Assignment 2: Structure from Motion
Computer Vision 2

Bogdan Floris
bogdan.floris@student.uva.nl

Philipp Lintl
philipp.lintl@student.uva.nl

Max Schlogel
max.schlogel@student.uva.nl

1 Introduction

The structure from Motion procedure allows to reconstruct 3D coordinates of objects and scenes, that are photographed in several variations. In this assignment, we start with the implementation of several variations of the eight point algorithm.

Later on we will use the eight point algorithm to reconstruct a 3D representation from 2D images. For better understanding of the data we will start by visualizing the matches between consecutive images. Then we will apply 'structure from motion' for the reconstruction and then try out different methods to improve our results.

2 Fundamental Matrix

2.1 Implementational details

In this section, we apply different versions of the eight point algorithm with which we can represent images of the same scene as a 3D projection. In order to obtain the therefore required fundamental matrix, we need to extract feature points and respective descriptors on the grayscale and gaussian blurred version of the input images. We decided to use `SIFT` and adjusted its parameters for edge and contrast to improve the quality of keypoints. These keypoints then are extracted in the images subject to 3d reconstruction and matched to yield a set of shared keypoints in two images. We used Python and specifically opencv's `cv2` package to perform the keypoint extraction and matching of the descriptors. The latter were matched with k-Nearest-neighbors with k being 2.

2.2 Observations regarding keypoint extraction

When using the `SIFT` descriptors to come up with a matched set of interest points, the parametrization of the interest point extractor, as well as the filter ratio regarding the matched points decide over the amount and quality of key points which are crucial to the quality of the fundamental matrix. With quality we especially refer to interest points being detected in the background. As the goal is to reconstruct the house, we want to avoid detected points in the background. The filter ratio test refers to the procedure, which considers the ratio of the closest neighbor to the second closest neighbor as described in Lowe [2004].
At first, we look at the detected interest points and respective matches for image 1 and 10 if the default values of `SIFT` are not changed and the filter ratio is set to 1, so all matched points are kept. As seen in Figure 1, a lot of interest points in the left background are detected, which then influence the matching. So, in order to come up with a proper fundamental matrix to reconstruct the 3D scene,

Submitted to the Spring 2019 UvA Computer Vision 2 course

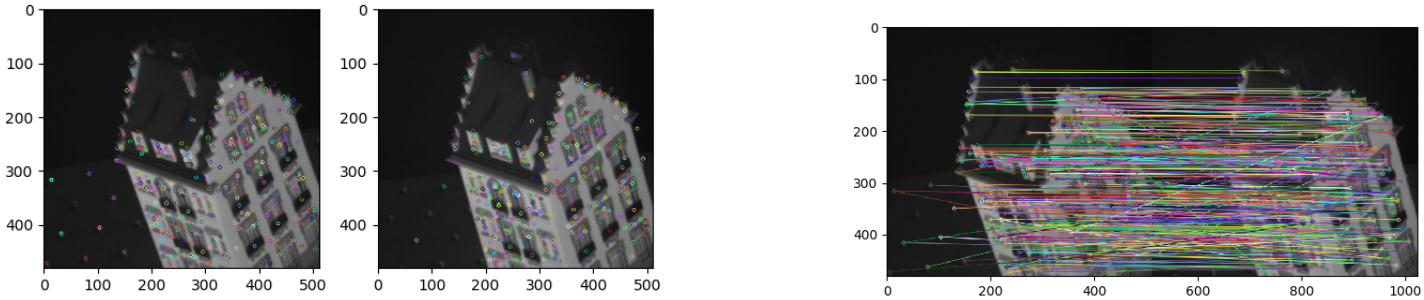

Figure 1: Detected interest points for default `SIFT` parameters and matched interest points with no match filtering applied.

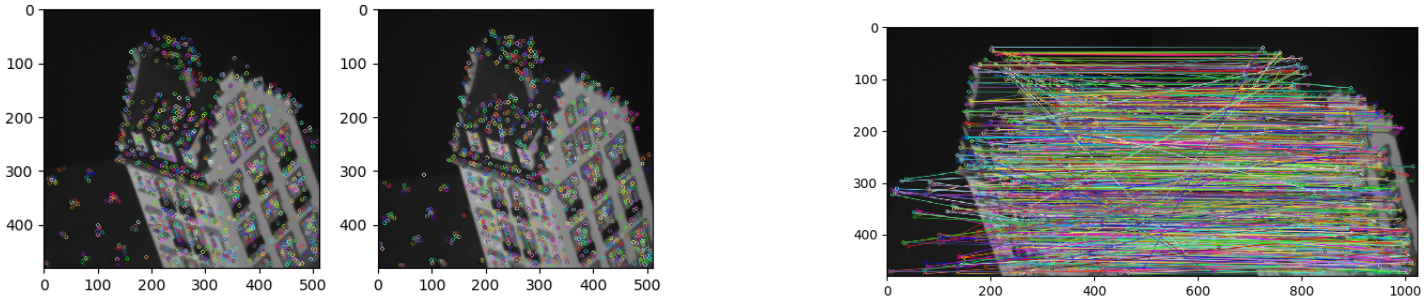

Figure 2: Detected interest points for smaller `SIFT` parameters and matched interest points with no match filtering applied.

we want to get rid of these background points but at the same time not want to disregard too many interest points in total.

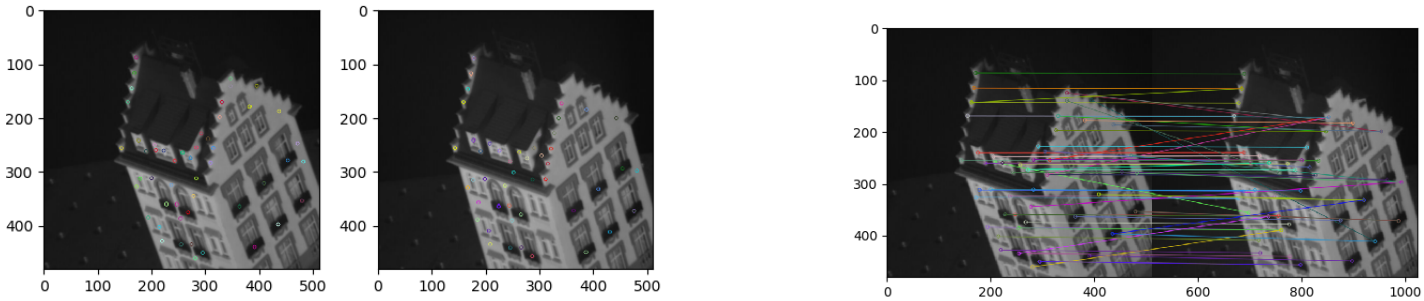

Figure 3: Detected interest points for larger `SIFT` parameters and matched interest points with no match filtering applied.

First, we change the default parameters of the `SIFT` detector. For once, we set `contrastThreshold` to a smaller (Figure 2) value, which results in more keypoints. This is not interesting, as the more interest points in general are detected, the more background points are found as well. A larger (Figure 3) value yields no background points, but also much less interest points in general. Eventhough the matched points are all on the house, we decide to for now move on with the default values, as simply too many key points are not considered in the first place.

A second way to increase the quality of the interest points involves the distance of the descriptors to the first and second neighbor. We draw on the ratio test described in Lowe [2004] and a value of 0.5. Figure 4 depicts the new results. We observe only one matched keypoint in the back, compared to a few in Figure 1. Therefore a ratio of 0.5 to filter the matched keypoints together with default `SIFT` parameters is our choice.

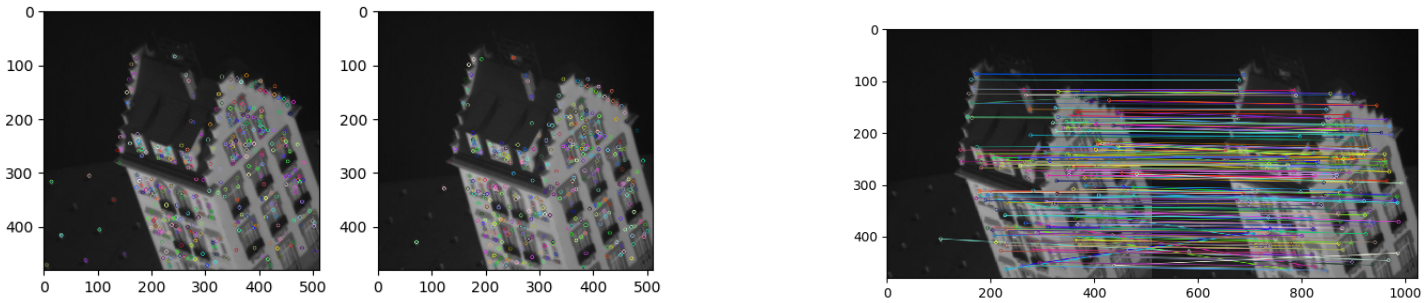

Figure 4: Detected interest points for default `SIFT` parameters and matched interest points with a match filter ratio of 0.5.

Finally, we tried to increase the contrast parameter slightly, which lead to no background points being detected, as well as less mistakenly matched interest points (Figure 5). This parametrization is the basis for the following experiments regarding the fundamental matrix.

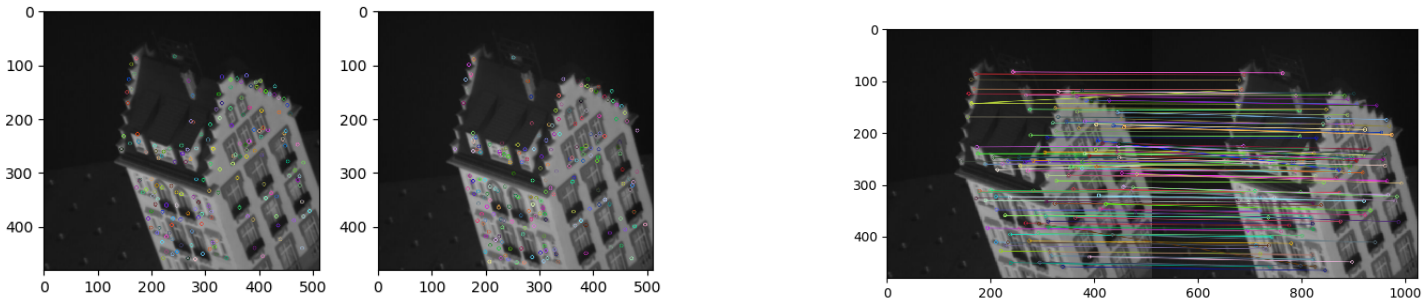

Figure 5: Detected interest points for slightly larger contrast `SIFT` parameter and matched interest points with a match filter ratio of 0.5.

2.3 Fundamental Matrix

After representing the key points in homogeneous coordinates, we obtain the fundamental matrix with the homogeneous linear equation (equation (1) on the assignment sheet). Then, matrix A is contructed according to the assignment sheet, with which we can build fundamental matrix F as the columns with respect to the smallest eigenvalue of matrix V, which resulted from a SVD of A. Based on this, follows the normalized version. It basically repeats the same steps with the difference that the data is normalized beforehand, as described in Hartley [1995]. This step helped to decrease the error of the transformation significantly. For the third alternative, we draw on the RANSAC algorithm to determine the best 8 points to construct the fundamental matrix from. The according error function is the sampson distance and a threshold value of 10e-4 was set to distinguish between in- and outliers. As the RANSAC version disregards outliers, we should obtain more accurate results. Out number of iterations were set to 1000. The fundamental matrix yielding the largest number of inliers is returned

together with the inliers that were outputted by that fundamental matrix.
We expect much worse results for both eightpoint and its normalized counterpart, as they cant disregard outliers and noise. Therefore, the RANSAC alternative should yield a much more accurate result.

2.3.1 Eight-Point Algorithm

Finally the results. As the original setting described above resulted in a lot of keypoints, we add one result with a filter ratio of 0.2 to illustrate the results better. In Figure 6, we clearly see a bad performance, as the eight point algorithm is not in line with the epipolar constraint, as the lines do not pass through the respective key points from the other image.

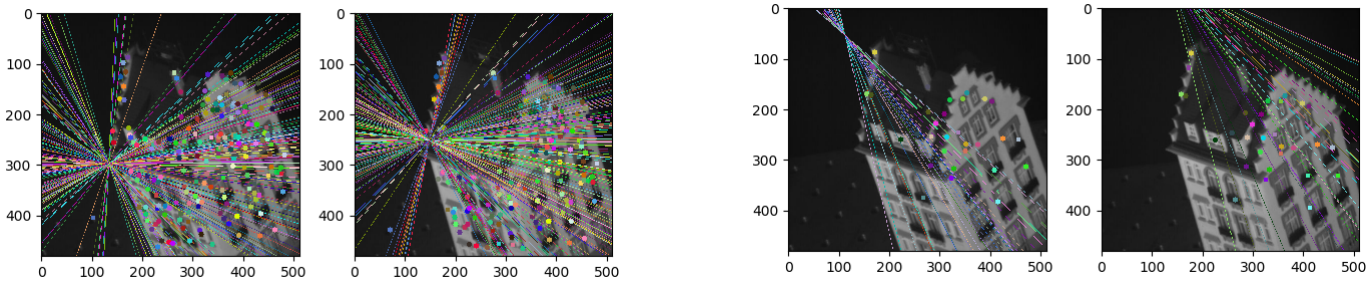

Figure 6: Eight point algorithm results for 0.5 and 0.2 filter ratio.

2.4 Normalized Eight Points algorithm

The same is done with the normalized version of the eight point algorithm. In Figure 7, we see a more accurate result, as the epipolar lines tend to hit the points more frequently than before.

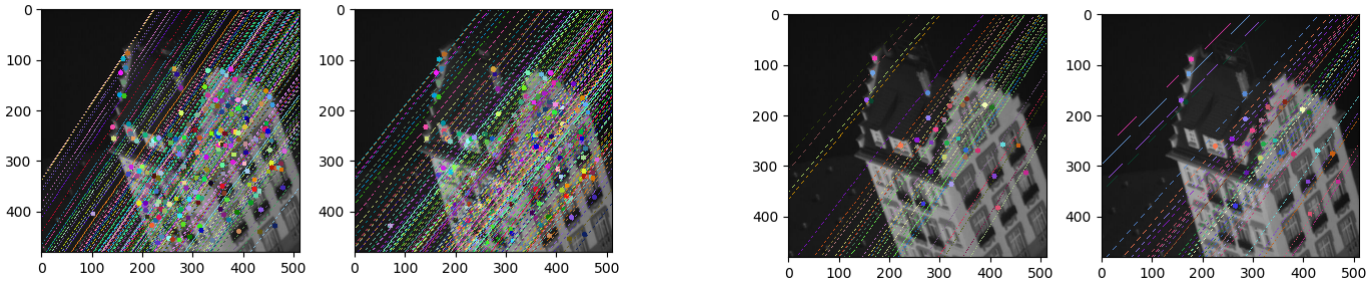

Figure 7: Normalized Eight point algorithm results for 0.5 and 0.2 filter ratio.

2.5 RANSAC Normalized eight point

Finally, the RANSAC algorithm yields the most accurate results, as seen in Figure 8. It probably comes back to less keypoints of higher quality, which leads to a more accurate calculation of the fundamental matrix.

3 Chaining

In this section we will compute and visualize the point-view-matrix for the 49 consecutive house images. We started with images 1 and 2 and continually added the matches for the following image pairs. One important thing to note is that we only compared the matches of one pair with the matches

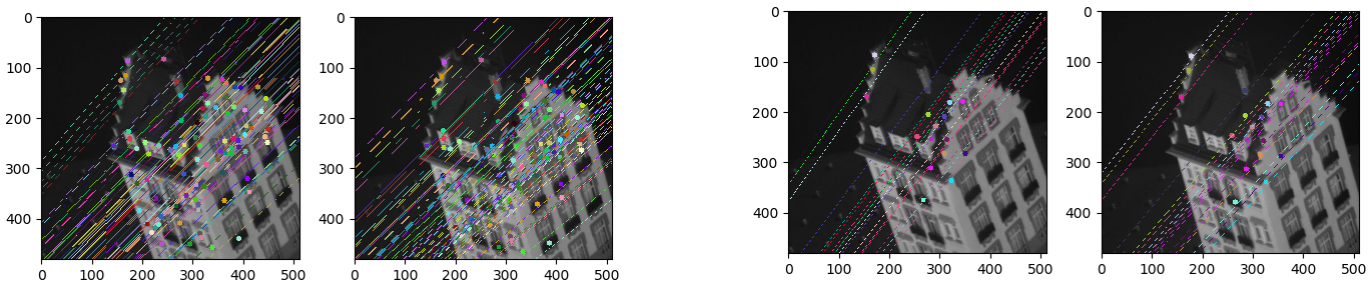

Figure 8: RANSAC Normalized Eight point algorithm results for 0.5 and 0.2 filter ratio.

of the previous pair for continuing the match graphs. So if we encounter a point/match in a later pair, which has already been encountered in an earlier pair (not the right previous pair), we will not add this match to the earlier match graph. This is also visible in the point-view-matrix in figure **??**. We decided to do it this way, because of multiple reasons. Firstly we assumed it to be the purpose of the consecutive pair-wise comparison of the matches. Secondly, when we extract the dense matrix later on, we will extract a matrix not containing any 0s, so the matches in question will not be of interest for us anyways. Thirdly, comparing each new match with all old matches is a computational task that increases exponential with the number of matches. What can we observe in the resulting point-view-matrix? The first thing which is clearly visible is the fact that the matrix does not have any entries in the top right part above the diagonal, except some few ones at the top. This is due to the fact that all consecutive matches that 'come around' back at the top and find matches in image 1 do not find fitting matches in image 2 that were not already discovered at the beginning of the chaining algorithm. The next thing is the almost diagonal structure of the point-view-matrix. This shows that the number of new matches found in new consecutive image pairs is almost the same for every new image pair. In general our point-view-matrix is rather sparse.

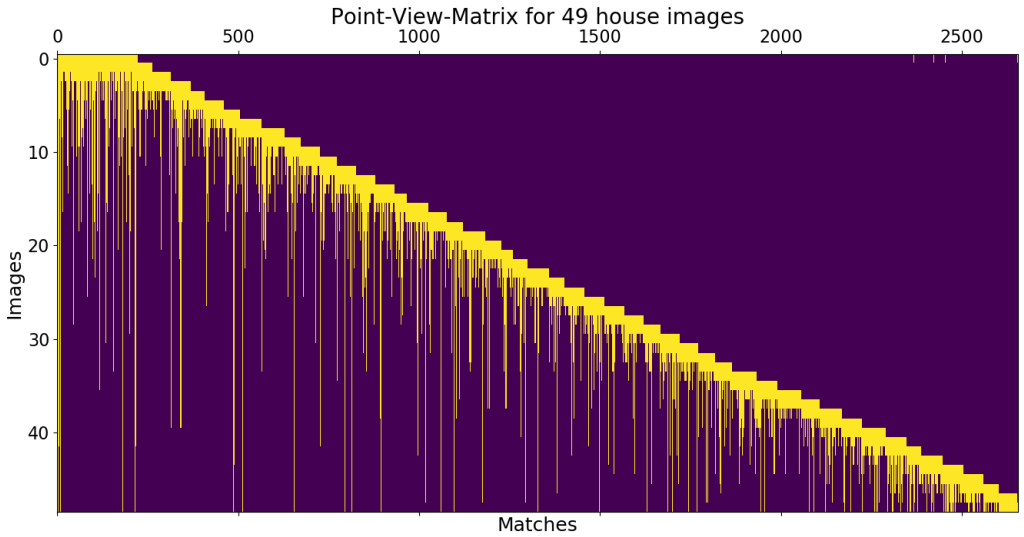

Figure 9: Point-View-Matrix

When comparing it with PointViewMatrix.txt we see that the pre-existing txt-file has a point-view-matrix which has no missing entry, 215 matching points for 101 images. This means PointView-

Matrix.txt represents are much more dense and also larger matrix in both dimensions. As we can see in the in figure **??**, we do not get a similar number of images with such a high number of matches (especially since we only have 49 images, but even the ratio is not as large as in the given PointViewMatrix.txt file). The barplot shows, how many consecutive matches we find for the number of consecutive images, if the first match is found in the first image. The last part is important, since when taking the dense submatrix for exercise 5 ('structure from motion'), we only consider matches, that were found in the first image. One way to improve the density of our point-view-matrix, could be to increase the number of matches found in the images, thus increasing the raw number of matches between consecutive images and therefore the overall length of the match graphs. This would per so not make the matrix denser, since we then have many matches that do not propagate into the next image pairs, but we could delete matches with very short match graphs and thus only keep the relevant ones.

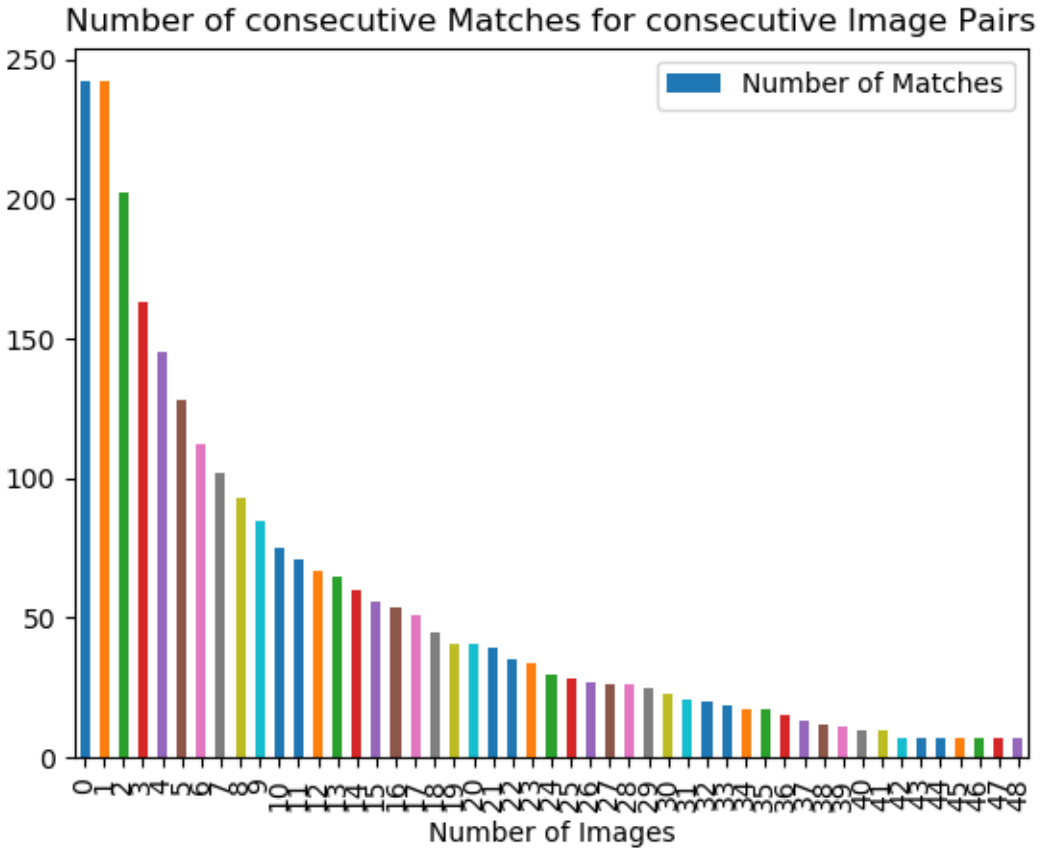

Figure 10: Nr. of Image vs. Nr. of Matches

4 Structure from Motion

In this section we will apply 'Structure from Motion' to extract 3D-points from images taken from different perspectives. We will compare four different settings. The first one will use one single dense block from the point-view-matrix we got in the previous section. For the next two we will use the procrustes-algorithm to stitch multiple results together, one is using three consecutive images, while the other one is using four consecutive images. The last version will be based on the pre-existing file PointViewMatrix.txt.

4.1 One single dense block

Here we extract one single dense block from the point-view-matrix we got in the previous section. We chose to take as many images as are necessary to get around 80 matches, which is 9 images. We decided to go for this number as it seemed to be a good trade-off between number of images/perspectives and number of matches found, when looking at figure **??**. The results can be seen in figure 11. There seems to be some geometrical structure that could represent the walls of the house. Some outliers exist, which do not fit in the image, but there are only a few of those.

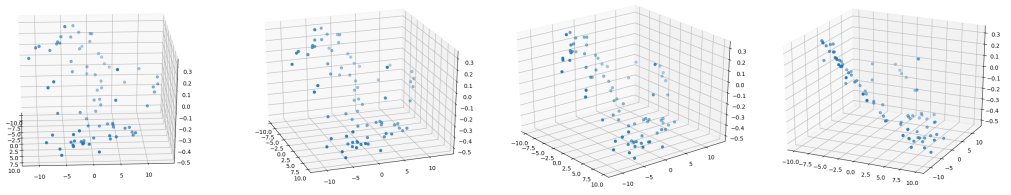

Figure 11: 4 different perspectives on a dense submatrix of part 3

4.2 Stitching with three consecutive images

In figure 12 we see the results for stitching with three consecutive images. For the stitching we used the procrustes algorithm. Unfortunately the original structure from the house or from the point clouds in the previous subsection (the dense submatrix) cannot be observed. This is probably due to errors in the stitching algorithm, that we could not find, despite a lot of effort we put it.

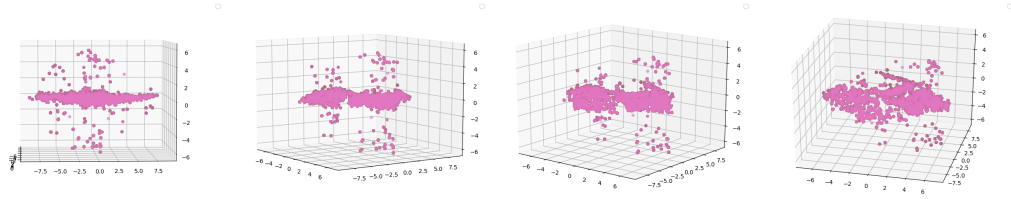

Figure 12: 4 different perspectives on the stitched points with 3 cons. images

4.3 Stitching with four consecutive images

Similar to the case with three consecutive images, we could not observe reasonable structure in the stitching, when using four consecutive images (see figure 13). This might also be due to the error in the stitching function.

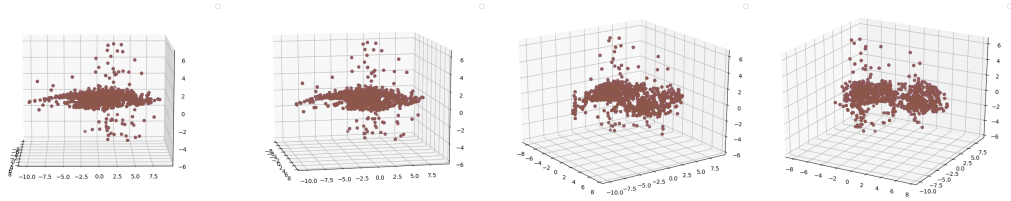

Figure 13: 4 different perspectives on the stitched points with 4 cons. images

4.4 Given PointViewMatrix.txt-file

In this subsection we present the results for the given point-view-matrix. Here, in figure 14, the structure we could observe from the dense submatrix is visible much clearer.

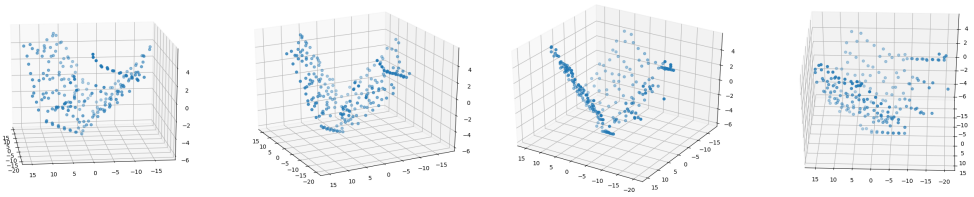

Figure 14: 4 different perspectives on point cloud of the PointViewMatrix data

5 Additional Improvements

To improve our results, we added Affine Ambiguity Removal and applied it to all four cases. In the cases of the dense submatrix and the given PointViewMatrix, we could not observe any noticable changes in the point clouds. For both cases of stitching we found changes in the point clouds(figure 15 and figure 16), but unfortunately the removal of the ambiguity did not resolve our problems from above.

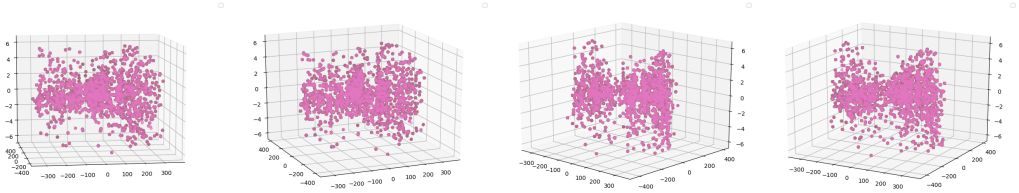

Figure 15: 4 different perspectives on the stitched points with 3 cons. images using ambiguity removal

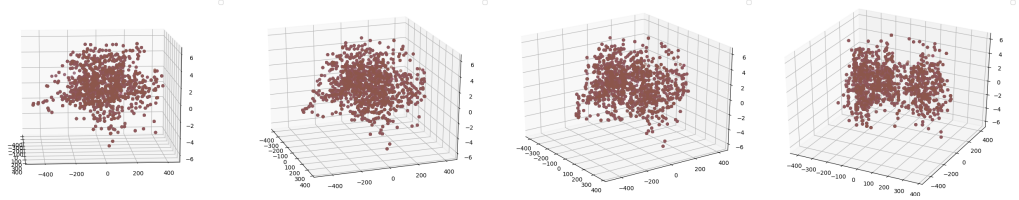

Figure 16: 4 different perspectives on the stitched points with 4 cons. images using ambiguity removal

6 Conclusion

The first part of this assignment revealed the importance of the quality of keypoints for the performance of the eight point algorithm. In particular, changing hyperparameters, the filter ratio for matching and using the RANSAC variant to obtain keypoints lead to more accurate results.

In the next section we applied the chaining algorithm, where we created a match graph structure for the matches found in the house images. Visualizing the graph gave us insights into the structure of the matches and how we could find a dense submatrix. In the last part of this assignment we used this submatrix to create a measurement matrix. Here we used factorization and 'structure from motion' to find the corresponding 3D-points of the matches and reconstruct the a 3D version of the house from the 2D images. Another way to deal with sparse point-view-matrices is to use smaller submatrices and stitch the resulting point clouds together. We applied this method, but unfortunately were not able to get good results. Also applying affine ambiguity removal did not help there.

—

All members contributed equally to the assignment.

References

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

Richard I. Hartley. In defence of the 8-point algorithm. In *ICCV*, 1995.