CMSC733-Project 3: Structure From Motion

Rohan Chandra Swati Singhal

***Abstract*—Structure from Motion from multiple views using bundle adjustment. We used a different solver algorithm for non-linear triangulation. We also iclude a pruning algorithm after non linear triangulation that helps in reducing re-projection error.**

I. VisualSfM Output

Following is the attached result from visual sfm.

Fig. 1. Visual SfM output

As you can see, the results from the software are not perfect. Our algorithm outperforms the result we get from the software.

II. Two View Reconstruction

We perform the following algorithm:

A. RANSAC on matching pairs

Raw Feature Matching:

Fig. 2. Feature matching before RANSAC

After RANSAC:

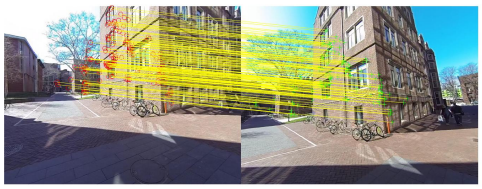


Fig. 3. Feature matching after RANSAC

B. Fundamental Matrix Computation

F =

$$\begin{bmatrix} -1.8459e^{-07} & -7.8618e^{-6} & .0024 \\ 8.9358e^{-06} & 1.2166e^{-07} & -0.0056 \\ -0.0035 & 0.0039 & 1 \end{bmatrix}$$

C. Essential Matrix Computation

E =

$$\begin{bmatrix} -0.0119 & -0.9265 & -0.2957 \\ 0.9906 & 0.0062 & 0.0425 \\ 0.1305 & -0.2226 & -0.0656 \end{bmatrix}$$

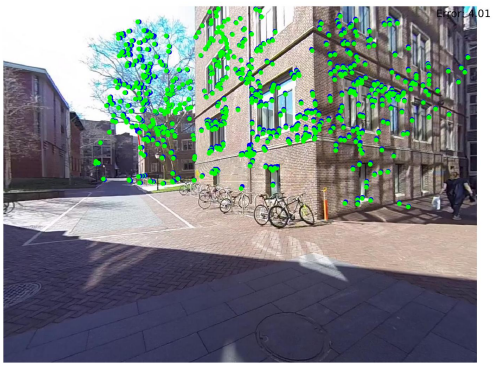
D. 2D Reprojection After Non - Linear Triangulation

This was the first roadblock that we encountered. For a long time we were getting a huge reprojection error. This was solved by fixing two things:

The first problem that I noticed was that our fundamental matrix constraint was much larger than ϵ. As you know, $x^T F x^{'} = 0$ ensures that the points x and $x^{'}$ lie on the epipolar plane and project to a 3D point. The problem was that in addition to the corrections suggested in the pdf, we were required to normalize the fundamental matrix by setting the last value to one.

The second fix that we made to our code was to do efficient data processing.

After getting mixed re-projection results, we realized we needed to filter our results. We thus only retained points that converged according to our non linear optimization.



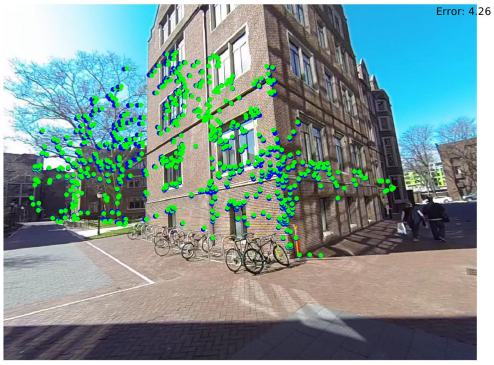


Fig. 4. 3D reprojection of 2D points. Blue indicate original points. Green indicate projected points. Error is displayed at the top

An interesting observation is that using the Levensberg-Marquardt algorithm on a linux machine generated sparse 3D points after the pruning step. This may be because due to some hardware configuration or alternate parameters, the solution set converged for very less points and our pruning algorithm rejects those points.

In order to get around this particular problem, we used the SNOPT solver in the TOMLAB solver library which gave significantly better results.

E. 3D visualiztion of Linear and Non Linear Triangulated Points

Figure 5 shows the initial results of our algorithm.

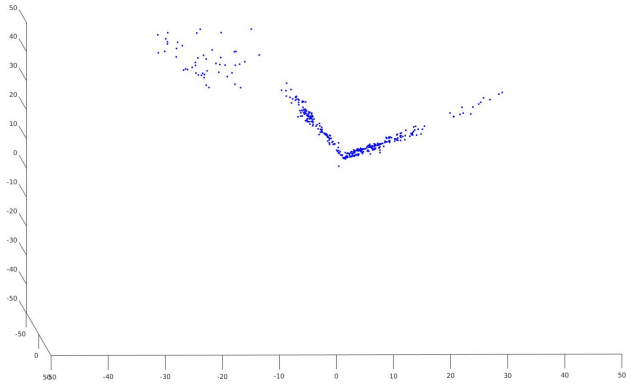


Fig. 5. Top View Reconstruction for 2 views

The two view reconstruction is sparse. Following this stage, we register the rest of the images in succession using pose estimation and apply bundle adjustment to get the following reconstruction.

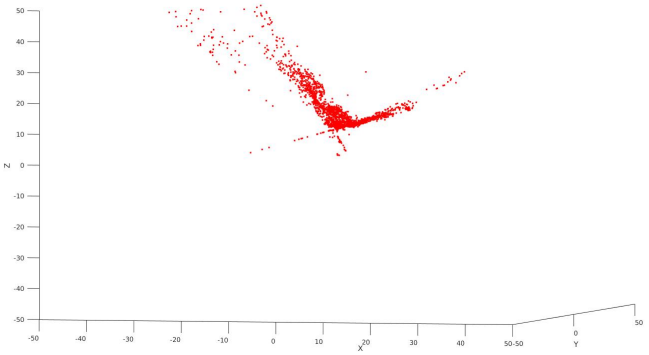


Fig. 6. Top View Reconstruction after Bundle Adjustment

The dense point cloud is a serious improvement over the sparse one.

It is worth mentioning that the bundle adjustment package was not easy to install and use. It took several days of painstaking software engineering to get the package to work. Due to compatibility issues with Mac computers, we installed SBA (the bundle adjustment package) on a linux machine.

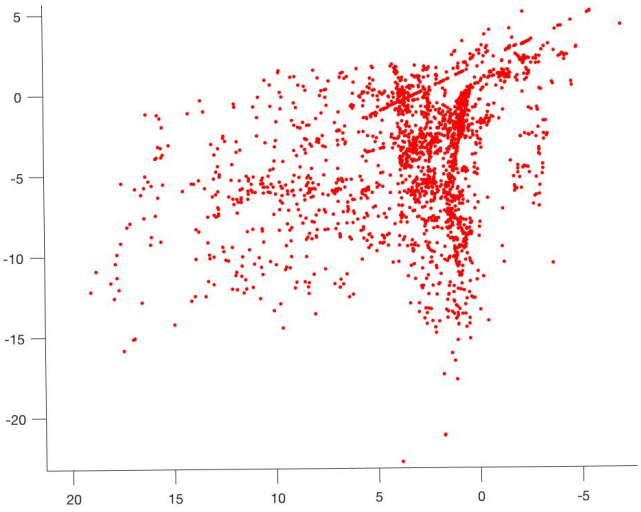


Fig. 7. Oblique View Reconstruction after Bundle Adjustment

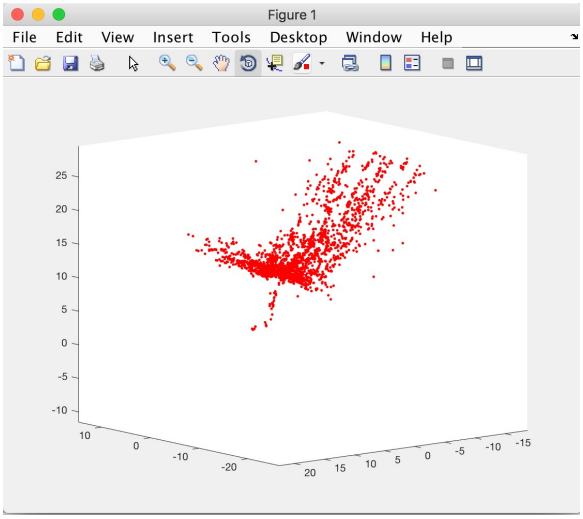


Fig. 8. Side View Reconstruction after Bundle Adjustment

III. Extra Credit

We used my iPhone 4S to capture the following images for extra credit:

Fig. 9. Images for Extra Credit

Selection of relevant and proper images is extremely important for performing reconstruction. Relevance would ensure we recover a structure that can be recognized easily from the point cloud and proper entails using images with 'good' feature points. It took us several tries to gather the images that we were satisfied with.

After securing the images that we wanted, we used the vl_feat library to perform feature detection and matching. We found that the vl_feat library considerably outperforms the SURF points from MATLAB. We procured over 2000 matching points from the former while obtaining only 400 from the latter.

The results of RANSAC can be shown below:

Fig. 10. RANSAC after feature matching

Apart from feature matching, we were required to calibrate the camera before taking images. In our project, we were given both the point matches and the camera matrix, K. In order to calibrate the camera and calculate the intrinsic matrix, we followed the steps in homework 1 of CMSC 733 where we use a checkerboard of square size 21.5 mm. Following calibration, correction was made for lens distortion.

After the required processing steps of calibration and feature matching, we apply our algorithm for calculating triangulated and refined 3D points. The re-projection error is shown in fig. 11. The blue points(under the green) indicate the original points and the green points indicate the re-projected points. The fact that you see so very little blue is an indicator that our non-linear triangulation worked perfectly.

Fig. 11. 3D reprojection of 2D points. Blue indicate original points. Green indicate projected points. Error is displayed at the top

For the purposes of demonstrating our algorithm we show the front view of the reconstructed laptop. Notice the right side in fig. 12. The two red columns and the 3 little red clouds of points on top correspond to their green counterparts in fig. 11. The lines of code on the laptop screen are also quite visible in our experiment.

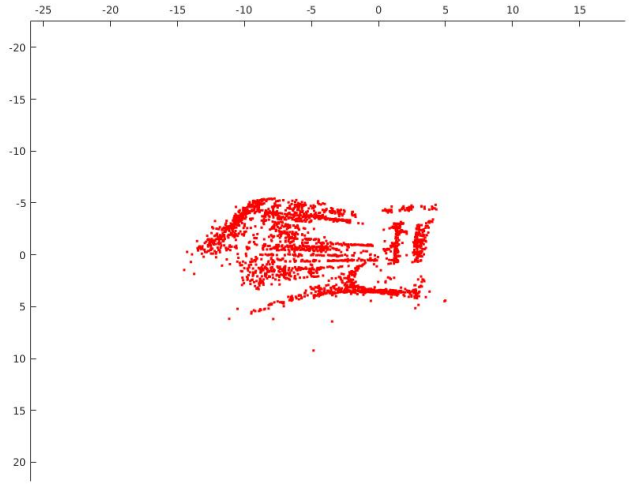


Fig. 12. Front View. Notice the familiarity