Buildings built in minutes

Using 1 late day

Gokul Hari
M. Eng. Robotics
University of Maryland
College Park, MD, 20742
Email: hgokul@umd.edu

Sakshi Kakde
M. Eng. Robotics
University of Maryland
College Park, MD, 20742
Email: sakshi@umd.edu

I. Structure from motion

In this project, we reconstructed a 3D scene and simultaneously obtained the camera poses with respect to the scene, with a given set of 6 images from a monocular camera and their feature point correspondences.

Feature Matching, Fundamental Matrix and RANSAC

The feature points correspondences from all the text files were parsed to an appropriate format to be utilised for the rest of the pipeline. By estimating Fundamental matrix and performing RANSAC using epipolar constraint, we estimated the inlier - feature correspondences to use in the later steps. The inliers obtained after in the getInliersRANSAC procedure is shown in 4, The red lines indicate all matches while the green lines indicate all inliers from the matched features.

Next, we computed the Fundamental matrix only for images 1 and 2, using the normalized 8 point algorithm. The 8 point algorithm involves solving a linear solver matrix that is generated by stacking the kroenker product between 8 point correspondences, using singular value decompostion. RANSAC helps to obtain a better estimate of the fundamental matrix. The fundamental matrix between image 1 and 2 is given as,

$$\begin{bmatrix} -4.040e^{-07} & -1.025e^{-05} & 2.586e^{-03} \\ 1.250e^{-05} & -5.788e^{-07} & -4.935e^{-03} \\ -4.077e^{-03} & 2.768e^{-03} & 1.000e^{+00} \end{bmatrix}$$

Essential Matrix and solving for camera poses

Using the fundamental matrix and the camera intrinsic matrix, we estimate the essential matrix E. E is decomposed to obtain 4 mathematically possible poses (rotation R and translation C matrices), that corresponds to the pose of the second image as the first image is considered to be in reference alignment (R = 3×3 identity matrix and C = 3×1 zero matrix)

Linear Triangulation and recovering correct pose

Given a point correspondence and the projection matrix estimated using the camera pose (R,C) and intrinsics (K) parameters, we can perform linear triangulation to obtain the 3D location of the point correspondence.

We perform linear triangulation for all the 4 poses to recover the corresponding 3D points of each pose. Out of the 4 poses and 3D points, we only select the pose and the 3D point cloud that obeys the chierality condition – the idea that depth of the 3D points must be positive. This is also called as the depth positivity constraint. Thus we choose the only physically possible pose using cheirality check and linear triangulation.

Non Linear Triangulation

Now, having obtained the 3D points from Linear triangulation from the selected pose, we can refine these 3D points by performing non linear optimization. We use the linear triangulation output as the initial guess and perform the optimization by minimizing the reprojection error between actual points and reprojected points. We use the trust region field method in scipy.optimize function. While performing this optimization, we learnt that running the optimizer for each point correspondence led to faster convergence compared to running the optimizer once for all the set of estimated 3D points.

Reprojection Error

The reprojection error is computed as,

$$\rho = \sum_{j=1,2} (u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}\tilde{X}})^2 + (v^j - \frac{P_2^{jT}\tilde{X}}{P_3^{jT}\tilde{X}})^2 \quad (1)$$

where j corresponds to each of two cameras/ images, u^j and v^j are the actual image coordinates of image j.

Till here, we have registered the camera pose and 3D point of images 1 and 2. We proceed to estimate the camera poses with respect to the reference alignment for the remaining images and further estimate the 3D world points.

Linear PnP, Ransac and Non linear optimization

Given the intrinsic camera matrix K , image points x and corresponding 3D points X, we can compute the pose R,C parameters, which is termed as the perspective-n-point problem. We normalize the 2D points x by performing $K^{-1}x$, to remove the effect of the intrinsics Now, we need atleast 6 2D image points and corresponding 3D points to form a linear PnP solver matrix which can be solved using singular value decomposition (svd) to obtain the 3×4 pose matrix RT which is composed of the rotation and translation components. Note that the normalisation of 2D image points has removed

the K from the 3×4, which otherwise would've yielded the projection matrix. The first 3 columns of *RT* matrix is the rotation matrix, which is supposed to be orthonormal. Practically, there maybe errors in maintaining orthonormality, so we enforce it by decomposing using svd and multiplying only the U and V^t matrices. The translation vector t is obtained by performing - $R^{-1}(RT_4)$ where RT_4 is the 4th column in pose matrix *RT*.

The PnP estimates with just 6 points is highly erroneous and a better estimate can be obtained with RANSAC with the reprojection error threshold chosen to be 5. In addition to this, the pose R and T, obtained after RANSAC are further refined by minimizing the reprojection error using non linear least squares optimization with trust region field method using scipy.optimize.least squares library. To fasten the optimization, we convert the rotation matrix R to a 1×4 quaternion vector along with the translation matrix t (1×3) while passing to the optimizer as a parameter (1×7) under optimization .

A. *Bundle Adjustment*

We noticed that the non linear optimization of pose and 3D points, were yielding coarse-level refinement, (i.e) larger the reprojection error, more effective optimization. We proceeded to perform bundle adjustment, in order to further optimize all estimated camera pose parameters and the 3D points together to achieve 3D reconstruction until that particular camera under registration.

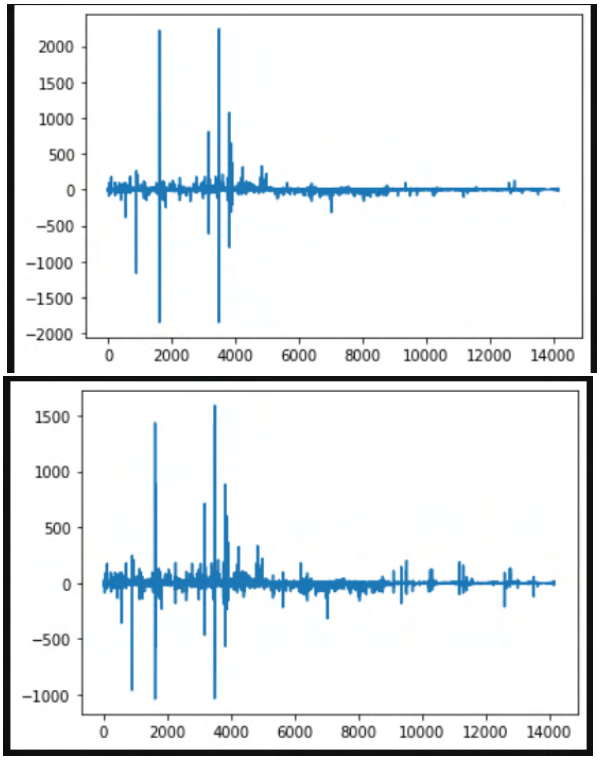
To perform Bundle Adjustment for a given set of poses, 3D points (parameters under optimization) and corresponding 2D points, we need to create a sparsity Matrix that records whether a 2D point observation belongs to the particular parameter. For example, consider there are N image points, N_{3d} world points, n_C cameras (since number of image files provided were 6, maximum n_C will be 6), where each camera has 6 extrinsic parameters, (Rotation: roll, pitch, yaw; Translation: cx,cy,cz). The sparsity matrix M_{ba} has dimensions $2 * N \times (N_{3d} * 3 + n_C * 6)$. If image point at index 12 in N corresponds to world point at index 12 in N_{3d}, then the elements of matrix M_{ba}, that relate them, will be 1.

The sparse matrix is sent as the jacobian sparsity parameter to the least squares optimizer from scipy using 'trf' method. We were able to notice significant coarse level refinement by plotting the residuals of N observations,

Before including the bundle adjustment section in our pipeline, we initially ran bundle adjustment for all the 6 images and 3d points put together in our unit tests, and the residuals are shown in figure 1. Notice that large residuals at range of 2000 to -2000 had reduced to below 1500.

The top view of the 3D points is shown in figure 3 Notice that the some of the 3D points (green) obtained before bundle adjustment tends looks to be sparsely distributed in regions interior to the building where, the 3D points (blue) obtained after bundle adjustment is seen to be more finely distributed and accurate.

[a]



[b]

Fig. 1. (a) Residuals before Bundle Adjustment, (a) Residuals after Bundle Adjustment

II. VisualSFM : A Visual Structure from Motion System

VisualSFM is a GUI application for 3D reconstruction using structure from motion (SFM). The results obtained using the software are shown in fig. 2

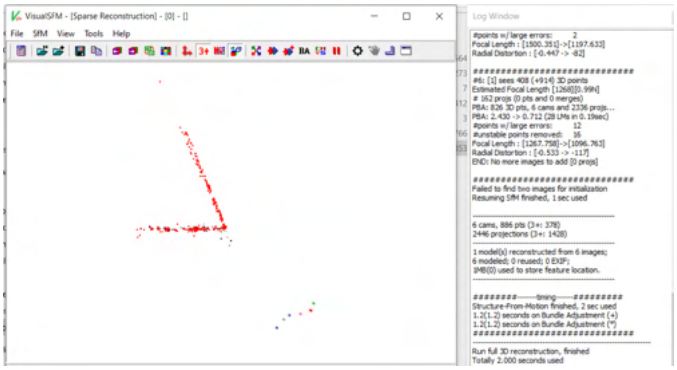


Fig. 2. 3D points using VisualSFM software

III. Analysis

1) Getting the correct set of feature matches in an important step, as it affects the F matrix calculations. We observed that the RANSAC does not always give the same results. Despite running it for 2000 iterations, there is no consistency is the output of RANSAC.

2) If the F matrix is noisy, then the extracted camera pose will be noisy. These camera poses are further used for triangulation. Thus, a noisy set of feature matches can result into noisy results of triangulation. Though we are using non-linear optimizers at various stages, we think this problem will still remain.
3) We observed that the output results and the obtained re-projection error matrix are not same for all the test runs. At times, we observed that the PnP error for a few images became too high. The non-linear PnP did reduce the error, but still the error values were relatively high. The exact cause of this behaviour is unknown. We think it is because of the noisy F matrix itself.

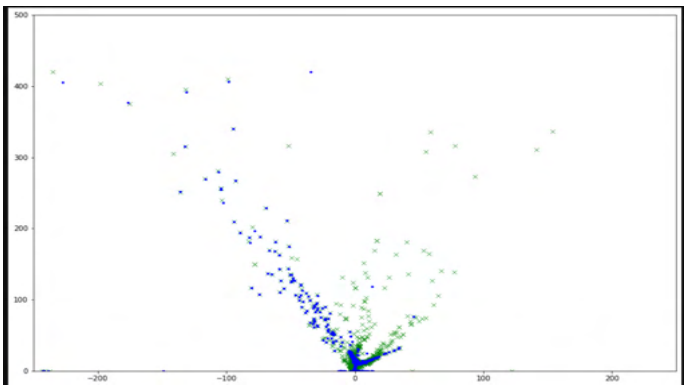


Fig. 3. 3D points without bundle adjustment (green); 3D points after bundle adjustment (blue)

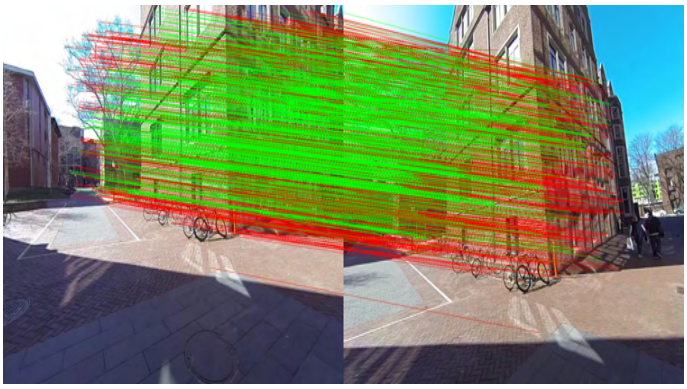


Fig. 4. matches between images 1 and 2

IV. Results

By estimating poses using Non linear PnP, adding 3D points using Non linear Triangulation, and further refinement of poses and 3D points using bundle adjustment, we estimated and added the remaining poses and 3D points for the remaining images.

The inliers obtained after in the getInliersRANSAC procedure is shown in 4, The red lines indicate all matches while the green lines indicate all inliers from the matched features. The final reconstructed 3D points are seen in reprojected

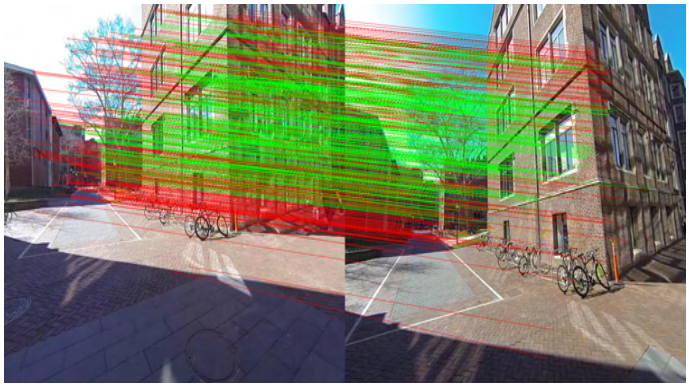


Fig. 5. matches between images 1 and 3

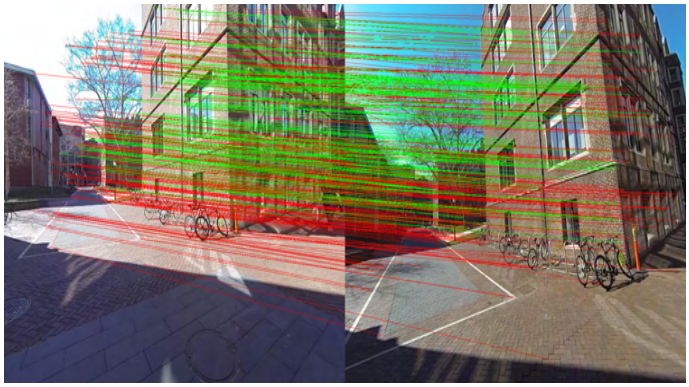


Fig. 6. matches between images 1 and 4

References

[1] http://www.cs.cmu.edu/ 16385/s19/lectures/lecture10.pdf
[2] https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/sdai30/index.ht
[3] https://www.youtube.com/watch?v=RR8WXL-kMzA
[4] http://www.cs.cmu.edu/ 16385/s19/lectures/lecture10.pdf

	Linear PnP	Non-Linear PnP	Linear Triangulation						Non-Linear Triangulation						Bundle Adjustment					
Image ID			0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
1	NA	NA	14.17	NA	NA	NA	NA	NA	14.09	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	510.31	309.02	54.17	199.19	NA	NA	NA	NA	54.00	193.57	NA	NA	NA	NA	600.40	66.26	71.96	NA	NA	NA
3	7,395.25	1,157.65	168.00	450.07	186.37	NA	NA	NA	166.93	432.92	175.78	NA	NA	NA	1,027.67	804.68	114.96	149.22	NA	NA
4	8,566.39	157.05	NA	NA	131.80	10.60	NA	NA	NA	NA	74.24	10.52	NA	NA	927.31	749.60	109.34	72.90	12.87	NA
5	15,940.78	909.83	NA	NA	142.03	256.93	19.89	NA	NA	NA	99.38	19.96	19.50	NA	907.57	738.68	902.67	208.14	12.57	21.83

Fig. 7. Reprojection Error(mean square distance) after each step

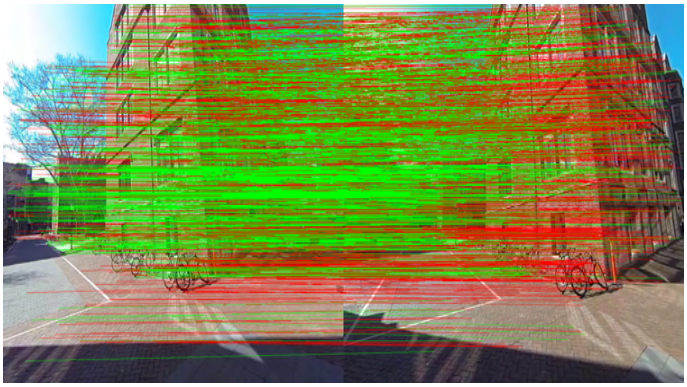


Fig. 8. matches between images 2 and 3

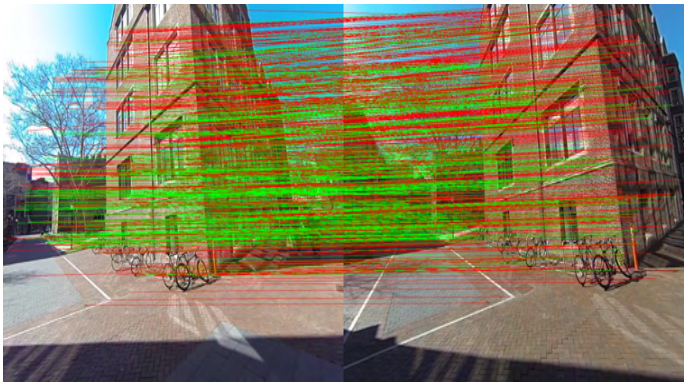


Fig. 9. matches between images 2 and 4

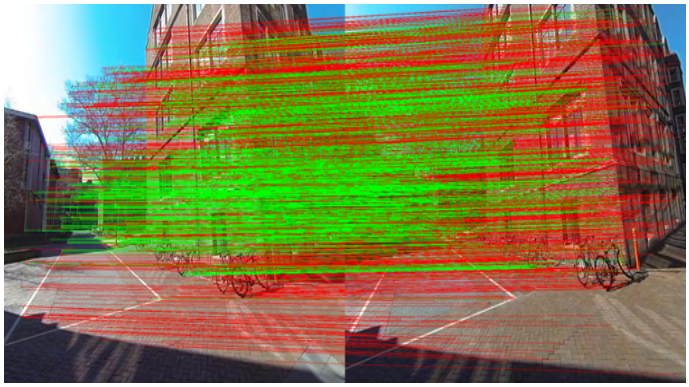


Fig. 10. matches between images 3 and 4

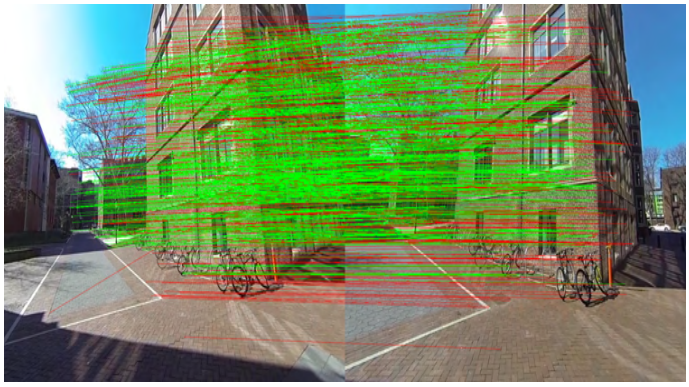


Fig. 11. matches between images 3 and 5

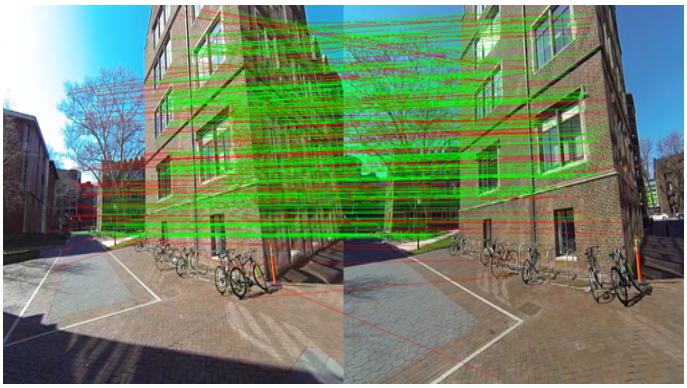


Fig. 12. matches between images 3 and 6

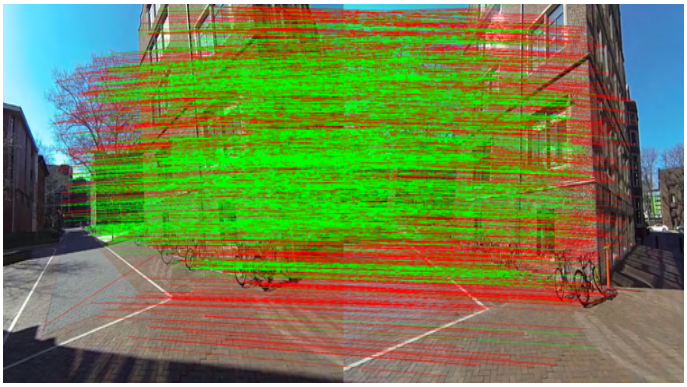


Fig. 13. matches between images 4 and 5

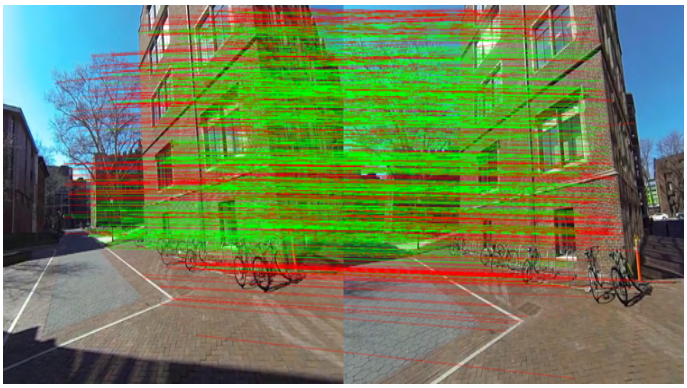


Fig. 14. matches between images 4 and 6

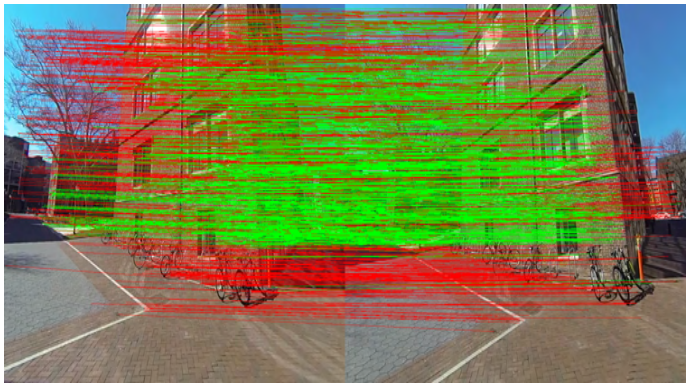


Fig. 15. matches between images 5 and 6