Computer Vision 2: Assignment 2 Report

Ya-Chu Yang
(11571276)
qwerthjkl45@gmail.com

Tomas Fabry
(11868414)
tomasfabry1@gmail.com

Alexander Hustinx
(11246545)
alexanderhustinx@gmail.com

1 Fundamental Matrix

In this section, we calculated the fundamental matrix, which is used to obtain the epipolar lines. First, we used SIFT to get corresponding points in each image, and performed RANSAC to estimate the homography between images. Further steps to obtain the fundamental matrix will be described below.

1.1 Eight-point Algorithm

We choose eight point corespondences in each pair of image to build a matrix A. Through finding the SVD of A and only extracting the two largest singular values and the corresponding vectors to force rank of fundamental matrix to be two, we can obtain three new matrices, which are multiplied together to get the fundamental matrix.[1]

1.2 Normalized Eight-point Algorithm

All points are normalized in this approach before constructing matrix A to improve the stability. Points are multiplied with transformation matrix T and T' to ensure their mean is zero before performing eight-point algorithm. Note that, after getting the fundamental matrix, the matrix should be denormalized again.

1.3 Normalized Eight-point Algorithm with RANSAC

This approach is performing normalized eight-point Algorithm via a RANSAC-based approach. The whole process is calculating a fundamental matrix through normalized eight-point algorithm and using Sampson distance with threshold 0.5 to count the number of inliers, which are corresponding points that agree with the fundamental matrix. The whole process in our experiment is iterated 100 times to pick the best fundamental matrix leading to the largest set of inliers.

1.4 Results

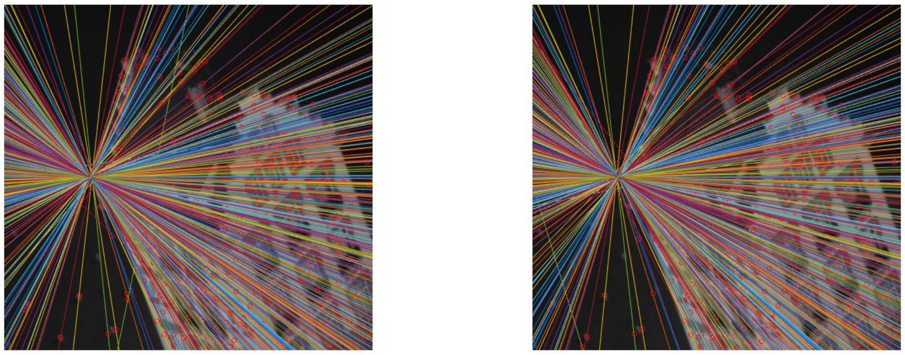


Figure 1: Epipolar lines on 1^{st} and 2^{nd} frame using the basic eight-point algorithm

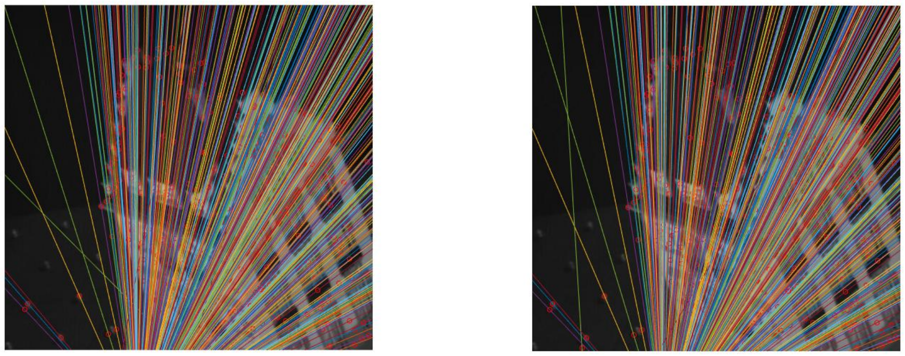


Figure 2: Epipolar lines on 1^{st} and 2^{nd} frame using the normalized eight-point algorithm

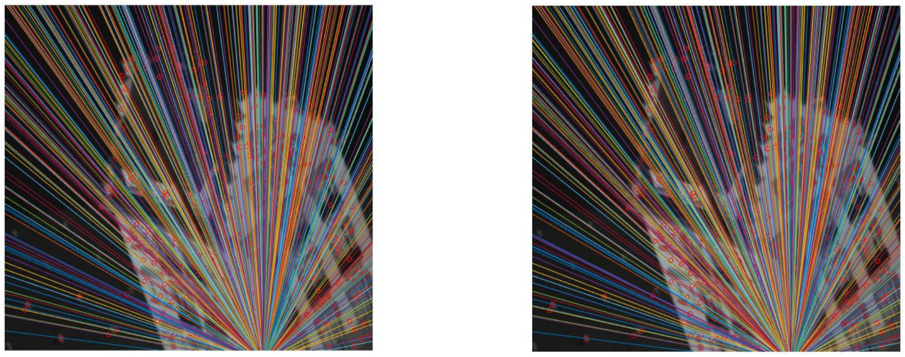


Figure 3: Epipolar lines on 1^{st} and 2^{nd} frame using the normalized eight-point algorithm with RANSAC

For these three approaches, the epipolar constraint is satisfied when every points lies on their corresponding epipolar lines.[2] However, epipolar lines obtaining from the first two approaches, the eight-point algorithm and the normalized eight-point algorithm, are not plausible since lines are not intersecting at same epipoles (see Figures 1 and 2). The last approach seems much more reasonable, because all epipolar lines intersects at the same point (see Figure 3), which means that the point of interest in both cameras and the centers of two camera lenses lie on a single 3D plane.

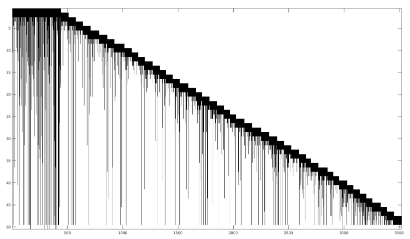
Further, we can see some improvement from the first to second method, and the second to third method. The first approach, is without proper data processing, thus, the result can be very sensitive to the noise. As we can see in Figure 1, the estimation of the camera pose is not really reasonable. Therefore, proper data normalization is added in the second method. The performance seems much better compared to the first one. However, this method attempts to minimize algebraic error($X^{'}FX = 0$) from eight points instead of finding the fundamental matrix, minimizing the geometric error. As a result, the estimation of camera pose still is not plausible (see Figure 2).

To further solve this problem, RANSAC is performed based on the second approach, to create a normalized eight-point algorithm with RANSAC. Through RANSAC, a better fundamental matrix is found, minimizing Sampson distance between two corresponding points in two images. In Figure 3, we can see much more accurate the estimation of position of the epipole is than the first two methods.

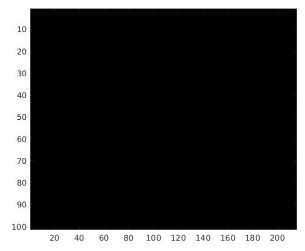
2 Chaining

In this section, we construct a point-view matrix (PVM) to visualize the matching process in Section 1. This matrix is a logical matrix in our implementation. The rows of the matrix represent frames and the columns represent the 3D points. Thus, if one 3D point is contained in two consecutive frames, the elements at the point column and the two frame rows in the matrix will be marked as one; otherwise, it will be zero.

2.1 Results



(a) Point-view matrix from the house dataset



(b) Point-view matrix from PointViewMatrix.txt

Figure 4: Point-view matrix

2.2 Results Analysis

For the point-view matrix derived from the house dataset, it seems like a diagonal matrix, and each frame row overlaps with their corresponding next frame row, meaning interest points in one frame can find their matching points in the next frame. Thus, this matrix somehow describes the matching process from previous section. Furthermore, we can see the first several columns of matrix are much more dense than other columns. That might be caused by some salient interesting points existing in every frames, for instance, some corner at the rooftop that we can observe in multiple frames.

Though this point-view matrix describes the matching process in some way, however, we can see that the matrix is sparse, and the value of the elements above the diagonal in the matrix are all zero (see Figure **??**), which means that there aren't any corresponding points in all previous frames to the interesting points in the current frame. This situation is rare, thus, we would say the matrix from the house dataset is not really plausible.

2.3 Comparison between two Point-View Matrices and Improvement

The point-view matrix from the supplied txt-file is much more dense than the one from house dataset. We can see that the denser point-view matrix solves the problem that we mentioned in the previous sub section, that the value of the left top elements are all zero.

To have better 3D reconstruction, having a dense point-view matrix is necessary. Thus, to change the sparse matrix(4a) to a dense matrix, finding the corresponding points to the current frame should not only be limited to the previous single frame. The interesting points in the current frame should be matched with the points from the first frame to the last one frame. With this improvement, the matrix from the house dataset could become denser.

3 Structure from Motion

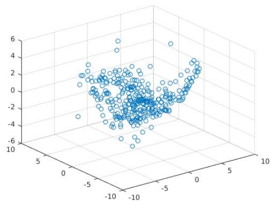
In this section, we show the results to our structure from motion (SFM) algorithm. We show results when obtaining the point cloud from a single dense block, when obtaining the point cloud from 3 and 4 consecutive images, the point cloud constructed from the supplied PointViewMatrix txt-file and we show this for two different structure-motion decompositions. The two decompositions we used were:

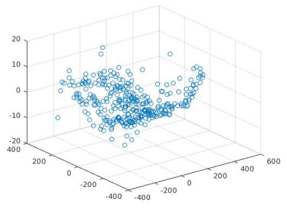
$$M = U_3 W_3^{\frac{1}{2}} \quad S = W_3^{\frac{1}{2}} V_3^T \tag{1}$$

$$M = U_3 \quad S = W_3 V_3^T \tag{2}$$

Note, the variables in the decompositions stand for the basic variables used in SVD and SFM. Finally, we analyze these results and our findings.

3.1 Results


(a) Using decomposition 1 with set size 3


(b) Using decomposition 2 with set size 3

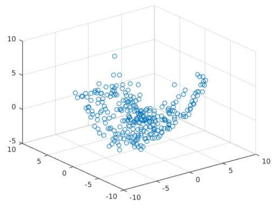
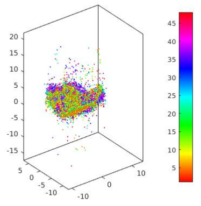
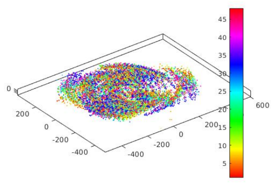

(c) Using decomposition 1 with set size 4

Figure 5: Point cloud obtained from 1 single dense block from house dataset


(a) Using decomposition 1 with set size 3


(b) Using decomposition 2 with set size 3

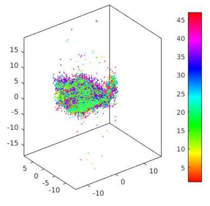
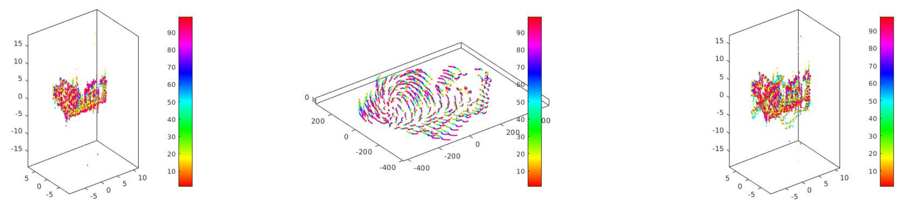

(c) Using decomposition 1 with set size 4

Figure 6: Point cloud obtained from mutiple dense blocks from house dataset



(a) Using decomposition 1 with set size 3

(b) Using decomposition 2 with set size 3

(c) Using decomposition 1 with set size 4

Figure 7: Point cloud obtained from mutiple dense blocks from PointViewMatrix.txt

3.2 Result analysis

The different decompositions shown in Figures 5a and b display seemingly comparable results except for the second decomposition b, which have a much higher value although it holds the same structure as the others. The reason why they all look similar to each other might be the points clouds are generated by one single block from PVM. Thus, the difference between three images are hard to differentiate.

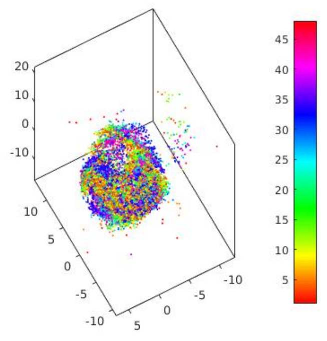
As can be seen from Figures 6a and c, and Figure 7a and c, they show that the point clouds appears less noisy for set size 3 than for set size 4. The results are contrary to our thoughts that larger set size contributes better results. However, we have only tried set size 3 and 4, the relative performance of these two set size are not that huge. Thus, we only can tell if the larger set size leads to better performance by increasing the difference between two set size. Further more, from Figures 7a and b, we can see difference caused by affine ambiguity. These two are all similar to the house, however, one is more flat than the other one.

Compared to the our results, the supplied PointViewMatrix txt-file seems more detailed. As explained in Section 2, we assume this is at least partially caused by our sparse PVM compared to their dense PVM.
Again, the sparsity of our PVM, compared to the supplied PVM can be seen quite well in Figures 6b and 7b.

3.3 Additional improvements

For additional improvement, we try to get a denser point-view matrix. The chaining process does not only find the matching points between points in previous one frame and the current frame, it also traces back more previous frames to find the corresponding points in the current frame.

3.4 Results


(a) Before improvement

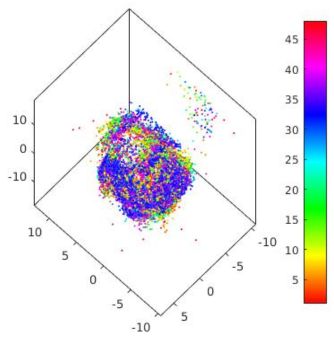

(b) After improvement

Figure 8: Point cloud from the house dataset with set size 3 and decomposition 1

3.5 Result analysis

As we can see from Figure 8, at the certain angle, we can see the house structure after the improvement slightly clearer compared to the one without the improvement. The reason why the result doesn't get a huge improvement might be the lack of data, leading to the incomplete and rough house. Also, the other potential reason is that our matching algorithm doesn't find enough matching points in two frames. Thus, it is hard to build a dense point-view matrix.

4 Self-evaluation

All team members have contributed to the report as well as implementation.

References

[1] Andrew Zisserman Richard Hartley. *Multiple View Geometry in computer vision*. 2004.

[2] Wikipedia. Epipolar geometry — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Epipolar%20geometry&oldid=821304424`, 2018. [Online; accessed 14-May-2018].