Project III: Structure From Motion Using 2 late days

Naitri Rajyaguru
email: nrajyagu@umd.edu

Angelos Mavrogiannis
email: angelosm@umd.edu

***Abstract*—This project focuses on the problem of Structure from Motion. Structure from Motion is a technique of estimating three dimensional structures from 2D images taken from different positions.In this project, 6 images taken from different view points are provided and matching features for each image is given. The problem is solved using classical methods of computer vision.**

I. Phase I: Traditional Approach

The algorithm to address the problem of Structure From Motion is given below:

- Feature Matching and Outlier Rejection using RANSAC
- Estimation of Fundamental Matrix
- Estimation of Essential Matrix from Fundamental Matrix
- Estimation of Camera Pose from Essential Matrix
- Perform Triangulation using Cheirality condition
- Perform Non-Linear Triangulation for refining 3D points
- Estimation of pose using PnP
- Bundle Adjustment for pose and 3D points refinement.

A. Feature Matching and Outlier Rejection using RANSAC

The feature point correspondences of all images are provided and outlier rejection was accomplished by computing error of estimated fundamental matrix. We use 8 point algorithm to estimate fundamental matrix. The algorithm is stated as follows:
1) From the matches that are computed, 8 pairs are taken to estimate initial fundamental matrix.
2) Before computing initial F matrix, we normalize image coordinates and compute is done using equations below:

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix}$$

$$\begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

3)All of the pairs from both the images are tested against this fundamental matrix $x' * F * x^T$ and error was computed.
4)If the error is below some threshold value then that point pair will be inlier and corresponding Fundamental matrix will be considered as good.
5) The above steps were repeated for 1000 iterations and outliers were discarded.
The fundamental matrix computed is given below:

$$\begin{bmatrix} [-2.12959983\text{e}-07 & -9.82079979\text{e}-06 & 2.75965943\text{e}-03] \\ [1.20285739\text{e}-05 & -6.86549435\text{e}-07 & -3.90275056\text{e}-03] \\ [-4.46212938\text{e}-03 & 1.89118235\text{e}-03 & 9.99976832\text{e}-01] \end{bmatrix}$$

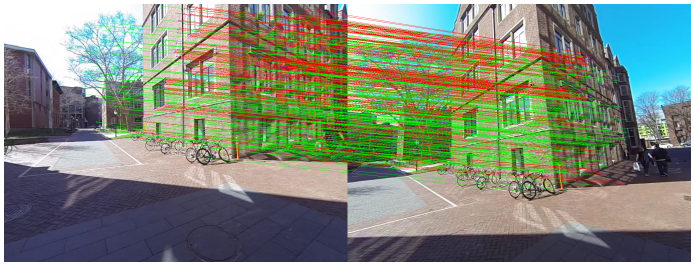

Fig. 1: Feature Matching with outliers and inliers

B. Estimation of Essential Matrix

If we have calibrated cameras, we can get essential matrix from which we can easily estimate pose. The Essential Matrix can be estimated by:

$$E = K^T F K$$

The estimated Essential matrix is:

$$\begin{bmatrix}[-0.01082897 & -0.72431028 & -0.2659097] \\ [0.88083521 & -0.05612574 & 0.44911254] \\ [0.15204238 & -0.61718941 & -0.14066586]]\end{bmatrix}$$

C. Estimation of Camera Pose

The Essential matrix is composed of Rotation and Translation of camera. If we decompose Essential matrix using singular value decomposition then we get four camera pose configurations.

$$C_1 = U(:,3) \text{ and } R_1 = UWV^T$$
$$C_2 = -U(:,3) \text{ and } R_2 = UWV^T$$
$$C_3 = U(:,3) \text{ and } R_3 = UW^TV^T$$
$$C_4 = -U(:,3) \text{ and } R_4 = UW^TV^T$$

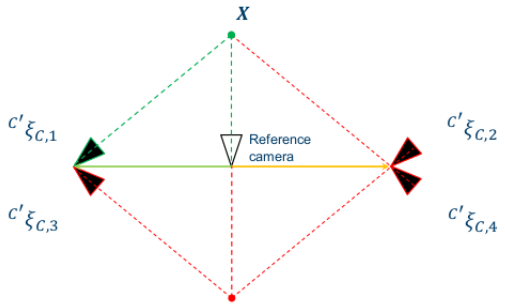

Fig. 2: Four ambiguous poses

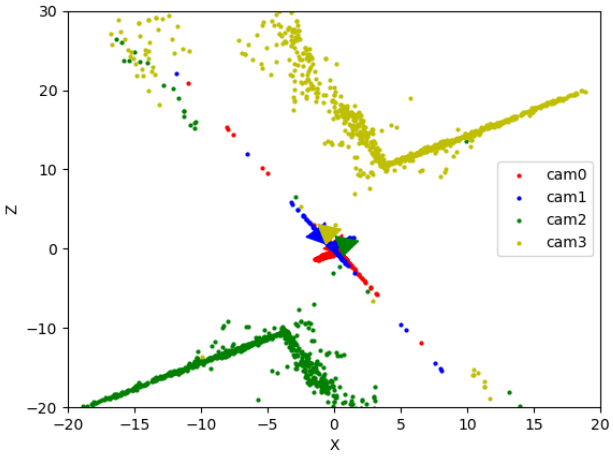

Fig. 3: Four ambiguous poses with 3D points

D. Triangulation Check for Cheirality Condition

To recover the correct pose of the camera among the four possible conditions,we try to construct 3D point corresponding to the feature match among two images. For the four possibilities, 3D points for each of them are computed and the point set which verifies the Cheirality condition, is the correct pose. The concept behind this is that depth of 3D points must be positive which is called depth positivity constraint.

$$r_3(\mathbf{X} - \mathbf{C}) > 0 \text{and} z > 0$$

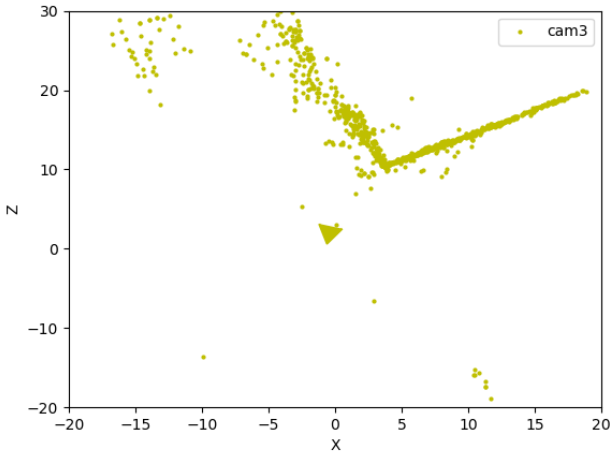

Fig. 4: Selected pose and points

E. Non-Linear Triangulation

We have points form Linear triangulation and also have extracted best pose and corresponding points. To refine these points, we perform non linear optimization. 3D points estimated from the linear triangualtion and use reprojection error to optimize using *scipy* library. As seen in the figure, the reprojection loss difference before and after non-linear optimization is 30.47202072528099 and 30.197910855974285 respectively.

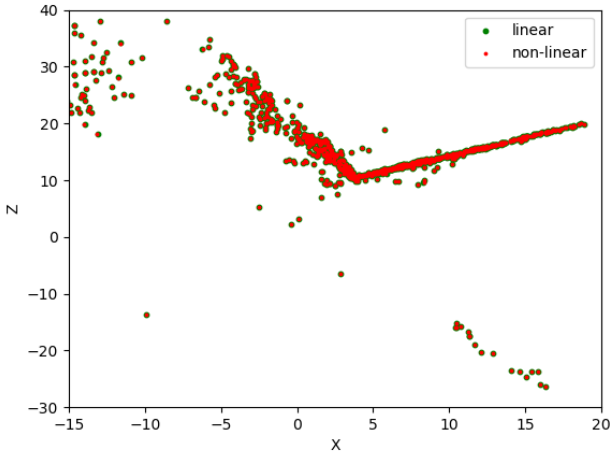

Fig. 5: Linear v/s Non Linear Triangulation

F. Linear Camera Pose Estimation

Having acquired a set of n 3D points X from the scene with known 2D correspondences x in the images and an intrinsic camera matrix K, we are now able to estimate the 6-DOF camera pose using linear least squares. The camera pose can be expressed in the form of three parameters for the rotation (roll, pitch, yaw) and a 3-dimensional vector that represents the translation of the camera with respect to the world. This general pose estimation problem is known as Perspective-n-Point (PnP) and can be solved if $n \geq 3$. We begin by normalizing the image coordinates by multiplying them with the inverse of the intrinsic camera matrix K^{-1} and solve the resulting linear least squares problem with Singular Value Decomposition. Let u and v be the normalized image coordinates, $\tilde{X}$ the corresponding homogeneous point in the world coordinate system and $P = R\begin{bmatrix} I_{3\times3} & -C \end{bmatrix}$ the camera projection matrix. Then we can write the following equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} \tilde{X} \tag{1}$$

which can get reduced to the following least square form:

$$\begin{bmatrix} 0_{1\times4} & -\tilde{X}^T & v\tilde{X}^T \\ \tilde{X}^T & 0_{1\times4} & -u\tilde{X}^T \\ -v\tilde{X}^T & u\tilde{X}^T & 0_{1\times4} \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0 \tag{2}$$

where P_i^T is each row of the camera projection matrix $P \in \mathbb{R}^{3\times4}$. This equation concerns a single 3D point with its 2D image correspondence. Using this method we need $n \geq 6$ point correspondences to solve the PnP problem, so we vertically stack $n = 6$ matrices in the form of the 3×12 matrix on the left hand side of the previous equation and solve for P using SVD. The solution is given by the last column of the matrix V that results from SVD, from which we can extract the rotation matrix R and translation vector T. This solution does not enforce orthogonality of the rotation matrix, which is a basic property of $SO(3)$, so we apply a corrective action by further decomposing R with another SVD. We did a similar decomposition for correcting a rotation matrix when implementing Zhang's calibration method [1]. Finally, we check the value of the determinant of the resulting matrix R and consider $R = -R$ and $T = -T$ if $det(R) = -1$. We can solve for C using the expression $C = -R^T T$. This procedure was implemented in $LinearPnP.py$.

G. PnP RANSAC

In order to remove outliers incorporated from the linear PnP solution and acquire a more robust camera pose, we use RANSAC with the result from linear PnP as an initial estimate. The linear PnP method is based on minimizing the algebraic error, but here we estimate a refined camera pose through additionally imposing a threshold based on the reprojection error:

$$e = \left(u - \frac{P_1^T \tilde{X}}{P_3^T \tilde{X}}\right)^2 + \left(v - \frac{P_2^T \tilde{X}}{P_3^T \tilde{X}}\right)^2 \tag{3}$$

We run RANSAC for a maximum of 600 iterations with a threshold for a point to be an inlier equal to 10, based on its reprojection error value. This was implemented in $NonlinearPnP.py$ which iteratively calls the function $LinearPnP$.

H. Nonlinear PnP

A more efficient method to enforce the orthogonality of the rotation matrix R is by representing it as a 4-dimensional quaternion q which provides a more elegant representation of rotations in 3D space. As an additional step to further refine the estimation of the camera pose, we use a nonlinear least squares solver from the $scipy$ library to minimize the reprojection error. The minimization is nonlinear due to the quaternion parameterization that we introduced. The loss function for the optimization can be written as follows:

$$\min_{C,q} \sum_{i=1,J} \left(u^j - \frac{P_1^{jT} \tilde{X}_j}{P_3^{jT} \tilde{X}_j}\right)^2 + \left(v^j - \frac{P_2^{jT} \tilde{X}_j}{P_3^{jT} \tilde{X}_j}\right)^2 \tag{4}$$

This nonlinear optimization was implemented in $NonlinearPnP.py$. The pose obtained from Non-Linear PnP is more refined and is used for triangulation of image points. The result of Linear triangulation of the image 3 with respect to image 2 and its pose is shown in the image below.

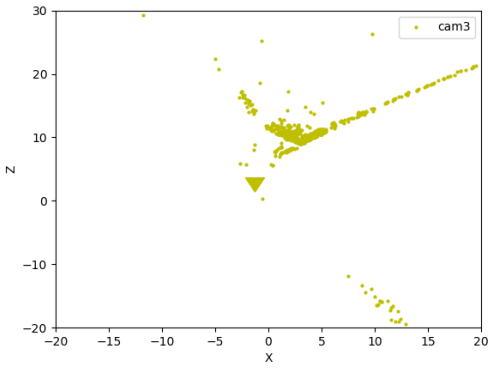

Fig. 6: Linear Triangulation of image 3

I. Visibility Matrix

The final step for refining the camera pose consists of the method of Bundle Adjustment, which involves simultaneously refining the camera poses and 3D points by minimizing the following reprojection error over $C_{i=1}^I$, $q_{i=1}^I$, and $X_{j=1}^J$:

$$\min_{\{C_i,q_i\}_{i=1}^I,\{X\}_{j=1}^J} \sum_{i=1}^{I} \sum_{j=1}^{J} V_{ij} \left(u^j - \frac{P_1^{jT} \tilde{X}}{P_3^{jT} \tilde{X}}\right)^2 + \left(v^j - \frac{P_2^{jT} \tilde{X}}{P_3^{jT} \tilde{X}_j}\right)^2 \tag{5}$$

where V_{ij} is the visibility matrix. This is a $I \times J$ matrix in which an element v_{ij} is equal to one if the j^{th} point is visible from the i^{th} camera and zero otherwise and is jointly implemented in $Wrapper.py$ and $BuildVisibilityMatrix.py$.

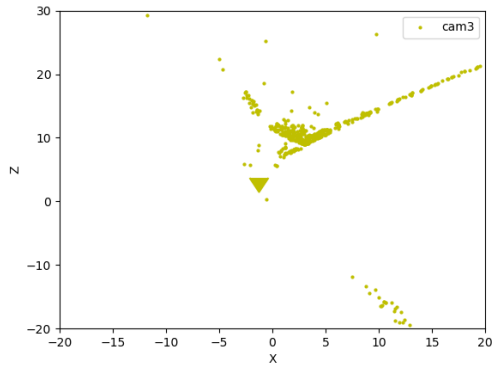

Fig. 7: Non Linear Triangulation of image 3

J. Bundle Adjustment

Using the visibility matrix that we constructed previously, and the large-scale bundle adjustment in *scipy*, we implement bundle adjustment in $BundleAdjustment.py$.

II. Visual SFM

The results obstained using VisualSFM GUI for 3D reconstruction of the scene is given below:

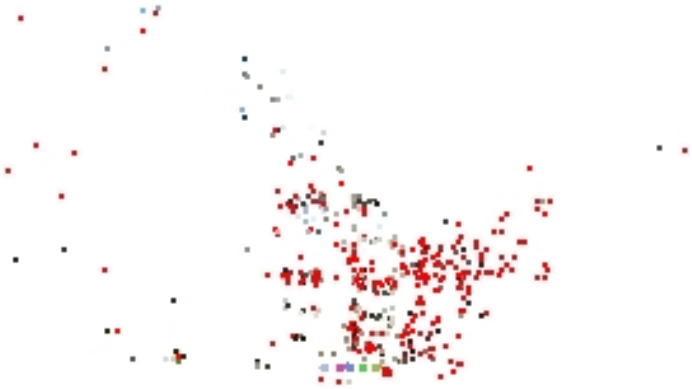

Fig. 8: Visual SFM output

III. Challenges Faced

- The pose estimation from initial frames depends on fundamental matrix and hence while computing the same major variations were observed. Hence, Fundamental matrix was estimated twice;first using matches and second time using inliers.
- The inter matches from different images caused problem during pose estimation as they estimated wrong pose. This problem is yet to be fixed but for ordered images the code works fine.
- RANSAC during PnP causes lots of variations and results in error if any parameter is changed and hence it might cause a problem.
- Optimization using Bundle Adjustment takes considerable time and reprojection errors after it seemed spurious and hence not included in this report.

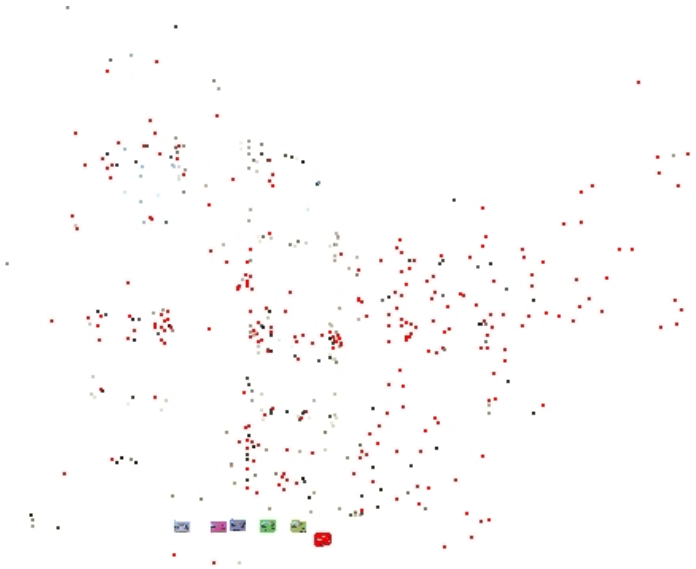

Fig. 9: Visual SFM output

IV. Results

The error table is shown below: As image 4 onwards, appropriate estimation of pose has not been observed and hence errors are huge.

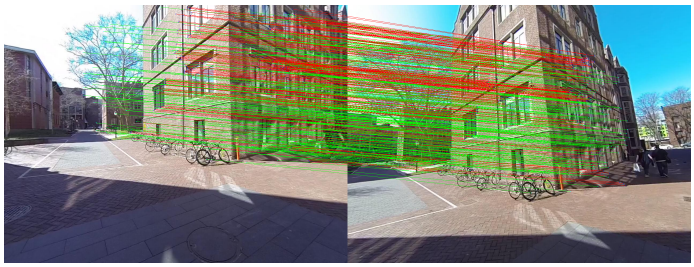

Fig. 10: Feature Matching and RANSAC inliers 1 2

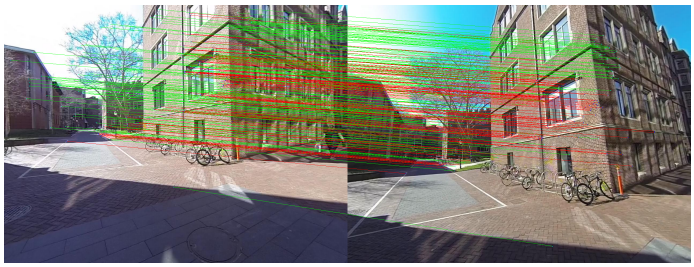

Fig. 11: Feature Matching and RANSAC inliers 1 3

References

[1] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

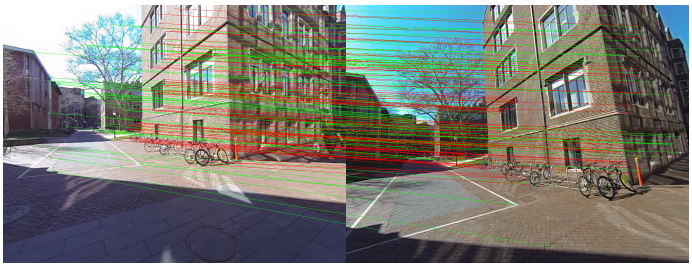

Fig. 12: Feature Matching and RANSAC inliers 1 4

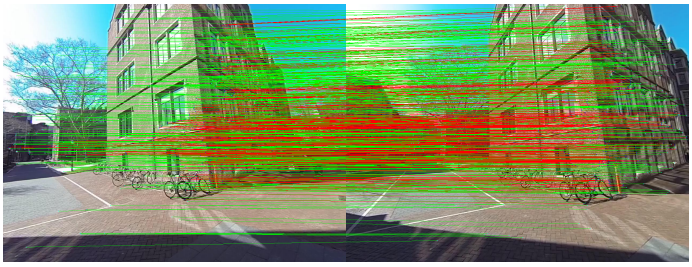

Fig. 13: Feature Matching and RANSAC inliers 2 3

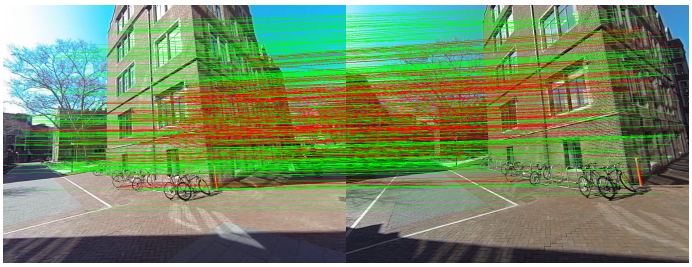

Fig. 14: Feature Matching and RANSAC inliers 2 4

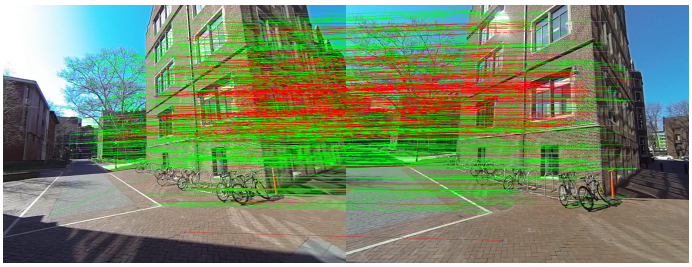

Fig. 15: Feature Matching and RANSAC inliers 3 5

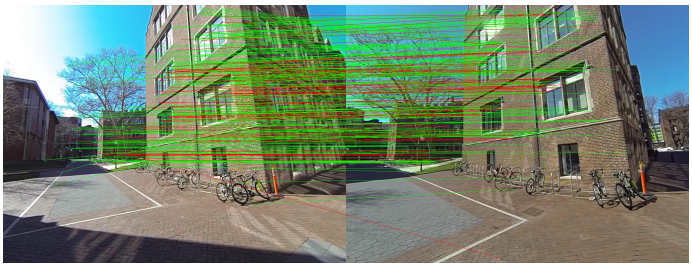

Fig. 16: Feature Matching and RANSAC inliers 3 6

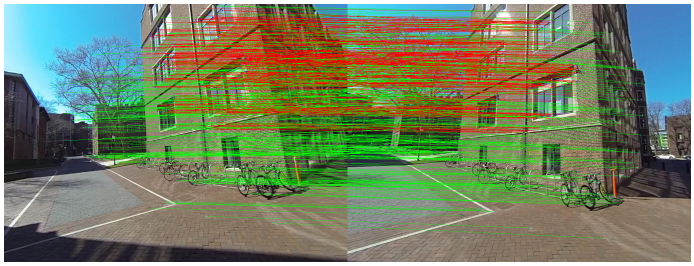

Fig. 17: Feature Matching and RANSAC inliers 4 5

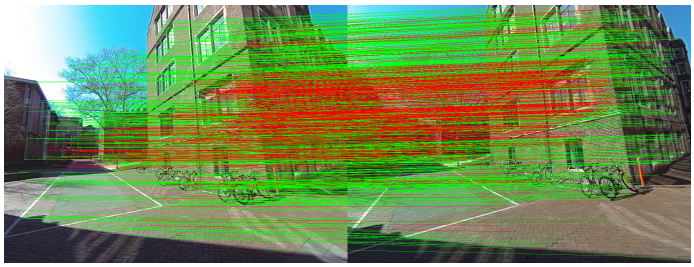

Fig. 18: Feature Matching and RANSAC inliers 3 4

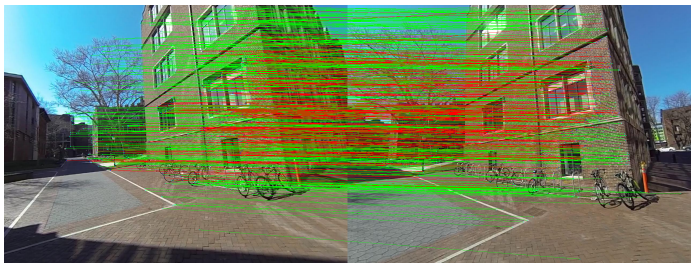

Fig. 19: Feature Matching and RANSAC inliers 4 6

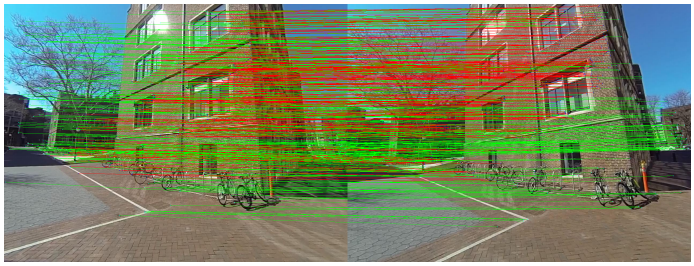

Fig. 20: Feature Matching and RANSAC inliers 5 6

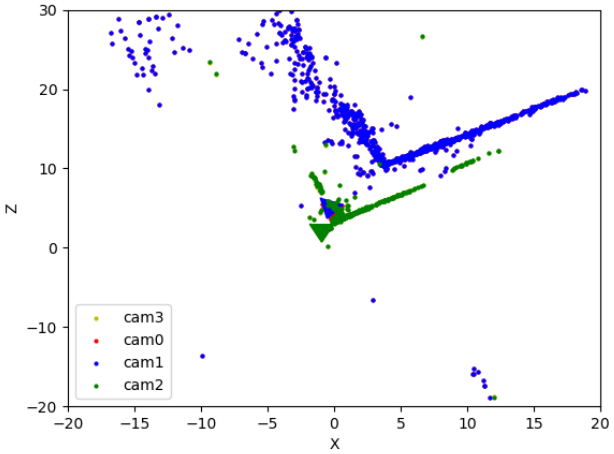

Fig. 21: Plot of points and camera poses till image3

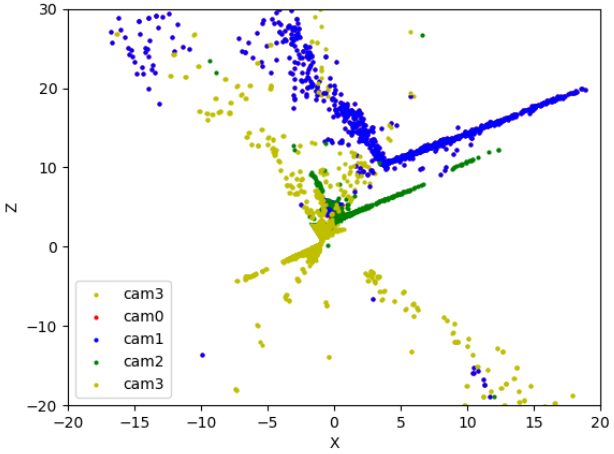

Fig. 22: Plot of points and camera poses till image4

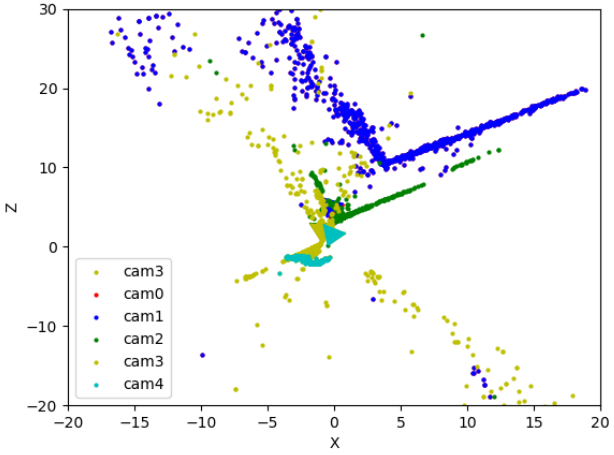

Fig. 23: Plot of points and camera poses till image5

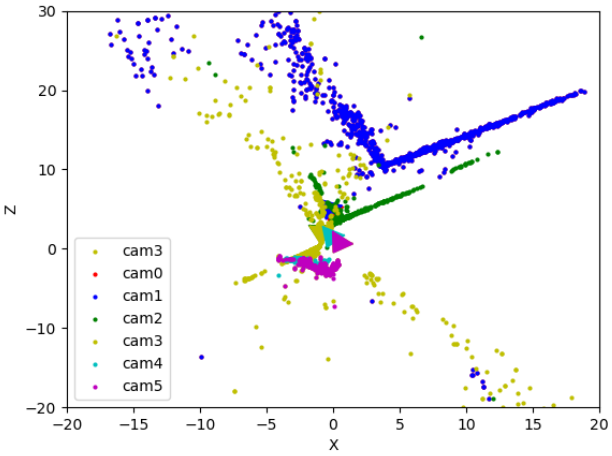

Fig. 24: Plot of points and camera poses till image6

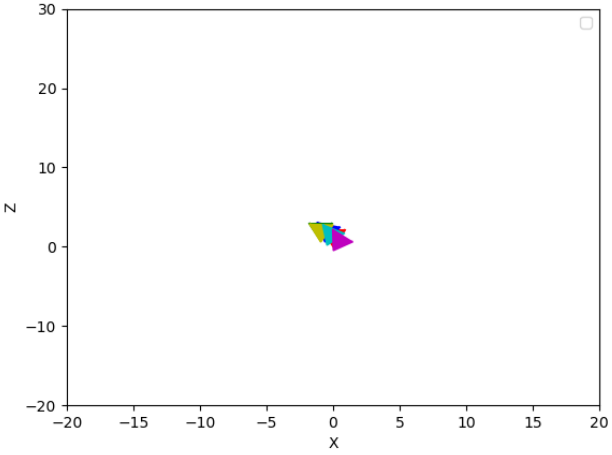

Fig. 25: Plot of all camera poses

Image Number	Linear Triangulation Error	Non-Linear Triangulation Error	PnP Ransac Error	Non Linear PnP Ransac Error	Bundle Adjustment error
1	NA	NA	NA	NA	NA
2	30.472020725280994285	30.197910855974285	NA	NA	NA
3	66.4400405243	66.0254	1052.1184855420129	266.87464513923953	5.00228
4	136.18376	114.009	15251.1000886822195	5590.512068920818	1.158143
5	19.437696	18.40522	3881.5723166883395	735.061296002159	1.1943750
6	79.32340	76.6775	1416.0052799047885	268.8781893080119	1.2085

Fig. 26: Error Table