Project 3 Report
An SfM Approach

Ameya Patil
Department of Computer Science
University of Maryland
College Park, Maryland 20740

Sigurthor Bjorgvinsson
Department of Computer Science
University of Maryland
College Park, Maryland 20740

I. Introduction

The aim of this project was to create a 3D model of a scene from multiple 2D images of the scene captured from different perspectives and having common regions between them. To start with, we were given 6 images of a street scene with a building in it, 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. Estimating Fundamental Matrix

RANSAC was performed on the provided point correspondences to get refined correspondences(inliers) as shown in Figures 1 and 2. The fundamental Matrix was estimated using the inliers between the first and second image only. The points were normalized and the 8-point algorithm was used to solve for the matrix. The singularity constraint was enforced to change the rank of the matrix from 3 to 2. The computed fundamental matrix was then denormalized to get the correct matrix. The Fundamental matrix between image 1 and 2 is as shown below:

$$\begin{bmatrix} -7.050727e^{-07} & -1.541574e^{-05} & 3.785774e^{-03} \\ 1.963694e^{-05} & -1.456382e^{-06} & -5.253717e^{-03} \\ -6.296857e^{-03} & 1.920373e^{-03} & 1.346693 \end{bmatrix}$$

III. Essential Matrix from Fundamental Matrix

The Essential Matrix was simple to calculate as it involved adding the presence of the camera matrix to the Fundamental matrix. The Essential matrix is as shown below:

$$\begin{bmatrix} -0.026652 & -0.665084 & -0.303668 \\ 0.841332 & -0.066702 & 0.501199 \\ 0.201903 & -0.660052 & -0.157527 \end{bmatrix}$$

IV. Estimate Camera Pose from Essential Matrix

By assuming that the first camera is located at the world coordinate system origin and aligned with the world coordinate system, and by factorizing the Essential matrix, 2 rotation matrices and 2 translation vectors were generated, from which 4 possible camera orientations were generated for the second image. Using these camera orientations, we created 4 projection matrices $P = K \begin{bmatrix} R & t \end{bmatrix}$ and used them to generate real world points from all the image points using the Linear Triangulation function.

V. Triangulation

A. 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.

B. Cheirality Check

Using the 4 estimated camera poses for the second camera and the triangulated 3D world points, cheirality check was performed to get the best camera pose. The correct camera pose would have the fewest real world points positioned behind the camera in Z direction. Here we had a lot of confusion regarding the last column in U. In the project page, it states that this is C - the camera location vector, but what we learned was that this is actually the translation vector - t.

C. Non-Linear Triangulation

Given 2 projection matrices, image points and matching real world points, we wanted to minimize the projection error on both images by optimizing the real world points. This was done using the LM optimizer API in scipy. Our first attempt was to optimize all of the points together but later found out that was unnecessary. Instead we minimized the error for each real world point, leading to much faster convergence.

VI. 3.6. Perspective-n-points

The following steps were performed while registering the remaining images, to estimate the camera pose for them and to further refine the world points.

A. PnP RANSAC

For a certain image to be registered, we found a previous pose estimated image, with which it had most inlier matches, instead of always finding inliers with the first image (assumed to be taken from world origin). This was done so as to have more matches to work with. For example, image 1 had no matches with image 6 and this was causing issues. Using the inlier matches with a pose estimated image, we could find the 2D image to 3D world point correspondences for this new image. We then performed RANSAC to estimate a pose for the new image using LinearPnP.

B. Linear Camera Pose Estimation

In this section, we needed to find a rotation and translation matrix that would satisfy a set of real world points projected to a set of image points. First we normalized the image points to remove the effect of the camera matrix by multiplying with the inverse of the camera matrix. Next we created a linear solver matrix to solve for the rotation and translation.

Our attempts to enforce orthogonality on the R matrix was found to increase the projection error. Best results were obtained when we retrieved the R and t matrix right out of the last row of V^T. This in fact did not matter because after the Non-Linear PnP the algorithm converged to the same projection error.

C. NonLinear PnP

In the Non-Linear PnP we were trying to minimize the projection error by optimizing the rotation and translation using 3D to 2D correspondences found before. A mistake we learned here was that we attempted to generate the projection matrix before passing it to least_squares and optimizing that matrix. This resulted in a camera matrix being embedded in the projection matrix and leading to bad optimization. We ended up passing the rotation and translation matrix to the least square and then building the projection matrix in the cost function where the camera matrix was taken in as an argument, instead of being a part of the initial guess.

The reprojections of the triangulated 3D world points after optimizing the estimated camera pose are as shown in Figure 3

Further, new 3D world points were triangulated for the pair of cameras used in the above step, and were optimized using non-linear optimization. The results are shown in Figure 4

VII. Bundle Adjustment

A. Visibility Matrix

We created a dictionary called traj where the key was a real world point and the value was an array of tuples, imageIndex (in our list) and the 2D image coordinates to which the real world point was mapped. Using this dictionary, we generated the Visibility matrix where each row is an imageIndex and the column represented the index of the real world coordinate. Using the traj dictionary, this was simple.

B. Bundle Adjustment

For every new addition of image to the pool, we updated the 3D world points. This could have affected the 3D world to 2D image mappings of other images as well, so we need to perform bundle adjustment which basically tries to optimize all the variables - the camera poses for all images registered so far in each iteration, and the 3D world points - together. Since in this case, the variables to be optimized were too many, we had to optimize the refinement algorithm itself. We did this by constructing a sparse jacobian matrix which tells the optimizer function about the dependency of a particular residual on a particular variable, so that it does not spend time computing gradients for unrelated residuals and variables. This sparse jacobian matrix was sent as additional input to the $scipy.least\_squares$ optimizer.

The reprojections of the triangulated 3D world points after bundle adjustment are as shown in Figure 5

VIII. Surprises

The matches data we received was tainted with points with one-to-many connections. In the end we removed all matches that had repeated points in matches that had already been read and stored.

Real world points needed to be able to change for images that had been processed. For example if we have point A in image 1, point B in image 2 and point C in image 3. After processing image 1 and 2, point A and B point to a real world point X, when image pair 2 and 3 get processed, B and C is identified as a inlier. After PnP had been done and Linear triangulation was done, point B and C had a real world point P. What we did was that when B was saved with another real world point, we moved all image points to the new real world point. So once image 2 and 3 was finished being processed, A now pointed to P instead of X. We believe this was the correct approach to have more residuals per world point.

IX. Results

In the table I we show the Linear PnP and NonLinear PnP for images 3 to 6 paired with the image with which it had most inlier matches. Linear Triangulation and NonLinear Triangulation is performed between the two images in the pair, and the average reprojection error before and after bundle adjustment is calculated for all images that have been processed so far. For example, image pair 3 and 4 displays reprojection error of all world points found in images 1 to 4.

In the end, we were able to get a good average pixel reprojection error. We did not understand the increase in pixel error in pairs 4/5 and 5/6 after the NonLinear PnP (bolded in table). The least_square showed the final cost less than the initial cost but our error went up.

References

[1] http://cmp.felk.cvut.cz/cmp/courses/TDV/2013W/lectures/tdv-2013-07-anot.pdf

[2] https://www.cs.utah.edu/ srikumar/$cv_spring2017_files$/Lecture3.pdf

[3] http://cis.upenn.edu/ cis580/Spring2016/Lectures/cis580-18-coursera-2016-SfM-fulll.pdf

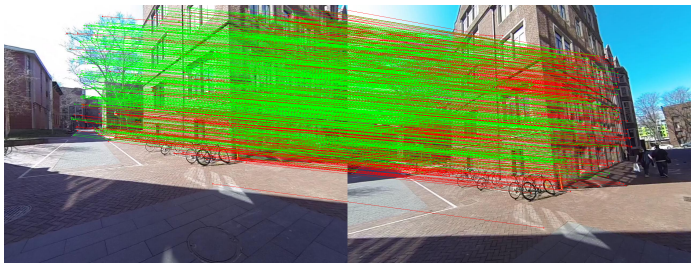
[4] http://ai.stanford.edu/ birch/projective/node20.html

[5] https://www.uio.no/studier/emner/matnat/its/UNIK4690/v16/forelesninger/$lecture_73$-pose-from-epipolar-geometry.pdf

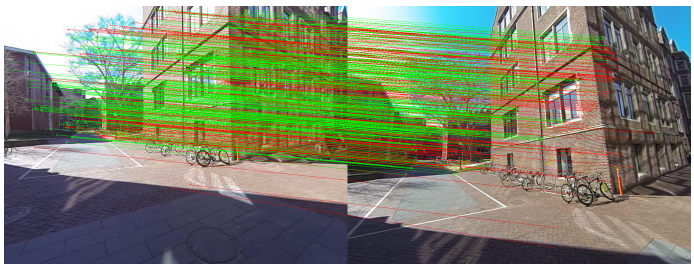
[6] https://scipy-cookbook.readthedocs.io/items/bundleadjustment.html

	Average Euclidean Pixel Projection Error					
Image Pair	Linear PnP	NonLinear PnP	Linear Triangulation	NonLinear Triangulation	Before BA	After BA
Image 1 and 2	n/a	n/a	0.887073	0.867780	n/a	n/a
Image 2 and 3	2.771641	2.500534	0.848535	0.823843	0.956535	0.269139
Image 3 and 4	1.881009	1.854282	0.331476	0.319151	0.444658	0.319146
Image 4 and 5	**2.066174**	**2.095174**	0.317244	0.298908	0.550520	0.344769
Image 5 and 6	**2.553068**	**3.867218**	**0.502613**	**0.509479**	0.599114	0.376498

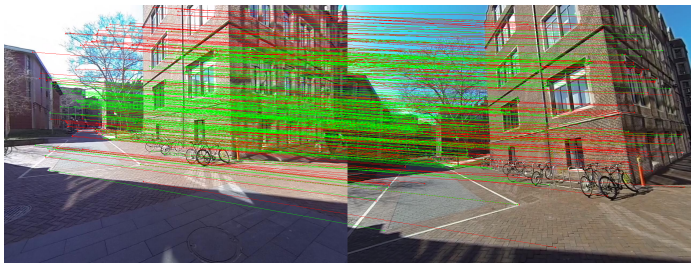
TABLE I: Table showing average euclidean pixel projection error for all images



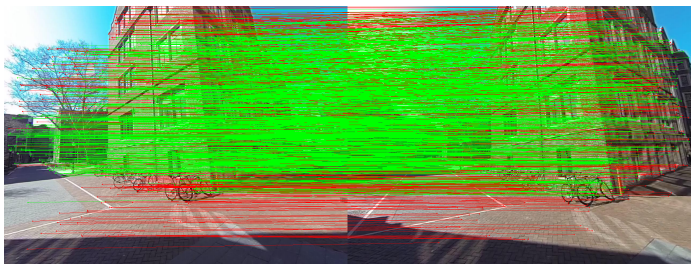
(a) Feature correspondences between images 1 and 2



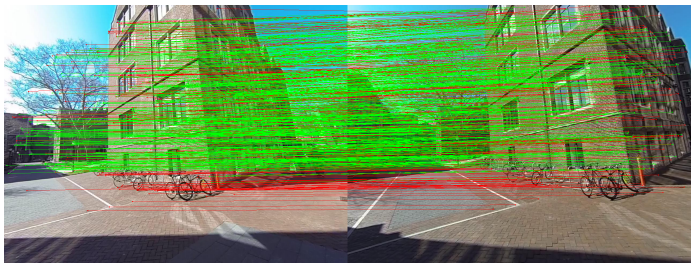
(b) Feature correspondences between images 1 and 3



(c) Feature correspondences between images 1 and 4

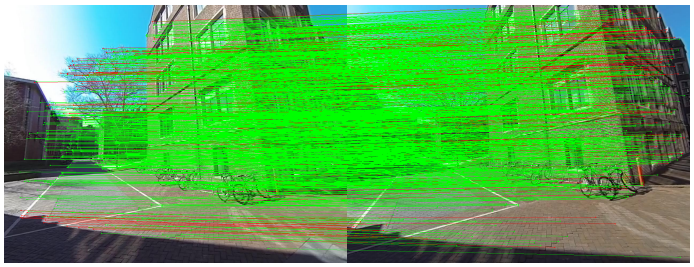


(d) Feature correspondences between images 2 and 3

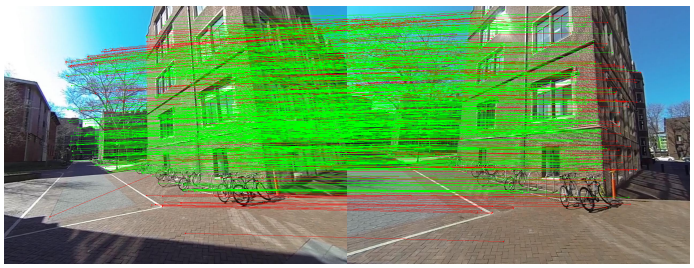


(e) Feature correspondences between images 2 and 4

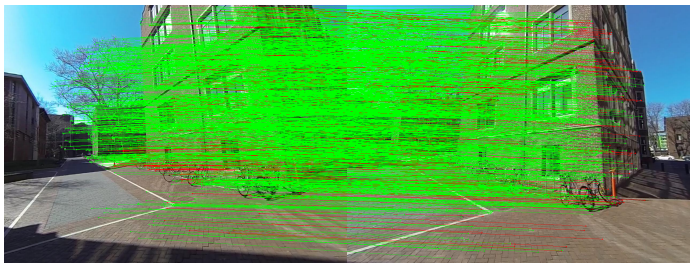
Fig. 1: Feature correspondences between images after RANSAC. Green lines show the accepted correspondences and red lines show the rejected ones.



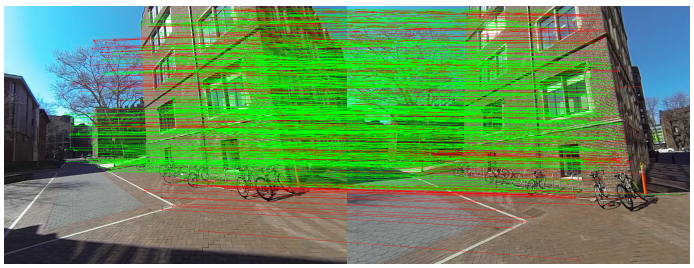
(a) Feature correspondences between images 3 and 4



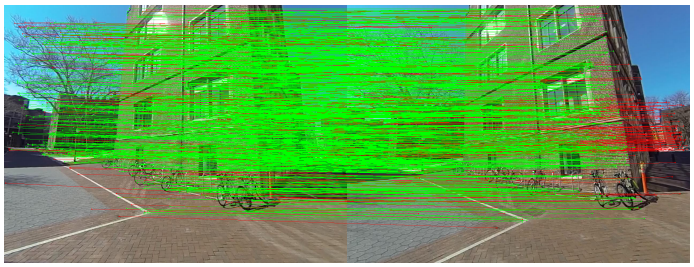
(b) Feature correspondences between images 3 and 5



(c) Feature correspondences between images 4 and 5



(d) Feature correspondences between images 4 and 6



(e) Feature correspondences between images 5 and 6

Fig. 2: Feature correspondences between images after RANSAC. Green lines show the accepted correspondences and red lines show the rejected ones.

(a) Reprojections for Images 3 and 4 after optimized PnP

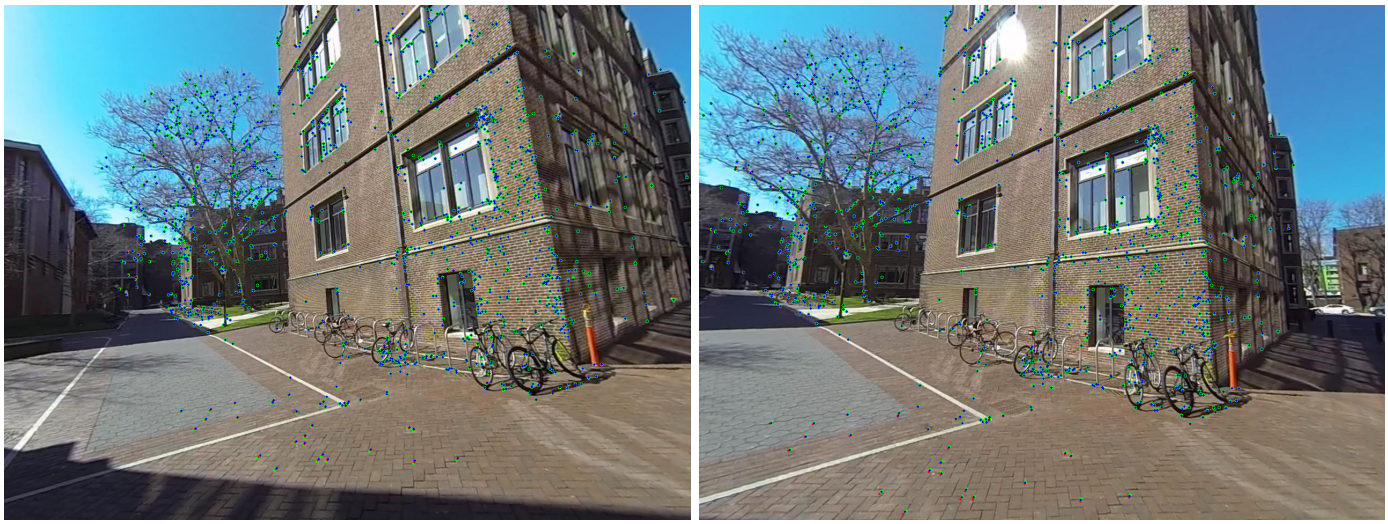
(b) Reprojections for Images 5 and 6 after optimized PnP

Fig. 3: Reprojections after PnP step to get camera poses. Green points mark the **actual image 2D points**, red points are the **reprojected points after linear PnP** and blue points are the **reprojections after nonlinear PnP**

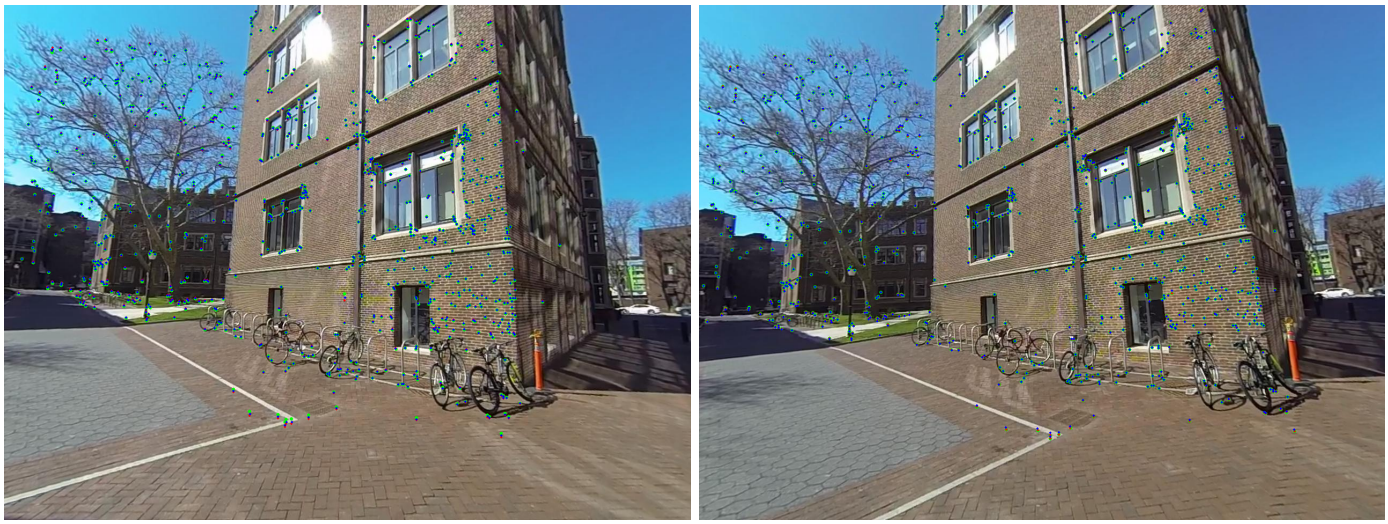
(a) Reprojections for Images 1 and 2 with newly triangulated world points

(b) Reprojections for Images 2 and 3 with newly triangulated world points

(c) Reprojections for Images 3 and 4 with newly triangulated world points



(d) Reprojections for Images 4 and 5 with newly triangulated world points

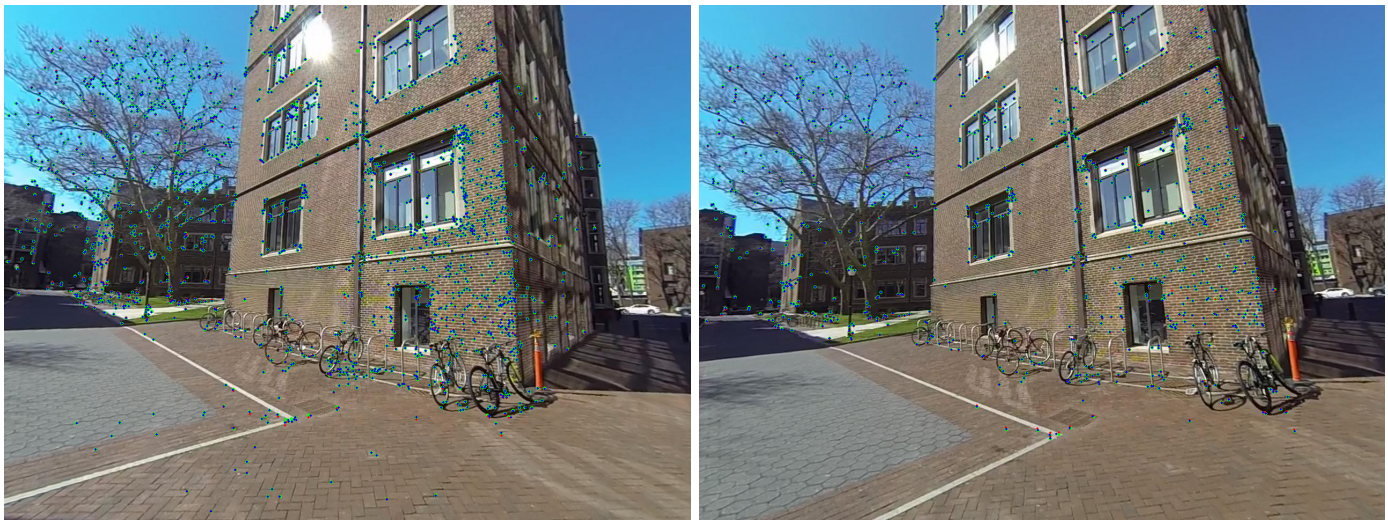


(e) Reprojections for Images 5 and 6 with newly triangulated world points

Fig. 4: New triangulated world point reprojections, after PnP step. Green points mark the **actual image 2D points**, red points are the **reprojected points after linear triangulation** and blue points are the **reprojections after nonlinear triangulation**

(a) Reprojections for Images 1 and 2

(b) Reprojections for Images 3 and 4



(c) Reprojections for Images 5 and 6

Fig. 5: Reprojections after the final bundle adjustment step, performed after all the images were registered. Green points mark the **actual image 2D points**, red points are the **reprojected points before bundle adjustment** and blue points are the **reprojections after bundle adjustment**