Computer Vision 2 - Assignment 2

Bob Borsboom (10802975), Louis van Zutphen (11851910), Reinier Bekkenutte (13438557)

May 3, 2021

Link to Github: https://github.com/RBekkenutte/CV2/tree/main/Assignment%202

Introduction

In this assignment, we were tasked with creating structure from motion, given a dataset of a rotating house. The idea is that we have multiple images of the same house, but from a slightly different angle. If we can find the corresponding keypoints between these slightly rotated images, we can construct the Point View Matrix of this dataset. This matrix combines the different views and their corresponding keypoints. With that PVM, we can create a 3D reconstruction of the house in question, combining the multiple views. This report is split into three key sections. In the first section we will elaborate on how we computed the fundamental matrix, which is needed to find the mapping between two pictures that have different camera positions. After that, in section two, we will discuss the Point View Matrix we came up with. Section three consists of our attempts to create actual structure from motion, using our Point View Matrix. In section four we will discuss some possible additional improvements and section six consists of the self-evaluation. Finally we will conclude this assignment and with that, our paper.

3. Fundamental matrix

3.0.1 Method

In order for us to create the fundamental matrix, we first had to find keypoints matches between two given frames. This can be done in a number of ways and a number of parameters can be changed to find the optimal setup. First we quantitatively find the best threshold for the distance filter on our keypoints. After that we will compare two keypoint detectors; SIFT and Akaze, to find which one suits our task best.

When a keypoint detector finds matches, each match has two descriptors which in their turn, have a certain distance that accompanies them. When using a matcher like K-nearest neighbors, this distance is a good value to filter on, since it essentially tells us how far apart the descriptors are between the matched keypoints. In figure 1 below you can see the matches of images 0 and 20 of the house dataset, for different thresholds of the distance.
In order to estimate the F matrix (fundamental matrix) we sample at least 8 matches between two frames. We can then fill the matrix A which have the different point on the vertical axis and on the horizontal axis it describes the characteristics in terms of the x and y coordinates. Using the Singular Value Decomposition we can then calculate the fundamental matrix. This matrix tries to transform the images in such a way that the distance between the images is minimized.

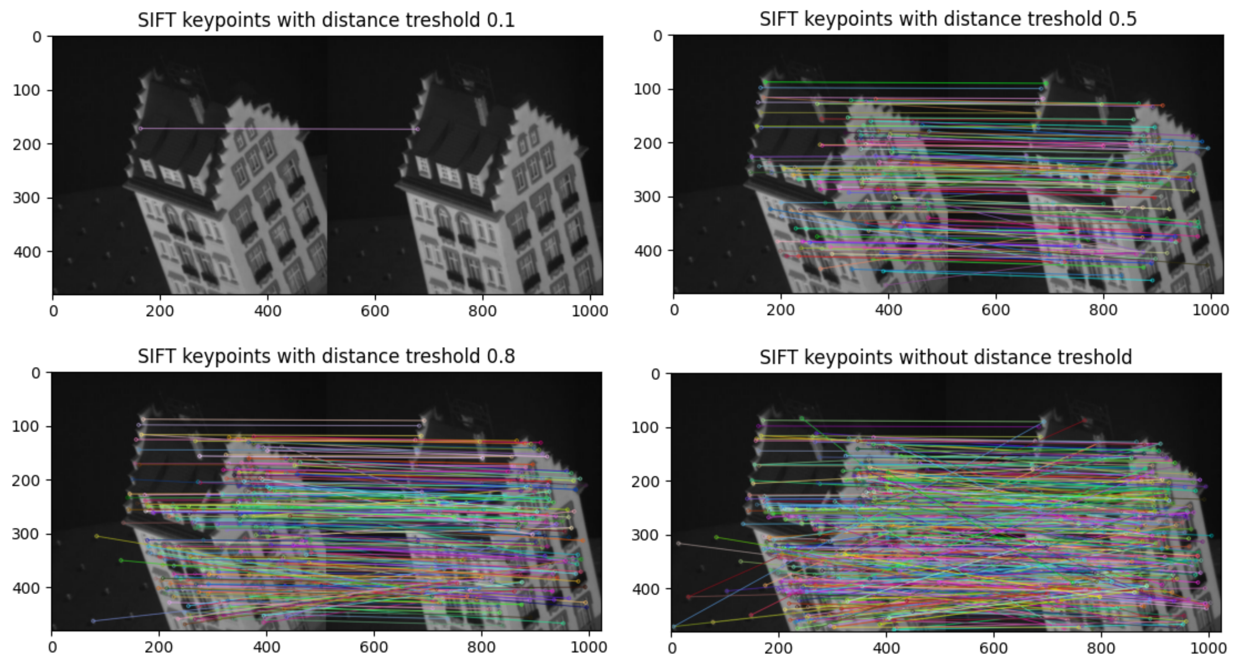
SIFT keypoints with distance treshold 0.1
SIFT keypoints with distance treshold 0.5
SIFT keypoints with distance treshold 0.8
SIFT keypoints without distance treshold

Figure 1: SIFT Keypoint matches for different distance thresholds

As can be seen above, the distance threshold is a good filter for the keypoints, since it regulates the amount of distance that is allowed between the matches. With that, it also implicitly leaves out the outliers, since their distance is inherently high. This is visible in the comparison above.

Next to changing the distance threshold, we also tried to use a different feature detector, Akaze. This detector has more expressive power and with that, is able to detect better feature. A comparison between SIFT and Akaze can be found in figure 2 below.

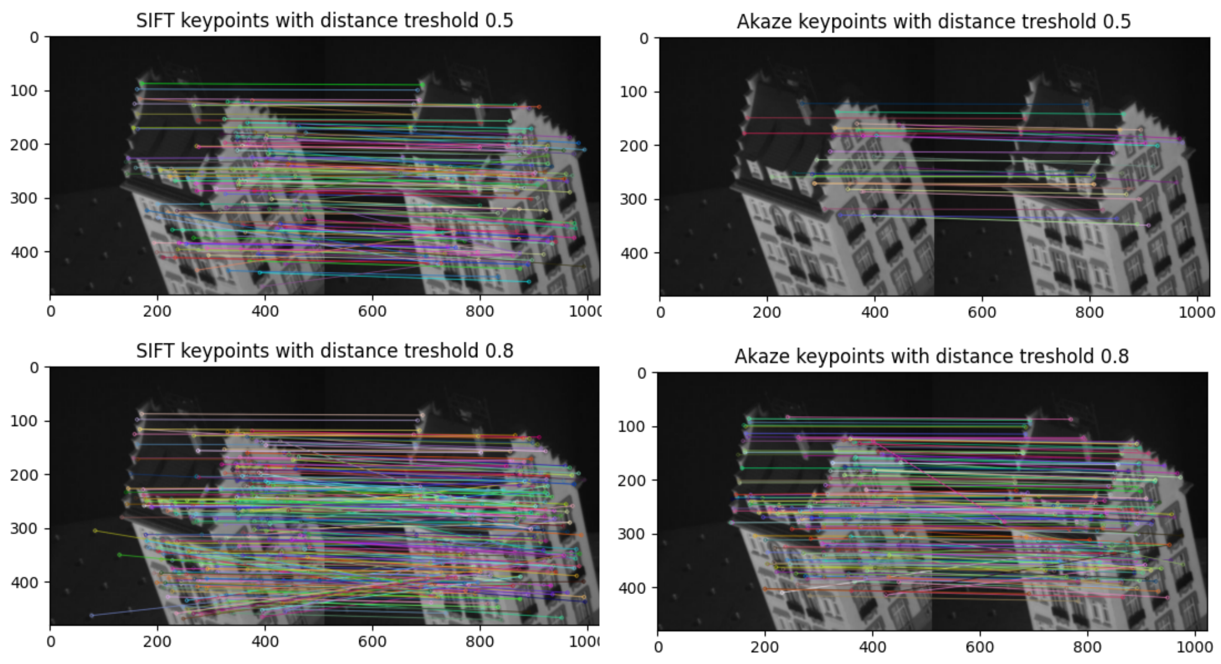
SIFT keypoints with distance treshold 0.5
Akaze keypoints with distance treshold 0.5
SIFT keypoints with distance treshold 0.8
Akaze keypoints with distance treshold 0.8

Figure 2: Difference in keypoint matches between SIFT and Akaze for distance thresholds 0.5 and 0.8

3.1 Eight-point algorithm

The eight-point algorithm will estimate the fundamental matrix between two frames. The fundamental matrix is a 3x3 matrix which relates corresponding points between two frames (rotation and translation parameters).

3.2 Normalized eight-point algorithm

Given the research of (1) it turns out that the performance of the eight-point algorithm improves when the input data is normalized. In this part of the assignment we have normalized the data using the transformation matrix T. We used the formulas as given in the assignment. After the transformation the mean value of the data is zero and the average distance to the mean is close to $\sqrt{2}$

3.2.2 Find a fundamental matrix

Using the normalized data the eight-point algorithm estimated a new fundamental matrix F.

3.3 Normalized eight-point algorithm using RANSAC

In this part of the assignment we tried to find the best possible F matrix between two images given the Sampson distance. This metric tells us what the distance is between the matching points of two frames after applying the fundamental matrix. Ideally the distance should be as low as possible. We used the following approach:

1. Perform the following steps 100 times
2. Sample randomly eight matching points between two frames
3. Calculate the F matrix with the normalized data
4. Apply the Fundamental matrix
5. Calculate the amount of inliers (a point is an inlier when its distance is smaller then a threshold)
6. Save the F matrix

After the 100 steps use the best F matrix (based on the highest amount of inliers.

3.4 Analysis

3.4.1 Analysis Eight-point algorithm

As stated above the fundamental matrix matrix is estimated with the use of the original input data. At least 8 matching points are selected for calculating the fundamental matrix. Looking at the result in figure 3 it can be seen that the epipole is inside the image according to these keypoints. However, this is not the case in the real world, indicating that these results are lacking.

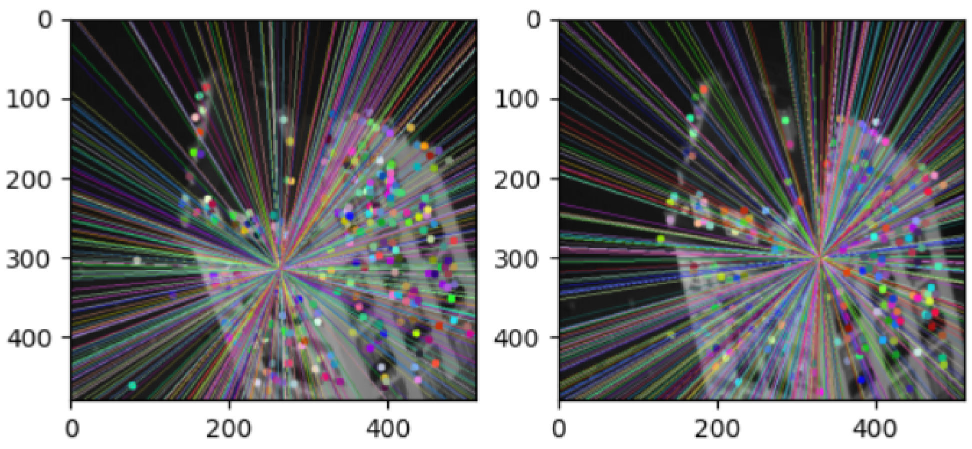

Figure 3: Epipolar lines based on the fundamental matrix using the eight-point algorithm. Frame 0 (left) and 20 (right) are selected.

3.4.2 Analysis Normalized eight-point algorithm

As stated above, before calculating the fundamental matrix between two frames the input data is normalized. After normalization the mean is zero and the average distance to the mean is $\sqrt{2}$. Looking at the result in figure 4 it can be seen that the epipole is not inside the image (compared to the unnormalized data). This means that the eight point algorithm improved based on the normalized data. It can also be seen that a lot of keypoint lie on the epipolar lines.

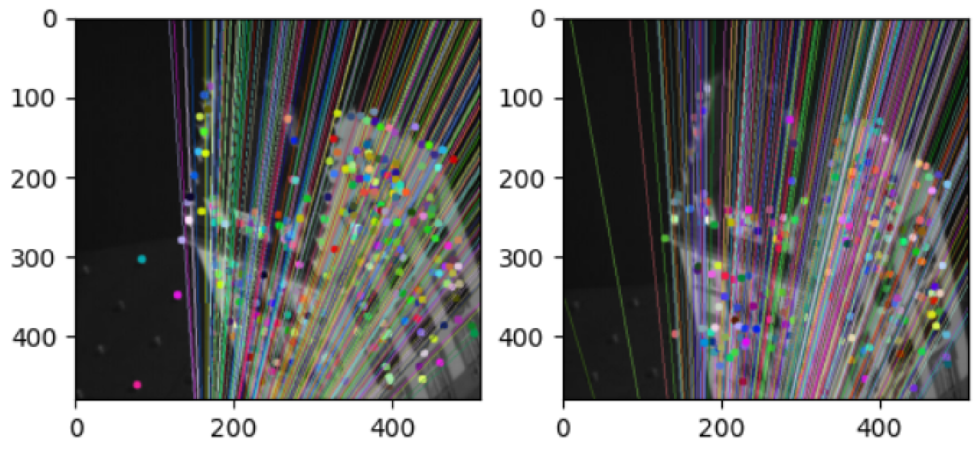

Figure 4: Epipolar lines based on the fundamental matrix using the normalized eight-point algorithm. Frame 0 (left) and 20 (right) are selected.

3.4.3 Analysis Normalized eight-point algorithm using RANSAC

As states above the best fundamental matrix is saved when calculating the fundamental matrix 100 times with randomly selecting 8 points. The best matrix is saved based on the amount of inliers which is calculated using the Sampson distance metric. The different Sampson distances tried are 0.01, 0.1, 0.5, 1. Looking at the result in figure 5 it can be seen that different Sampson distances results in different epipolar lines and epipoles. For the Sampson distance of 0.01 (upper left) the epoipole lies inside the image which is not correct. The best Sampson distance is 1 the result can be seen in the upper right corner. Almost all the epipolar lines go over the building (and thus not the background) and the keypoint lie inside the line.

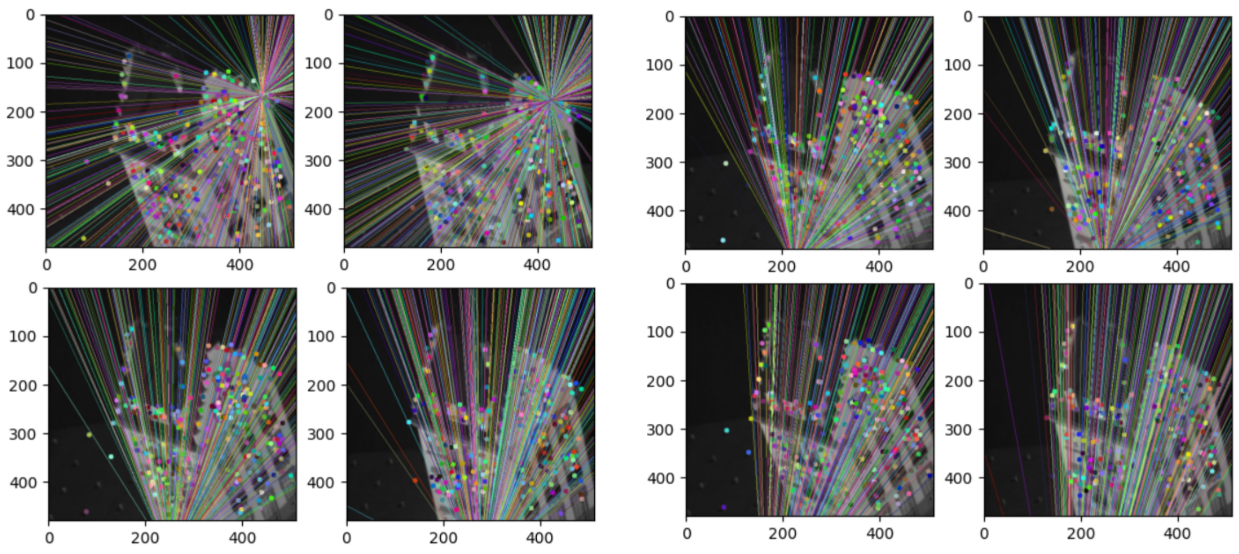

Figure 5: Epipolar lines based on the fundamental matrix using the normalized eight-point algorithm with different Sampson distances. Sampson distances from left to right, top to bottom: [0.01, 0.1, 0.5, 1]. Frame 0 (left) and 20 (right) are selected.

4. Chaining

4.1 Point View Matrix

Inspired by the teddy bear example from (2) and the assignment the point view matrix is implemented. The point view matrix is a overview of which matching key points are visible in each images. On the y axis the images are visible (x and y points). On the x axis the number of unique matching key points are visible. Ideally it would be beneficial if a lot of matching key points appear in a lot of images. Because then this information can be used for making a 3D reconstruction.
The point view matrix is implemented as follow: Between every consecutive frame (1-2, 2-3 .. 48-49) the matching key points are found. The matching keypoints are filled in the point view matrix on the corresponding image rows (x and y) and corresponding keypoint column. If a keypoint is found which has already been obtained in earlier frame(s) then this keypoint is filled in the corresponding column. Otherwise, a new column is added for said keypoint.

Next to the regular point view matrix we also created a ”densified version” of the point view matrix. Because the regular point view matrix could be very sparse we added an additional filtering rule. We filter out columns which are less then 6 rows long. In other words, a matching keypoint must at least appear in 3 images (every image has 2 rows in the point view matrix, one for the x coordinate and another one for the y coordinate).

4.2 Analysis

Looking at figure 6 it can be seen that there are almost 4.000 matching keypoints found which occur in at least two images (49 images in total). How ever, the point view matrix is quite sparse as most keypoints only occur in a few images. There are a few keypoint which occur in a lot of images as can be seen as the long vertical lines. We created a second point view matrix with the filter rule that a key point must at least occur in three images. This result can be seen in figure 6 (right). It can be seen that this matrix looks more dense then figure 6 (left). Looking at the amount of keypoint (horizontal axes) it can be seen that there are less keypoints (+/- 1500 vs +/- 3700). So roughly half of the keypoints occurs in less then three images.

It could be possible to make the point view matrix even more denser but on the other side you then will also lose a lot of keypoints. We found that was the best trade off between the amount of keypoints and sparsity

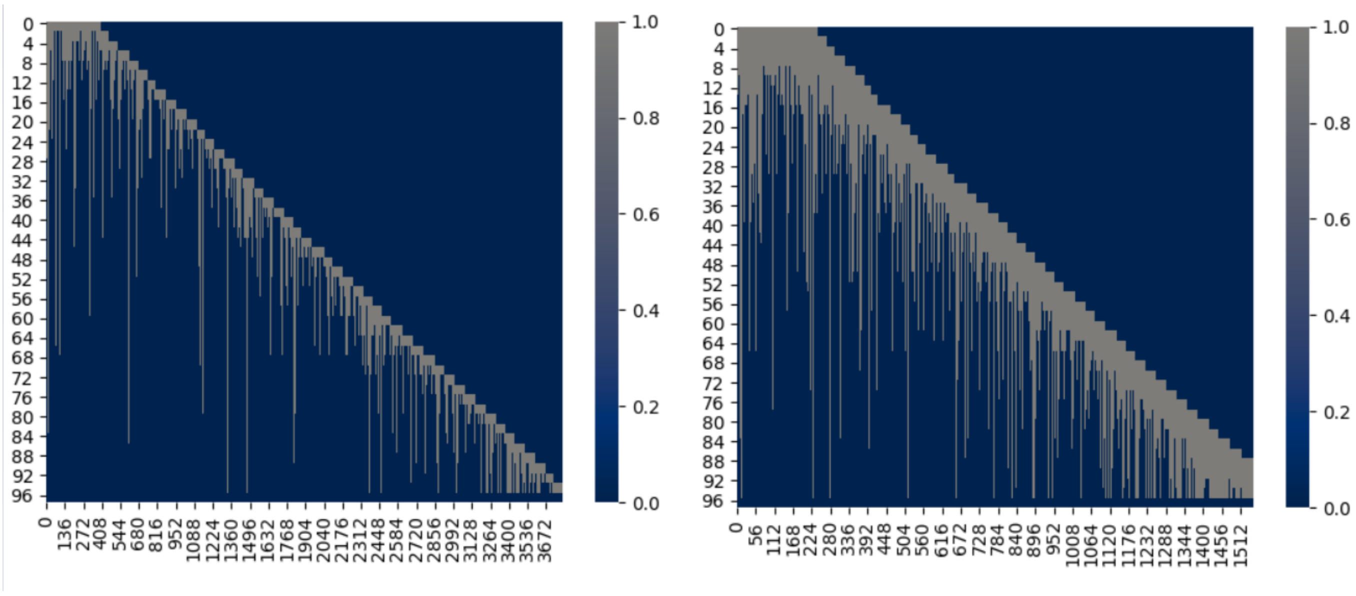

Figure 6: On the y axis the different images (x and y points) can be found. On the x axis the different matching key points can be found. Left is the sparse verion (where a keypoint appears in at least 2 images), right the dense version (where a keypoint appears in at least 3 images).

5. Structure from motion

5.1 Factorize and stitch

Since the point view matrix is sparse, first dense blocks are filtered out from this matrix. These dense blocks, called the measurement matrices, are normalized and decomposed into Structure and Motion matrices. These are then used for the 3D reconstruction, for which we have used the Procrusted function in SciPy[1] to merge the 3D points. The number of overlapping points between two consecutive blocks is used to procrust the Structure matrices for the corresponding points.

5.2 Analysis

We were tasked with recreating the 3D structure of the house dataset, by using three different matrices:

- Only the first dense block of our PVM described above
- Multiple dense blocks of our PVM described, consisting of either three or four consecutive images
- The dense PVM given in *PointViewMatrix.txt*

In the following sections we will show the respective results and elaborate on them.

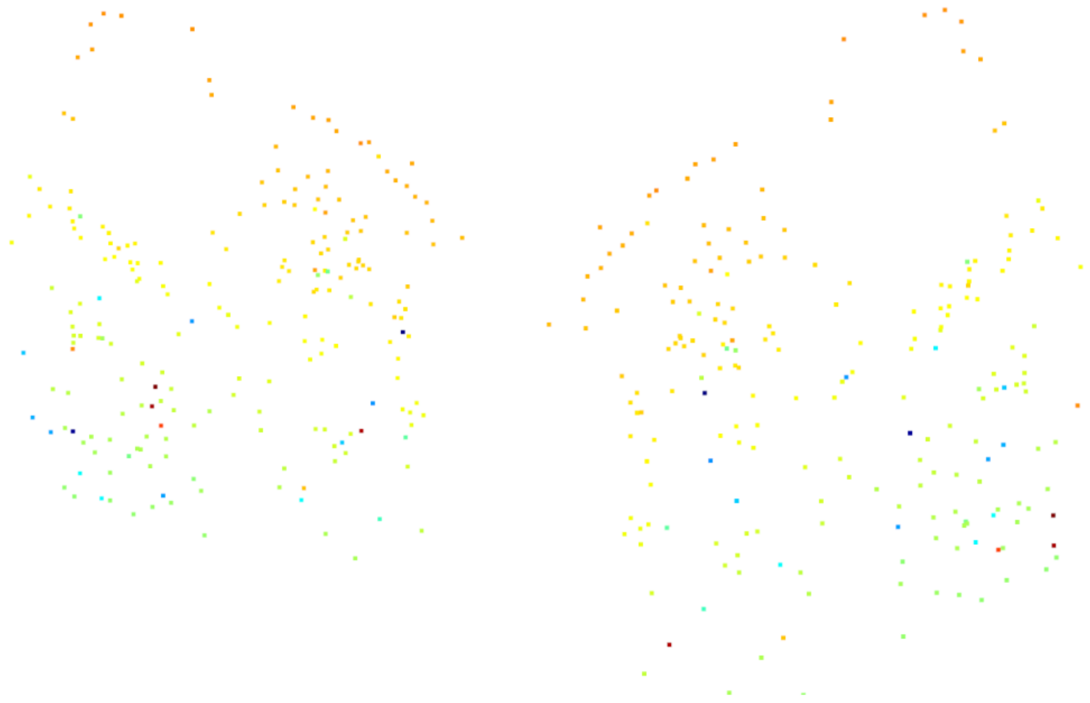

Figure 7: 3D point cloud obtained from 1 single dense block. Keypoints must at least appear in 3 images.

As can be seen in figure 7, it is hard to obtain a truthful reconstruction of the house, using only the first dense block. This is expected, since the algorithm only receives a very small part of the data. When looking closely, however, some contours of the house are visible, like the roof.

[1]https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.orthogonal_procrustes.html

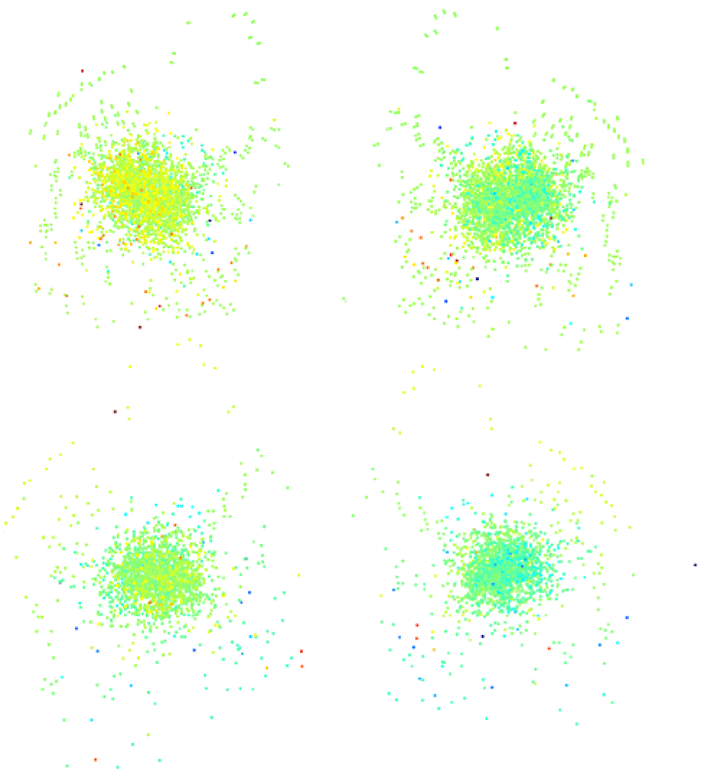

Figure 8: 3D point cloud obtained by iteratively selecting dense blocks from the point view matrix. Keypoints appearing in at least 3 images (above) and keypoint appearing in at least 4 images (below).

When comparing figure 8 with figure 7 it can be immediately be seen that multiple points are plotted. However, similar contours as in figure 7 are shown here, with a lot of the points being plotted near the origin of the point cloud. This leads us to the conclusion that our algorithm has some shortcomings, which we were not yet able to find.

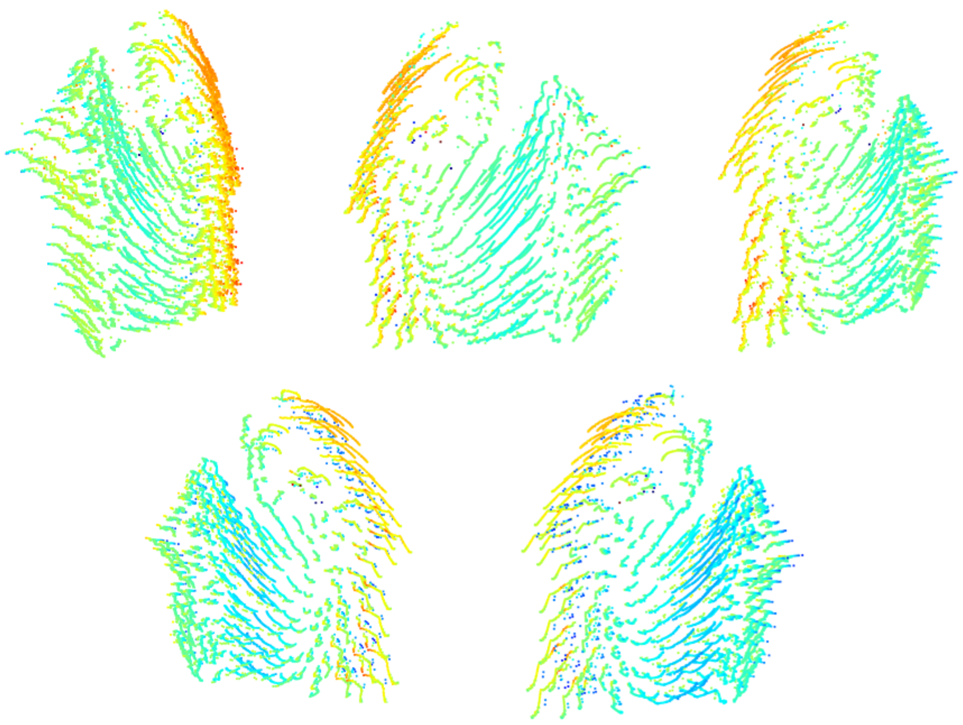

Figure 9: 3D point cloud obtained from PointViewMatrix.txt using keypoints appearing in at least 3 images (above) and keypoint appearing in at least 4 images (below).

If we take the data that was given to us in the assignment, the dense matrix in *PointViewMatrix.txt*, we can see that our algorithm finds some correspondences between the points. As a house is clearly visible in the above pictures. However, a strong 3D structure is still not completely visible, as a lot of the points are simply plotted behind each other. This is again a hint that our algorithm does not work like it is supposed to.

6 Additional improvements

For improving the result we have tried two methods.
As already stated in section 4.1 and 4.2 we have tried to make the point view matrix less sparse. We applied a filter rule which says that a matching keypoint should at least appear in three images. Comparing the original point view matrix and our improved point view matrix in figure 10 it can be seen that the matrix has become less sparse. The most sparse columns are filtered out. On the other side less keypoint are left in the densified version (+/- 3600 vs 1500). The less keypoints are left over, the more difficult it becomes to correctly estimate the motion between frames. But the advantage of the dense version is that the keypoints it receives appear in more images on average.
Another improvement is that we used another keypoint detector. Instead of SIFT we used the Akaze keypoint detector. We also discussed this in section 3.0.1. When using the Akaze keypoint detector, we did get a better Point View Matrix. In the following figure this PVM is shown, both the normal version and the densified version.

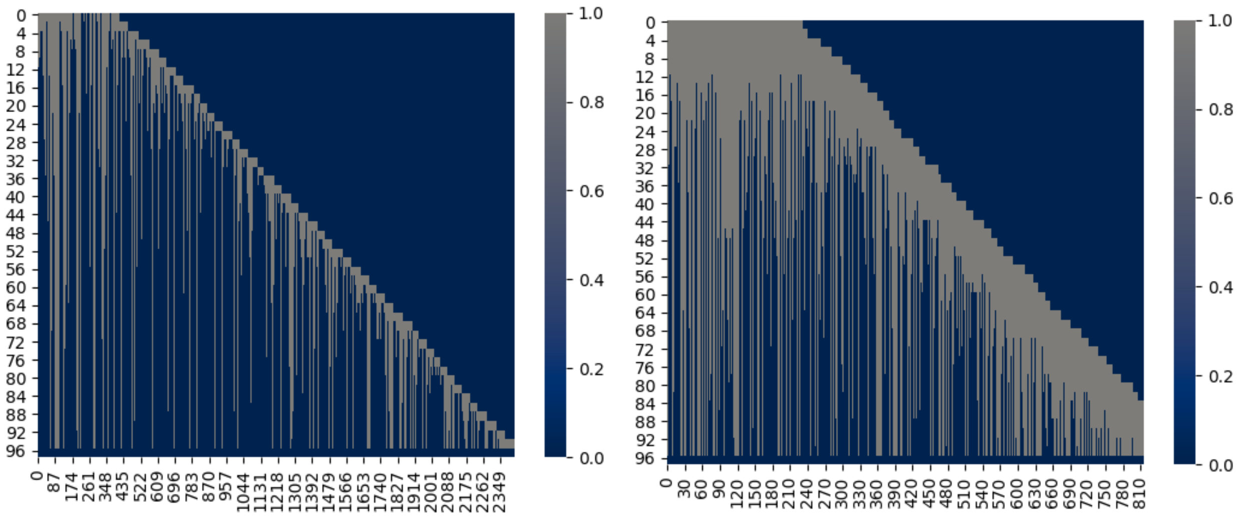

Figure 10: Both the default (left) and the densified (right) Point View Matrices when using the Akaze descriptor

What can be seen is that, when using the Akaze descriptor, more keypoints are found that persist through the entire dataset, when comparing to SIFT. This is related to the densified version, since the diagonal has more width. This should mean that better estimates can be made when using the dense blocks, since the blocks are bigger. However, when we gather the results we still do not get a very different reconstruction than we do with SIFT. This would lead us to the conclusion that our implementation has some errors and inconsistencies, which we were not yet able to find.

7 Self-Evaluation

We did all the work together. We met up via discord and zoom. Screenshare was used to do all the coding together and several free online whiteboard tools were used to sketch out ideas. Afterwards we together wrote the report.

Conclusion

In this assignment we studied structure from motion using the house dataset. Where the house dataset exist of a range of house images where each house is rotated a little bit. We studied the structure from motion first in estimating the fundamental matrix using the eight-point algorithm (with and without normalized data) and Ransac with the Sampson distance. As second we investigated the point view matrix which describes the overlapping matching point between frames. We tried two different versions of the point view matrix, namely sparse and dense. Lastly, we used the dense blocks from the point view matrix in order to estimation the motion between frames and to reconstruct a 3D point cloud from the house dataset.

References

[1] L., H.R.: In defense of the eight-point algorithm. Ieee transactions on pattern analysis and machine intelligence, vol. 19 (1997)

[2] Rothganger. F, Lazebnik, S.: 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. International Journal of Computer Vision 66 (2006)