Computer Vision 2
Assignment 2: Structure from Motion

Riaan Zoetmulder
MSc Artificial Intelligence
University of Amsterdam
riaanzoetmulder@gmail.com

Minh Ngo
MSc Artificial Intelligence
University of Amsterdam
nlminhtl@gmail.com

1 Fundamental Matrix

In the first part of the assignment we calculated the Fundamental Matrix. We firstly detected matching interest points in the image pairs using SIFT. Secondly, we randomly picked 9 points and ran the eight point algorithm, the normalized eight point algorithm and the normalized eight point algorithm with RANSAC. A brief description of each of the algorithms is given shortly. Using the Fundamental matrix we obtained, we could calculate the epipolar lines. Furthermore, we checked the correctness of the fundamental matrix by checking whether the following was true: $\vec{p}^T \mathbf{F} \vec{p} = 0$ and rank(F) = 2.

1.1 Eight Point Algorithms

1.1.1 The Eight point algorithm

In order to run the eight point algorithm we constructed the matrix A, ran a singular value decomposition, and obtained the matrix F from the column in the V matrix corresponding to the smallest eigenvalue. This method can however be improved, by making sure that all of the images have the same origin. This can be achieved by normalization of the interest points.

1.1.2 The Normalized Eight point algorithm

In order to ensure that the mean of the points is 0 and the average distance to the mean is $\sqrt{2}$ we multiply the points by a transformation matrix T and T'. We than create the matrix A and perform a singular value decomposition on it. The matrix $\tilde{F}$ is obtained from the last column of V. This is restructured into a 3x3 matrix. We perform another singular value decomposition on $\tilde{F}$. Set the smallest singular value to 0, and recalculate $\tilde{F}$ using $\tilde{F} = \tilde{U}\tilde{D}\tilde{V}^T$. Lastly, we denormalize this matrix. This still leaves us to deal with getting a good sample from the points, such that we calculate the fundamental matrix that maximizes the amount of inliers. We use RANSAC to deal with this problem.

1.1.3 The Normalized Eight point algorithm with RANSAC.

RANSAC is an algorithm that selects n random points from a population of points. It than calculates the fundamental matrix using the normalized eight point algorithm. Whichever fundamental matrix ensures the most inliers, is stored. If a fundamental matrix is found that has more inliers, the previous one is replaced. This continues for a preset amount of iterations.

Epipolar lines

An epipolar line is the line of intersection of the epipolar plane with the image plane. This line necessarily goes through the epipole, therefore all of the epipolar lines must converge at some point.

We ran the normalized eight point algorithm with ransac for 1000 iterations. We set the threshold for inliers to 1. The resulting epipolar lines are shown in Figure 1. Because all of the lines intersect at the epipole we believe that this result is correct.

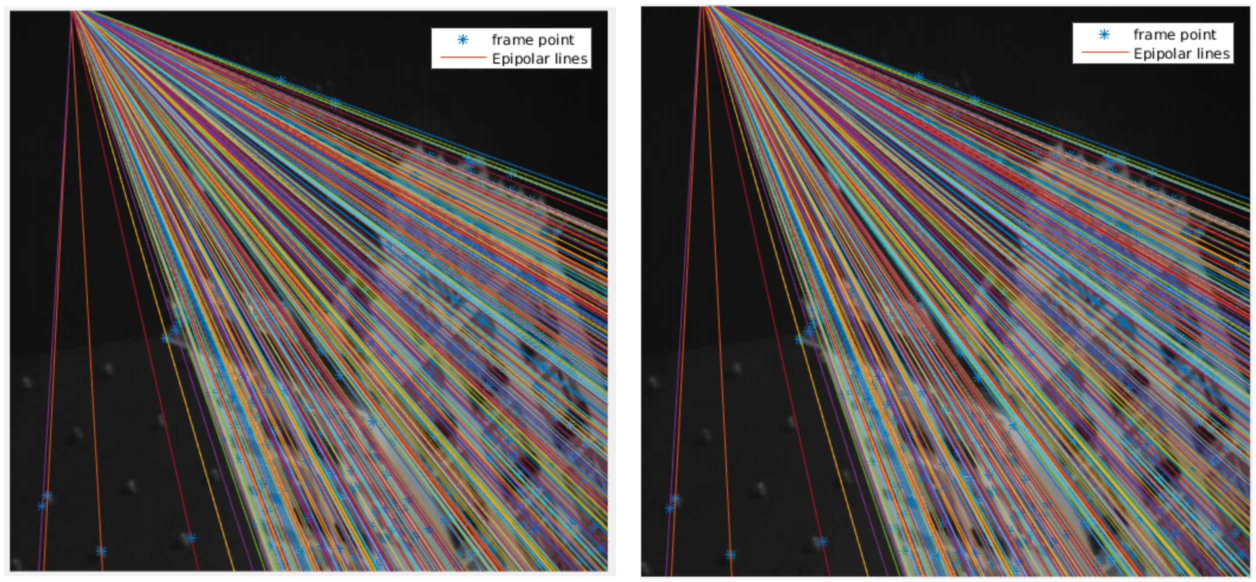

Figure 1: Epipolar lines on images 1 and 2 of the House.

2 Chaining

In order to perform the basic chaining we iterated over all of the image pairs and performed the matching process using the fundamental matrix that was found by RANSAC. For each newly matched point that didn't already have a column, we introduced a new column in the point view matrix. If this point was found in another image, we did not add a column but we noted for this image that the point was also found there. When we found multiple matches for a point, this point was ignored. A point view matrix computed from all pair of consecutive frames is presented in the figure 2.

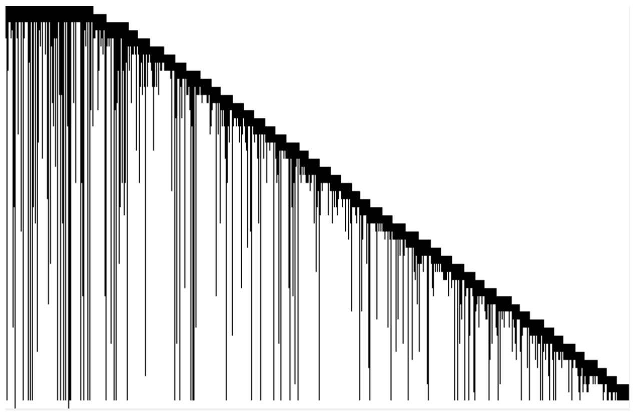

Figure 2: Point view matrix for all pair of frames of the House dataset

3 Structure from Motion

We follow the pipeline introduced from the lecture of Jan van Gemert. We are implementing an iterative algorithm. In each iteration we are taking M consecutive rows of the point view matrix represented matches from K+1 pairs of frames into consideration and find a dense block of them by taking columns where there are all coordinates presented. We normalizing coordinates by substracting mean and then construct a measurement matrix D of the size $2 * M \times N$ where N is a number of points. The D matrix is decomposed later into components $U \times W \times V^T$. The first 3 columns of U and V, and the 3x3 top-left block of W are taken into account to compute a motion and shape matrix. Takashi and Kanade are computing them in the following manner:

$$M = U \times W^{1/2}$$

$$S = W^{1/2} \times V^T$$

For our case we have scaling parameter for the Z axis:

$$S(3,:) = 10 \times S(3,:)$$

since it better reconstructs the shape from our dataset. With $S = W^{0.5} \times V^T$ a flatter shape was obtained [Fig 3]. We have experimented with 2, 3 and 4 consecutive images for 3D point cloud estimation. The obtained insight was that more consecutive images are taken into account - less outliers will be, but less 3D points can be derived at the same time, therefore there should be some trade-off between those criteria.

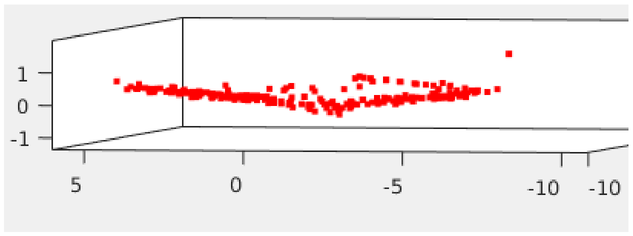

Figure 3: 3D house shape reconstructed as $S = W^{0.5} \times V^T$

4 Additional improvements

4.1 Taking several frames in consideration to build PVM

We have introduced an approach to deal with sparsity of the point view matrix. Instead of considering only (N-1)-th and N-th frame in each iteration we are determining keypoint matches between frame pairs {N-1, N} and {N-2, N} and consequently running RANSAC for both of them. We firstly consider inliers from the first pair for building the point view matrix. If no matches from the (N-1)-th frame has been found in the previous row of PVM then we consider the (N-2)-th frame. If the point has been matched then we feel the particular point of the N-th frame in this column and set the point in the (N-1)-th frame as an average between points from (N-2)-th and N-th frames. It can be understood that we assumed that a considered keypoint has moved smoothly from the (N-2)-th frame to the N-th frame via the (N-1)-th frame. This approach has significantly reduced the sparsity of the point view matrix [Fig 4, 5].

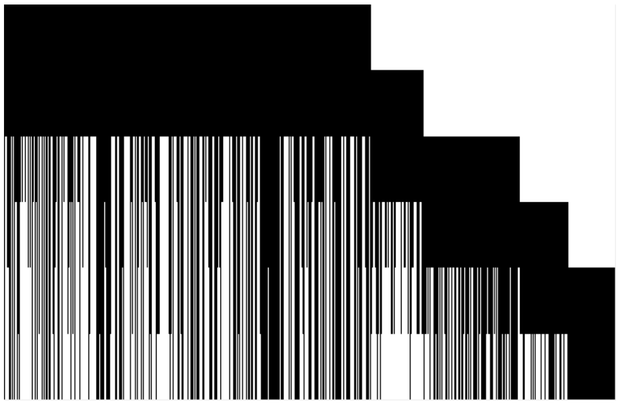
Figure 4: Point view matrix of the House dataset without taking (N-2)-th frame in consideration.

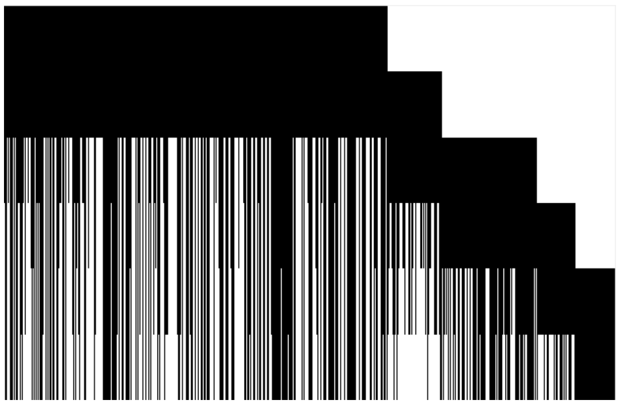
Figure 5: Point view matrix of the House dataset with taking (N-2)-th frame in consideration.

4.1.1 Background keypoints suppression

The house dataset is relatively trivial and doesn't have a lot of outliers or keypoints situated on the background. We have dealt with them by increasing a threshold of the SIFT keypoints matching algorithm.

For the Teddy Bear dataset, this is more complicated, because the background is moving. We assumed that these movements are small in comparison to the object movement and consequently threshold the distance between matched SIFT keypoints to be not so small. This solution has significantly increased the amount of inliers of the RANSAC algorithm.

4.2 Stitching algorithms

We also experimented with different stitching algorithms namely Iterative Closest Point and Procrustes Analysis. In the experiment scenario algorithms didn't provide significantly different results. Stitched point clouds are provided in figures 6 and 7.

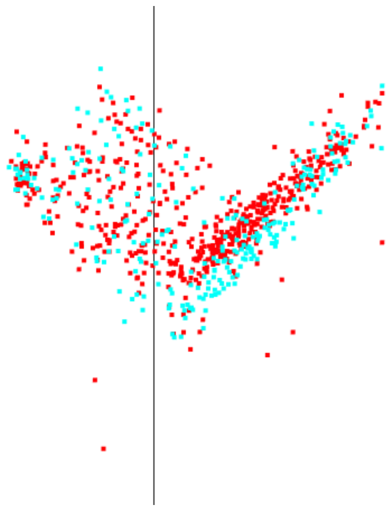
Figure 6: Stitched 3D point cloud of the house dataset generated from 3 consecutive rows of the point view matrix using ICP algorithm.

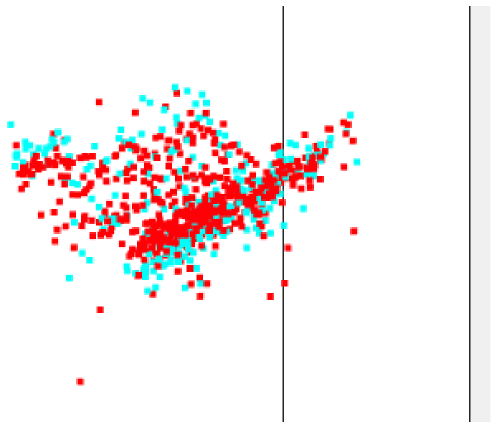
Figure 7: Stitched 3D point cloud of the house dataset generated from 3 consecutive rows of the point view matrix using Procrustes Analysis

5 Experiments & Results

Experiments have been performed to reconstruct the 3D coordinates from the House dataset. A reconstructed shape of the house is introduced in the figure 8 captured from different perspectives. Two perpendicular walls and a roof plane can be detected, since there were more keypoints detected on the house walls they are more distinguishable in our experiment results.

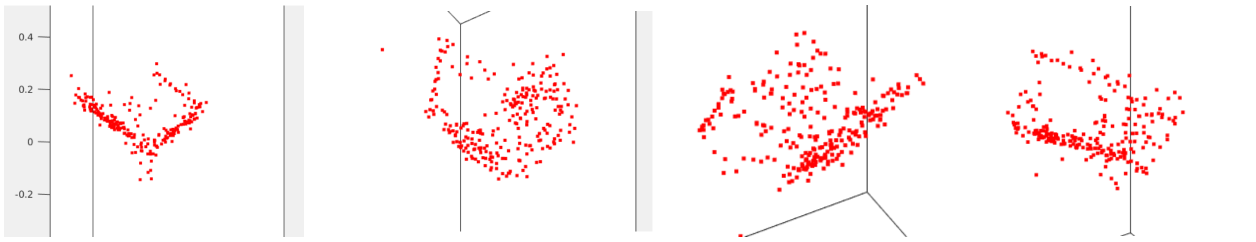

Figure 8: Unstitched point cloud for the house. Roof is clearly visible. Results are presented without stitching to have a distinctiveness in shape.

In addition to the mandatory part of the assignment we have experimented with more advanced dataset with the teddy bear. Our algorithm introduced additional improvements to deal with background outliers and sparsity of the point view matrix and has been able to reconstruct the Teddy Bear shape introduced in the figure 9. However, because of the surface noise I have not managed to stitch Teddy Bear point clouds together with Procrustes Analysis nor with ICP, therefore it will be left as an open question for future investigation.

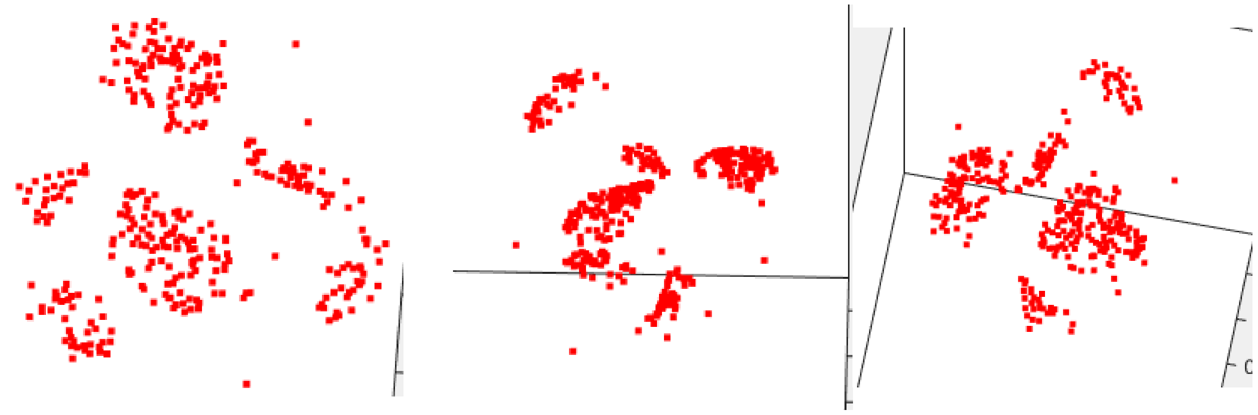

Figure 9: Unstitched point cloud for the bear. Head, feet, arms and belly are clearly visible.