Computer Vision, Assignment 4 - Model Fitting and Optimization

David Stålmarck

February 2025

Exersice 1

For two cameras:

$$P_1 = [A_1 \ t_1] \quad \text{and} \quad P_2 = [A_2 \ t_2]$$

If they share the same camera center $\mathbf{c}$, we know that:

$$A_1\mathbf{c} + t_1 = 0 \quad \text{and} \quad A_2\mathbf{c} + t_2 = 0.$$

Assuming A_i is invertible for both matrices, we get:

$$\mathbf{c} = -A_1^{-1}t_1 = -A_2^{-1}t_2.$$

Any 3D point X using our cameras can be described as:

$$X = \mathbf{c} + \lambda\mathbf{d}$$

This is just saying the point X is some distance λ away from the camera center in the direction of $\mathbf{d}$.

Inserting X and using $A_i\mathbf{c} = 0$ we get:

For camera 1:

$$\mathbf{x}_1 \sim A_1X = A_1(\mathbf{c} + \lambda\mathbf{d}) = \lambda A_1\mathbf{d}$$

For camera 2:

$$\mathbf{x}_2 \sim A_2X = A_2(\mathbf{c} + \lambda\mathbf{d}) = \lambda A_2\mathbf{d}$$

Since we assumed A_1 is invertible we get $\mathbf{d} \sim A_1^{-1}\mathbf{x}_1$ from $\mathbf{x}_1 \sim A_1\mathbf{d}$. Putting that together with $\mathbf{x}_2 \sim A_2\mathbf{d}$ we get:

$$\mathbf{x}_2 \sim A_2A_1^{-1}\mathbf{x}_1$$

This shows that the transformation between the two image planes is given by:

$$H = A_2A_1^{-1}$$

Alternatively the opposite transformation $\mathbf{x}_1 \sim A_1A_2^{-1}\mathbf{x}_2$ which gives:

$$A_1A_2^{-1} = H^{-1}$$

Exercise 2

A 2D homography has 8 degrees of freedom since a 3×3 matrix (9 parameters) is defined up to scale.

Determining a homography requires 4 point correspondences because each provides two equations.

To achieve 98% confidence in RANSAC with 10% outliers, the number of required iterations is:

$$1 - (1 - 0.90^4)^k \geq 0.98 \quad \Rightarrow \quad k \geq 4.$$

Computer Exercise 1

I used the given pictures a.jpg (A) and b.jpg (B).

Using the pyton cv2-library with their detectAndCompute() function for a SIFT-object, and for finding the inliers using cv2.findHomography, I got:

- Number of SIFT features in A: 1521
- Number of SIFT features in B: 1632
- Number of matches: 542
- Number of inliers: 264 out of 542

Below is the resulting panorama created using SIFT and RANSAC:

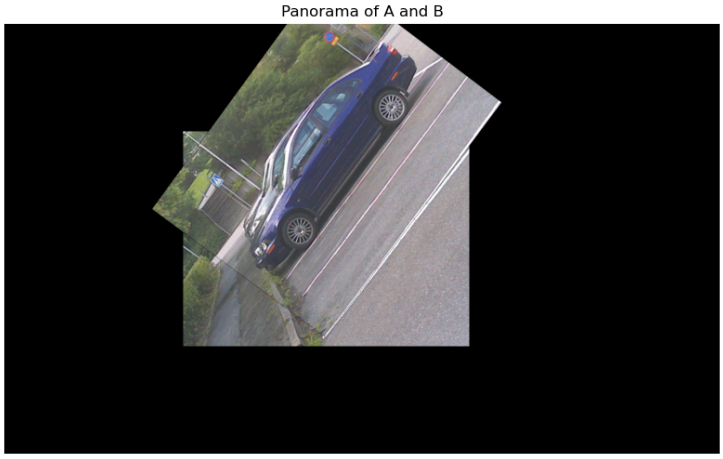


Figure 1: Panorama of images A and B.

1 Exercise 3

An Essential matrix has 5 degrees of freedom because it encodes 3 parameters for rotation and 2 for the direction of translation (its scale is undefined).

Unlike the 2D homography, we only get one equation from the epipolar constraint and therefore the minimal number of point correspondences needed is equal to the degrees of freedom, that will say 5.

To achieve 98% confidence in RANSAC with 10% outliers, the number of required iterations is:

$$1 - (1 - 0.90^5)^k \geq 0.98$$

Solving this shows that we need about 5 iterations.

2 Computer Exercise 2

The total number of inliers identified after RANSAC estimation was **1465**.

The computed RMS reprojection error was **0.3487 pixels**.

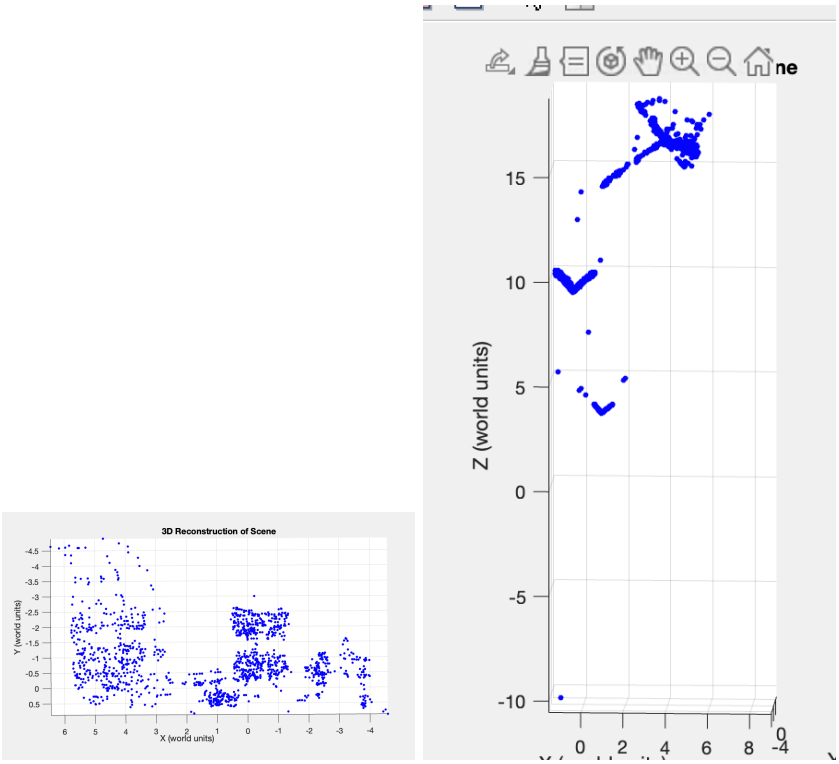


Figure 2: 3D Reconstruction: (Left) Reconstruction in the x-y plane. (Right) 3D visualization to highlight depth structure.

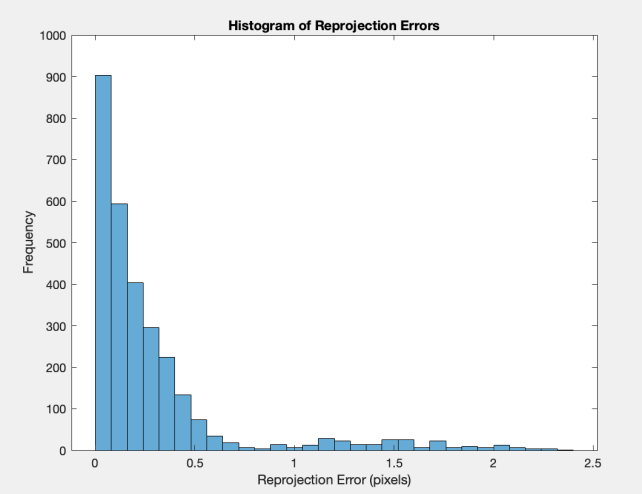


Figure 3: Histogram of Reprojection Errors.

3 Computer Exercise 3

I got stuck a very long time on this one not getting a good result until I had a proper adaptive search for choosing gamma. Really cool to see the improvements afterwards!

Final RMS Error: 0.2959

This value was after 1000 iterations, however, we could see additional small improvements beyond this. I tried running it for many more iterations and got about 0.28 after about 3000 iterations.

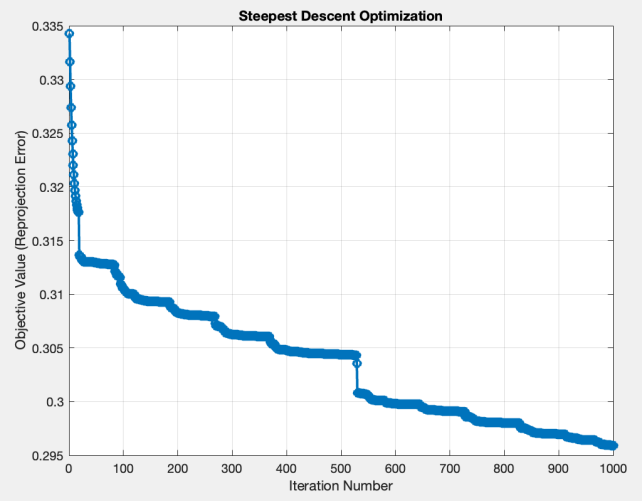


Figure 4: RMS value after n iterations using Steepest Descent Optimization with an adaptive search of choosing gamma.

4 Computer Exercise 4

Final RMS Error: 0.1956

Interestingly enough, already after 5 searches it started convering and didn't improved less than 0.0001 after 350 additional iterations.

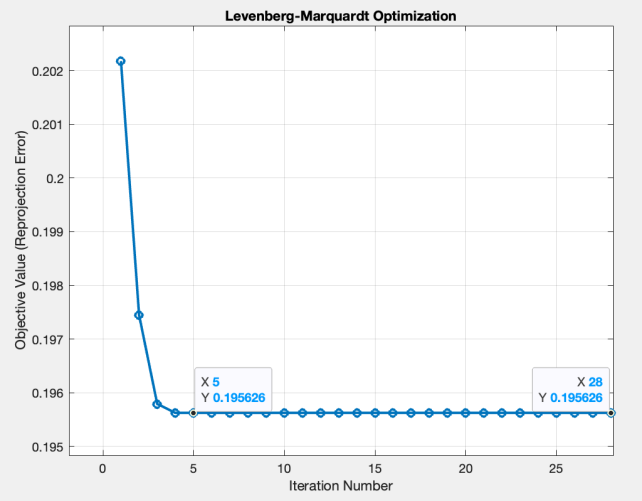
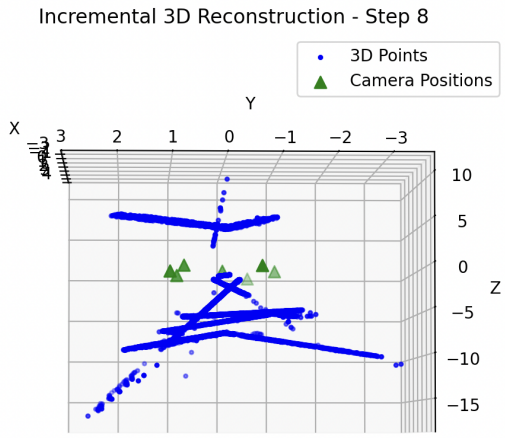
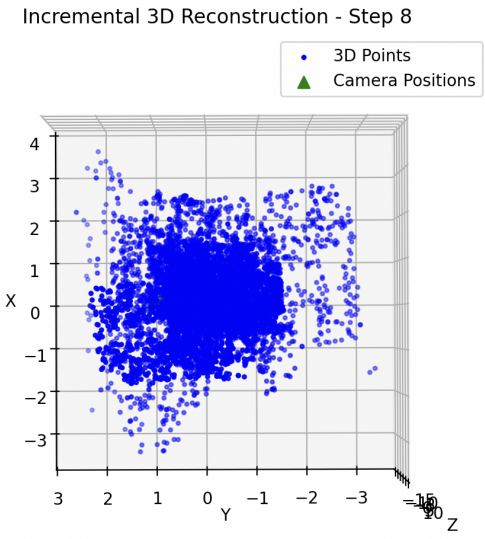


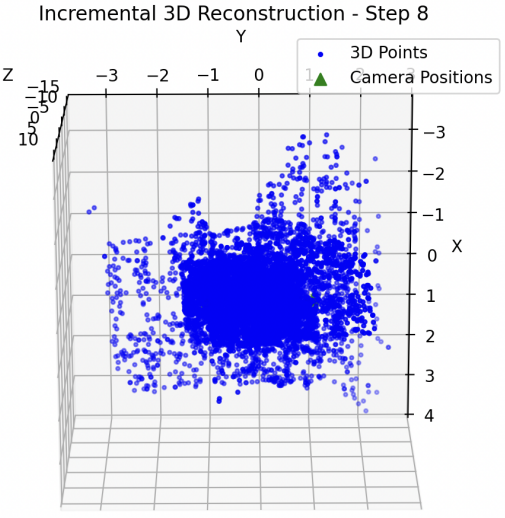
Figure 5: RMS value after n iterations using Levenberg-Marquardt Optimization with an adaptive search for choosing lambda.

5 Computer Exercise 5

I started doing this in matlab but got very few inline matches and therefore switched back to python and their cv2.SIFT. I was not able to get a great reconstruction, but by removing some outliers and lowering Lowe's Ratio get an okay result. However, then after about 10 steps it went downhill and it started to match false points with each other, see picture taken after each step (step 1 is picture 1 and 2, step 2 is picture 2 and 3, step N is picture N and N+1)







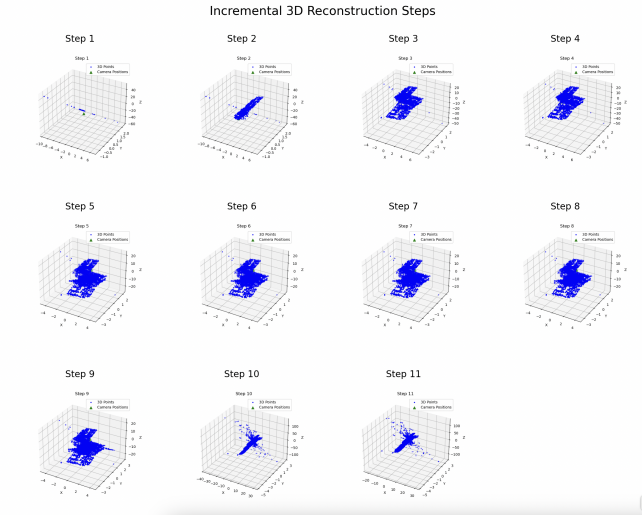


Figure 6: Incrementally how the 3D reconstruction looked like.