Project 2 : Structure from Motion (SfM)

Abhijeet Sanjay Rathi
M.S. Robotics
Worcester Polytechnic Institute
Email: asrathi@wpi.edu
Using 1 Late Day

Anuj Jagetia
M.S. Robotics
Worcester Polytechnic Institute
Email: ajagetia@wpi.edu
Using 1 Late Day

***Abstract*—This project seeks to utilize traditional computer vision techniques to reconstruct 3D scenes and estimate camera poses via a method known as Structure from Motion (SfM). SfM leverages a sequence of 2D images to reconstruct the 3D structure of a scene, creating point cloud-based 3D models.**

I. Introduction

This project focuses on constructing a 3D scene from 2D images and simultaneously obtaining the camera poses of a monocular camera. This procedure is called Structure of Motion (SfM). We create an entire rigid structure from a set of images with different view points (or equivalently a camera in motion). 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 steps employed for this task are: estimating fundamental matrix, essential matrix, triangulating 3D points, performing perspective-n-points and then bundle adjustment

II. 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.

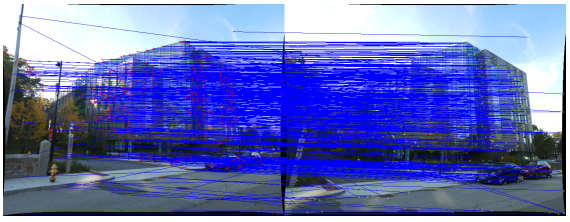


Fig. 1. *Feature Matching before RANSAC*

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 $(x_i^{'T} F x_i = 0)$.

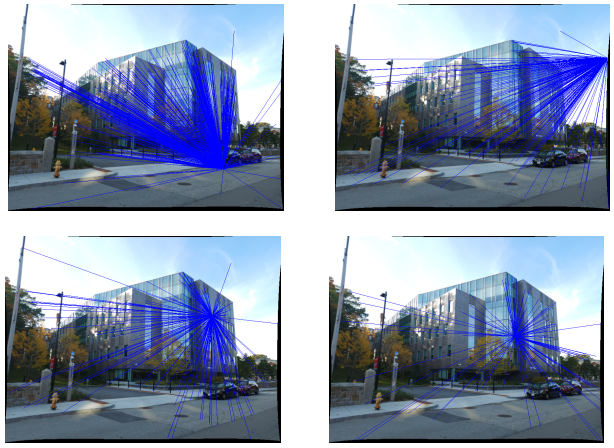


Fig. 2. Epipolar Lines on Images

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.

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.

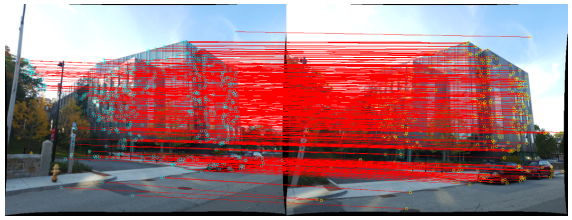


Fig. 3. *After applying RANSAC*

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 $P = KR[I_{3\times3} - C]$.

$$E = UDV_T$$

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

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).

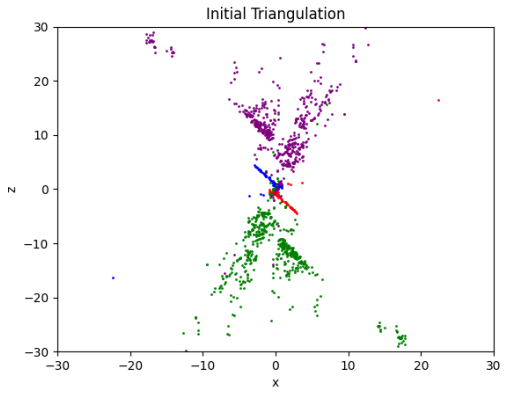


Fig. 4. *Linear Triangulation*

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.

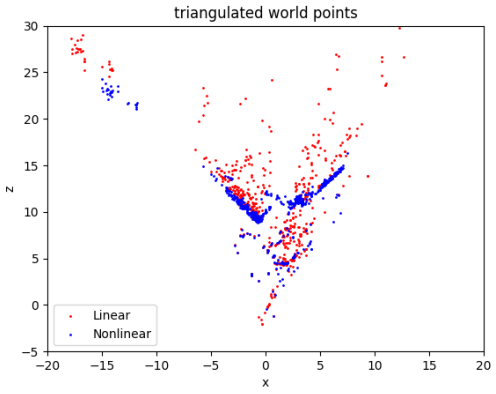


Fig. 5. *Linear and Non-Linear Triangulation Between Image 1 and 2*

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.

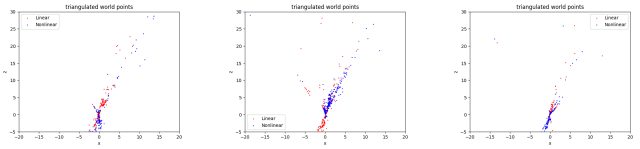
H. Bundle Adjustment and Visibility Matrix

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

$$C^I_{i_{i=1}}, q^I_{i_{i=1}} and X^J_{j_{j=1}}$$

This minimization can be solved using a nonlinear optimization function *scipy.optimize.leastsq* it will be slow as the number of



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

Fig. 6. Linear and Non-Linear Triangulation

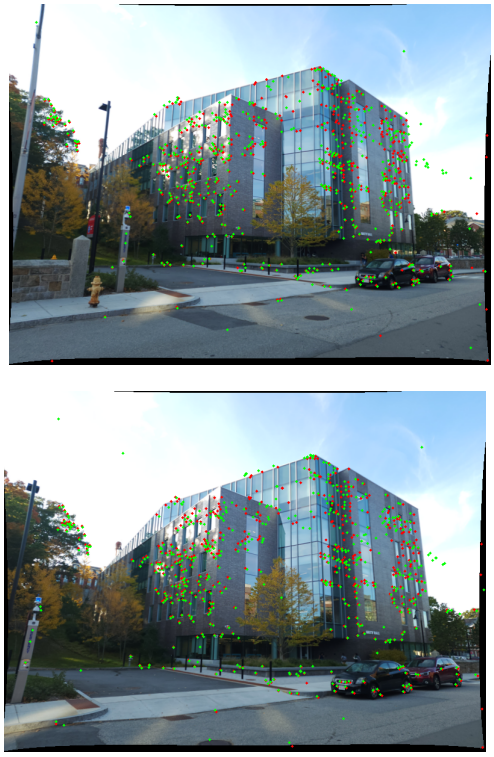


Fig. 7. Linear Re-Projection

parameters are more. This method enhances accuracy, consistency, and reliability of the final 3D models by removing outliers from initial reconstructions through iterative optimization.

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:

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

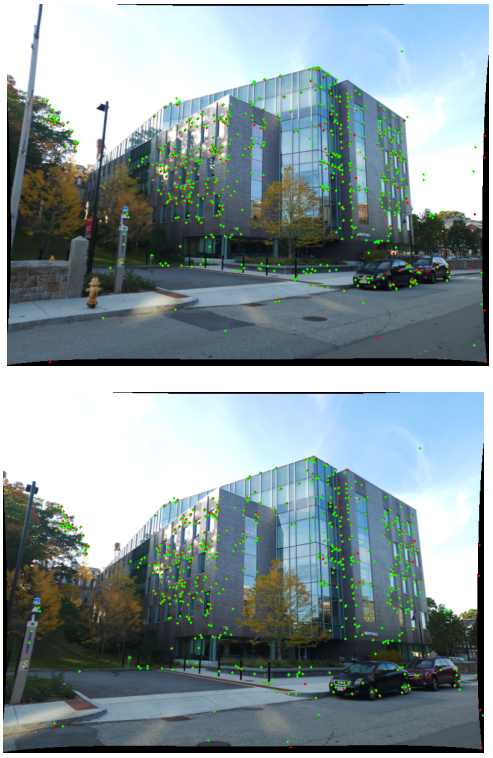


Fig. 8. Non-Linear Re-Projection

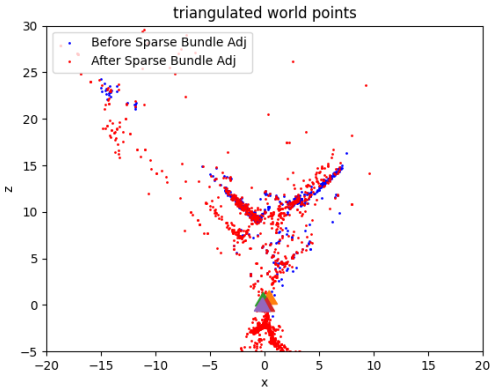


Fig. 9. *Before and After Bundle Adjustment on Images 1, 2, 3*

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

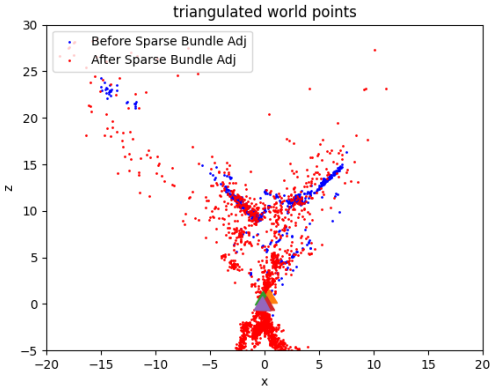


Fig. 10. *Before and After Bundle Adjustment*