Structure from Motion (SfM)

Ajith Kumar Jayamoorthy
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, U.S.A.
ajayamoorthy@wpi.edu

Shiva Kumar Tekumatla
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, U.S.A.
stekumatla@wpi.edu

***Abstract*—This document consists of the project implementation of 3D reconstruction of a scene from multiple 2D images. The traditional Structure from Motion (SfM) method is implemented. The results and observations have been recorded in this document.**

I. Introduction

The main objective of the project is to generate a 3D model of a particular scene from multiple 2D images captured from different perspectives. Two methods have been studied and implemented in this project. In the first method, called the Structure from Motion (SfM), 6 images have been given to start with, a text file describing the 2D image point correspondences between all possible image pairs and the calibration matrix of the camera used for capturing the images.

II. Traditional Approach - Structure from Motion (SfM)

The traditional method consists of following steps:

1) Feature Matching and Outlier rejection using RANSAC
2) Estimating Fundamental Matrix (F)
3) Estimating Essential Matrix (E)
4) Estimate Camera Pose from Essential Matrix
5) Check for Cheirality Condition using Triangulation
6) Perspective-n-Point
7) Bundle Adjustment

A. Feature Matching and Outlier rejection using RANSAC

Initially, there are 6 images provided which are from different camera perspectives of the same building with fixed camera parameters. Each pair of images have lot of pixels common. The figure 1 has one of the sample input images.

Fig. 1. Camera Perspective 1 (Unity Building, WPI) [1]

Feature matching are given in text files and they are predetermined by feature descriptors. The given data tends to be noisy and contains outliers and therefore we will remove the outliers using RANSAC. Given a pair of images long with the matches, we use RANSAC with the 8-point algorithm to estimate the fundamental matrix between them.

B. Estimating Fundamental Matrix (F)

In this method we randomly sample 8 points and then we estimate the fundamental matrix F. Then we count the number of points that satisfy the epipolar constraint (x'Fx $\approx$ 0) and finally select the fundamental matrix that results in the largest number of inlier correspondences. The Fundamental matrix estimated using the above method is as follows:

$$F = \begin{bmatrix} -6.4121e-07 & -2.4365e-05 & 1.3846e- \\ 2.8041e-05 & 1.6928e-06 & -3.5683e-02 \\ -1.5631e-02 & 3.4325e-02 & 1.0 \end{bmatrix}$$

C. Estimating Essential Matrix (E)

We estimate the essential matrix by from the fundamental matrix by using the equation E= K^TFK, where K is the camera calibration matrix or camera intrinsic matrix. As in the case of F matrix computation, the singular values of E are not necessarily (1,1,0) due to the noise in K. This has been corrected by reconstructing it with (1,1,0) singular values by using SVD, as suggested in the reference website [1]. The Essential Matrix is evaluated and is given below:

$$E = \begin{bmatrix} -0.01404208 & -0.45120026 & 0.21074795 \\ 0.51477257 & 0.0415464 & -0.82473738 \\ -0.24096739 & 0.8640367 & 0.01652605 \end{bmatrix}$$

D. Estimate Camera Pose from Essential Matrix

The first camera is considered to be the origin of the global/world coordinate system. The second camera data is taken into consideration and based on the essential matrix, four different configurations of the second camera from the first camera are calculated in the form of translation vector (C) and Rotation matrix (R). They are evaluated using singular value decomposition as follows:

$$\mathbf{E} = UDV^T$$

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

The four configurations are calculated as follows:

$$\begin{aligned} C1 &= U(:,3); & R1 &= UWV^T \\ C2 &= -U(:,3); & R2 &= UWV^T \\ C3 &= U(:,3); & R3 &= UW^TV^T \\ C4 &= -U(:,3); & R4 &= UW^TV^T \end{aligned}$$

E. *Cheirality Condition using Triangulation*

*1) **Linear Triangulation**:* Using the 4 estimated camera poses, we found out the 3D world points corresponding to the matches between the two images. This was done by performing SVD on a system of linear equations to minimize the L1 distance between a projected 3D point and the actual 2D image point.

*2) **Cheirality condition checking**:* After that using the generated 3D work points and the camera poses, the cheirality check was performed to get the best camera pose by using the condition $r_3(\mathbf{X} - \mathbf{C}) > 0$, where r_3 is the third row of the Rotation matrix (Z-axis of the camera), X is the 3D global co-ordinates and C is the translation vector of the camera centre in global co-ordinates. The configuration which satisfies the Linear triangulation condition is visualized as shown in figure 3

*3) **Non-Linear Triangulation**:* Given two camera poses and linearly triangulated points X, we can optimize the re-projection error of the 3D points in the image plane. We accomplish this task by using the optimize function from the scipy library given as follows **scipy.optimize.least_squares**

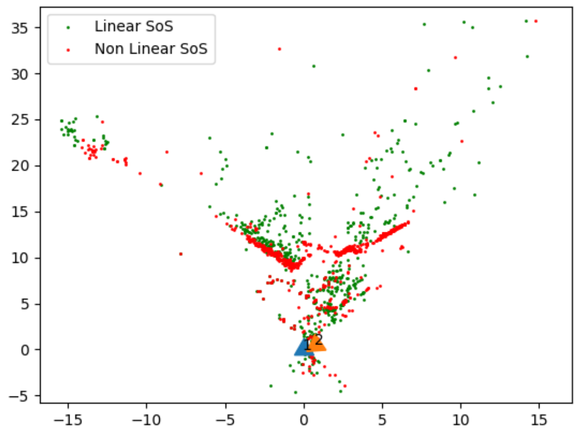

Fig. 2. Linear and Non-Linear Triangulation for first two camera poses [1]

F. *Perspective-n-Points (PnP)*

Since we have a set of n 3D points in the world, their 2D projections in the image and the intrinsic parameter; the 6 DOF camera pose can be estimated using linear least squares. This fundamental problem, in general is known as Perspective-*n*-Point (P*n*P). For a solution to exist, the number sets of 3D points used should be greater than or equal to three i.e. $n \geq 3$

*1) **Linear Camera Pose Estimation (Linear PnP**:* Given the correspondence between the image points(x) and the world points(X) along with the intrinsic parameters of the camera (K), we first calculated the inverse of the intrinsic parameter matrix to normalise the image points. Then we solve the system of equations by using the Ax = 0 to to get linear least squares solution with SVD.

*2) **PnP RANSAC**:* Linear PnP 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.

*3) **Non-Linear PnP**:* In the Non-Linear PnP we were trying to minimize the projection error by optimizing the rotation matrix and translation vector using the correspondences given before. The optimization is done same as in case of Non-Linear triangulation using the scipy.optimize.least_squares function. The re-projections of the triangulated 3D world points after are as shown in figure 4

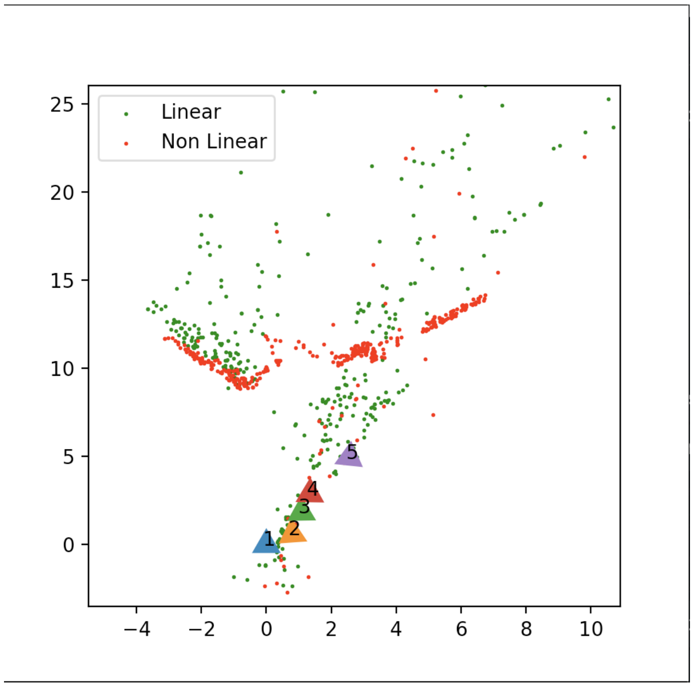

Fig. 3. PnP results before Linear and Non-Linear Triangulation [1]

G. *Bundle Adjustment*

1) Visibility Matrix: We create a matrix based on the camera poses and the visibility of the given world point from that particular camera pose. The rows of the matrix represent the camera pose index and the columns represent the index of the real world co-ordinates. For every world point index, 0 if marked if there is no visibility from a particular camera index and 1 is assigned if the point is visible from that particular camera index.

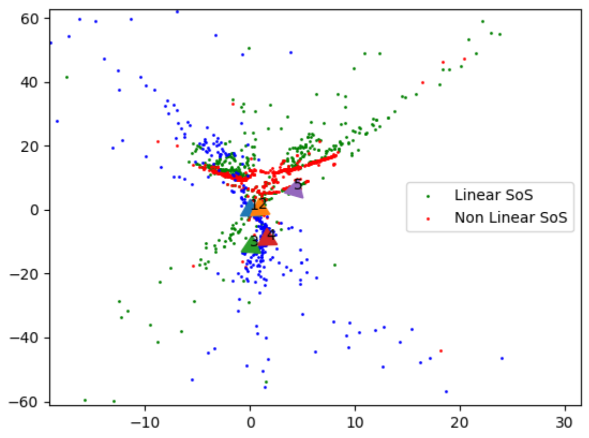

Fig. 4. Camera Pose estimation after PnP optimization and Triangulation [1]

2) Bundle Adjustment Implementation: After the Non-Linear PnP, the new Rotation matrix and translation vector obtained are further optimized. The input the bundle adjustment function are the World coordinates (X), the new camera co-ordinate x, the camera parameter, rotation matrix (R), translation vector (C) and the visibility matrix (V). Further optimization of the rotation matrix (R) and translation vector (C) are performed by using the least_squares method from the scipy.optimize library. The error function considered is same as the re-projection error used in the previous optimization problems. Based on this error the optimization process is done and new values of R and C are obtained. Along with this parameters, new global co-ordinates set (X) is also passed as the output.

REFERENCES

[1] https://rbe549.github.io/fall2022/proj/p3/
[2] https://drive.google.com/drive/folders/1lrDkQanWtTznf48FCaW5lX9ToRdNDF1a