Assignment 2: Structure from Motion

Diego van der Mast
12222933

Vasiliki Vasileiou
13640496

Sameer Ambekar
13616064

Abstract

In this assignment we implement the Structure-from-Motion algorithm to recover three-dimensional structures from 2D images

1 Introduction

In this assignment, we first calculate the fundamental matrix using three different methods, and then we compare these methods. The matching process in this first step is performed across pairs of views. Consequently, we implement the point-view matrix to represent point correspondences for different camera views and visualize them into a single match graph structure. In the following step, we use the previous point-view matrix to recover a 3D structure from 2D images. We then investigate other methods that could potentially improve the results. Finally, we use COLMAP an industrial tool that works with real-world data. Using this tool we perform various experiments and observe how the changes we did, affect the results.

2 Data

In this assignment, we use the house dataset, which consists of 49 consecutive frames of the same house with some noisy background information. By creating a video of these images we observe slight differences in the camera position between a frame and its next one.

3 Fundamental Matrix

The projection transformation (homography) can be estimated by the following steps:

1. Detect interest points in each image.
2. Characterize the local appearance of the regions around interest points.
3. Get a set of supposed matches between region descriptors in each image.
4. Perform RANSAC to estimate the homography between images.
5. Estimate the fundamental matrix [1] for the given two images.

In this part of the assignment, before we use RANSAC to estimate the fundamental matrix, we implement two other algorithms to determine it. The former is the Eight-point Algorithm and the latter is its normalized version. Both of them are described in more detail in the following sections.

For the first three steps the process we follow is the same. In particular, we first load two images and we translate them in gray-scale. Subsequently, we implement the calculate_keypoint_matching function which gets as arguments the two images and returns the keypoints of each images and their matches. For the calculation of the keypoints we use SIFT features (OpenCV) and for the matching process we use the BFMatcher (OpenCV). The extracted keypoints are converted to homogeneous coordinate system by adding ones as the third dimension of the points.

3.1 Eight-point Algorithm

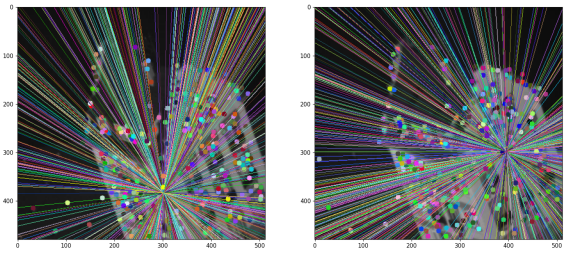
Based on the equation (1), where x_i and y_i are the x,y coordinates of the pair points (p, p') we construct the matrix A. This is followed by the Singular Value Decomposition of the A, from which we obtain the matrix V ($A = UDV^T$). The entries of F are the components

( ), .

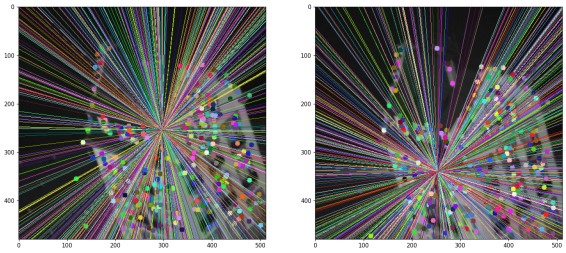
of the column of V corresponding to the smallest singular value.

$$\underbrace{\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx'_n & x_ny'_n & x_n & y_nx'_n & y_ny'_n & y_n & x'_n & y'_n & 1 \end{bmatrix}}_{A} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0 \qquad (1)$$

The last step of the algorithm is the same for the three modifications we applied (in sections 3.1, 3.2, 3.3). Specifically the fundamental matrix should be singular (rank of two), which is not true in every case of our implementation. Consequently, we fisrt calculate the SVD of F, we set the smallest singular value in the diagonal matrix Df to zero in order to obtain the corrected matrix D_f to zero in order to obtain the corrected matrix D'_f. Finally, we recompute the final F by the equation $F = U_f D'_f V_f^T$.



(a) Consecutive Frames 01, 02.



(b) Non Consecutive Frames 04, 12.

Figure 1: Epipolar lines of Normal Eight-Point Algorithm in consecutive and non consecutive images. Left image contains epipolar lines based on the points of right hand image while right image contains epipolar lines based on the points of left hand image.

3.2 Normalized Eight-point Algorithm

The normalized version of the Eight-point Algorithm leads to the stability of the result [1]. We apply a similarity transformation to the set of points pi so that their mean is 0 and the average distance to the mean is $\sqrt{2}$. This transformation is described by the following equations:

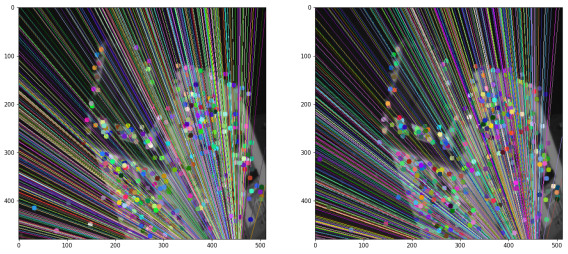
$$m_x = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad m_y = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad d = \frac{1}{n}\sum_{i=1}^{n} \sqrt{(x_i - m_x)^2 + (y_i - m_y)^2}, \text{ and}$$

$$T = \begin{bmatrix} \sqrt{2}/d & 0 & -m_x\sqrt{2}/d \\ 0 & \sqrt{2}/d & -m_y\sqrt{2}/d \\ 0 & 0 & 1 \end{bmatrix}$$

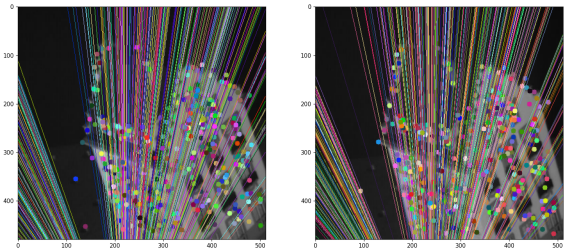
Then we apply the Eight-point Algorithm to the normalized points as exactly we described in the previous section (3.1). The final step is the step of the de-normalization where the final F is calculated by the equation: $F = T'^T \widehat{F}' T$.

3.3 Normalized Eight-point Algorithm with RANSAC

The results of the previous section are visibly better. However, they can be improved even more by removing the outliers using the RANSAC method in the normalized 8-point algorithm. In particular, we perform the following process multiple times (selected 300):



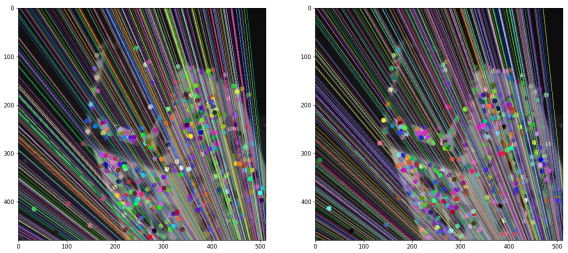
(a) Consecutive Frames 01, 02.



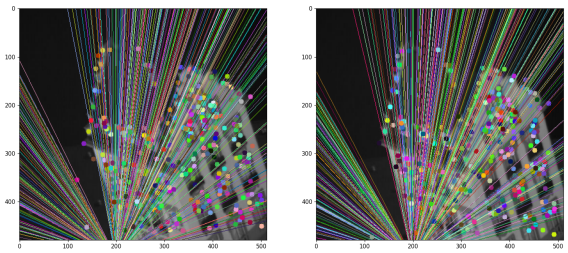
(b) Non Consecutive Frames 04, 12.

Figure 2: Epipolar lines of Normalized Eight-Point Algorithm in consecutive and non consecutive images. Left image contains epipolar lines based on the points of right hand image while right image contains epipolar lines based on the points of left hand image.

- Pick randomly 8 point correspondences.
- Calculate the fundamental matrix.
- Count the number of the inliers based on the Sampson distance an a set threshold (selected 0.3).
- Keep the case of the highest number of inliers in order to calculate the final fundamental matrix.
- Follow the steps of section 3.2



(a) Consecutive Frames 01, 02.



(b) Non Consecutive Frames 04, 12.

Figure 3: Epipolar lines of Normalized Eight-Point Algorithm with RANSAC in consecutive and non consecutive images. Left image contains epipolar lines based on the points of right hand image while right image contains epipolar lines based on the points of left hand image.

3.4 Analysis

As we can observe in the previous images (Figure 1), the normalization of the 8-point algorithm shows better results than a simple method. As we can observe in the case of continuous frames and the case of non-continuous the epipolar is located inside the image, something undesirable. When we use the same algorithm on the normalized points we notice a particular improvement in the results (Figure 2). In particular, more epipolar lines intersect the points of the images. Finally, using RANSAC to filter out the outliers we are getting even better results (Figure 3). In all cases the non-consecutive cases give the worst results compared with the consecutive, however, in these cases, the improvement is higher.

	Consecutive frames 01-02	**Non consecutive frames 04-12**
Keypoints 1st frame	543	566
Keypoints 2nd frame	548	555
Matching points before RANSAC	434	390
Matching points after RANSAC	383	42

Table 1: Matching keypoints before and after applying RANSAC

In the above table, we can observe the number of points per image and match before or after applying RANSAC. As we waited the number of inliers is smaller when we perform RANSAC and even smaller if it is performed on the case of the non-consecutive frames. Finally, it is important to note that the number of inliers after the use of the RANSAC depends on the selected threshold, the number of iterations, and the matching methods we use. We tried to select the most proper values to achieve the best result.

4 Chaining

Coordinate points from pairs of matches correspond to a surface point on the actual object. To represent this, we create a matrix where coordinate points across from different views for the same surface point are connected. This is commonly done using a point-view matrix, where the rows represent images and the columns represent surface points. To implement this, a matrix of size $2M \times N$, where M represents the number of views and N is the number of surface points. We multiply M by 2 because the X and Y coordinates are put on separate rows in the following way:

$$\begin{matrix} x_1(1) & x_2(1) & \cdots & x_N(1) \\ y_1(1) & y_2(1) & \cdots & y_N(1) \\ & & \ddots & \\ x_1(M) & x_2(M) & \cdots & x_N(M) \\ y_1(M) & y_2(M) & \cdots & y_N(M) \end{matrix} \tag{1}$$

To construct the matrix, we use matched points from consecutive pairs of views $(1-2, 2-3, 3-4, \cdots, 49-1)$. When a pair contains coordinate points that are not yet present in the matrix, a new column is added to contain the coordinate points. If they are, we add the coordinate points to the respective column. In figure 4 a sparsity pattern of the point-view matrix is shown. It becomes clear that half of the matrix contains zeroes. Additionally, some columns do not contain many coordinate points, indicating that the corresponding surface point is not present in many views. As these surface points will not add a lot of information to the reconstruction, we only keep the columns that have more points than a specified threshold. In our case, this threshold is defined as a product of a scaling value and the average amount of points per column. An example of a matrix made using this method is presented in figure 5. This matrix can be referred to as a dense matrix. Compared to the supplied sample point view matrix visualized in figure 6

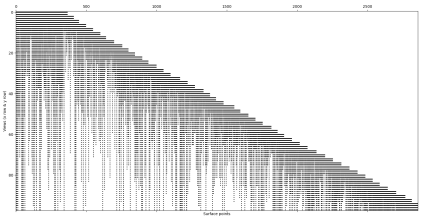


Figure 4: Sparse point view matrix

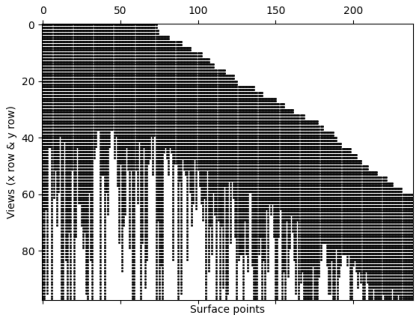


Figure 5: Dense point view matrix

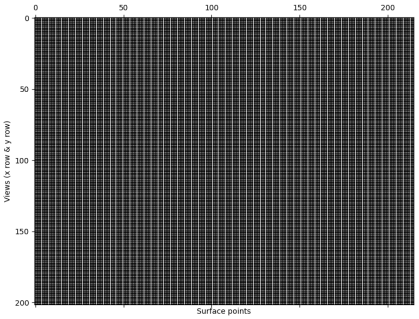


Figure 6: Ground truth point matrix

5 Structure from Motion

Structure From Motion (SFM) is a technique by which we estimate the 3d structure from sequence of images of 2d images that are captured in motion. Given set of images, It begins by finding correspondence between the images. Existing techniques such as SIFT, SURF feature detectors can be used to extract the features and be matched using brute force techniques. Followed by usage of RANSAC that helps in removing the outliers. The refined features are now made use of by storing them separately. In this section we aim to construct the 3d point cloud using the point view matrix. The sources for PVM matrix are: Given Matrix and obtained using the chaining section of assignment.

5.1 Factorize and Stitch

We make use of following steps to obtain the 3d structure of the model.

- The given matrix is read and filtered by removing 0s and NaN value since they act as noise for the algorithm and a desne block is created.
- Now the Structure (S) and Motion (M) are obtained by making use of SVD as explained in the lecture. For every consecutive 3rd or 4th dense block that has been created we obtain the Structure and Motion matrix using SVD algorithm as mentioned the lecture and append all the values to a list named 'all-S'.

$$\begin{aligned} M &= U_3 W_3^{1/2} \\ S &= W_3^{1/2} V_3^T \end{aligned} \tag{2}$$

- We now iterate over the all-S values and pass the next frame from the all-S list to the Scipy procrustes function to obtain 2 matrices. The firs output matrix will be standardised version of 1st Structure matrix and the 2nd output obtained is the orientation of 2nd frame that best fits the 1st frame.
- We now append the obtained matrices to a numpy array in an iterative manner. This array is now transposed and viewed through visualisation functions such by Open3d, matplotlib libraries etc.

We obtain the results visualized in figure 7 by making use of our point view matrix. We observe that that due to sparse matrix the 3d reconstruction is not very clear but still includes basic structure of the house in the form of edges.

Figure 7: House reconstruction with 3 and 4 consecutive images and using a single dense block on generated point-view matrix

5.2 Analysis

We obtain Fig. 8 by making use of every consecutive 3rd frame. We observe that the reconstruction is better than using our own matrix since the given matrix has all the required features present in abundance and less sparsity. Similarly, we obtain Fig. 9 by using every

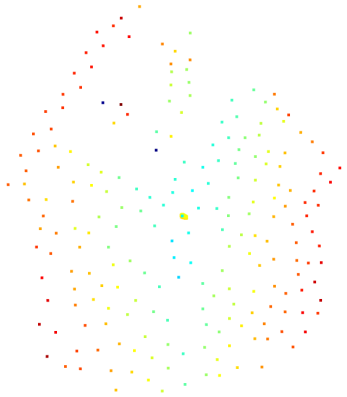


Figure 8: 3d point clouds view obtained using every 3rd frame from given PointViewMatrix.txt and can observe the image of the house similar to the given dataset images

4th frame and observe that the 3d model is worse than the previous because of lack of points for the stitch compare

Due to absence of Camera calibration data the 3d point cloud cannot be further improved. Existing software such as OpenSFM make use of data such as Camera model, EXIF data to obtain the fundamental matrix and construct a better 3d point cloud model. We obtain Fig. 10 by using the given point view matrix as a single dense block. Initially we extract S and M matrix from the single dense block. Since we cannot use just one Structure S matrix to construct the 3d model, we append the structure S matrix 'n' times (where 'n' refers to the number of rows of the point view matrix . We now have a repeated list of 'S' matrix that is used in an iterative way to stitch and obtain the point cloud as shown in Fig. 10. We observe that the structure repeats itself in terms of shape.

6 Additional improvements

We improve the existing results by making use of a 'threshold' to filter the points. Points below the threshold will be preserved whereas the points above the threshold will be removed. This method works because it would ensure that the scale of values of the given points stay amongst a close proximity. Since the block is being normalized after this, it would ensure more and more points are centered close to the mean and not far away from the mean. We observe from Fig. 11 that the number of points has decreased drastically compared to the baseline result in Fig. 7. Due to this decrease in points, it becomes hard to discern the house. We theorize

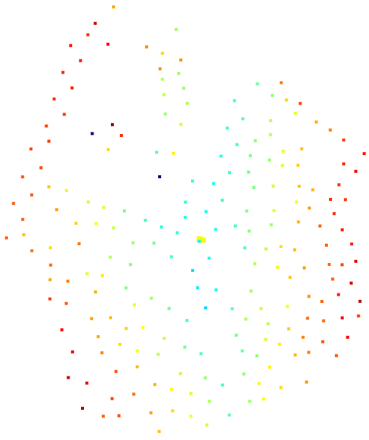


Figure 9: 3d point clouds view obtained using every 4th frame from given PointViewMatrix.txt and can observe the image of the house similar to the given dataset images

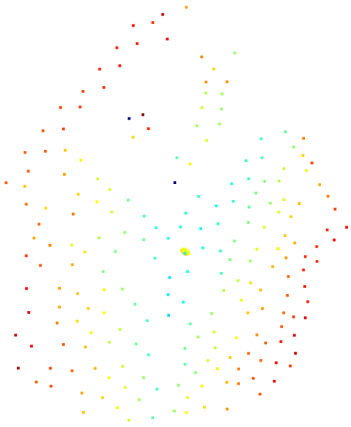


Figure 10: 3d point clouds view obtained using just one single block and stitching the S together.

this is happens because with less points in the point clouds, the amount of constraints decrease leading to a less accurate stitching process. Also, unlike the usual dataset that are used for 3d point cloud generation we do not have access to useful information like Camera model details, Exif data, Focal length, Depth information etc which are vital for producing good visual 3d results.

To investigate how a industrial grade software would design the 3d point cloud we make use of Colmap and generate the point cloud from the same given house dataset and obtain results as shown in Fig. 12. Clearly, we observe that compared to Fig. 16 where there is access to additional information for the Sarphati dataset.Specifically, the 3d output obtained Fig. 16 lacks interior structural information, detailing, is constructed using less points, lacks filling the structure with necessary information and is far away from the marker points.

7 Real-world Data and Industry Tool

1. COLMAP [2] [3] is a tool that can recover 3D structure based on sets of images. It does this through a general-purpose SfM and Multi-view stereo pipeline. Multi-view stereo is used in an effort to recover 3D geometry of a scene with the result of SfM. This is achieved by computing a dense point cloud of the scene based on depth and normal maps of sets of images, after which algorithms are deployed to reconstruct the actual surface. Through the GUI, the software is very straightforward to use. In this work, a dataset containing 116 images of the Sarphati monument in Amsterdam was used to create a 3D reconstruction. This was done once for all frames and once for every fourth frame to compare the performance.
2. In order to obtain a sparse model of the dataset with the SfM and MvS pipeline we are follow the next steps:

Figure 11: House reconstruction with 3 and 4 consecutive images and using single dense block on generated point-view matrix with density improvement

Figure 12: House dataset reconstructed using COLMAP tool

- In the first step, feature detection/extraction finds sparse feature points in the image and describes their appearance using a numerical descriptor. This is visualized in figure 13.
- In the second step, feature matching and geometric verification finds correspondences between the feature points in different images. The sequential matching was selected while our dataset consists of sequential images of the monument.

Figure 13: COLMAP extracts SIFT points

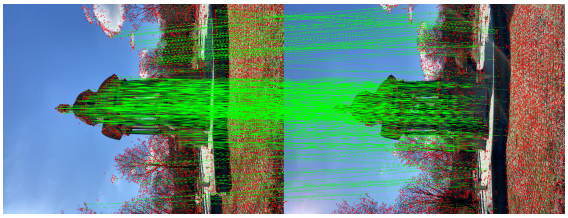


Figure 14: Consecutive frames have visual overlap and there is no need to match all image pairs exhaustively

- In the third step, the scene is incrementally extended by registering new images and triangulating new points. The results are visualized in "real-time" during this reconstruction process.

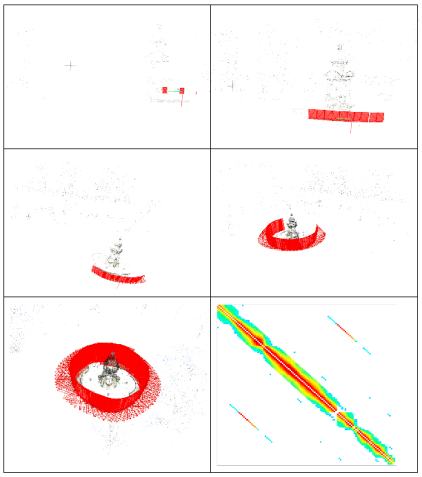


Figure 15: Results for first 3, 10, 20, 50 frames respectively in first two lines, final result and colmap chain respectively in last line.

- In the final step, MVS pipeline recovers denser scene geometry. It produces depth and normal maps for all registered images. Consequently, it fuses the depth and normal maps into a dense point cloud with normal information and finally, it estimates a dense surface from the fused point cloud.

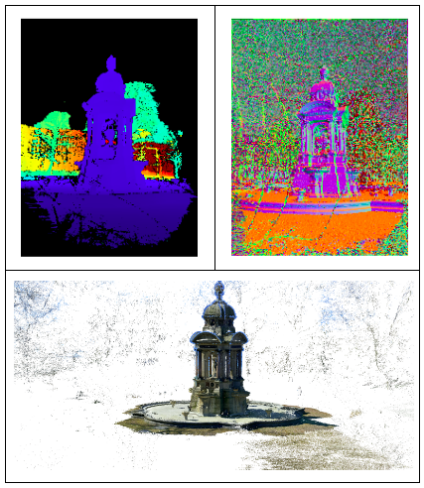


Figure 16: Depth map, normal map and dense representation respectively

3. In the images we notice that there are many background points. This is due to the information contained in some images of our data set. Therefore, this limitation of information would allow us to reduce these errors in our final result. This is already confirmed when we compare the use of the MvS pipeline, which uses depth maps, with the simple SfM pipeline. Therefore,

in a similar way, and therefore offering higher improvement, masking or semantic segmentation methods could work. At the same time we can apply some adjustments to our data to improve it. In some data, there is not enough sharpness, and the light is too bright, there is also camera distortion, and so on.

4. We construct a 3d model using every 3rd image and skipping the images that occur in between them. We observe from Fig. **??**, Stereo models generated as shown in Fig. 17 and Fig. 18 that the reconstruction is not perfect and lacks the features that are necessary to make it look complete compared to Fig. 16. We observe that the reconstruction lacks detailing and features when compared to the model obtained using all the images.
5. The process described in the previous section reduces the number of background pixels in the reconstruction due to a reduction in the amount of data used to execute the reconstruction. While the background noise is reduced, the amount of detail is also reduced as described 7.4.

Figure 17: Obtained by using every 3rd image from the given Sarphati dataset.

Figure 18: Obtained by using every 3rd image from the given Sarphati dataset.

8 Self-Evaluation

We shared the points equally and worked collaboratively on questions that we found difficult to implement individually.

References

[1] R. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[2] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.