Computer Vision

Assignment Report 3: Fundamental Matrix, Camera Calibration, Triangulation

Maastricht University

Department of Data Science and Knowledge Engineering

Authors

Nicky Acheila, i6202817

Georgios Patsiaouras, i6198785

1 Introduction

In this assignment we study stereo image reconstruction of multiple (here two) views of the same scene. In the first part, we provide an implementation of the fundamental matrix, a technique that associates corresponding points of image pairs. The fundamental matrix F is calculated based on two sets of matching points of the two images (views), where for each of these points p, p' of Image 1 and 2 respectively we have that Fp denotes the epipolar line in Image 1 where p' should lie. In more detail, we use the 8-point algorithm which utilizes the coplanarity constraint to create linear equations that determine the values of the fundamental matrix, and then use singular value decomposition, instead of basic calculations, to overcome the rank deficiency problem and optimize the resulting matrix.

In the next part, our focus turns to the Random sample consensus (RANSAC) method, which is broadly used to compute and optimize the parameters of a mathematical model. In this context, we optimize the fundamental matrix to fit two sets of points derived by the SIFT descriptor implementation of the MATLAB VLFeat pacakage. Our goal is to distinguish between inliers and outliers, due to the inaccuracy of the SIFT method, while using the fundamental matrix as an estimator of the model fitness. In this process, we attempt to determine the influence of the different RANSAC parameters to the final result.

Finally, we implement the algebraic solution of the linear triangulation, where we compute 3D intersection points out of their 2D projections derived by the provided pairs of images. In reality, the computed points are only estimations, due to the fact that the matching points are not perfect, and so the linear triangulation method takes into account that our data is noisy.

2 Implementation

2.1 Part 1 - Fundamental Matrix Estimation

In the first part of the assignment we were requested to implement a fundamental matrix estimation function. This function takes as input pairs of matches of two images and tries to estimate a fundamental matrix in order to compute from the matches of one image the epipolar lines in the next image.

The implementation is relatively easy as we build the A matrix (as instructed by theory), then we take the singular value decomposition (SVD) of A and find the eigen vector that corresponds to the minimum eigen value that gives the intermediate step fundamental matrix and finally we apply SVD to this matrix in order to pull the minimum eigen value calculated to 0. Afterwards we rebuild the final fundamental matrix F by multiplying the decomposed matrices. Finally, we calculate the product AF which provides the distances from the epipolar lines as seen in Figure 2.2 and we display the overall mean squared error which is approximately 0.000161.

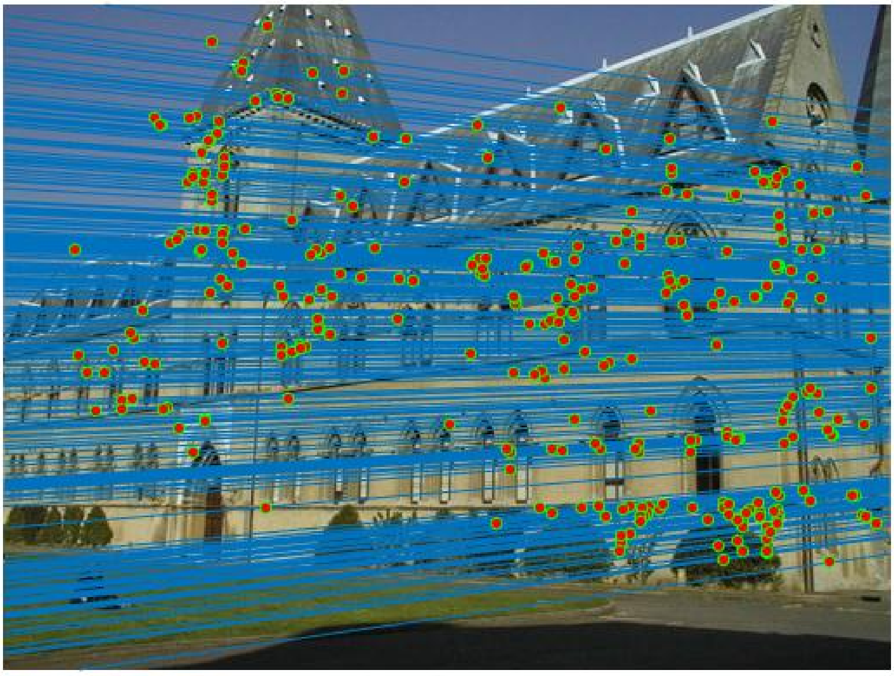

Figure 2.1: Epipolar lines calculated with the estimated fundamental matrix

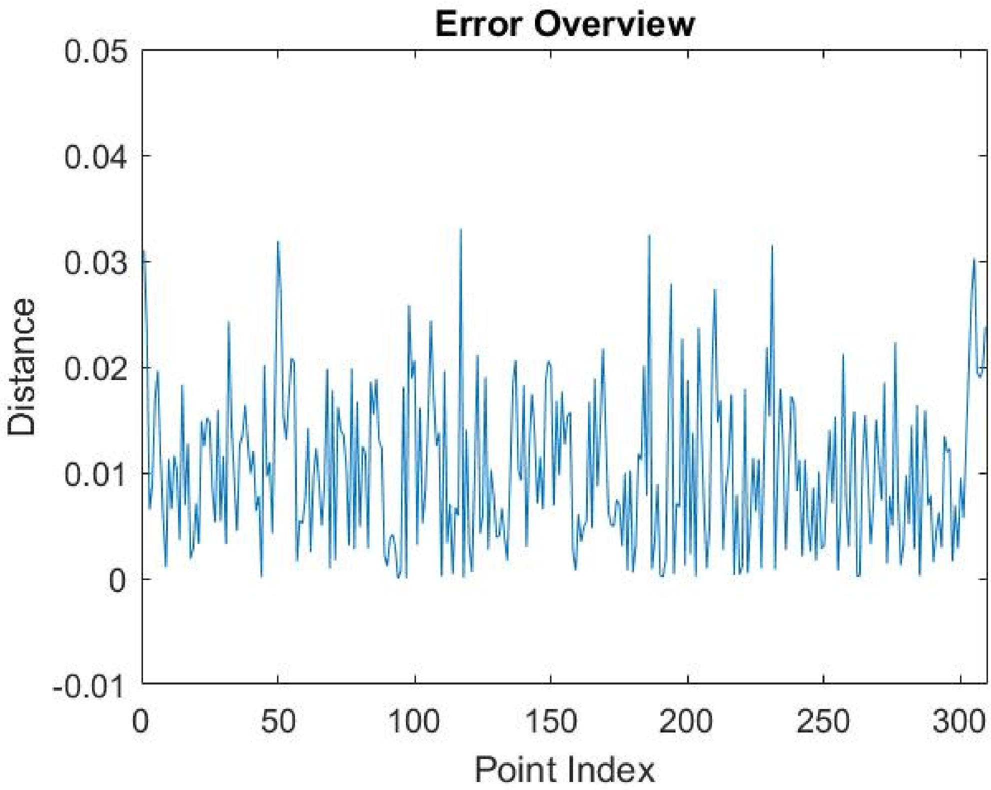

Figure 2.2: Distances from epipolar lines per point

2.2 Part 2 - Fundamental Matrix Estimation with RANSAC

In part 2 we were requested to bring this estimation closer to a real life application. In real life applications, we do not have absolute knowledge over point correspondences between two images. We rely on features detection algorithms which often result in noisy and unreliable correspondences. In this part we are going to use SIFT descriptors to retrieve matching points and then RANSAC, alongside our fundamental matrix function from part 1, to extract inliers and make our correspondences more reliable.

RANSAC is an iterative algorithm that samples a number of points s out of a set, fits a model on this sample, calculates the error of the current model and finally decides on the model that fits as much of the points of the set as possible. In this application, the model is the fundamental matrix, while the xFx' product is considered the error for each pair of points x,x'. To discern between inliers and outliers the algorithm uses a threshold d denoting the acceptable distance (error) from the model that a point is still considered an inlier. As for an acceptable model, it depends on the threshold T that specifies the minimum amount of inliers necessary for an admissible model.

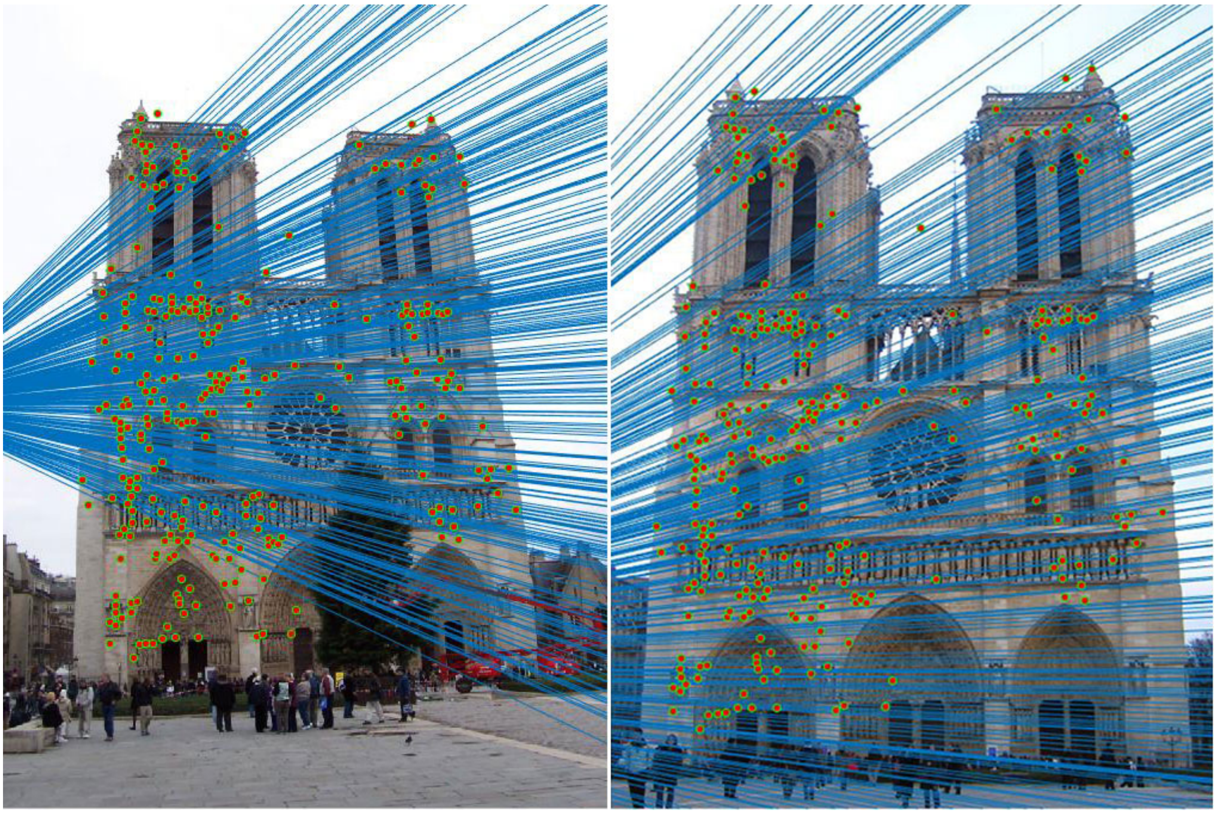

Figure 2.3: Epipolar lines in Notre Dam picture with d threshold 1 and T threshold 150

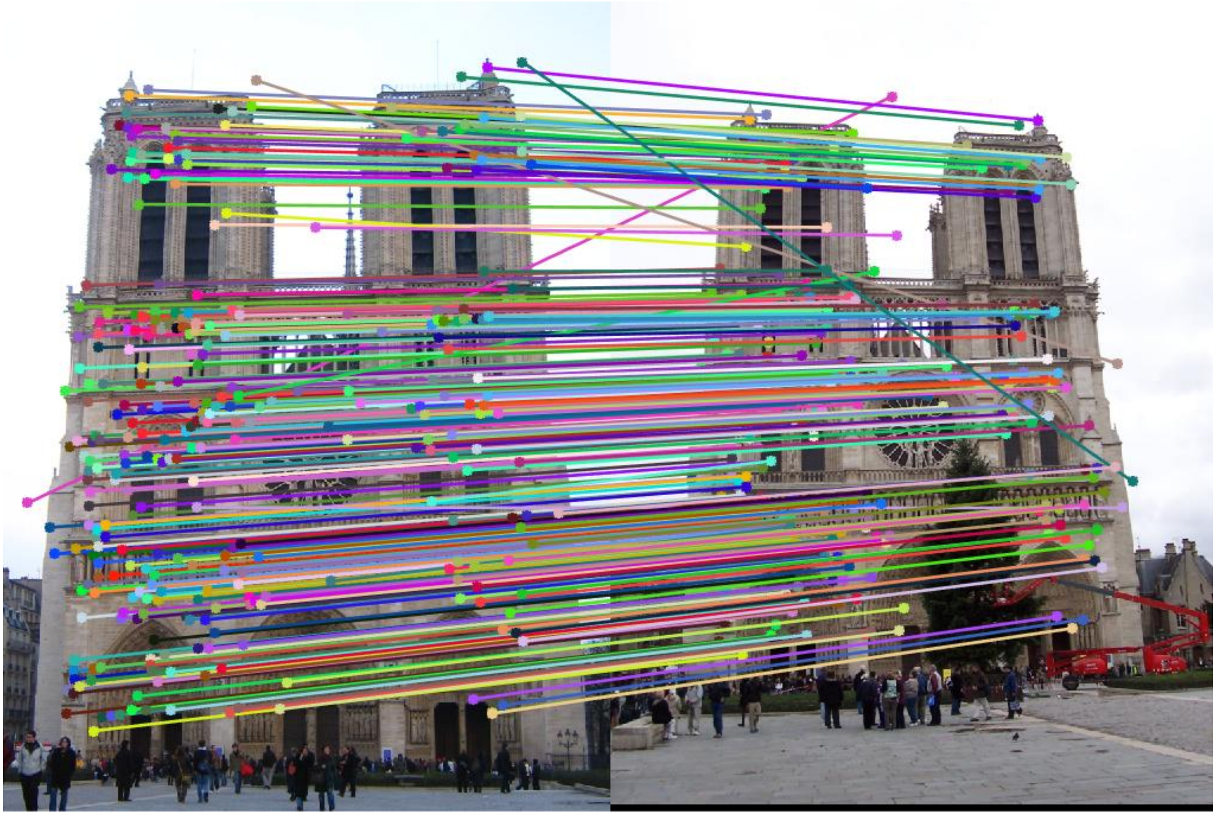

Figure 2.4: Notre dame correspondences before

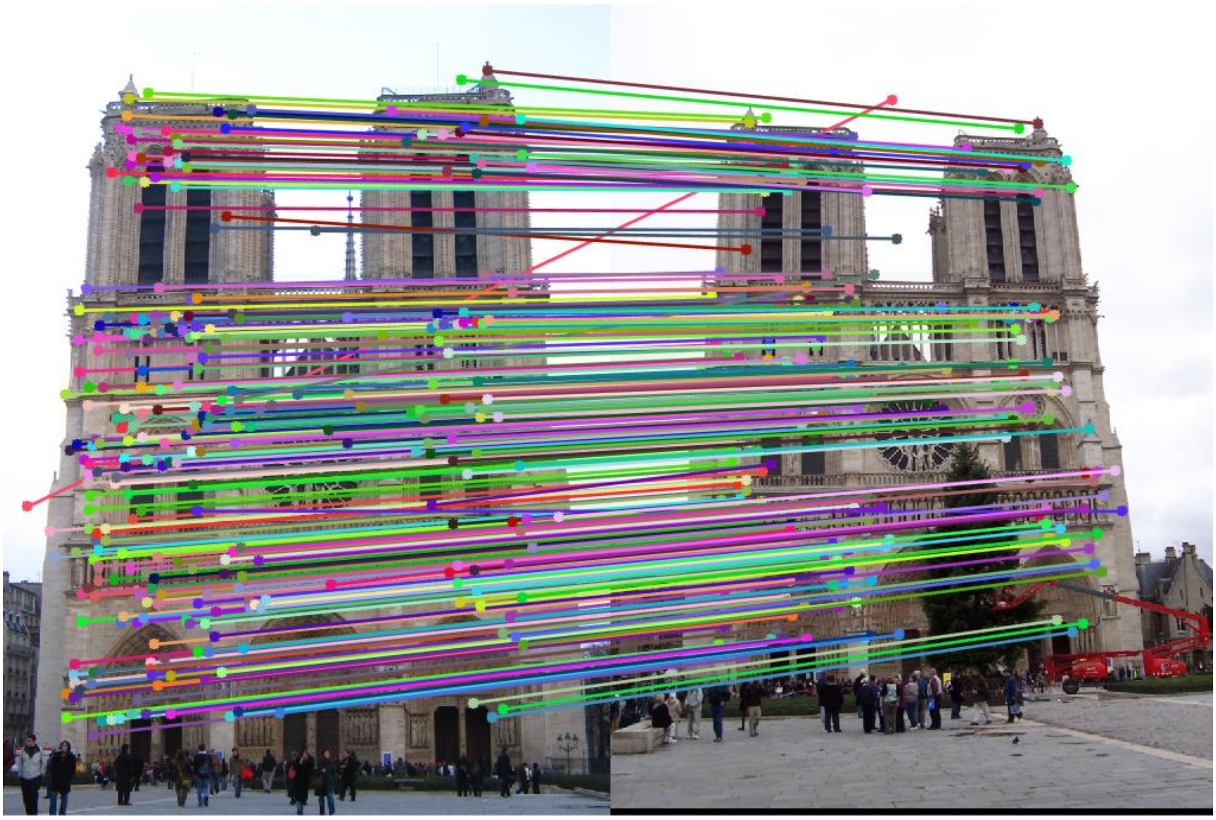

Figure 2.5: Notre dame correspondences after applying RANSAC with $d = 1$ and $T = 150$

2.3 Part 3 - Triangulation (Algebraic Solution)

Triangulation is defined as the procedure to reconstruct a 3D set of points out of two or more different projections (images) of it. In this part we use two images of the same location taken by two calibrated cameras set in different angles, while their camera matrices are provided. By having this information, for each point of our image, we can calculate the 3D coordinates based on the two cameras by using the camera matrices.

The algorithm implementation is pretty simple and straight forward. First we calculate the 3D coordinates of the two cameras so that we can project them in world coordinates along side with the rest of the points. In order to do that we take the singular value decomposition of the camera matrices and then retrieve the last vector, which is of size 4, divided by its last element in order to make it in the type of (x,y,z,1). Then this vector is the camera 3d coordinates, and we do this procedure for both matrices.

Following the same logic, we iterate over all points and after calculating the A matrix and its SVD we retrieve the 3D coordinates and collect them in a matrix. To represent the outcome we are plotting two different figures. The first figure is giving us lines drawn from each camera to each point in order to visualize the projection. And the second figure is a scatter plot of the coordinates of the points and cameras in the world 3D space.

In figures 2.6 and 2.7 we can see the 3D coordinates generated by our algorithm for the library image. In the appendix in figures 0.1, 0.2, 0.3 and 0.4 we can see the same figures (different RANSAC values) with fixed axes.

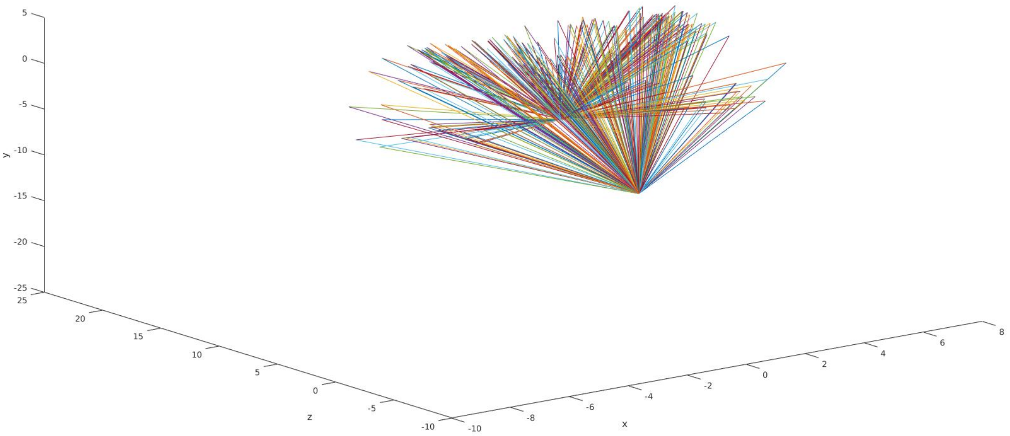

Figure 2.6: Library image: Lines drawn from camera center points to each of the points

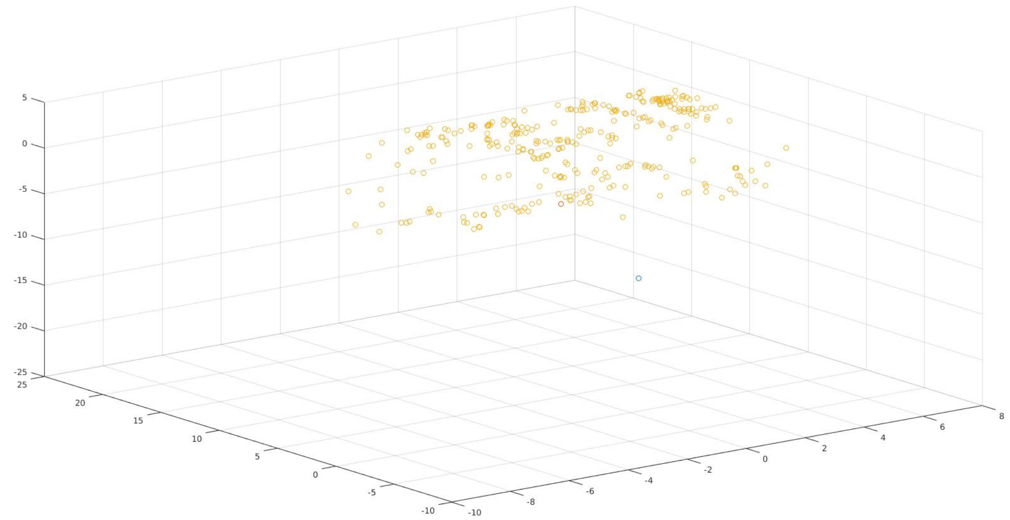

Figure 2.7: Library image: 3D Coordinates of points and camera centers

In addition, in figures 2.8 and 2.9 we can see the 3D coordinates generated by our algorithm for the house image.

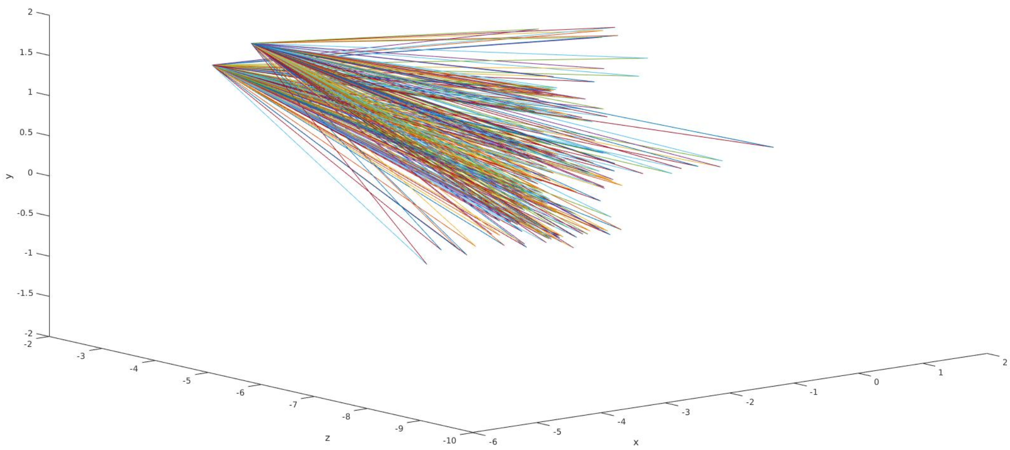

Figure 2.8: House image: Lines drawn from camera center points to each of the points

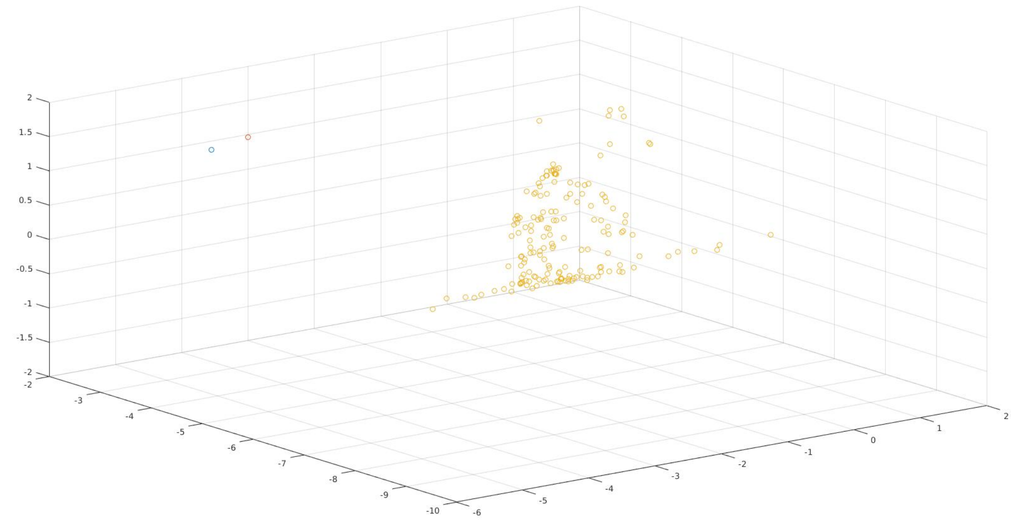

Figure 2.9: House image: 3D Coordinates of points and camera centers

3 Experiments

3.1 Effect of RANSAC hyper parameters

In this experiment we investigate the effect of the hyper parameters of the RANSAC algorithm. The RANSAC algorithm was introduced in section 2.2.

At this point the algorithm has acquired 308 possibly matching features which have to be divided to inliers and outliers and so we tried two different situations. The first one is changing the d when T and s remain the same. In table 3.1 we can see that higher values of d, as it was expected, give us a highest number of inliers. But after a certain value, which in this example was $d = 1.5$ the inliers are including all the matching which could mean that the algorithm is creating more generic and inaccurate models. In the experiment process, we observed that although there is a clear analogous relation between d and the amount of inliers, the random sampling plays a bigger role in determining the outcome as we can have that for example with $d = 1.2$ there were instances that we had 81 or 174 inliers, meaning that in some cases the random sampling did not manage to return the points that fit the best possible fundamental matrix.

The second situation we tried was changing the s value. We tried 4 different values with $d = 1$ and $s = 6, 7, 8, 9$. In table 3.2 the values of inliers seem like they are proportionally increasing when s is increasing, which makes sense considering that a bigger sample provides us with a better model.

d	T	Inliers
0.5	5	35
1	5	74
1.5	5	203
2	5	304
10	5	308

Table 3.1: Effect of different values of d on the inliers found

d	T	s	Inliers
1	5	6	8
1	5	7	43
1	5	8	148
1	5	9	308

Table 3.2: Effect of different values of T on the inliers found

To showcase the randomness of results we present in Figure 3.1 the distribution of inlier amount with respect to error (residual). For the most part, we see that the role of T is rather insignificant with respect to the end result. As for d there is a tendency for a smaller amount of inliers with small error, but still the values are significantly scattered. The bottom right graph is the only one that is clearly indicative of how the number of matching points per sample s is to be optimized, as in all cases we get an amount of 307 or more inliers with very small error.

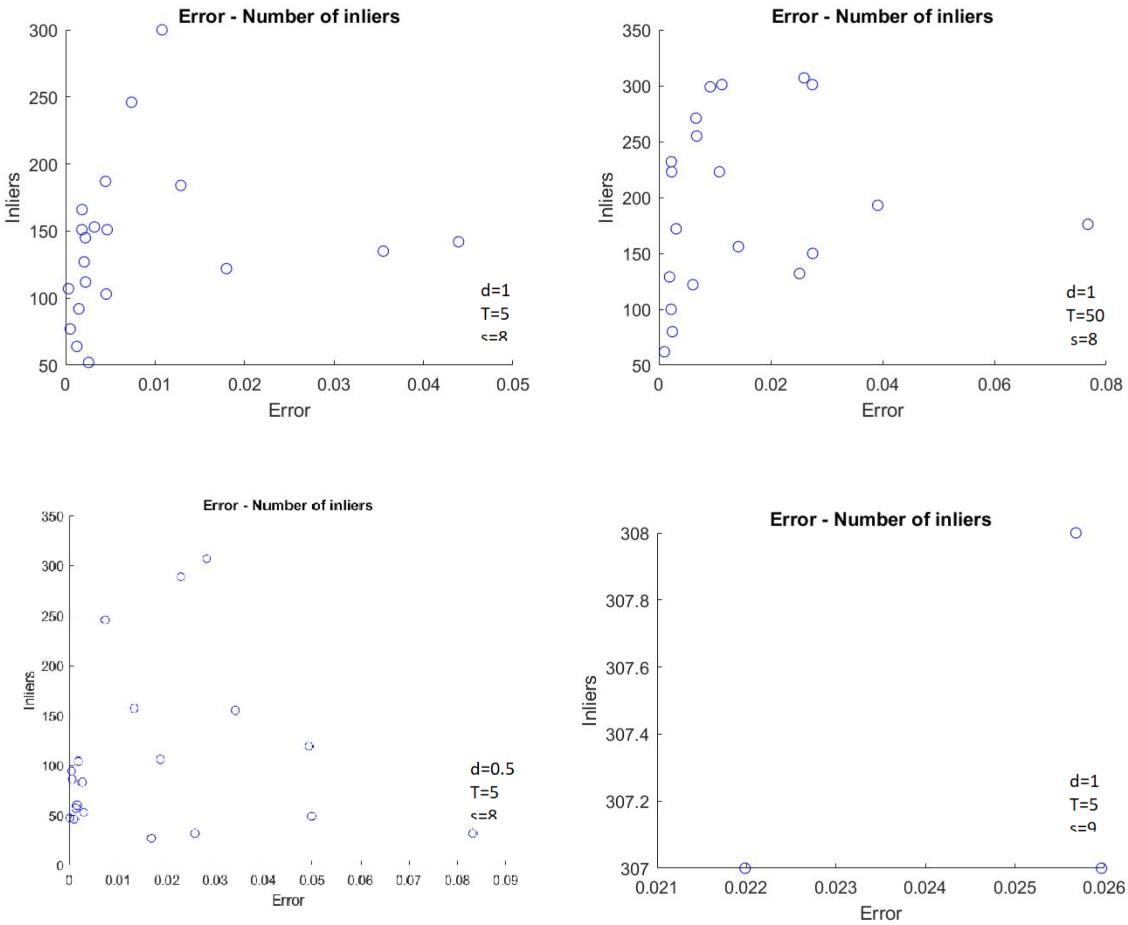

Figure 3.1: Randomness in results

4 Conclusion

We implemented a fundamental matrix estimator and a RANSAC method that utilizes it to find an optimal fundamental matrix for our set of matching points , calculated by the SIFT algorithm, and divide them to inliers and outliers. Our experiments show that the random sampling plays a great role to the results, meaning that there are "good" samples that can provide us with a really well fitting fundamental matrix and vice versa. This conclusion also agrees with the fact that the number of matches s is directly related to the outcome, as the more points for the fundamental matrix estimation the more robust the result. One other way to optimize our results is to allow for more iterations. Finally, we implemented the algebraic solution of the linear triangulation problem, which could be in another context generalized in 3D reconstruction with multiple views.

5 Appendix

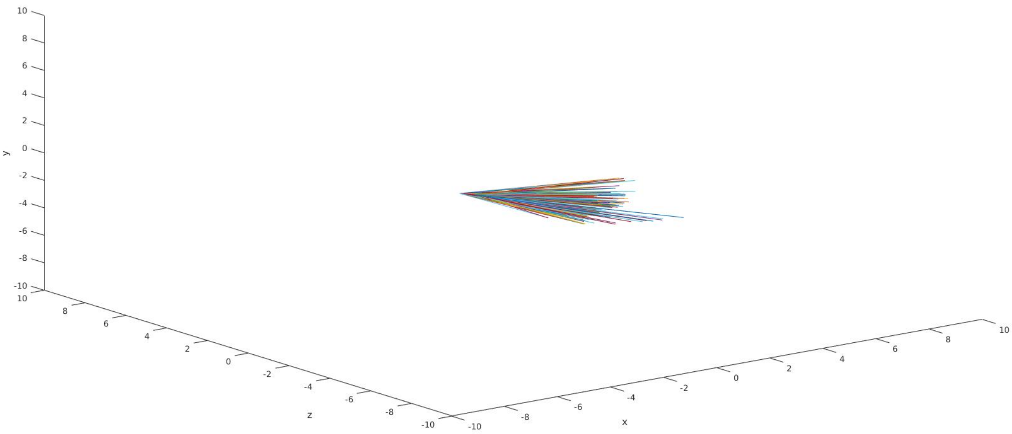

Figure 0.1: House image: Lines drawn from camera center points to each of the points

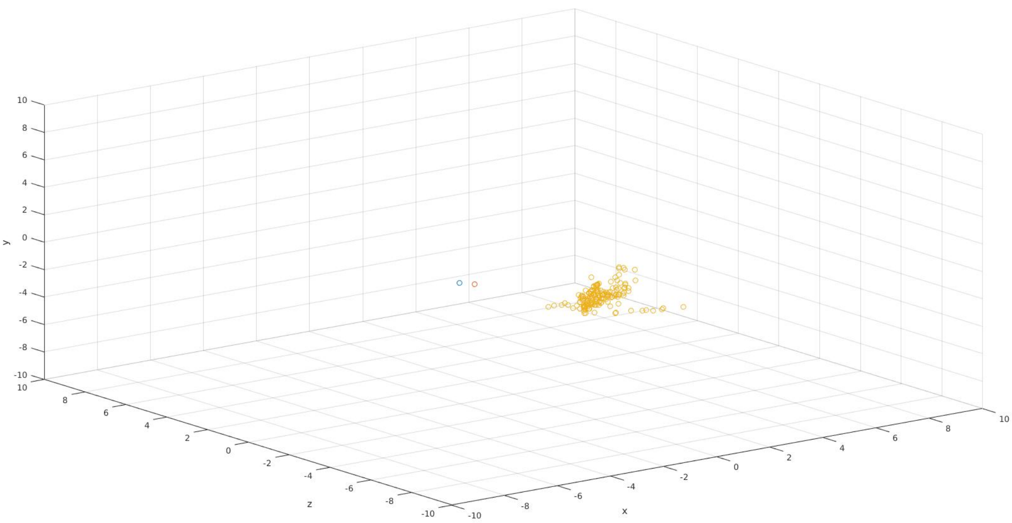

Figure 0.2: House image: 3D Coordinates of points and camera centers

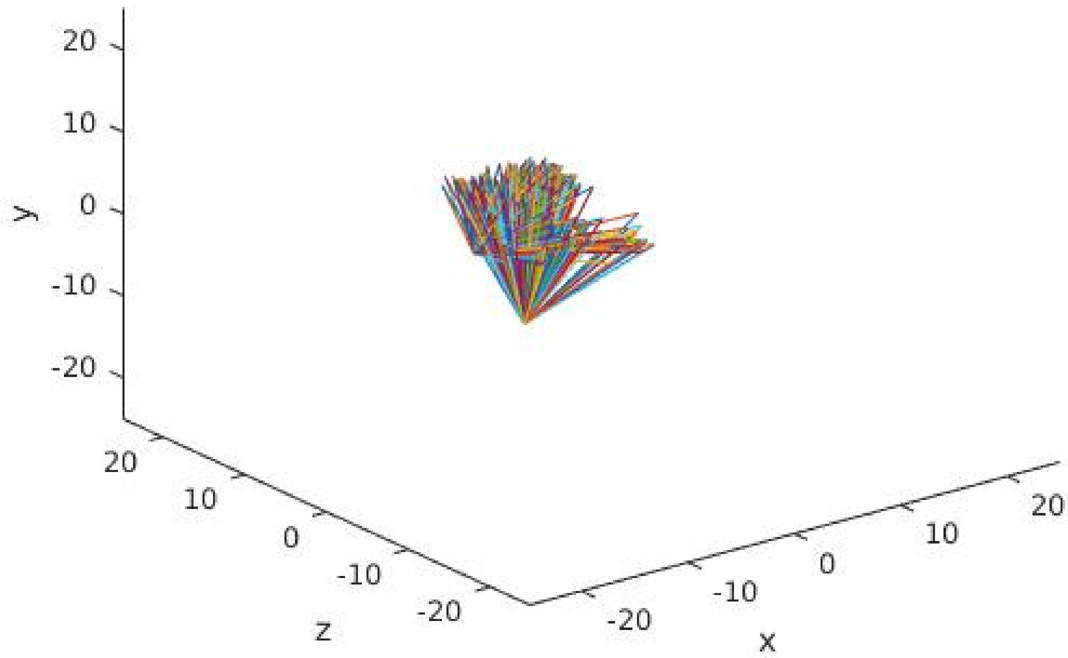

Figure 0.3: Library image: Lines drawn from camera center points to each of the points

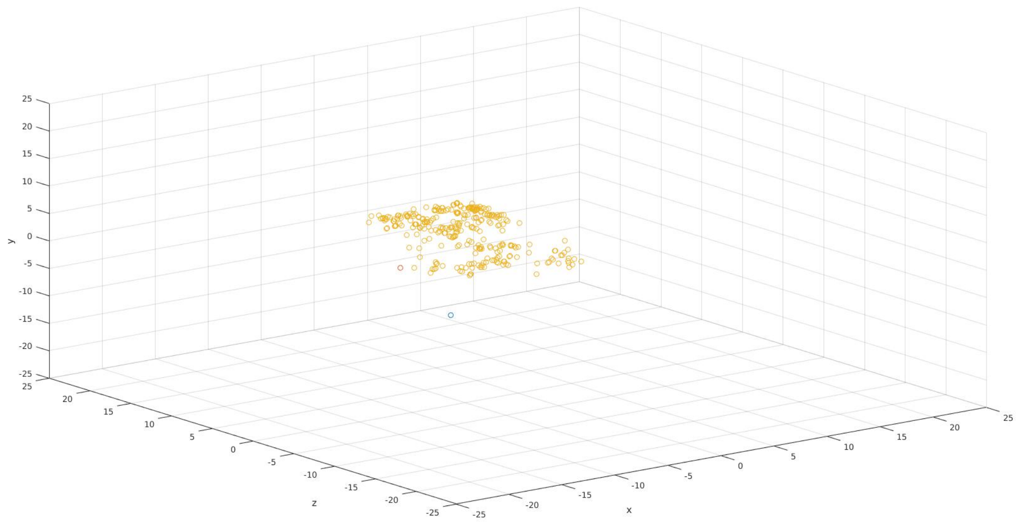

Figure 0.4: Library image: 3D Coordinates of points and camera centers