Computer Vision - FMAN85
Assignment 2 - Spring 2020

Camera Calibration and DLT Method

Hicham Mohamad, hi8826mo-s
hsmo@kth.se

January 30, 2020

1 Introduction

In this assignment, we are going to study and get familiar with the basic elements of **camera calibration** and Direct Linear Transformation (DLT) method. We will solve the **resection** and **triangulation** problems using DLT and compute inner parameters using RQ factorization. In addition we will try out SIFT for feature detection/matching.

The resulting plots and values in this report are obtained by running the implemented Matlab script `ass2_CE1_CE2.m` and `ass2_CE3_CE4_CE5.m`.

2 Calibrated vs. Uncalibrated Reconstruction

In this section, we recall the **camera equations**

$$\lambda \mathbf{x} = P\mathbf{X} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, 1)$, $\mathbf{X} = (X_1, X_2, X_3, 1)$ and P is a 3×4 matrix. The vector $\mathbf{X}$ represents a 3D point and $\mathbf{x}$ is its projection in the image. The 3×4 matrix P contains the parameters of the camera that captured the image. It can be decomposed into $P = K[R \quad t]$ where R and t encodes orientation and position of the camera, respectively, and K contains the **inner parameters**.

A camera $P = K[R \quad t]$ is called **calibrated** if the inner parameters K are known.

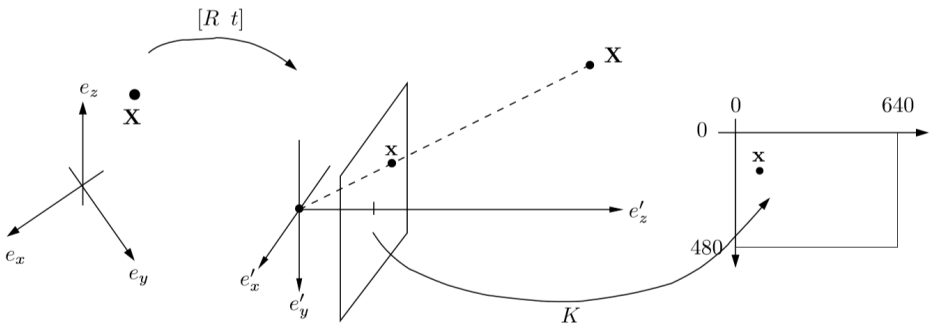


Figure 1: The different coordinate systems and mappings.

2.1 Exercise 1

In this task, we need to show that when estimating structure and motion (3D points and cameras) simultaneously, under the assumption of **uncalibrated cameras**, there is always an unknown projective transformation of 3D space that cannot be determined using only image projections.

Suppose that a set of image projections $\mathbf{x}_{ij}$ are given, i.e. of scene point j in image i. It is assumed that these images come from a set of 3D points $\mathbf{X}_j$, which are unknown. Similarly, the position, orientation and calibration of the cameras P_i are not known. A **reconstruction** task is to find the camera matrices P_i, as well as the 3D points $\mathbf{X}_j$ such that the camera equations is written as

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j \tag{2}$$

By assuming an **unknown** projective transformation T, we can construct a new solution such that

$$\tilde{P}_i = P_iT^{-1} \quad \text{and} \quad \tilde{\mathbf{X}}_j = T\mathbf{X}_j, \tag{3}$$

thus, we write

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j = P_iT^{-1}T\mathbf{X}_j = \tilde{P}_i\tilde{\mathbf{X}}_j \tag{4}$$

In this way, if $\mathbf{X}$ is the estimated 3D-points, then we show that a new solution can always be obtained from T$\mathbf{X}$ for any projective transformation T of 3D space. *This means that given a solution we can apply any projective transformation to the 3D points and obtain a new solution.* In this case, we can conclude that there is **projective ambiguity** present.

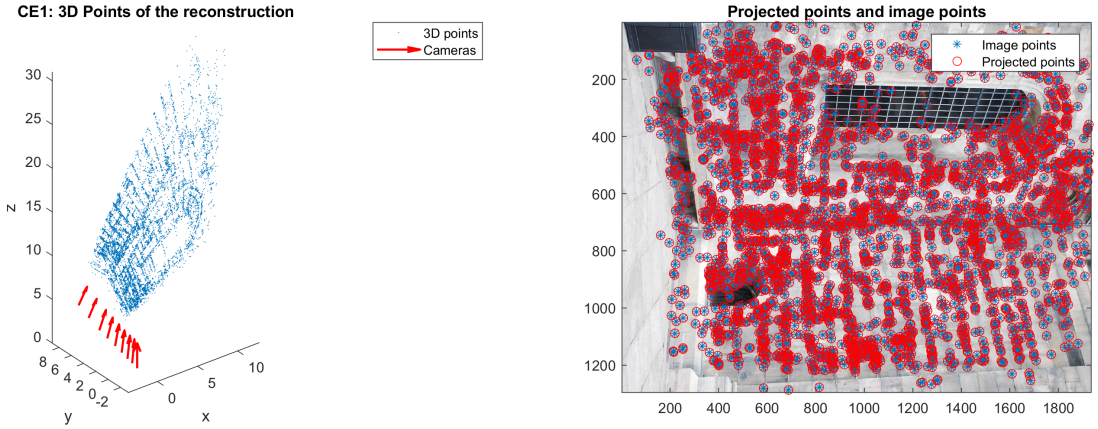
2.2 Computer Exercise 1

In this section, we consider the file `compEx1data.mat` that contains the 3D points of the reconstruction $\mathbf{X}$, the camera matrices P, the image points $\mathbf{x}$ and the filenames `imfiles` of the images. Specifically, we have

- $\mathbf{X}$ is a 4×9471 matrix containing the homogeneous coordinates for all 3D points,
- $\mathbf{x}\{i\}$ is a 3×9471 matrix containing the homogeneous coordinates of the image points seen in image i (NaN means that the point has not been detected in this image).
- $P\{i\}$ contains the camera matrix of image i and `imfiles{i}` contains the name of that image.

2.2.1 Projections and the corresponding image points

The resulting plot the 3D points of the reconstruction is shown in Figure 2a. We use the Matlab file `plotcams.m` to plot the cameras in the same figure. It looks like a reasonable reconstruction.



(a) (b)

Figure 2: a) The 3D points of the reconstruction using uncalibrated cameras. b) Plot of the image, the projected points, and the image points in the same figure.

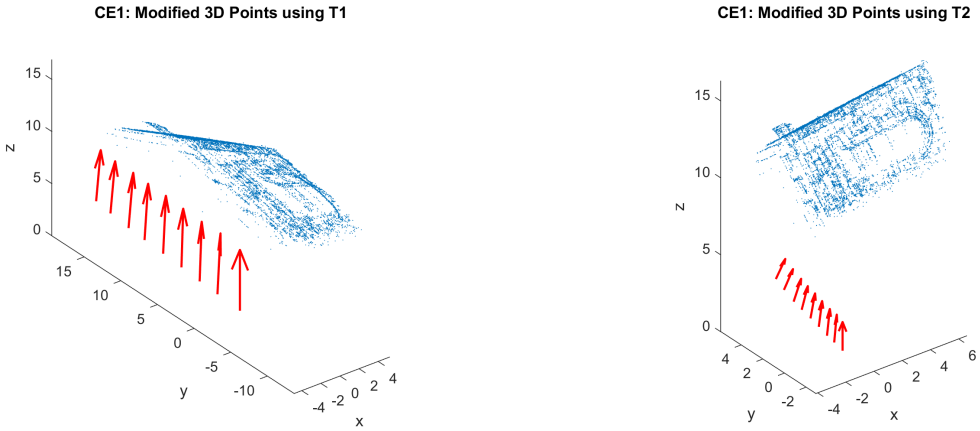
After projecting the 3D detected points into one of the cameras (camera 1), we plot the image, the projected points, and the image points in the same figure as shown in Figure 2b. As usual we should divide by the third coordinate before plotting by using the implemented function `pflat`. As we can notice, the projections appear to be close to the corresponding image points.

2.2.2 Two projective transformations T_1 and T_2

In this section, we use the following two projective transformations to modify the 3D points and cameras (as in Exercise 1) so that two new projective solutions are obtained.

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/10 & 1/10 & 0 & 1 \end{pmatrix} \quad T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/16 & 1/16 & 0 & 1 \end{pmatrix} \tag{5}$$

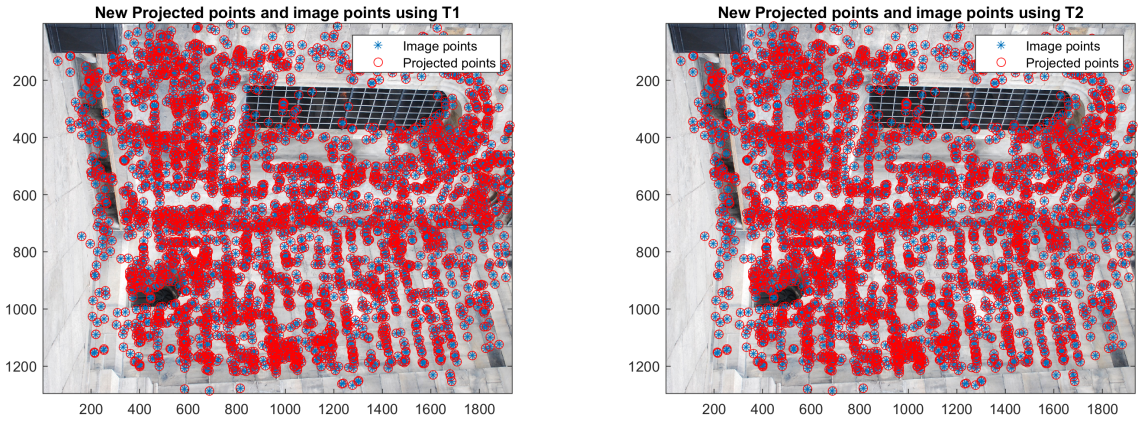
Now after dividing the points by the fourth coordinate by using `pflat.m` function, we plot the 3D points and cameras in the same figure for each of the new solutions, as shown in Figure 3. By comparing them to that in Figure 2, it seems that these new 3D plots look distorted. This effect returns to the fact that projective transformations do not necessarily preserve angles or parallel lines.



(a) (b)

Figure 3: The 3D points and cameras in the same figure for each of the new solutions with T1 and T2 respectively

After projecting the new 3D points into one of the cameras, we plot the image, the projected points, and the image points in the same figure, as illustrated in Figure 4. As we expect, the new projections doesn't appear to have changed.



(a) (b)

Figure 4: Plots of the image, the projected points, and the image points in the same figure for each of the new solutions with T1 and T2 respectively

2.3 Exercise 2

when we use calibrated cameras, we can not get the same projective distortions as in Computer Exercise 1, since we actually remove the projective ambiguity. This solution is called **Euclidean reconstruction**

which is determined up to a similarity transformation.

3 Camera Calibration

The normalization is carried out by applying the homography K^{-1} to the image coordinates

3.1 Exercise 3

We recall that a camera $P = K[R \quad t]$ is called **calibrated** if the inner parameters K are known. In this task, we suppose that a camera has got the inner parameters

$$K = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

The inverse of K is computed by

$$K^{-1} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \frac{1}{\det(K)} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ -fx_0 & -fy_0 & f^2 \end{pmatrix}^T = \frac{1}{f^2} \begin{pmatrix} f & 0 & -fx_0 \\ 0 & f & -fy_0 \\ 0 & 0 & f^2 \end{pmatrix} = \begin{pmatrix} 1/f & 0 & -x_0/f \\ 0 & 1/f & -y_0/f \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

In addition, this matrix can be factorized into

$$K^{-1} = \underbrace{\begin{pmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}}_{B} \tag{8}$$

The geometric interpretation of the the transformations A is to rescale the coordinates by the focal length f ,whereas B translates them by the principal point (x_0, y_0).

When normalizing the image points of a camera with known inner parameters we apply the transformation K^{-1}. In this context, this operation means calibrating the camera.

The inner parameters, i.e. the K matrix transforms the image points according to

$$\begin{pmatrix} fx + x_0 \\ fy + y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{9}$$

In this way, we can see that the principal point (x_0, y_0) end up in $(x_0(f+1), y_0(f+1))$

A point with distance f to the principal point can be expressed by $(x_0 + f/\sqrt{2}, y_0 + \sqrt{2})$. Thus, this point ends up in

$$\begin{cases} x = fx + x_0 = f(x_0 + f/\sqrt{2}) + x_0 = x_0(f+1) + f^2/\sqrt{2} \\ y = fy + y_0 = f(y_0 + f/\sqrt{2}) + y_0 = y_0(f+1) + f^2/\sqrt{2} \end{cases} \tag{10}$$

Now we suppose that for a camera with resolution 640×480 pixels we have the inner parameters

$$K = \begin{pmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \implies K^{-1} = \begin{pmatrix} 1/f & 0 & -x_0/f \\ 0 & 1/f & -y_0/f \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/320 & 0 & -1 \\ 0 & 1/320 & -240/320 \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

Now we can normalize the points $x_1 = (0, 240)$, and $x_2 = (640, 240)$ as

$$\tilde{\mathbf{x}}_1 = K^{-1}\mathbf{x}_1 = \begin{pmatrix} 1/320 & 0 & -1 \\ 0 & 1/320 & -240/320 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 240 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \tag{12}$$

$$\tilde{\mathbf{x}}_2 = K^{-1}\mathbf{x}_2 = \begin{pmatrix} 1/320 & 0 & -1 \\ 0 & 1/320 & -240/320 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 640 \\ 240 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \tag{13}$$

The angle α between the viewing rays projecting to these points can be calculating by using the scalar product formula

$$\cos\alpha = \frac{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}_2}{||\tilde{\mathbf{x}}_1||||\tilde{\mathbf{x}}_2||} = 0 \tag{14}$$

From this we can conclude that $\alpha = 90$.

Same camera center and principal axis

The camera $K[R \quad t]$ and the corresponding normalized version $[R \quad t]$ have the same camera center

$$\begin{cases} K[R \quad t]C = 0 \\ [R \quad t]C_n = 0 \end{cases} \implies C = C_n = R^T t \tag{15}$$

They have also the same principal axis because we know that $V = R_3^T$, i.e. the third row. Also the matrix K has the third row $K_3 = (0, 0, 1)$ so the multiplication by K does not actually affect the result.

3.2 Exercise 4

In this exercise we consider the camera

$$P = \begin{pmatrix} 1000 & -250 & 250\sqrt{3} & 500 \\ 0 & 500(\sqrt{3} - \frac{1}{2}) & 500(1 + \frac{\sqrt{3}}{2}) & 500 \\ 0 & \frac{-1}{2} & \frac{\sqrt{3}}{2} & 1 \end{pmatrix} \tag{16}$$

If the focal length is $f = 1000$ and the principal point is $x_0, y_0) = (500, 500)$, we need to normalize the camera. When the skew is zero and the aspect ratio is one, we can write the inner matrix K by

$$K = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1000 & 0 & 500 \\ 0 & 1000 & 500 \\ 0 & 0 & 1 \end{pmatrix} \tag{17}$$

$$\implies K^{-1} = \begin{pmatrix} 1/f & 0 & -x_0/f \\ 0 & 1/f & -y_0/f \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/1000 & 0 & -1/2 \\ 0 & 1/1000 & -1/2 \\ 0 & 0 & 1 \end{pmatrix} \tag{18}$$

Then normalizing the camera means multiplying P by K^{-1}

$$\tilde{P} = K^{-1}P = \begin{pmatrix} 1/1000 & 0 & -1/2 \\ 0 & 1/1000 & -1/2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1000 & -250 & 250\sqrt{3} & 500 \\ 0 & 500(\sqrt{3} - \frac{1}{2}) & 500(1 + \frac{\sqrt{3}}{2}) & 500 \\ 0 & \frac{-1}{2} & \frac{\sqrt{3}}{2} & 1 \end{pmatrix} \tag{19}$$

Thus we get

$$\tilde{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & 0.5 & 0 \\ 0 & -0.5 & 0.866 & 1 \end{pmatrix} \tag{20}$$

If the images are of size 1000×1000 pixels, we can normalize the corners of the image (0,0),(0,1000),(1000,0), (1000,1000) and the center (500,500) by applying

$$\tilde{\mathbf{x}}_1 = K^{-1}\mathbf{x}_1 = \begin{pmatrix} -1/2 \\ -1/2 \\ 1 \end{pmatrix} \tag{21}$$

$$\tilde{\mathbf{x}}_2 = K^{-1}\mathbf{x}_2 = \begin{pmatrix} -1/2 \\ 1/2 \\ 1 \end{pmatrix} \tag{22}$$

$$\tilde{\mathbf{x}}_3 = K^{-1}\mathbf{x}_3 = \begin{pmatrix} 1/2 \\ -1/2 \\ 1 \end{pmatrix} \tag{23}$$

$$\tilde{\mathbf{x}}_4 = K^{-1}\mathbf{x}_4 = \begin{pmatrix} 1/2 \\ 1/2 \\ 1 \end{pmatrix} \tag{24}$$

$$\tilde{\mathbf{x}}_0 = K^{-1}\mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{25}$$

4 RQ Factorization and Computation of K

As mentioned in the lecture note, we usually use K for the right triangular matrix and R for the orthogonal matrix. Given a camera matrix $P = [A \quad a]$ we want to use RQ-factorization to find K and R such that $A = KR$, where

$$\begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}, \quad A = \begin{pmatrix} A_1^T \\ A_2^T \\ A_3^T \end{pmatrix}, \quad R = \begin{pmatrix} R_1^T \\ R_2^T \\ R_3^T \end{pmatrix} \tag{26}$$

such that R_1, R_2, R_3 and A_1, A_2, A_3 are 3×1 vectors containing the rows of R and A respectively.

4.1 Exercise 5

We can verify by matrix multiplication that

$$A = KR = \begin{pmatrix} A_1^T \\ A_2^T \\ A_3^T \end{pmatrix} = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} R_1^T \\ R_2^T \\ R_3^T \end{pmatrix} = \begin{pmatrix} aR_1^T + bR_2^T + cR_3^T \\ dR_2^T + eR_3^T \\ fR_3^T \end{pmatrix} \tag{27}$$

If we consider

$$P = \begin{pmatrix} 800/\sqrt{2} & 0 & 2400\sqrt{2} & 4000 \\ -700/\sqrt{2} & 1400 & 700/\sqrt{2} & 4900 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 1 \end{pmatrix} \tag{28}$$

we need to find R_3 (R is an orthogonal matrix) and f. From the third row of 27 we see that

$$A_3 = fR_3 \tag{29}$$

Since the matrix R is **orthogonal** R_3 has to have the length 1. We therefore need to select

$$f = ||A_3|| = \sqrt{1/2 + 1/2} = 1 \quad \text{and } R_3 = \frac{A_3}{||A_3||} = \begin{pmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{pmatrix} \tag{30}$$

Using the second row of the camera matrix, and the fact that $R_3 \perp R_2$, $||R3|| = ||R2|| = 1$, we need to compute R_2, d, e. When R_3 is known we can proceed to the second row of 27. The equation

$$A_2 = dR_2 + eR_3 \tag{31}$$

tells us that A_2 is a linear combination of two orthogonal vectors (both of length one). Hence, the coefficient e can be computed from the scalar product by projecting A_3 onto R_3

$$e = A_2^T R_3 = \begin{pmatrix} -700/\sqrt{2} & 1400 & 700/\sqrt{2} \end{pmatrix} \begin{pmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{pmatrix} = 350 + 350 = 700 \tag{32}$$

Then, we can write

$$A_2 = dR_2 + eR_3 \implies dR_2 = A_2 - eR_3 \implies d = ||A_2 - eR_3|| = \| \begin{pmatrix} -700/\sqrt{2} \\ 1400 \\ 700/\sqrt{2} \end{pmatrix} - \begin{pmatrix} -700/\sqrt{2} \\ 0 \\ 700/\sqrt{2} \end{pmatrix} \| \tag{33}$$

Now we can find d

$$d = \sqrt{0 + (1400)^2 + 0} = 1400 \tag{34}$$

Now we can find R_2

$$R_2 = \frac{A_2}{d} - e\frac{R_3}{d} = \begin{pmatrix} -1/2\sqrt{2} \\ 0 \\ 1/2\sqrt{2} \end{pmatrix} - \begin{pmatrix} -700/\sqrt{2} \\ 0 \\ 700/\sqrt{2} \end{pmatrix} \tag{35}$$

When R_2 and R_3 is known we similarly use the first row of 27 to compute b and c from

$$A_1 = aR_1 + bR_2 + cR_3 \tag{36}$$

By projecting A_1 onto R_3 we get

$$c = A_1^T R_3 \tag{37}$$

Then we project A_1 onto R_2 we get

$$b = A_1^T R_2 \tag{38}$$

Finally, we can compute R_1, a from

$$A_1 - bR_2 - cR_3 = aR_1 \tag{39}$$

What is the focal length, skew, aspect ratio, and principal point of the camera?

The resulting matrix K is not necessarily of the form

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{40}$$

since element (3,3) might not be one. To determine the individual parameters, focal length, principal point etc. we therefore need to divide the matrix with element (3,3). As we know, this does not modify the camera in any way since the scale is arbitrary.

4.2 Computer Exercise 2

Using `rq.m` to compute the RQ factorization of a matrix, we can compute K for one camera in each of the solutions we obtained in computer exercise 1 and we get

$$K1 = \begin{pmatrix} 2394.0 & 0 & 932.4 \\ 0 & 599.5 & 628.3 \\ 0 & 0 & 1 \end{pmatrix} \quad K2 = \begin{pmatrix} 2394 & 0 & 932.4 \\ 0 & 2398.1 & 628.3 \\ 0 & 0 & 1 \end{pmatrix} \tag{41}$$

As we can see, `K1` and `K2` don't represent the same transformation. Actually, the two matrices don't differ by a scale factor.

5 Direct Linear Transformation DLT

The problem of determining the camera matrix P from known scene points $\mathbf{X}_i$ and their projections $\mathbf{x}_i$ is called the **resection** problem. To solve the problem we will use the method called **Direct Linear Transformation** (DLT). This method formulates a homogeneous linear **system of equations** and solves this by finding an approximate **null space** of the system matrix. The goal is to solve the camera equations

$$\lambda_i \mathbf{x}_i = P\mathbf{X}_i \quad i = 1, \cdots, n \tag{42}$$

$$\begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} \tag{43}$$

where p_i are 4×1 vectors containing the rows of P. The first equality in the system in 42 can be written as

$$\begin{cases} \mathbf{X}_1^T p_1 - \lambda_1 x_1 = 0 \\ \mathbf{X}_1^T p_2 - \lambda_1 y_1 = 0 \\ \mathbf{X}_1^T p_3 - \lambda_1 = 0 \end{cases} \tag{44}$$

where $\mathbf{x}_i = (x_i, y_i, 1)$. In matrix form this can be written as

$$\begin{bmatrix} \mathbf{X}_i^T & 0 & 0 & -x_i \\ 0 & \mathbf{X}_i^T & 0 & -y_i \\ 0 & 0 & \mathbf{X}_i^T & -1 \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{45}$$

If we include all the projection equations in one matrix we get a system of the form

$$\begin{bmatrix} \mathbf{X}_1^T & 0 & 0 & -x_1 & 0 & 0 & \cdots \\ 0 & \mathbf{X}_1^T & 0 & -y_1 & 0 & 0 & \cdots \\ 0 & 0 & \mathbf{X}_1^T & -1 & 0 & 0 & \cdots \\ \mathbf{X}_2^T & 0 & 0 & 0 & -x_2 & 0 & \cdots \\ 0 & \mathbf{X}_2^T & 0 & 0 & -y_2 & 0 & \cdots \\ 0 & 0 & \mathbf{X}_2^T & 0 & -1 & 0 & \cdots \\ \mathbf{X}_3^T & 0 & 0 & 0 & 0 & -x_3 & \cdots \\ 0 & \mathbf{X}_3^T & 0 & 0 & 0 & -y_3 & \cdots \\ 0 & 0 & \mathbf{X}_3^T & 0 & 0 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \implies Mv = 0 \tag{46}$$

In most cases the system $Mv = 0$ will not have any exact solution due to noise in the measurements. Therefore we will search for a solution to the minimization problem

$$\min_{||v||^2=1} ||Mv||^2 \tag{47}$$

We refer to this type of problem as a **homogeneous least squares** problem.

5.1 Exercise 7

When using DLT it is often advisable to normalize the points before doing computations. Suppose the image points $\mathbf{x}$ are **normalized** by the mapping N by

$$\tilde{\mathbf{x}} \sim N\mathbf{x} \tag{48}$$

and that we compute a **camera matrix** in the new coordinate system, that is, we obtain a camera $\tilde{P}$ that solves

$$\tilde{\mathbf{x}} \sim \tilde{P}\mathbf{X} \tag{49}$$

Now we can compute the camera P that solves the original problem

$$\mathbf{x} \sim P\mathbf{X} \tag{50}$$

from $\tilde{P}$ by writing

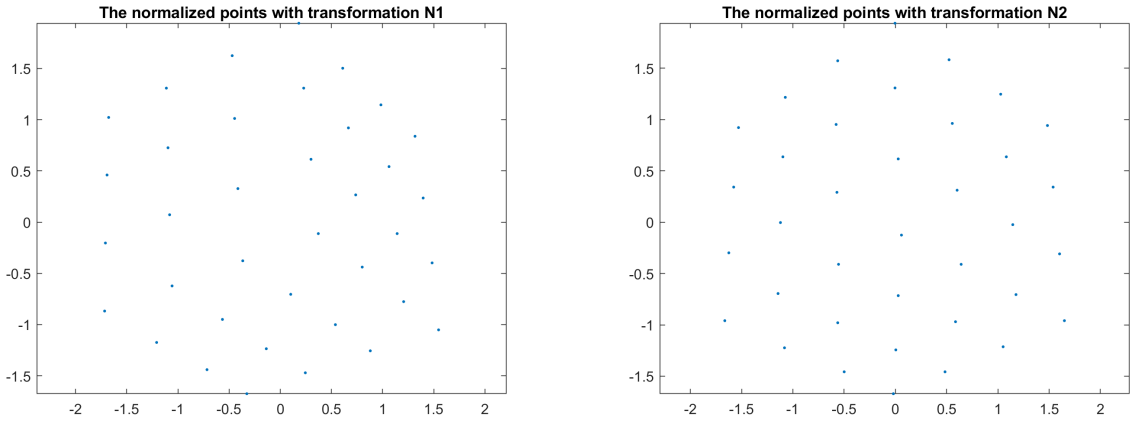
$$\tilde{\mathbf{x}} \sim N\mathbf{x} \sim \tilde{P}\mathbf{X} \implies \mathbf{x} \sim N^{-1}\tilde{P}\mathbf{X} \sim P\mathbf{X} \implies P \sim N^{-1}\tilde{P} \tag{51}$$

5.2 Computer Exercise 3

In this section we consider two images `cube1.jpg` and `cube2.jpg` of a scene with a Rubics cube. The file `compEx3data.mat` contains a point model `Xmodel` of the visible cube sides, the measured projections x of the model points in the two images and two variables `startind`, `endind` that can be used for plotting lines on the model surface.

5.2.1 Normalizing the measured points

Here we need to normalize the measured points by applying a transformation N that subtracts the mean of the points and then re-scales the coordinates by the standard deviation in each coordinate. The resulting plots of the normalized points are shown in Figure 5. As we can see it looks like the points are centered around $(0, 0)$ with mean distance 1 to (0,0).



(a) (b)

Figure 5: Plot of the normalized points by applying the transformation N.

5.2.2 DLT equations

First we set up the DLT equations in Matlab for **resectioning** as shown in Listing 1. Then by using SVD we can solve the resulting homogeneous least squares system

$$\min_{||v||^2=1} ||Mv||^2 \tag{52}$$

Listing 1: The code for Setting up the DLT equations for resectioning: matrix M.

```
1
2  Xnb = size(Xmodel,2);
3
4  M1 = zeros(Xnb*3, 4*3+Xnb);
5  for iM=1:Xnb
6      M1(3*(iM-1)+1,1:4) = [Xmodel(:,iM); 1]';
7      M1(3*(iM-1)+2,5:8) = [Xmodel(:,iM); 1]';
8      M1(3*(iM-1)+3,9:12) = [Xmodel(:,iM); 1]';
9      M1(3*(iM-1)+1:3*iM,12+iM) = -xtilde1(:,iM);
10 end
```

$S^T S$ is a diagonal matrix and contains the eigenvalues where can see actually that the smallest eigenvalue is close to zero. Then we can see also that $||Mv|| = 0.0151$. This return to the fact that $Mv = 0$ will not have any exact solution due to noise in the measurements.

5.2.3 The camera matrix

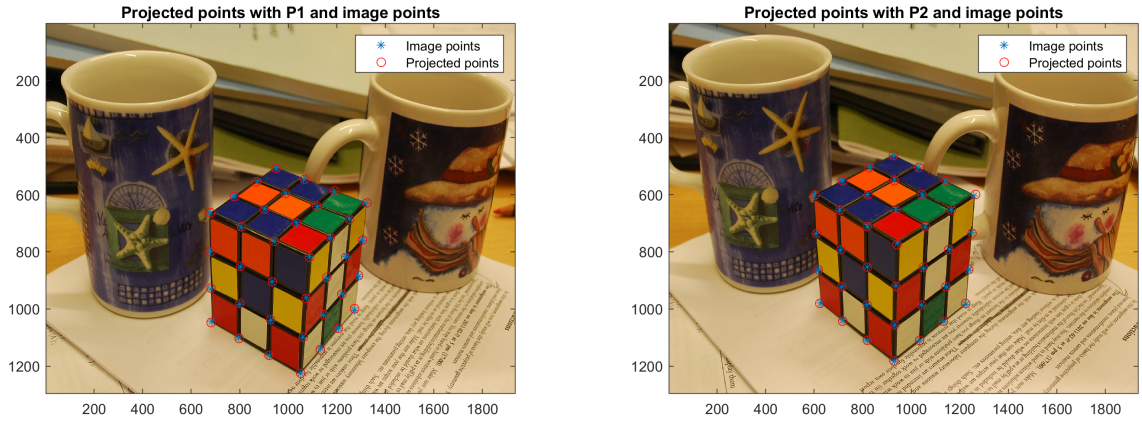
To extract the entries of the camera from the solution and set up the camera matrix we use the following Matlab instructions

```
1  P1tilde = reshape(-v1star(1:12),[4 3])';
2  % set up the camera matrix
3  P1 = N1\P1tilde;
```

In this way we make sure that we select the solution where the points are in front of the camera. (If $\mathbf{X}$ has 4th coordinate 1 then the 3rd coordinate of $P\mathbf{X}$ should be positive for X to be in front of the camera.)

5.2.4 Project the model points into the images

Before projecting the model points we transform the camera matrix to the original (un-normalized) coordinate system, as in Exercise 7. As illustrated in Figure 2b, we plot the measured image points in the same figure. It is clear that they are close to each other.



(a) (b)

Figure 6: Plot of the measured image points in the same figure with the projected points.

Also we plot the camera centers and viewing directions in the same plot as the model points in Figure 7. The result looks reasonable.

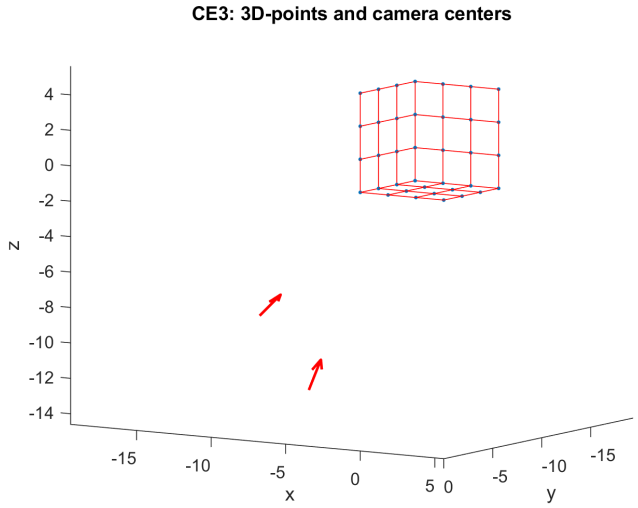


Figure 7: plot of the camera centers and viewing directions in the same plot as the model points.

5.2.5 The inner parameters

Here we compute the inner parameters of the first camera `K1` using RQ factorization `rq.m`. These must be the "true" parameters because we are dealing with a DLT problem where we are assuming that the scene points X_i and their projections x_i are known, and consequently there is **no ambiguity** as in Exercise 1.

$$K1 = \begin{pmatrix} 12.7600 & -0.0938 & 5.0043 \\ 0 & 12.7513 & 3.5222 \\ 0 & 0 & 0.0052 \end{pmatrix} \tag{53}$$

6 Feature Extraction and Matching using SIFT

In this section we try feature extraction using SIFT. After downloading and starting `VLFeat` from `http://www.vlfeat.org/download.html`, we start Matlab and run `vlsetup.m`.

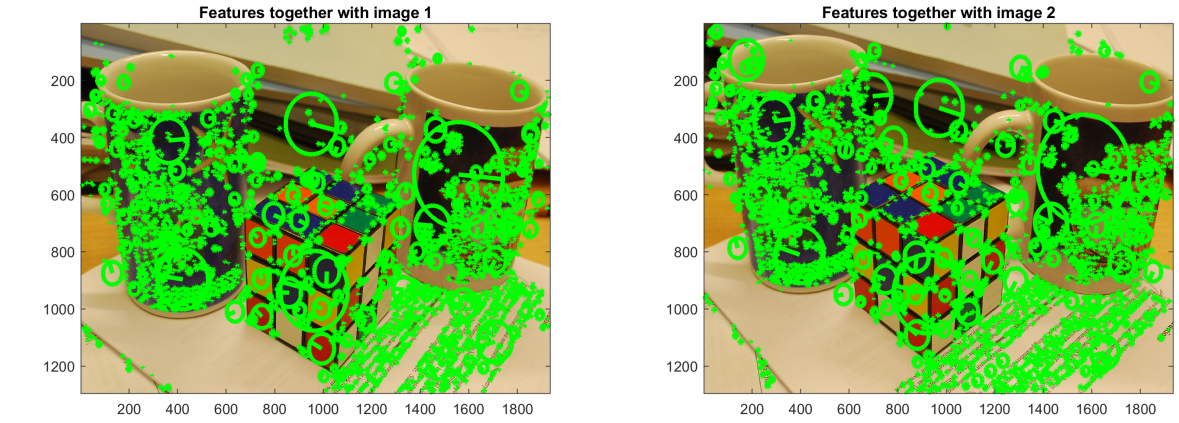
6.1 Computer Exercise 4

In this task, we load the images `cube1.jpg` and `cube2.jpg` then we Compute sift features using `vlfeat`. A plot of the features together with the images is shown in Figure 8.

The SIFT detector searches for **peaks** in scale space (similar to peaks in the **autocorrelation function**, see lecture notes). The second argument filters out peaks that are too small.

```
[f1 d1] = vl_sift( single(rgb2gray(imCube1)), 'PeakThresh', 1);
% match the descriptors
[matches ,scores] = vl_ubcmatch(d1,d2);
% We can now extract matching points using:
x1 = [f1(1,matches (1 ,:));f1(2,matches (1 ,:))];
x2 = [f2(1,matches (2 ,:));f2(2,matches (2 ,:))];
```

The vector `f1` contains 4 rows. The first two rows are the coordinates of the detected features. The second two contains an orientation and scale for which the the feature was detected.



(a) (b)

Figure 8: plot of the images cube1.jpg and cube2.jpg together with the features after computing SIFT using vlfeat.

With the following code, we can randomly select 10 matches and plot the two images next to each other and then plot lines between the matching points, as illustrated in Figure 9.

```
perm = randperm(size(matches ,2));
figure;
imagesc ([imCube1 imCube2]);
hold on;
plot([x1(1,perm (1:10)); x2(1,perm (1:10))+ size(imCube1 ,2)], ...
     [x1(2,perm (1:10)); x2(2,perm (1:10))] ,'-');
hold off;
title('Lines between the 10 matching points')
```

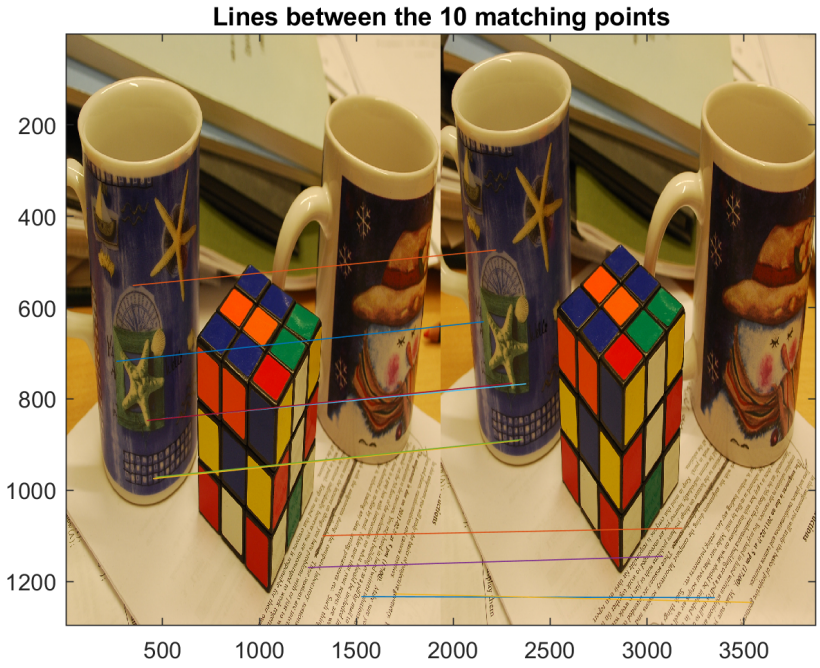


Figure 9: Plot of the two images next to each other and the lines between the 10 matching points selected randomly. It seems that all of the matches appear to be correct.

7 Triangulation using DLT

Triangulation is a method for finding the position of a 3D point $\mathbf{X}$ given that the projections $\mathbf{x}_i$ of this point in a number of images is known, as well as the camera matrices P_i.

If the camera matrix is known, then a 3D point projected in the corresponding image must lie on the 3D line that intersects the **image point** and the **focal point** of the camera. Hence triangulation can be seen geometrically as finding the intersection of a number of 3D lines, as shown in Figure 10. It is clear that we at least need two lines, and so at least projections of the 3D point in two images. We can describe our problem using the camera equations as

$$\lambda_i \mathbf{x}_i = P_i \mathbf{X}_i \quad i = 1, \cdots, n \tag{54}$$

The problem is linear in the unknown λ_i and $\mathbf{X}$, so we can find the least squares **solution** to this problem using **DLT method** by formulating it as

$$Mv = 0 \tag{55}$$

where

$$M = \begin{bmatrix} P_1 & -x_1 & 0 & \cdots & 0 \\ P_2 & 0 & -x_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & 0 \\ P_n & 0 & 0 & \cdots & -x_n \end{bmatrix} \quad v = \begin{bmatrix} \mathbf{X} \\ \lambda_1 \\ \lambda_2 \\ \vdots \end{bmatrix} \tag{56}$$

Actually, the Direct Linear Transformation DLT method consists in solving the minimizing problem

$$\min_{||v||^2=1} ||Mv||^2 \tag{57}$$

by computing the **singular value decomposition** $M = USV^T$ of M and let v be the last column of V .

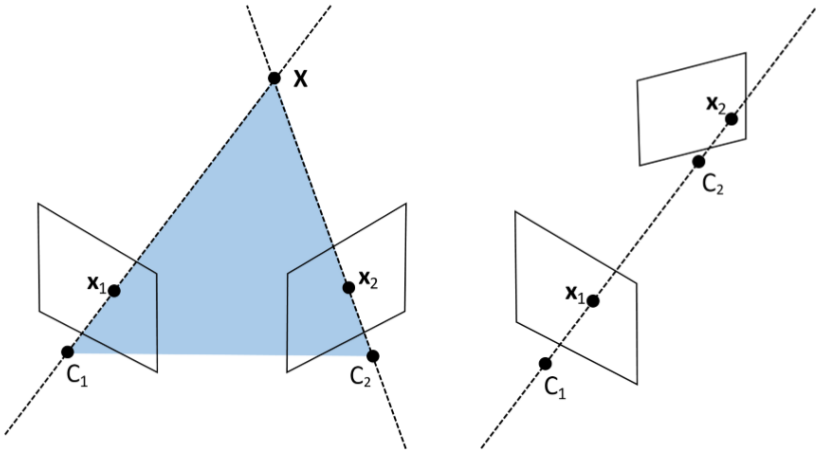


Figure 10: Illustration of triangulation. Left: The sought point X is in the intersection of the two viewing lines. Right: If the camera moves so that the two viewing lines intersect for all points the problem becomes degenerate and we cannot determine X.

7.1 Computer Exercise 5

Using the estimated cameras from Computer Exercise 3, we now triangulate the points detected in Computer Exercise 4.

First task to do here is to set up the **DLT equations** for triangulation, and then solve the **homogeneous least squares** system using SVD and getting v as the last column of V.

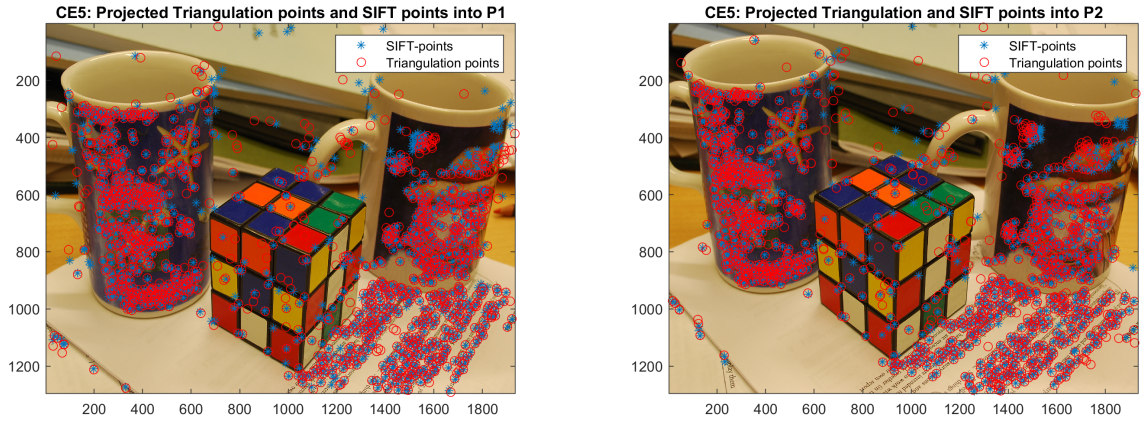
$$\begin{cases} \lambda_1 \mathbf{x}_1 = P_1 \mathbf{X} \\ \lambda_2 \mathbf{x}_2 = P_2 \mathbf{X} \end{cases} \implies \begin{bmatrix} P_1 & -\mathbf{x}_1 & 0 \\ P_2 & 0 & \mathbf{x}_2 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = 0 \tag{58}$$

As shown in Listing 2, we have to do this in a loop, once for each point.

Listing 2: The code for computing the triangulation using DLT.

```
1
2  % Set up the DLT equations for triangulation
3  X = [];
4  for j=1:size(x1,2)
5      M = [P1 -[x1(:,j);1] [0 0 0]';
6           P2 [0 0 0]' -[x2(:,2);1]];
7
8      % and solve the homogeneous least squares system
9      % do this in a loop, once for each point
10     [U,S,V] = svd(M);
11     vstar = V(:,end);
12     X = [X vstar(1:4,:)];
13 end
```

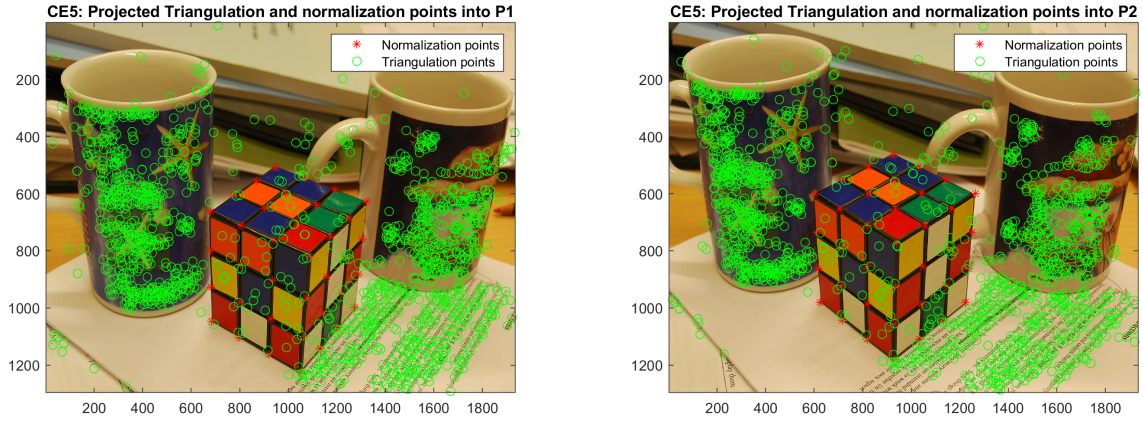
After projecting the computed points into the two images, we can compare them with the corresponding SIFT-points as shown in Figure 11.



(a) (b)

Figure 11: After projecting the points computed from DLT Triangulation, we compare them with the corresponding SIFT-points in the same figure into image cube1.jpg in a) and into image cube2.jpg in b).

Then we compare with the results we get when we normalize with inner parameters of the cameras, that is, using the inverse of K for normalization as illustrated in Figure 12.



(a) (b)

Figure 12: After projecting the points computed from DLT Triangulation, we compare them with the corresponding points from normalization task in Computer Exercise 3 in the same figure into image cube1.jpg in a) and into image cube2.jpg in b).

As we saw in Computer Exercise 4 a portion of the SIFT matches will be incorrect. Most of the time (but not always) this will result in triangulations with large error. In this task we need to compute the errors (both images) between the projected 3D points and the corresponding SIFT points and remove those points for which the error in at least one of the images is larger than 3 pixels. The implemented code is shown in Listing 3.

Listing 3: The code for finding the points with reprojection error less than 3 pixels in both images.

```
1
2 % Finds the points with reprojection error
3 % less than 3 pixels in both images
4 good_points = (sqrt(sum((x1 - x1proj(1:2,:)).^2)) < 3 &...
5                sqrt(sum((x2 - x2proj(1:2,:)).^2)) < 3);
6
7 % Removes points that are not good enough
8 goodX = X(:, good_points);
9 [m5,n5] = size(goodX);
```

```
homgoodX = zeros(m5,n5);
for j = 1 : n5
    homgoodX(:,j) = pflat(goodX(:,j));
end
```

At the end we plot the remaining 3D points, the cameras and the cube model in the same 3D plot in Figure 13. It is clear that we can distinguish the dominant objects, i.e. the cups and the paper.

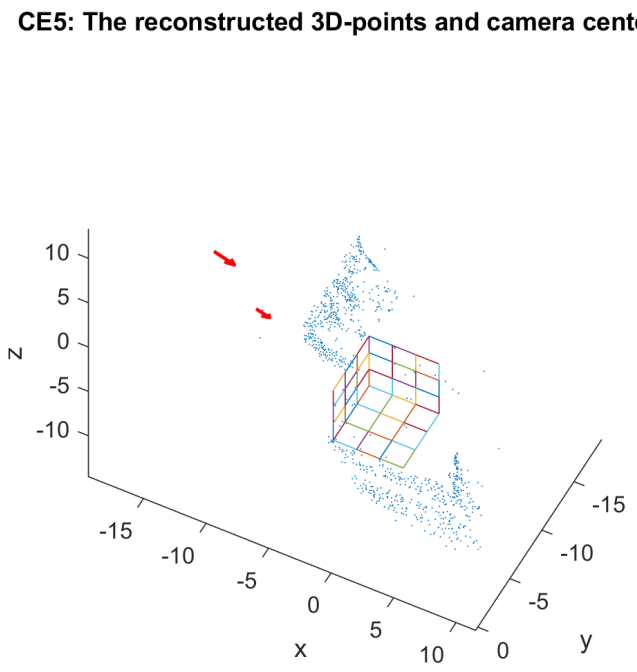


Figure 13: Plot of the remaining 3D points, the cameras and the cube model in the same 3D plot, after Removing those points for which the error in at least one of the images is larger than 3 pixels.

References

[1] Carl Olsson, Computer Vision - FMAN85, Lectures notes: `https://canvas.education.lu.se/courses/3379`

[2] Hartley, Zisserman, Multiple View Geometry, 2004.

[3] Szeliski, Computer Vision - Algorithms and Applications, Springer.