Computer Vision 2
Assignment 2

Martín de la Riva, Nuno Neto de Carvalho Mota
11403799, 11413344

May 2018

Introduction

In this assignment we will perform Structure From Motion, an algorithm to reproduce the shape of a 3d object by using SIFT descriptors while moving the camera along the object.

1 Fundamental Matrix

In this section we performed the Fundamental Matrix using (1) eight-point algorithm, (2) normalized eight-point algorithm and (3) normalized eight-point algorithm with RANSAC.

In the following images we show the epipolar lines of 2 images using the different methods. We first tested images 1 and 2. As they are so similar, the difference between methods cannot be fully seen. Here we use images 1 and 25, in which you can clearly see the improvement when changing methods. The two methods without RANSAC are completely off, being the normalized one somehow better, while the RANSAC method looks completely accurate.

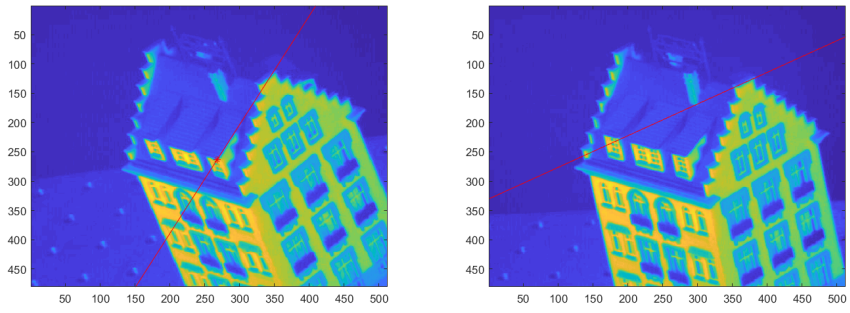

Figure 1: Epipolar lines with eight-point algorithm Fundamental Matrix.

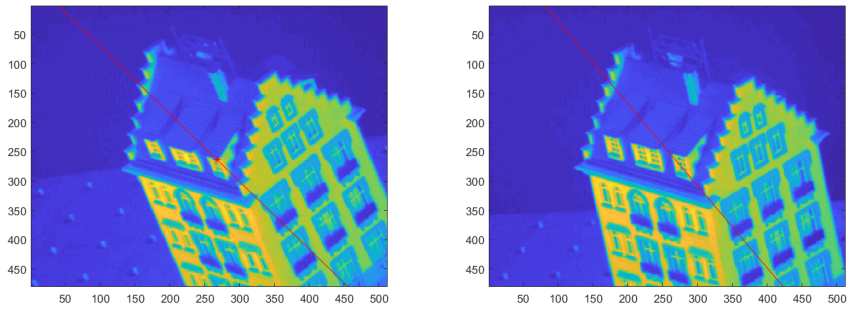

Figure 2: Result of epipolar lines based on normalized eight-point algorithm Fundamental Matrix.

For the RANSAC method we experimented with qualitative results and found 10^{-4} a good value for the threshold.

The function *fundamentalMatrix* draws the epipolar lines and returns the Fundamental Matrix of two images, given as a parameter which method to choose (each method is performed by a self-explanatory name function: *eightPointAlgorithm*, *normalizedEightPointAlgorithm* and *normalizedEightPointAlgorithmRANSAC*).

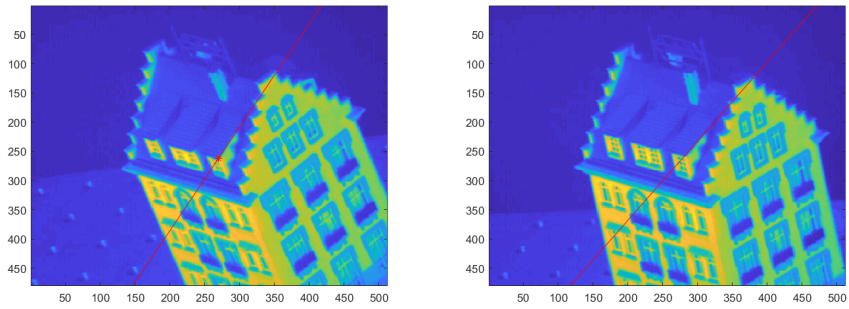

Figure 3: Result of epipolar lines based on normalized eight-point algorithm with RANSAC Fundamental Matrix.

2 Chaining

For the chaining method we developed the function chain(), which computes the match graph. This match graph has as the columns the different inlier points identified with the RANSAC method previously described, and as rows the x and y dimensions of each frame. It is interesting to verify that at the beginning most of the points are shared among the frames (due to the images being very

similar), while towards the end we start only being able to keep fewer points (Figure 4).
It is also important to notice that it is possible that a significant portion of the identified matching points might not be very accurate. This could, up to some degree, be improved by properly tuning vlfeat's hyper-parameters, in order to return fewer not very informative points (such as background noise) or even controlling the quality of the matching points. This hyper-parameter search was not performed, though, as it was not requested in the assignment.
Using the basic configuration a total of around 3800 distinct points were found in the match graph.

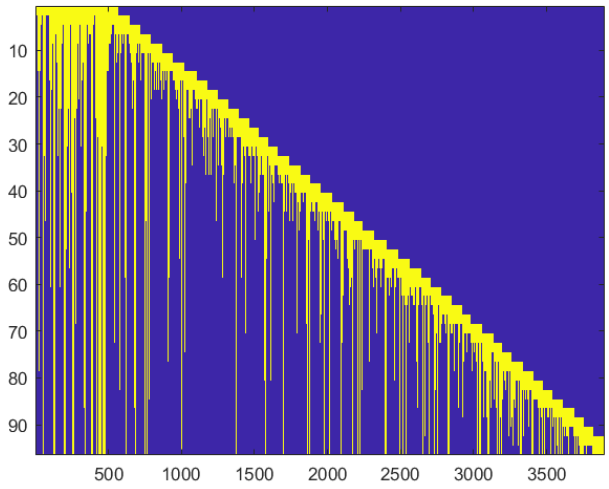

Figure 4: Visualization of the Point View Matrix. Columns are points and rows are images. Each point is labeled 1 if it's in the matrix, 0 if it is not.

Later on it was determined that a total of around 105 points were common to all frames, yielding the results presented in the next section.

3 Structure from Motion

On this part of the project we followed the implementation described in the assignment sheet provided. Unfortunately we had some troubles when attempting to stitch iteratively, but we weren't able to determine what caused this issue. We suspect that it might be due to the procrustes method returning an translation parameter for each individual point. However, since we determine the transformation using only the common points to the previously accumulated

point cloud and the point cloud associated with the new dense block, we cannot simply apply it to all new points. We tried to solve this issue by taking the mean of the translations provided by the procrustes, but perhaps our reasoning was not correct. As such, we do not show results for the iterative method, but they can easily be obtained my running main with the appropriate parameters specified.

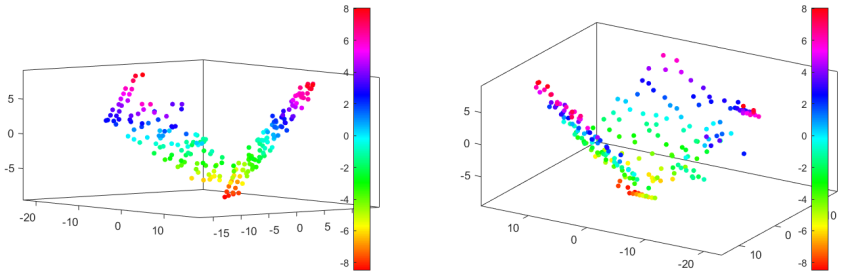

Figure 5: 3D final result on given *PointViewMatrix.txt*.

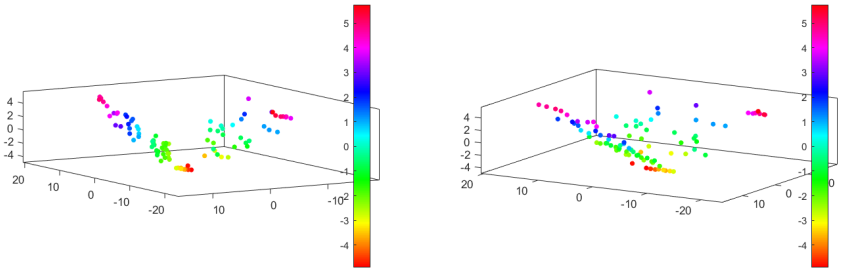

Figure 6: 3D final result on our generated Point View Matrix

4 Additional Improvements

We implemented no additional improvements, but some of them might be, for example, removing background noise (by tuning hyper-parameters or retaining only the features that are inside a bounding box on the house, for example) or better determining the matches between frames.

5 Running experiments

From file *main.m* all experiments can be performed. Select which assignment section to perform (changing variable *section* to *3.1*, *3.2*, *3.3*, *4* *5*). Specific parameters from each section can also be tuned.

6 Self-evaluation

Both students divided the work evenly both in documenting, implementing and writing the report.

7 Conclusion

In conclusion we were able to successfully implement the structure from motion algorithm (with exception to the iterative approach, which, again, we suspect is due to meaningless bug or the procrustes returning individual translation parameters (see above)). It is very interesting to see that even with keeping only the points that are common to all frames we get results very similar to the ones obtained the the point view matrix provided for the assignment.
Overall the method seems powerful and useful.