Assignment 2 - Epipolar Geometry and 3D Reconstruction

Nalet Meinen
Computer Vision

December 8, 2020

1 Part 1: The Eight Point Algorithm

1.1 Normalization Matrix

The first part was to calculate the Normalization Matrix. This is described on the lecture notes as follows:

> Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels

I began with calculating the centroid, then the mean squared distance. Important the mean of two norms results in $\sqrt{2}$, that is why we devide $\frac{\sqrt{2}}{\text{mean}}$.

1.2 Eight Points Algorithm

The formulas for the eight points algorithm where described in the slides from 23 - 35 (lecture 8). It is important to correctly calculate the matrix A, else the SVD is not possible. Slide 33 describes the algorithm very well and is straight forward. For understanding purposes this PDF from Standford helped me to understand it better intuitively.

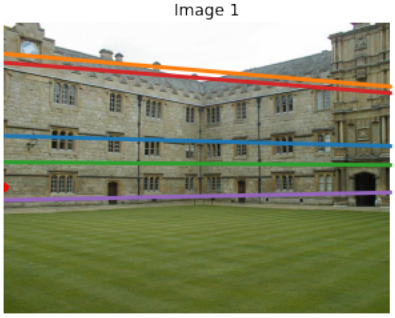

Figure 1: Result of the matched points

The results for the subsets of points were very accurate. In the first version of the code the plot of the line was slightly off where it should have been. This was corrected with `x @ F`. The reason could be a transpose somewhere, which I did not find in the time given.

1.3 Epipole and Epipolar Line

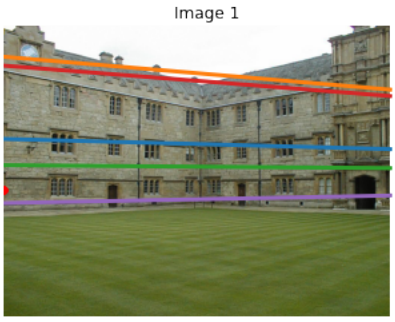

Figure 2: Epipolar line

Deriving a formula was easy. For that we convert the point itself to a line. With the multiplication with F we have then therefore the point/line in the new space. We know x. The MathLibPlot library can draw a line between two points. So I calculated the y points at the border of the image and let the library draw the line.

The task where I must plot the point where the epipolar lines intersect is wrong. Obviously this point is outside of the image as we would guess if I would have followed the lines. I guessed that if the task was written this way the intersection point should be on the image.

2 Part 2: 3D Model Reconstruction

2.1 Ransac

Implementing the RANSAC algorithm took a lot of time. Here I first started an approach without the fundamental matrix using only the distance, after merging the two points clouds in 2D.

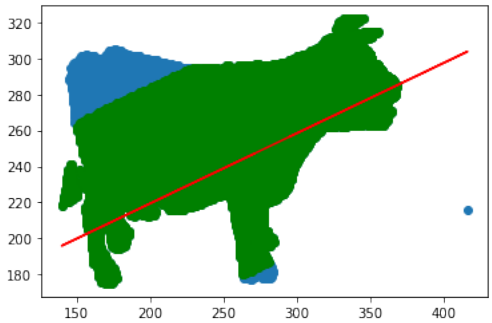

Figure 3: RANSAC inliners

The dataset included two points that were intended as outliners. Finding a good line was difficult and then fitting the line to the data, using then the inliers as the input for the eight-point algorithm. Later I did discard this idea. The correct way was to use the eight-point algorithm as the model itself and then use the least squared to calculate the distance to F. The most difficult part here was to find the correct parameters to get RANSAC working correctly. I only managed to get rid of one outlier instead of the two in the data.

2.2 Essential Matrix

The calculation of the essential matrix was mentioned in the slides. The formula $E = I^T \cdot F \cdot I$ was used.

2.3 Decompose Essential Matrix

For the decomposition of the essential matrix this PDF from Lunds university helped me. Important are the matrices W (rotation) and Z (skew). In the first attempt I tried to change the W in order to get the result aligned to the coordinate-system (positive-wise). then I saw that the provided image of the result is also "backward".

After enforcing rank two with diag($[1, 1, 0]$) and rerunning the SVD. After, we get the two rotations. Also, I got S the two skew matrixes. As said before this PDF describes this very well on page 4.

2.4 Reconstruction of the 3D Points

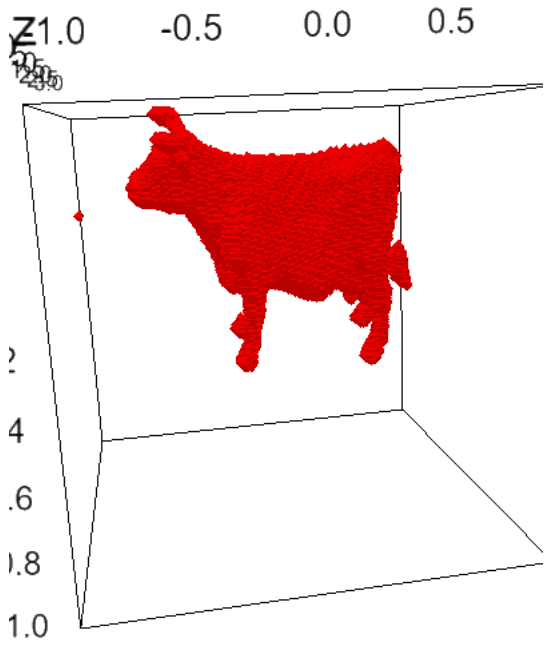

Figure 4: RANSAC inliners

As mentioned before, finding the different parameters for RANSAC was difficult and time-consuming. The decomposition is not unique. Depending on how I ran RANSAC, I had to change E to $-E$ and vice-versa. A good thing was that this issue was also mentioned on the piazza website. This result was one of the better ones. Unfortunately, I did not found a way to make also the second outliner disappear.

In part (b) I had to calculate the error of the model. I was provided with homogenous coordinates which had to be converted. $H_{\text{groundtruth}} = Hr_{\text{groundtruth}} Hl^{-1}_{\text{groundtruth}}$.

As written in the task, I converted the matrix in Euler angles. With the obtained data we then got the results:

- error on angles: 51.5°
- error on translation: 1.1