Computer Vision 2: Assignment 2

Angela van Sprang, Oleg Litvinov, Emily Mes

May 13, 2022

1 Introduction

In this assignment, we implement a Normalized Eight-point Algorithm with RANSAC to estimate the fundamental matrix for any given two images. Further, the chaining procedure is introduced to produce the patch-view matrix. Next, an algorithm to recover three-dimensional structures from 2D images via factorizing and stitching is implemented. At the end of the report, exemplar usage of the industrial tool COLMAP is shown.

2 Fundamental Matrix

The Fundamental Matrix is a matrix that captures the relationship between the corresponding points in 2 views, with unknown camera intrinsics. It is a 3×3 matrix with 7 degrees of freedom, namely rotation, translation and the internal parameters. Furthermore, an important feature of the Fundamental Matrix is that $rank(F) = 2$, meaning that it satisfies the coplanarity constraint $xFx' = 0$. The Fundamental Matrix can be calculated using the Eight-point Algorithm discussed in the following subsections.

2.1 Basic Eight-point Algorithm

The Eight-point Algorithm owes its name to the fact that it needs at least 8 corresponding pairs of points in both views to find the Fundamental Matrix. In practice, scale-invariant feature transform (SIFT) is used to find matches between both images. SIFT localizes and matches key points using the Difference of Gaussians. Once $n \geq 8$ corresponding pairs are obtained, we can follow the following steps of the Eight-point Algorithm:

- Given $n \geq 8$ point pairs, construct matrix $A = \begin{pmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx'_n & x_ny'_n & x_n & y_nx'_n & y_ny'_n & y_n & x'_n & y'_n & 1 \end{pmatrix}$
- Take the Singular-Value Decomposition (SVD) of Matrix $A = U \cdot D \cdot V^T$
- Since we want the column of V^T with the smallest singular value, we can get the last column, which is a vector with 9 elements. Reshape this vector in a 3×3 matrix F.
- Take the SVD of Matrix $F = FU \cdot FD \cdot FV^T$
- Correct the smallest singular value of FD to be 0, after which we obtain FD'. This step is to satisfy that $rank(F) = 2$.
- Compute the Fundamental Matrix $F = FU \cdot FD' \cdot FV^T$.

Once we have obtained the Fundamental Matrix, we can check the estimation by plotting their corresponding epipolar lines. Epipolar lines are the projection of the line from which the point in the other image could have originated. This will be done in the Analysis subsection 2.4.

2.2 Normalized Eight-point Algorithm

An improvement that can be done to the Basic Eight-point Algorithm discussed above is the normalization of the found matches (i.e. the point correspondences). The normalization entails applying a similarity transformation to the input points ensuring that their mean is 0 and the average distance to the mean is $\sqrt{2}$. The transformation matrix T can be found by:

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

where m_x denotes the mean of the x-coordinates, m_y denotes the mean of the y-coordinates, and d is denoted by $d = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - m_x)^2 + (y_i - m_y)^2}$. So all the input point correspondences are normalized before finding the Fundamental Matrix, whereafter the Fundamental Matrix should be denormalized again by making use of the previously found transformations.

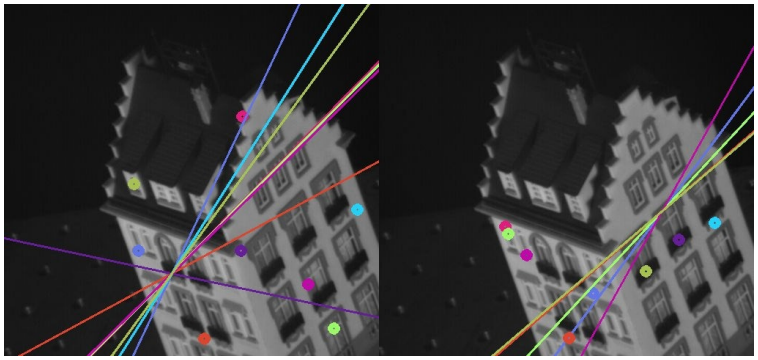
2.3 Normalized Eight-point Algorithm with RANSAC

A further improvement to the Normalized Eight-point Algorithm is the addition of RANSAC to the pipeline. The RANSAC-based approach includes multiple iterations in which randomly 8 points correspondences are selected from all point correspondences, whereafter a Fundamental Matrix is calculated and the number of inliers is counted. Inliers are the other correspondences that agree with this Fundamental Matrix and are calculated using the Sampson distance, meaning that if the Sampson distance is less than a certain threshold it counts as an inlier. The Sampson distance is defined as follows:

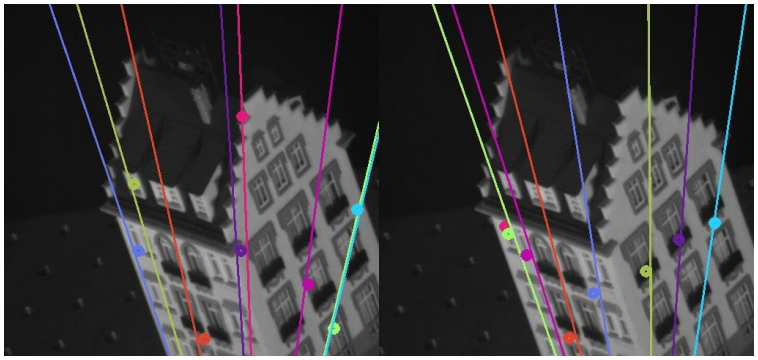
$$d_i = \frac{({p'_i}^T F p_i)^2}{(Fp_i)_1^2 + (Fp_i)_2^2 + (F^T p'_i)_1^2 + (F^T p'_i)_2^2}$$

When the number of inliers is higher thus far, the best Fundamental Matrix is replaced by the current Fundamental Matrix. The Fundamental Matrix that remains after all the iterations will then be used. The idea is that since outliers are not used in constructing the Fundamental Matrix, the matrix will be more robust and therefore the epipolar lines will be better. We performed RANSAC with 100 iterations and a threshold of 1.

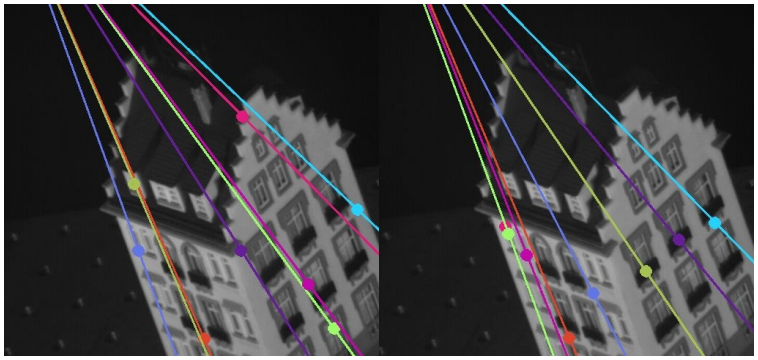
2.4 Analysis



(a) Epipolar lines after the basic 8-point algorithm.



(b) Epipolar lines after the normalized 8-point algorithm.



(c) Epipolar lines after the normalized 8-point algorithm with RANSAC.

Figure 1: The epipolar lines after the 8-point algorithm and its variants. The points in both images are the correspondences, and the corresponding lines are the found epipolar lines.

The figure shows the plotted epipolar lines for the 3 different Fundamental Matrix estimations that are found. The epipolar lines are the projections of the line from which its corresponding point could have originated. All the 3 situations use the same (by SIFT found) features and matches. However, we choose to plot only 8 points, since this is the minimum points required, and because otherwise, the plots would have been too chaotic and not at all interpretable. Furthermore, we can better compare the different 8-point algorithm variants when they have used the same features and matches. Ideally, we want to see the points aligning with the line of the same colour for both images, which is known as the epipolar constraint. In the epipolar lines found by the basic 8-point algorithm, we see that this is not the case, meaning that the Fundamental Matric is not using those points, and therefore not correct. However, we see that the epipolar lines all come

together in one point (the epipole), which is correct. In the normalized variant, we already see a lot of improvement. The epipolar lines go through or are very close to its corresponding point. Furthermore, we see that the lines will come together someplace below the image. However, there are multiple lines that are close to their corresponding point but do not align well, so there is still some improvement possible. We see that this improvement is definitely made in the last variant (normalized and RANSAC). All the points align perfectly with its corresponding epipolar line, and the lines will all come together above the house. It is remarkable that the epipole is on the other side of the house now. We can conclude that there is a significant difference between the basic and normalized variants. So much so that we go from implausible to plausible epipolar lines, and from not satisfying to satisfying the epipolar constraint. This difference is less significant between the normalized and normalized+RANSAC variants, but it is still clear that the latter performs the best of the variants and gives us good epipolar lines.

3 Chaining

To chain views from different cameras, we have constructed a point-view matrix (PVM) that represents point correspondences for different camera views.

3.1 Point-View Matrix

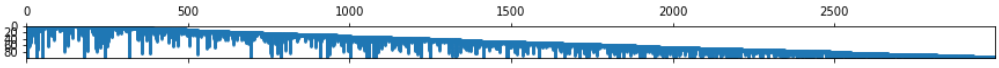
The point-view matrix can be used to chain multiple views, as it is constructed such that rows represent the images and columns represent the points. We find matches between two consecutive house images (omitting matching between 49 and 1 as our dataset is not of 360 degrees around the object), using the method described in earlier sections. When matching two consecutive images, we add a column to the point-view matrix for each newly introduced point. If a point is already introduced in the matrix, we add it to the previously defined point column. The PVM must be sparse since not all points are present in all images. The sparse parts of the matrix are filled with a placeholder value of $-e^{-4}$ (note that the value 0 cannot be used as a placeholder, since this could theoretically be the coordinate of a point in an image).

3.2 Analysis

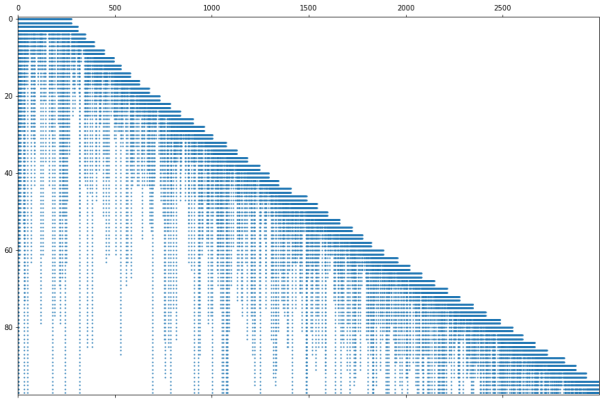
The obtained PVM is visualized in Figure 2. Here, points are displayed in blue and the sparse parts in white. We selected 500 matches between two images and applied a distance threshold of 1, i.e. the distance between the matched points should be less than 1. In total, the PVM contains 2998 points visible in (all or some of) the 49 images and is 90% sparse. The PVM has a clear diagonal and is lower triangular (i.e. the entries above the main diagonal are the placeholder) structure. This is to be expected, because when a new column is added, a new point is discovered that was not found in earlier images. Therefore, no points can be added above the newly added point (since we will not match the first images again).

We cannot directly compare our constructed PVM with the PVM provided in *PointViewMatrix.txt*, since the provided PVM contains more views, but fewer points. We can use it to conclude our PVM looks plausible since it contains similar values for points. If a dense block were constructed from our PVM, it looks similar to the provided PVM.

As previously noted, our PVM is not dense. In general, it is beneficial to have a PVM that is as dense as possible, because then many points are visible from many views and the views can be chained best. To improve the density, we could increase the number of returned matches or relax the distance threshold for a match. However, relaxing the threshold could also decrease the quality of the matches and introduce (more) noise. This is empirically studied in section 4.3.2.



(a) General overview, where the aspect ratio between the axes is 1.



(b) Detailed overview, where the vertical axis is stretched.

Figure 2: PVM obtained when taking 500 matches between two consecutive images (with a point distance of less than 1). A general and detailed overviews is given.

4 Structure from Motion

In this section, we use the point-view matrix to perform structure from motion (a technique for estimating three-dimensional structures from two-dimensional images). First, we explain the used method for implementation. Then, we give an analysis of the results and we explain some additional improvements.

4.1 Factorize and Stitch

In general, a dense point-view matrix can be regarded as the measurement matrix used in the factorization procedure of the affine structure from motion. After all, each point appears in all views, such that the matrix can be factorized directly to obtain the 3D configurations and camera positions of the points. Consider Figure 3 of the factorization of measurement matrix D, which is a matrix containing coordinates $\hat{x}_{ij}$ for point j from view i. The matrix can be factorized into camera positions and 3D configurations of the points.

Since our point-view matrix is sparse, we must find dense blocks to factorize and stitch. We perform the following steps:

1. Select a dense block from the point-view matrix.
   We find a dense block for each patch, by searching for all other patches that are visible in at least 3 or 4 views like explained in [2].

2. Normalize the point coordinates by translating them to the mean of the points in each view (in the dense block). Consider the result as the measurement matrix D.

3. Apply singular value decomposition (SVD) to the measurement matrix D.
   By applying SVD, we obtain the factorization $D = U * W * V^T$. The highest singular values are most relevant, so we take the upper left 3×3 block of W to obtain W_3. Also, we take the first 3 columns of U to create U_3 and the first 3 rows of V^T to obtain V_3^T. We use them to derive the motion and shape

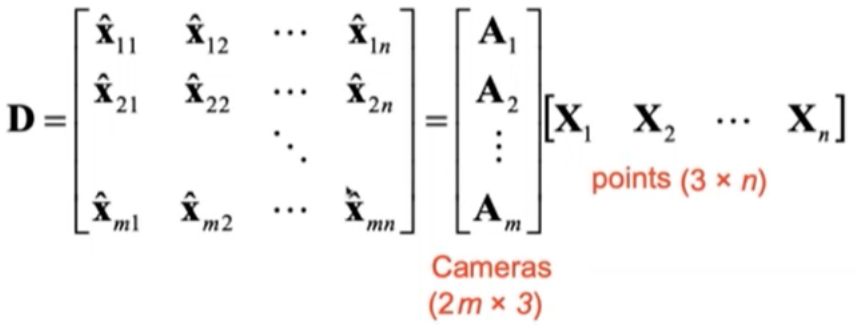


Figure 3: Factorization procedure of a measurement matrix D. (*Source: video tutorial 2 given with assignment.*)

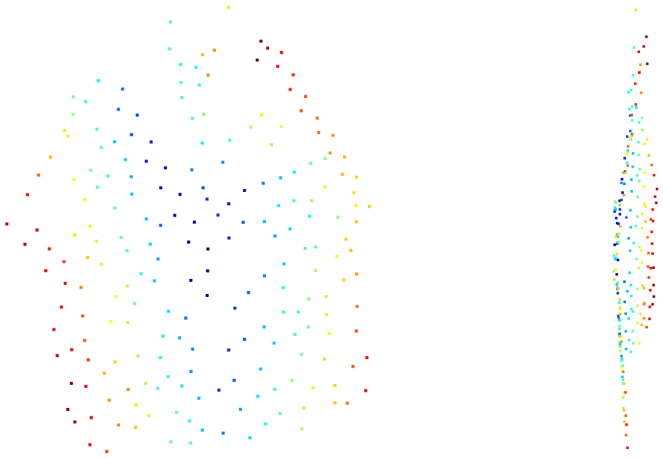
matrix as follows:

$$M = U_3 W_3^{\frac{1}{2}}$$
$$S = W_3^{\frac{1}{2}} V_3^T.$$

4. Repeat steps 1-3 for all dense blocks. Iteratively, we stitch each 3D point set to the main view using point correspondences (by Procrustes analysis). That is, we stitch the last point set to the second-to-last, and we stitch this combined point set to the third-to-last set, until they are all combined to the first point set.

4.2 Analysis

Unfortunately, all the point clouds produced with our PVM looked unstructured and meaningless. Therefore, we show the figure 5 produced with the given matrix.

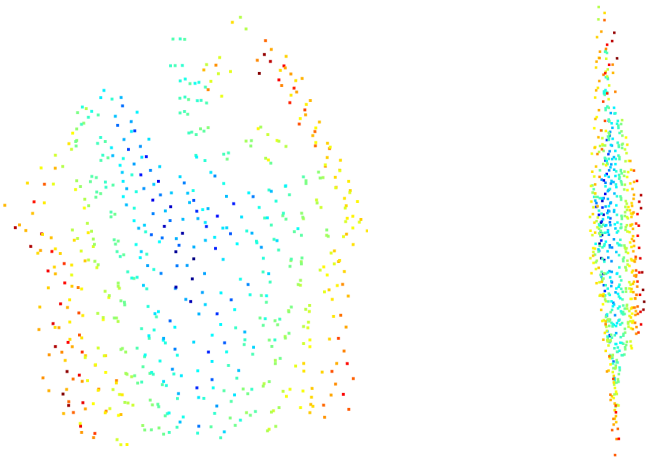


(a) Front view (b) Side view

Figure 4: 3D point cloud obtained from 1 single dense block from PointViewMatrix.txt

Before applying affine ambiguity removal (described in the next chapter) the 3d point cloud has significantly clearer depth while after the removal the house became almost flat.

While working on different point clouds alignments, we tried to split the given PVM into two parts vertically and horizontally and then reconstruct. The vertical split produced meaningless image while the horizontal one was similar to the original (Figure 5).



(a) Front view (b) Side view

Figure 5: 3D point cloud obtained from 2 dense block from PointViewMatrix.txt split horizontally

This alignment looks like the images do not complement each other but overlap.

We have also tried to align the third point cloud to the previous in two different ways: 1. align each new cloud to the first one; 2. and align each new cloud to the cumulative cloud from the previous steps. Both produced bad results for given PVM as well as for ours.

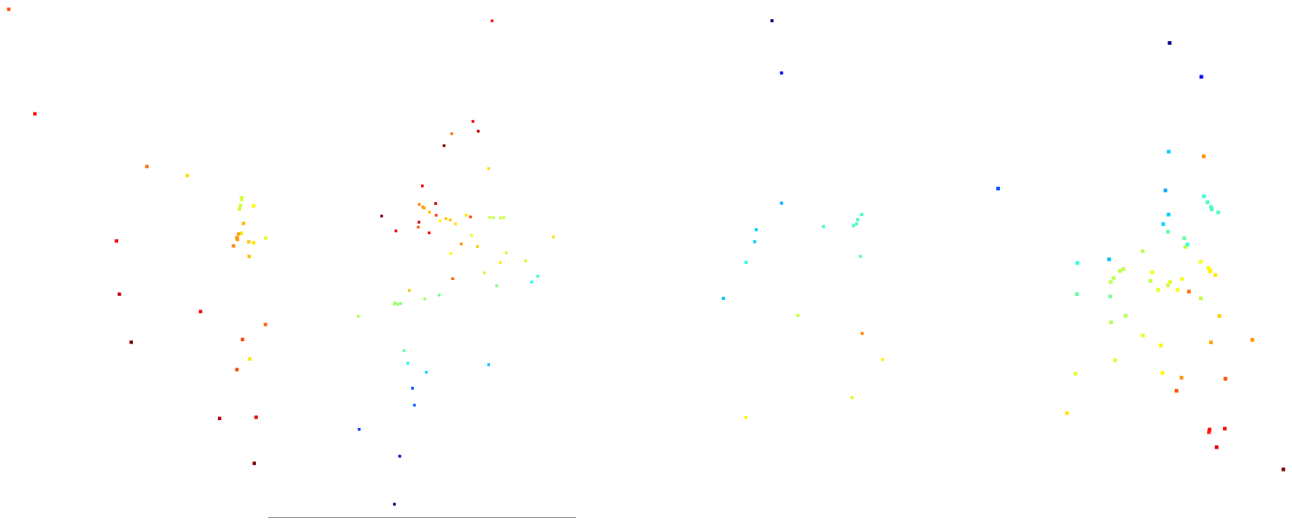
4.3 Additional Improvements

4.3.1 Affine ambiguity removal

The factorization of the measurement matrix is not unique, in other words, the motion and structure matrix can be transformed. For example, $D = M \times S$ also holds for $M \to MC, X \to C^{-1}X$. After all, we only have an affine transformation and we have not enforced any Euclidean constraints, like forcing the image axes to be perpendicular.

We aim to eliminate the affine ambiguity by forcing the image axes to be perpendicular and of unit length (orthographic projection). We do this by solving for CC^T in the linear equations: $ACC^TA^T = I$. We can recover C from CC^T using Cholesky decomposition. Finally, we update M and S: $M = MC$ and $S = C^{-1}S$.

Unfortunately, there is no significant improvement with affine ambiguity removal based on qualitative results. Consider Figure 6, which contain images of the point sets. It may be difficult to compare the point sets by looking only at 2D images, but it is clear that ambiguity removal does not have a major improvement on the 3D reconstruction. This may be the case because there is already a bug in our implementation, as the original results are not good as well. When performing an orthogonal projection on a scattered, random point cloud, it will probably still look like a scattered, random point cloud.



(a) Original: Front view (b) Original: Side view (c) Improved: Front view (d) Improved: Side view

Figure 6: Front and side view of original (without affine ambiguity removal) and improved (with affine ambiguity removal) results

4.3.2 Point-View matrix density improvement

The PVM matrix is sparse. To decrease its sparsity we applied a couple of approaches.

Hyper-parameter grid search was performed to find a combination which produces the best point matches in terms of the PVM matrix obtained. The parameters were the following: `filter_neighbours` - a boolean feature used for filtering based on the ratio of closest-distance to second-closest distance according to [1]; `distance_threshold` - was used in the assumption that the consecutive views do not really change much and the actual location of features can not change significantly due to that. The thresholds of [0.25, 0.5, 1, 10, 20] for L2-norm were tried; `n_points` - if the number of points matched after the all the filters if larger than `n_points`, only the `n_points` of matches with lowest descriptor distance will be selected. The values of [100, 200, 300, 400, 500, 600] were tried.

Via the hyper-parameter search, we achieved the sparsity of about 85% while the highest values obtained during the same procedure were more than 90% (same as originally obtained). The parameters producing the least-sparse matrix are the following: `distance_threshold`=20, `filter_neighbours`=False, `n_points`=500. This set is expected as all the filters are released.

This small improvement didn't lead to any difference in the following analysis. Therefore, we applied the second step of density improvement.

We removed each patch (column) of the PVM matrix if more than half of the images (rows) did not contain this patch. Doing that, we significantly squeezed the matrix and decreased sparsity to 29% as shown on figure 7.

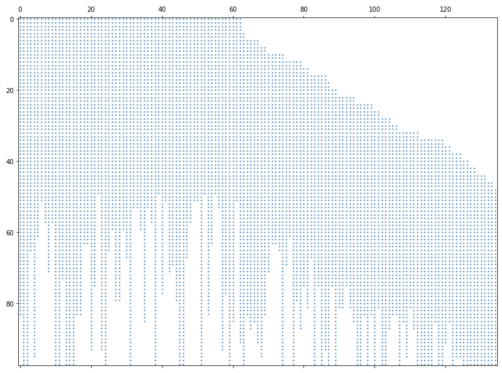
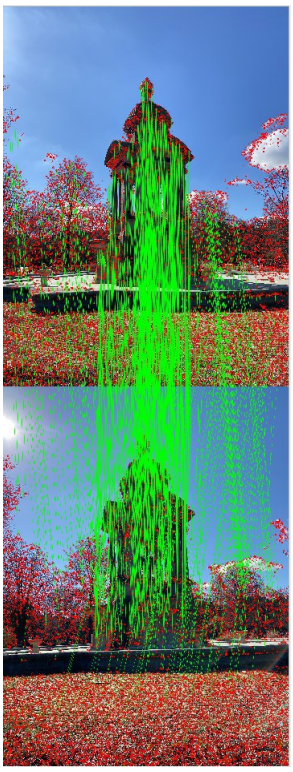


Figure 7: PVM after hyper-parameter search and removing patches which were presented on less than on a half of images

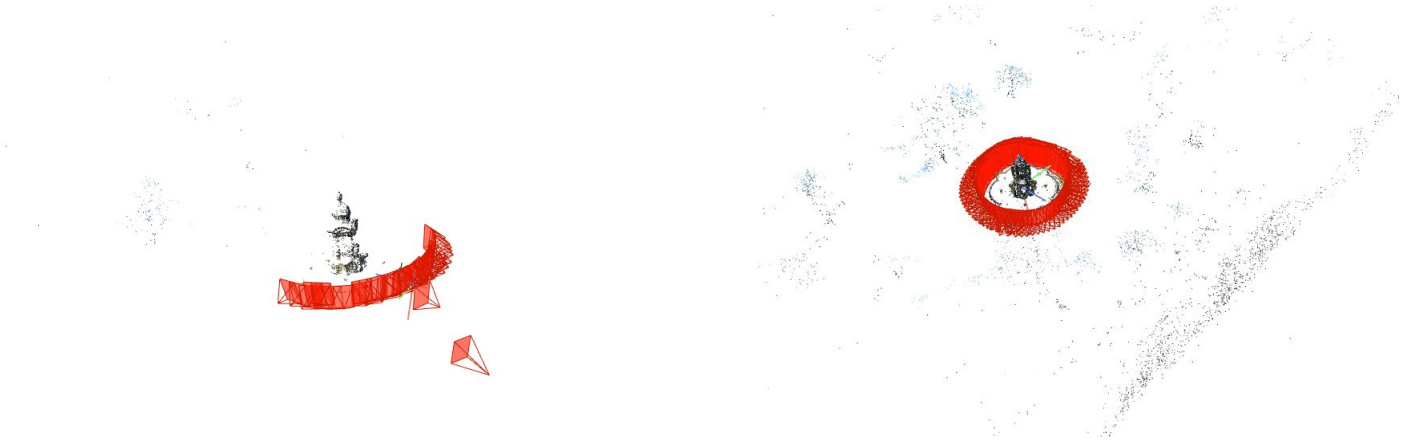
5 Real-world Data and Industry Tool

In this section, we will report our findings after using COLMAP, an industry tool to perform Structure-from-Motion (SfM). As a dataset, we used the provided Sarphati monument. COLMAP will stitch the multiple images together by matching the images. This is done by extracting the features of all images, whereafter the features are matched between images. Consequently, when COLMAP finds the same feature in multiple images, those will most likely have to be aligned. An example of feature extraction and matching of a single image pair can be seen below.



On inspection, the model is pretty correct, because we can clearly distinguish the building of the monument in the model. However, it is noticeable that in the model there is a lot of background noise we are not interested in. Therefore, we can make the model neater by masking the background of the input images. In that case, only background can not be detected as a feature and can therefore not be matched, meaning that we will not see it in the final model. Within the COLMAP GUI, it is in fact possible to select masks as well. However, we were not able to obtain the correctly corresponding masks in time. When the number of (adjacent) frames is reduced, we expect that the obtained model could be similar to the intermediate model shown above. Logically, only the parts of the monument that are captured on the images in the input set are represented in the obtained model. So for example, if we have a circular dataset and we only consider the first half of the images, we obtain a kind of front-facing model.

The SfM pipeline includes finding matches between all the images in the same block that are loaded simultaneously. Since our dataset consists of 116 images, and a blocksize of 50, we have 3 blocks in total that are matched. An intermediate model and the final obtained sparse model can be seen below.



(a) intermediate model

(b) final model

Figure 8: Intermediate model (left) and final obtained sparse model (right) of the Sarphati monument data set. Note that every red object is an estimated camera position.

6 Discussion

- The fundamental matrices calculated in our implementation are slightly different from the ones provided by the open-cv package by an order. However, since the obtained fundamental matrix using normalization and RANSAC satisfies the epipolar constrained, we do not think it is incorrect. More research could be done on this;
- Matrix densification was performed efficiently but the 3d reconstruction results did not improve;
- As an improvement for point registration, the alignment may be started from the largest sub-block as suggested in [2].

7 Conclusion

In this research paper we explained and showed the workings of the Eight-point Algorithm with multiple extensions, the chaining of different cameras to constuct the PVM, and additionally performed Structure-from-Motion. This entire pipeline was for recovering 3D structures from 2D images. Furthermore, 2 additional improvements are proposed and examined, namely affine ambiguity removal and PVM density improvement. Lastly, we used the COLMAP application to follow this same pipeline with real-world data. As an improvement, bug search is proposed. Different structure-motion decompositions comparison may also be a valuable future direction.

A Self-Evaluation

References

[1] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[2] Fred Rothganger et al. "3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints". In: *International Journal of Computer Vision* 66 (Mar. 2006), pp. 231–259. DOI: 10.1007/s11263-005-3674-1.