scene which contain sufficient world points seen by the original two cameras. It is assumed that outlier matching points exist from the world to the new images, thus RANSAC is used once more, checking the reprojection error between the known world points and new image points to returning the pose which has the least error for all points. This is also used as an initial condition to a nonlinear estimator to further refine the new estimated camera pose. It should be noted that for this process the camera orientations are represented as quaternions to provide better convergence of the estimator given the mathematical properties of quaternions not have discontinuity which exists in the Euler angle representation.

E. Bundle adjustment

Once all five camera poses of the provided images are registered in the scene, bundle adjustment will further refine the camera pose estimates and the 3D world points simultaneously by once again minimizing reprojection error. This makes use of the visibility matrix, a matrix of Booleans defining if a point can be seen by a camera in its image plane or not. This matrix can be large for many cameras and points but is generally of diagonal form and can thus be optimally iterated over for the necessary computations. After bundle adjustment the SfM is complete.

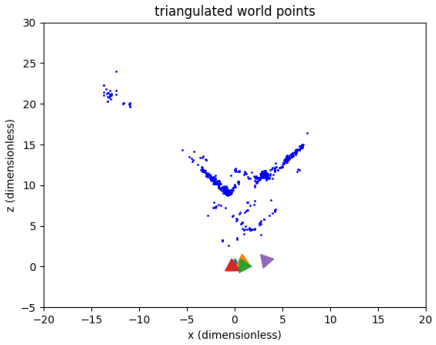

Fig. 4. Registered camera poses with nonlinear PnP

II. Deep Learning Approach
Neural Radiance Field(NeRF)

The neural radiance field is a Deep Learning apporach which is a combinations of the Neural Networks and Computer Graphics. The goals is to get a 3D represntation of the provided images of a particular object/scene using neural network and Computer Graphics techniques. The 3D coordinates **x,Y,Z** along with the viewing angles
textbftheta and
textbfphi forms the 5D input to the network fot training purpose.The network after processing gives an output as an **RGB** values which is the further processed to get the RGB map for volume rendering. The steps/Procedure for NeRF is as follows:

- Preprocessing Data
- Neural network
- Post Processing the Network output for better results.

A. Preprocessing Data

1) Getting the Ray: The given data is the image coordinates along with the Projection matrix for each image. The first thing is we convert those image points to World points using projection matrix and then create a ray passing through the both the points till. We will do this for each image pixel and

generate a certain amount of ray from it. No of rays is a hyperparameter to be tunned and is directly proportional to the quality of output and computing time

2) Sampling the Rays: Here we are trying to sample the ray using uniform sampling using the direction and origin of rays we got earlier. We can samlpe the rays both uniformyl and non-uniformly. For current scenario we have sampled the rays linearly and the output is decent. In this process we deliberately add noise to avoid overfitting and get a robust model for further testing.

3) Encoding the Ray: The positional encoding issimply encoding the found sample points in the sin and cos term at higher frequencies. Encoding in the higher number of frequencies gives us a better output as compared to wihtout encoding. However the higher the number of frequencies higher will the computing time as the input multiply with increase in number of frequencies. Here the number of higher dimnesion frequencies is an hyper parameter which can tunned to get better result.

B. Neural Network

After the Preprocessing is done we have simple Multi-layer preceptron as the Neural network, here we are free to use any no of layers and width of the network. For this project I have used a simple Multi-layer Perceptron with 3 basic layers.

Near point = 2
Far point = 6
Number of samples on each ray = 32
Learning rate =
Number of higher frequencies for positional encoding = 6
Image size = 100 × 100
Epochs/Iterations = 1000
Mini batch size = 1000
No of rays = 200 and 5000

Fig. 6. Network

C. Post Processing

The main goal of pos processing is to till filter and obtain a significant and legible data from the Netwokr output so that it can be used to Render the volume of the image. we used the output of RGB output of the Network to get ht RGB color and the volume density at a desired location. The output is given to the classical Rendering code which in turn generate a volume rendering and gives a color at particular point. The formula for volume rednering are as below

sampling parameter to we are doing uniform ded noise so that the belter results could be

$$C\sum_{i=1}^{N} T_i \alpha_i c_i$$

tter results and render ng positional encoding. as been used in NeRF. encoding. All the input e inpat it to the nerivork

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

Using volume density, we firs calculate the transmittance till Network which is just a giving us the output as

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

we will be using tha calcuated alpha and transimttance to get the volume density of the image and get the desired output for the same.

Fig. 7. Output 5000 rays

D. loss calculation

As we get the **RGB map** after post processing the data from the network we utilise that value to get the loss between the input and the ouptut image the loss MSE loss

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

This loss is then optmisied to get better results and run for no of interation

Fig. 8. Output- 200 rays

III. Result and Conclusion

- Phase 1: The 3D points triangulated and refined using the five images provided generated a mesh of points which resembled the part of the building facing the cameras.
- Phase 2: In the phase after 500 epochs the image output were not significantly change for the above menitioned parameters. If the model is trained for much better hyperparameters and larger epochs we will surely get better results.instead of using the 100 *100 image if we try using the original image it can give much better output as they will contain more data and more features. The Tensor shapes are of critical importance as one mistake may lead to failure.

References

[1] URL: https://en.wikipedia.org/wiki/Eight-pointalgorithm

[2] URL: https://colab.research.google.com/drive/1rO8xo0TemN67d4mTpakrKrLp03b9bgCX.

[3] DeTone, D., Malisiewicz, T., & Rabinovich, A. (2016, June 13). Deep image homography estimation. Retrieved February 4, 2023, from https://arxiv.org/abs/1606.03798

[4] Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., & Rother, C. (2018, March 21). DSAC - differentiable RANSAC for camera localization. Retrieved February 4, 2023, from https://arxiv.org/abs/1611.05705

[5] URL:https://colab.research.google.com/github/bmild/nerf/blob/master/tinynerf.ipynbscrollTo=R1avtwVoAQTu.

[6] Li, H., & Hartley, R. (2006). Five-point motion estimation made easy. RSISE, The Australian National University. Canberra Research Labs, National ICT Australia.