Project 3: Buildings built in minutes - An SfM Approach

(Using 7 Late Days.)

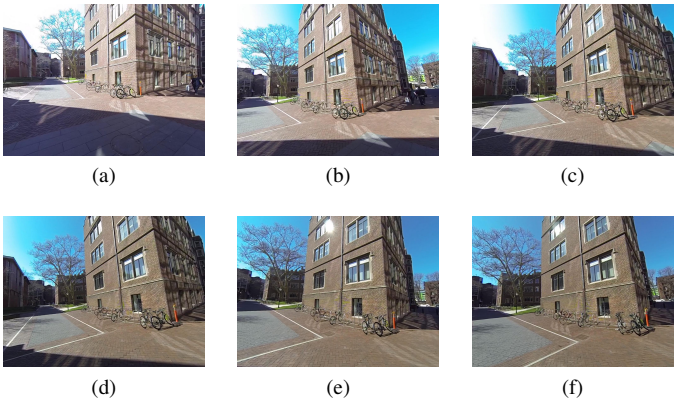
Rohith Jayarajan (115458437)
University of Maryland
College Park, Maryland 20740
rohith23@umd.edu

Rohitkrishna Nambiar (115507944)
University of Maryland
College Park, Maryland 20740
rohit517@umd.edu

***Abstract*—In this work, reconstruction of a 3D scene and simultaneous estimation of the camera poses of a monocular camera is studied. We explore the concepts of Structure from Motion (SfM) to create the entire rigid structure from a set of images with different view points.**

I. Feature Matching

A. Feature Detection



(a) (b) (c)

(d) (e) (f)

Fig. 1. Input Images

The first step towards feature matching is to understand how the two images are related geometrically. To do this, we first extract feature points that can be tracked from one image to other. We use SIFT feature points and show that these are tracked well across images. SIFT feature detector can be created in OpenCV[1] by using

Since we have the SIFT descriptors for each feature detected in both the images, we can compute feature matches between the two image by using the OpenCV function *cv2.BFMatcher()*. Since we only are concerned about the best pair matches, we use the function *bf.knnMatch()* with *k=2*. We further improve the quality of matches by using a ratio test with the ratio of te first best match and the second mathc to be less than a threshold of 0.8.

Now we have matches between the two images but all these matches are not correct matches. There are still some outliers which we need to process out.

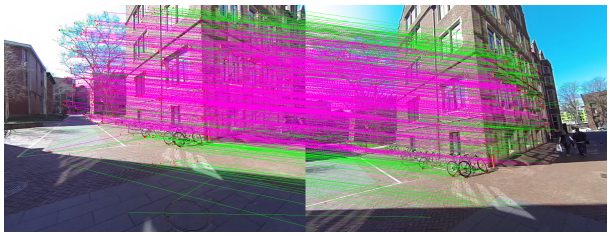


Fig. 2. Feature Matching with RANSAC

II. The Fundamental Matrix **F**

The F matrix is an algebraic representation of epipolar geometry. One important property of the Fundamental Matrix is that described in Equation 1. This is known as the epipolar constraint or the correspondence condition.

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

Since **F** is a $3x3$ matrix, we get a homogeneous linear system with 9 unknowns. To solve this system of equation, we need at least 8 point correspondences between two images.

To solve for **F** we cast the system into the format **Ax=0** whose solution is the smallest eigen-vector of A reshaped into a $3x3$ matrix. But another property of the Fundamental Matrix is that its rank is equal to 2. So it is necessary to set the third eigen value of **F** matrix to 0.

III. Outlier Rejection for Best F via RANSAC

Since the point correspondences are computed using SIFT feature descriptors, the data is bound to be noisy and contains several outliers as discussed in the previous section. Thus, to remove these outliers, we use RANSAC algorithm for a better estimate of the fundamental matrix

IV. Estimating **Essential Matrix** from Fundamental Matrix

After obtaining the best estimate for the Fundamental Matrix from the matching features in the two images, we need to compute the Essential Matrix **E**. The **E** matrix is related to

F matrix by $\mathbf{E} = K^T\mathbf{F}\mathbf{K}$, where **K** is the camera calibration matrix.

The rank of the Essential Matrix **E** is 2 but due to the noise present in **K** this is not the case in the obtained estimate. Hence, the rank of the Essential Matrix **E** is forced to 2 by setting the third eigen value of **E** to 0.

V. Estimate Camera Pose from Essential Matrix

Once we have obtained the Essential Matrix, we have to compute the camera pose which consists of 6 degrees-of-freedom (DOF).

The four camera pose configurations we get are:

1) $C_1 = U(:,3); R_1 = UWV^T$
2) $C_2 = -U(:,3); R_1 = UWV^T$
3) $C_3 = U(:,3); R_1 = UW^TV^T$
4) $C_4 = -U(:,3); R_1 = UW^TV^T$

where W = $\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $E = UDV^T$.

After this step, we need to make sure that ***det*(R)**! $= -1$. If this is not the case, then we need to correct the camera pose by setting $\mathbf{R} = -\mathbf{R}$ and $\mathbf{C} = -\mathbf{C}$

VI. Triangulation Check for Cheirality Condition

Given four camera poses, we need to identify which one of those is the correct camera pose. To do this, the first step is to triangulate 3D points using linear least squares.

To remove the disambiguity of identifying the correct camera pose, we check the cheirality condition i.e. the reconstructed points must be in front of the camera. A 3D point ***X*** is in front of the camera iff $r_3(X - C) > 0$ where r_3 is the third row of the rotation matrix. Not all triangulated points satisfy this condition due of the presence of correspondence noise. The best camera configuration, (**C,R,X**) is the one that produces the maximum number of points satisfying the cheirality condition.

VII. Non-Linear Triangulation

We computed the 3D points using Linear Triangulation but this estimate can be refined. Given, the initial estimate of the 3D point **X** and the pixel coordinate, minimizing the reprojection error (computed by measuring error between measurement and projected 3D point) will provide a better estimate of the world point. It is done by the Equation **??**.

$$\min_X \sum_{j=1}^{2} \left(u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}\tilde{X}} \right)^2 + \left(v^j - \frac{P_2^{jT}\tilde{X}}{P_3^{jT}\tilde{X}} \right)^2 \qquad (2)$$

where j is the index of each camera/successive image and $\tilde{\mathbf{X}}$ is the homogeneous representation of **X**. This minimization problem was optimized by using *scipy.optimize.least_squares* function in Scipy library.

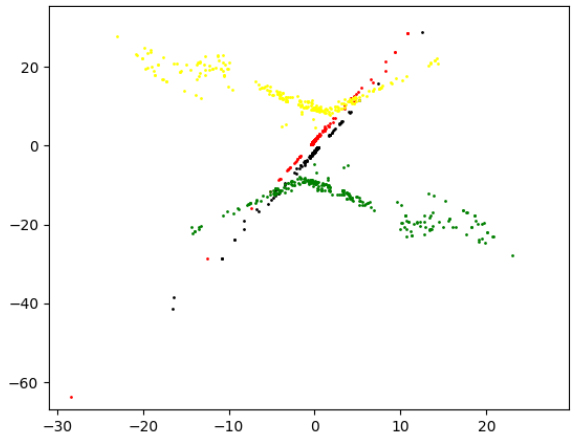


Fig. 3. Initial Triangulation with dis ambiguity for 4 camera poses.

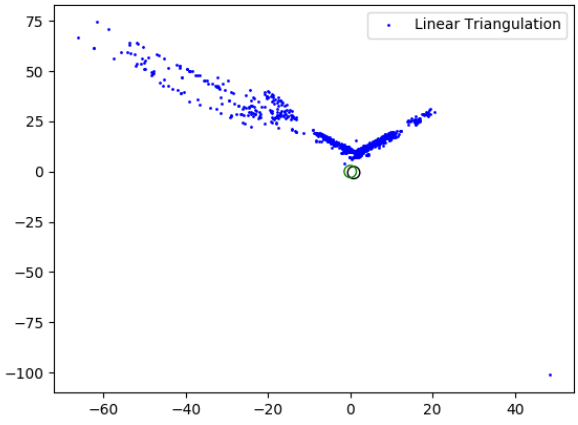


Fig. 4. Linear Triangulation selecting the correct camera pose.

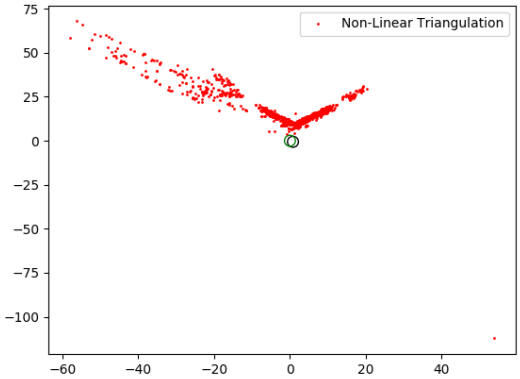


Fig. 5. Non-Linear Triangulation

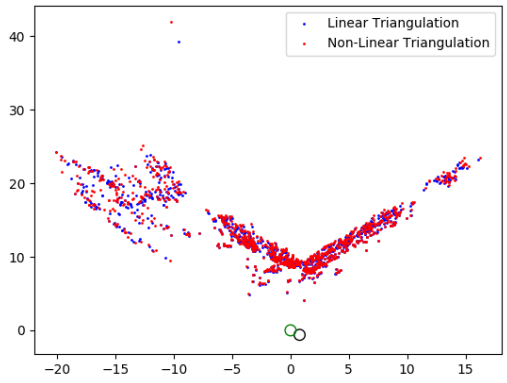


Fig. 6. Linear and Non-Linear Triangulation

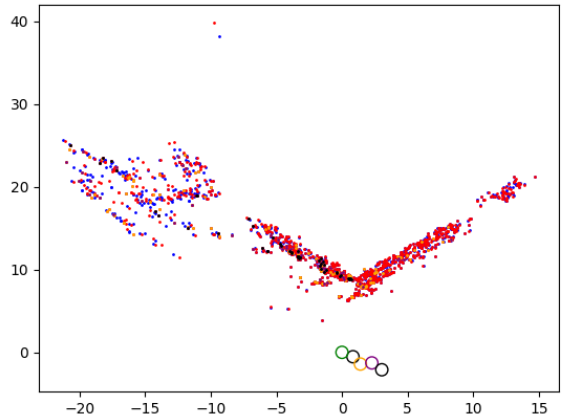


Fig. 7. Camera poses and scene reconstruction after PnP algorithm.

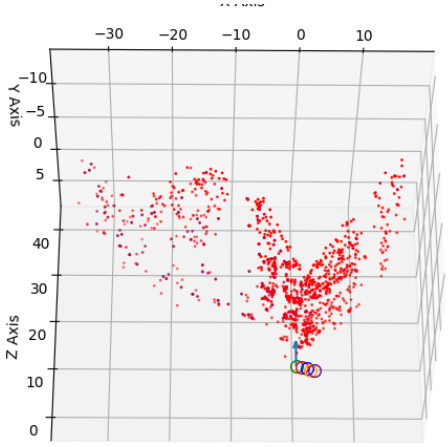


Fig. 8. Scene reconstructed in 3D

VIII. Linear Perspective-n -Points

We now have a a set of n 3D points in the world, their 2D projections in the image and the intrinsic parameters. WIth this the 6 DOF camera pose can be estimated using linear least squares. We need at least 6 point correspondences so as to get a unique solution. Given world points **X** and image points **x**, the 2D image points are normalized using the camera intrinsic matrix and the linear least squares system is solved for translation (**t**) and rotation (**R**) where $t = -R^T C$. The orthogonality of the rotation matrix is not enforced by the least squares solution, so it must be corrected. Also, rotation matrix should be corrected if the the determinant of the rotation matrix **R** = -1.

To get a robust estimate of the 6 DOF pose, RANSAC is used to overcome the errors involved.

IX. Nonlinear PnP

With the 3D-2D correspondences, and the linearly estimated camera pose. We can refine the pose that minimizes reprojection error given by the optimization in Equation 3

$$\min_{C,R} \sum_{j=1}^{J} \left(u^j - \frac{P_1^{jT} \tilde{X}}{P_3^{jT} \tilde{X}} \right)^2 + \left(v^j - \frac{P_2^{jT} \tilde{X}}{P_3^{jT} \tilde{X}} \right)^2 \quad (3)$$

where j is the index of each camera/successive image and $\tilde{\mathbf{X}}$ is the homogeneous representation of **X**. This minimization problem was optimized by using *scipy.optimize.least_squares* function in Scipy library.

X. Bundle Adjustment

The first step to bundle adjustment is to create the visibility matrix which is a $\boldsymbol{I} \times \boldsymbol{J}$ matrix, $\boldsymbol{V}$ where V_{ij} is one if the j^{th} point is visible from the i^{th} camera and zero otherwise.

Bundle adjustment is an optimization problem to refine both the initialized camera poses as well as the 3D points by minimizing a reprojection error function with the visibility matrix in it.

XI. V-SFM

We used V-SFM to compare our outputs. The reconstruction and camera poses generated can be seen in Fig. 9 and Fig. 10. We see that the outputs generated by V-SFM are much more accurate and represents the structure reconstructed more faithfully.

References

[1] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

Fig. 9. VSFM front view

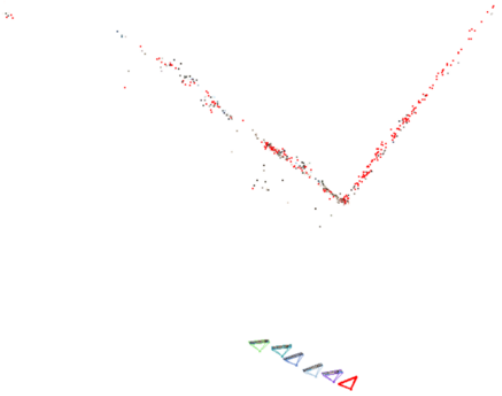


Fig. 10. VSFM Top View