Computer Vision 2 Assignment 2

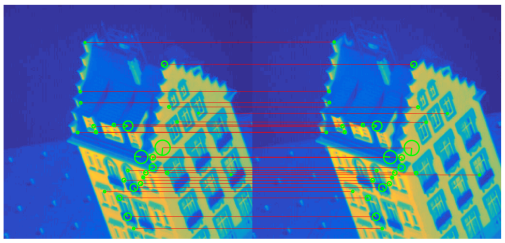
Kasper Bouwens Wolf Vos

May 2016

1 Fundamental Matrix

In this section we performed the normalized eight-point algorithm with ransac with different inlier tresholds and a different number of points to create the fundamental matrix.

Figure 1: Ransac with 8 points



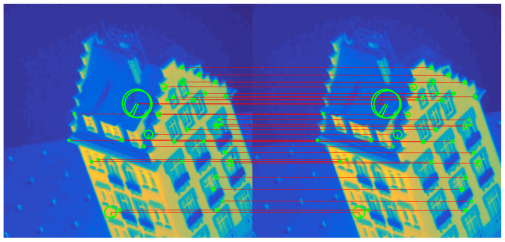
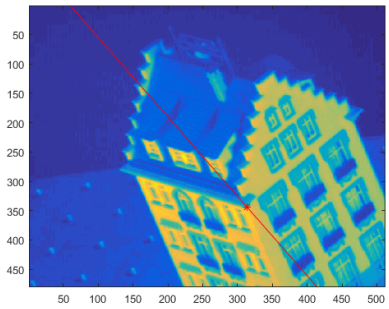
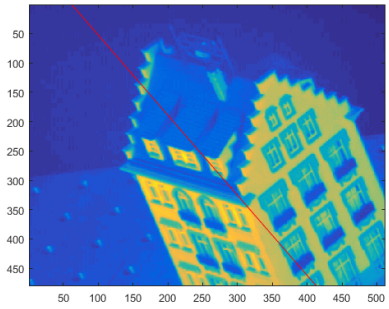


Figure 2: Ransac with 20 points

Performing RANSAC with twenty points could have resulted in a more accurate calculation of the fundamental matrix however, we are unsure about this.



(a) The figure in which we clicked



(b) The figure where the line was drawn

Figure 3: Epipolar lines

The treshold for the inliers was first randomly chosen to be 10^{-8}. We then found that our model was penalising the matches too hard and thus decided to set the treshold at 10^{-4}

The function SFM loads two images and calculates the descriptors, finds matches between them and (depending on the mode) normalizes the points and performs RANSAC. Calculating the fundamental matrix is done in the function FundamentalMatrix().

The last part was showing the epipolar lines. We included an example of this.

2 Chaining

For the chaining part we created a new function called mergescenes which constructed 4 matrices. One in which was the pointview matrix (which later seemed not to add any information), one with the indexes found int the matches in stead of ones and two for the x and y coordinates of the points in each frame. These could later be used in the structure from motion part. We eventually found that

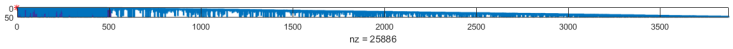


Figure 4: The structure of the pointview matrix

101 points where present across all images.

3 Structure from motion

In this part we combine the point view matrix (D) obtained in the previous part to create the structure (S) and the Motion (M) matrices with the getMS(pointView) function. If we remove all the columns (points) with 0 in them in the full D matrix we obtain an image of a house (if you use some imagination). Unfortunately the procrustes analysis didn't work as good as we hoped but an improvement for this function would be to run the procustes a couple of times with randomly samples points and then take the transformation with the smallest distance/error from the matching and use that transformation on all the data points. This will provide you with all the points (which it fails to do now) and it will do the transformation with a good estimate of the ideal transformation.

The structure from motion part works on the example pointView matrix and it also works on our dense pointview matrix with all the sparse columns removed. This can be seen in the figures below:'

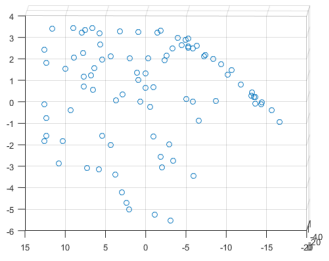


Figure 5: Side view of our dense matrix S

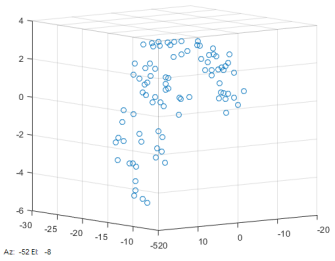


Figure 6: Top view of our dense matrix S

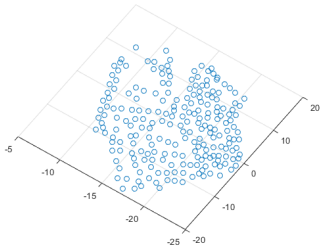


Figure 7: Example pointview matrix 1

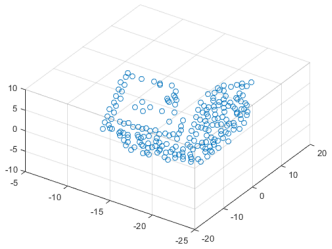


Figure 8: Example pointview matrix 2