Computer Vision - FMAN85
Assignment 1 - Spring 2020

Elements of Projective Geometry

Hicham Mohamad, hi8826mo-s
hsmo@kth.se

January 21, 2020

1 Introduction

In this assignment, we are going to exercise and get familiar with the basic elements of **projective geometry**. We are going to study the representations of points, lines and planes, as well as transformations and camera matrices.

2 Points in Homogeneous Coordinates

In general the homogeneous coordinates are the vectors of $\mathbb{R}^{n+1}$ to represent the **projective space** of n dimensions $\mathbb{P}^n$, and if the coordinate $n+1$ is not zero then we can interpret them as points of $\mathbb{R}^n$ by dividing with this coordinate.

If the third coordinate is zero we cannot divide by it and consequently there are elements in $\mathbb{P}^n$ that cannot be interpreted as points in $\mathbb{R}^n$. We call this type of point a **vanishing point** or a point at infinity.

2.1 Exercise 1

In this task, we need to determine the 2D cartesian coordinates of the points with the following triple homogeneous coordinates

$$\mathbf{x}_1 = \begin{pmatrix} 4 \\ -2 \\ 2 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 3 \\ -2 \\ -1 \end{pmatrix}, \quad \mathbf{x}_3 = \begin{pmatrix} 4\lambda \\ -2\lambda \\ 2\lambda \end{pmatrix}, \quad \lambda \neq 0 \tag{1}$$

The three **2D points** can be computed by dividing the given homogeneous coordinates by their third coordinates respectively. In this way, we get the respective points in $\mathbb{R}^2$

$$\mathbf{x}_1 = (2, -1), \quad \mathbf{x}_2 = (-3, 2), \quad \mathbf{x}_3 = (2, -1). \tag{2}$$

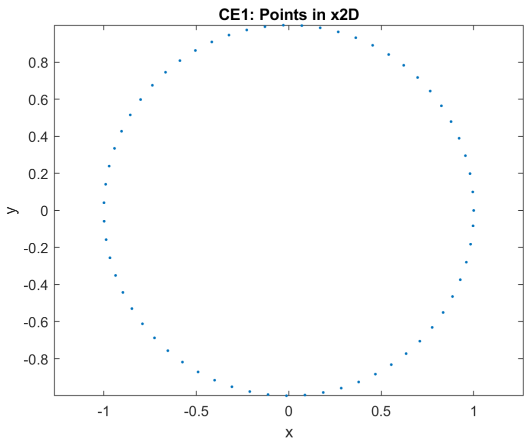
In the second part of this exercise, we need to give what is the interpretation of the point with homogeneous coordinates

$$\mathbf{x}_4 = \begin{pmatrix} 4 \\ -2 \\ 0 \end{pmatrix} \tag{3}$$

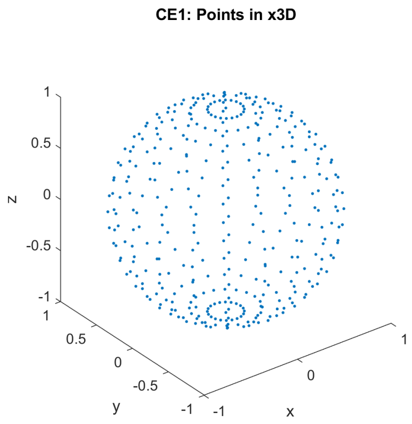
To interpret $\mathbf{x}_4$ geometrically we look at $(4, -2, \epsilon)$, where ϵ is a small positive number. Now, this point has non-zero third coordinate and is equivalent to $(\frac{4}{\epsilon}, \frac{-2}{\epsilon}, 1)$. In this way, we get a point with very large x- and y-coordinate. Making ϵ smaller we see that $\mathbf{x}_4 = (4, -2, 0)^T$ can be interpreted as a point infinitely far away in the direction $(4, -2)$, i.e. vanishing point.

2.2 Computer Exercise 1

After applying the implemented function `pflat()` to the points in x2D and x3D, we obtain the resulting plots shown in Figure 1



(a)



(b)

Figure 1: a) Plot illustrating the points in x2D after applying the implemented function pflat(). b) Plot illustrating the points in x3D.

3 Lines

From linear algebra we know that a line in $\mathbb{R}^2$ can be represented by the equation

$$ax + by + c = 0 \quad (a, b, c) \neq (0, 0, 0) \tag{4}$$

For a point $\mathbf{x} \sim (x, y, z)$ in $\mathbb{P}^2$ we instead use the modified formula

$$ax + by + cz = 0 \tag{5}$$

If x is a **regular point** with Cartesian coordinates (x, y), i.e. $\mathbf{x} \sim (x, y, 1)$ then the two equations in 4 and 5 give the same result.

3.1 Exercise 2

a) First we need to compute the homogeneous coordinates of the intersection in $\mathbb{P}^2$ of the two lines l_1 and l_2 and the corresponding point in $\mathbb{R}^2$.

$$l_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad l_2 = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}, \tag{6}$$

If x lies on both the lines so it must be a solution of the following system of equations

$$\begin{cases} l_1^T x = 0 \\ l_2^T x = 0 \end{cases} \implies \begin{cases} x + y + z = 0 \\ 3x + 2y + z = 0 \end{cases} \implies \begin{cases} x + y + z = 0 \\ -2y - 2z = 0 \end{cases} \tag{7}$$

Since we have no constraints on z we get

$$\begin{cases} x + y + z = 0 \\ -y - 2z = 0 \\ z = t \end{cases} \implies \begin{cases} x = t \\ y = -2t \\ z = t \end{cases} \tag{8}$$

Hence the homogeneous coordinates for $\mathbf{x} \in \mathbb{P}^2$ can be $(1, -2, 1)$. By dividing with the third coordinate we can interpret this element as a point in $\mathbb{R}^2$, namely $(1, -2)$.

b) Here we compute the intersection in $\mathbb{P}^2$ of lines l_3 and l_4

$$l_3 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \quad l_4 = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \tag{9}$$

$$\begin{cases} l_3^T x = 0 \\ l_4^T x = 0 \end{cases} \implies \begin{cases} x + 2y + 3z = 0 \\ x + 2y + z = 0 \end{cases} \implies \begin{cases} x + 2y + 3z = 0 \\ 2z = 0 \end{cases} \tag{10}$$

Since we have no constraints on y we get

$$\begin{cases} x + 2y + 3z = 0 \\ 2z = 0 \\ y = t \end{cases} \implies \begin{cases} x = -2t \\ y = t \\ z = 0 \end{cases} \tag{11}$$

Hence the representative of our intersection point $\mathbf{x} \in \mathbb{P}^2$ can be $(-2, 1, 0)$. In this case, we cannot interpret the result by dividing with the third coordinate because it is zero. This means that the lines are parallel and therefore do not intersect in $\mathbb{R}^2$. This type of point is called a vanishing point or a point at intinity, and the line $z = 0$ is called the vanishing line.

c) In this part we compute the line $l \sim (a, b, c)$ that passes through the points with Cartesian coordinates x_1 and x_2

$$x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \tag{12}$$

These points can be represented respectively by $x_1 \sim (1, 1, 1)$ and $x_2 \sim (3, 2, 1)$

$$\begin{cases} l^T x_1 = 0 \\ l^T x_2 = 0 \end{cases} \implies \begin{cases} ax_1 + by_1 + c = 0 \\ ax_2 + by_2 + c = 0 \end{cases} \implies \begin{cases} a + b + c = 0 \\ 3a + 2b + c = 0 \end{cases} \tag{13}$$

Since we have no constraints on c we get

$$\begin{cases} a + b + c = 0 \\ -b - 2c = 0 \\ c = t \end{cases} \implies \begin{cases} a = t \\ b = -2t \\ c = t \end{cases} \tag{14}$$

Hence the line $l \sim (a, b, c)$ can be $(1, -2, 1)$.

3.2 Exercise 3

From exercise 2, we had that $\mathbf{x} \sim (1, -2, 1)$ represents the intersection point in homogeneous coordinates of l_1 and l_2. Actually this point is in the null space of the matrix

$$M = \begin{pmatrix} 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{15}$$

where the null space of an $m \times n$ matrix A is the set

$$\mathcal{N}(A) = \{x \in \mathbb{R}^n ; Ax = 0\} \tag{16}$$

In fact, $M\mathbf{x} = 0$ can be seen as the scalar product of the vectors $(x, y, 1)$ and (a, b, c) of the two lines l_1 and l_2. Thus, we can write this in matrix form and get

$$\begin{cases} l_1^T \mathbf{x} = 0 \\ l_2^T \mathbf{x} = 0 \end{cases} \implies \begin{pmatrix} 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \implies M\mathbf{x} = 0 \tag{17}$$

If we use $(\lambda, -2\lambda, \lambda)$ with $\lambda \neq 0$ to represent $\mathbf{x}$ instead of $(1, -2, 1)$ we get other points in the null space.

3.3 Computer Exercise 2

In the file compEx2.mat there are three pairs of image points. As illustrated in Figure 2, we plot the image points in the same figure as the image.

For each pair of points, we can compute the line going through the points by using the Matlab function `null()`. Then we use the function `rital()` to plot the lines in the same image. These lines don't appear to be parallel in 3D !

Here we need to compute the point of intersection between the second and third line (the lines obtained from the pairs p2 and p3). This is obtained by using the Matlab function `null([l2,l3]')`

$$x23 = \begin{pmatrix} 0.0260 \\ 0.9997 \\ 0.0009 \end{pmatrix}$$

To plot this point in the same image, we divide by the third coordinate to get the 2D Cartesian coordinates

$$\begin{pmatrix} 028.3 \\ 1089.8 \\ 1.0 \end{pmatrix}$$

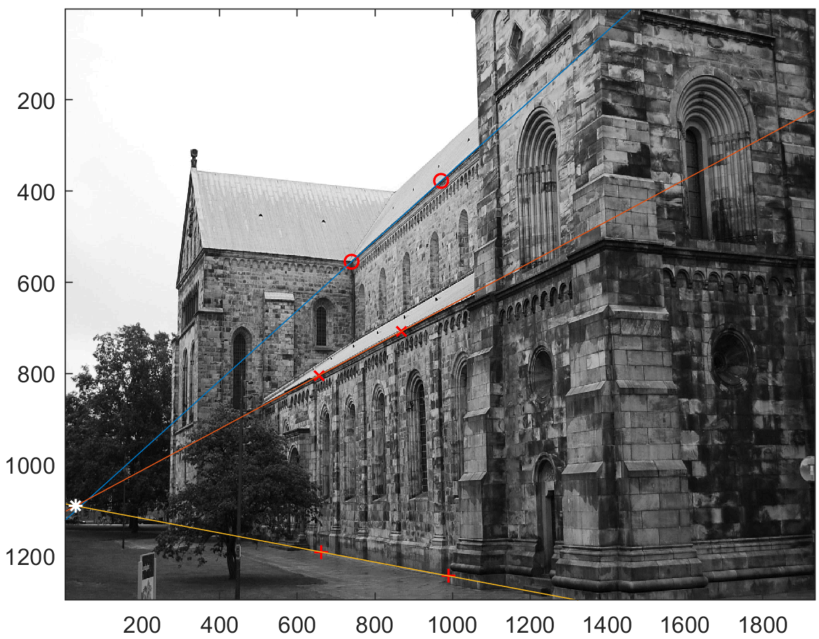


Figure 2: Plot of the image in Computer Exercise 2. There we can see the point of intersection between the second and third line (the lines obtained from the pairs p2 and p3) in the same image.

The distance between a 2D-point $x = (x_1, x_2)$ in Cartesian coordinates and a line $l = (a, b, c)$ can be computed using the distance formula

$$d = \frac{|ax_1 + bx_2 + c|}{\sqrt{a^2 + b^2}} \tag{18}$$

By using this formula we compute the distance between the first line and the intersection point and obtain

$$d = 8.2695$$

it is clear that the point is relatively close to zero, when the dimension of the image is almost 2000×1300.

4 Projective Transformations

A projective transformation is an invertible mapping $\mathbb{P}^n \to \mathbb{P}^n$ defined by

$$\mathbf{x} \sim H\mathbf{y} \tag{19}$$

where $x \in \mathbb{R}^{n+1}$ and $y \in \mathbb{R}^{n+1}$ are homogeneous coordinates representing elements of $\mathbb{P}^n$ and H is an invertible $(n+1) \times (n+1)$ matrix. Projective transformations are also often called **homographies**.

4.1 Exercise 4

In this exercise we consider the projective transformation H

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \tag{20}$$

a) Here we need to compute the transformations $y_1 \sim Hx_1$ and $y_2 \sim Hx_2$ where

$$x_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \tag{21}$$

By multiplying by H we get

$$y_1 \sim \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad y_2 \sim \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{22}$$

b) In this part we compute the lines $l_1 \sim (a_1, b_1, c_1)$ and $l_2 \sim (a_2, b_2, c_2)$ that pass through the points with homogeneous coordinates x_1, x_2 and y_1, y_2 respectively.

$$\begin{cases} l_1^T \mathbf{x}_1 = 0 \\ l_1^T \mathbf{x}_2 = 0 \end{cases} \implies \begin{cases} a_1 x_1 + b_1 y_1 + c_1 z_1 = 0 \\ a_1 x_2 + b_1 y_2 + c_1 z_2 = 0 \end{cases} \implies \begin{cases} a_1 + c_1 = 0 \\ b_1 + c_1 = 0 \end{cases} \tag{23}$$

Since we have no constraints on c_1 we get

$$\begin{cases} a_1 + c_1 = 0 \\ b_1 + c_1 = 0 \\ c_1 = t \end{cases} \implies \begin{cases} a_1 = -t \\ b_1 = -t \\ c_1 = t \end{cases} \tag{24}$$

The same we do for computing the line l_2

$$\begin{cases} l_2^T \mathbf{y}_1 = 0 \\ l_2^T \mathbf{y}_2 = 0 \end{cases} \implies \begin{cases} a_2 x_1 + b_2 y_1 + c_2 z_1 = 0 \\ a_2 x_2 + b_2 y_2 + c_2 z_2 = 0 \end{cases} \implies \begin{cases} a_2 = 0 \\ a_2 + b_2 + c_2 = 0 \end{cases} \tag{25}$$

Since we have no constraints on c_2 we get

$$\begin{cases} a_2 = 0 \\ b_2 + c_2 = 0 \\ c_2 = t \end{cases} \implies \begin{cases} a_2 = 0 \\ b_2 = -t \\ c_2 = t \end{cases} \tag{26}$$

Hence these lines can be represented respectively by $l_1 \sim (-1, -1, 1)$ and $l_2 \sim (0, -1, 1)$

c) By compute $(H^{-1})^T l_1$ we notice that

$$(H^{-1})^T l_1 = \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} = l_2 \tag{27}$$

d) In this last part of the exercise we show that **projective transformations preserve lines**. That is, for each line l$_1$ there is a corresponding line l_2 such that if $\mathbf{x}$ belongs to l_1 then the transformation $\mathbf{y} \sim H\mathbf{x}$ belongs to l_2, i.e. we want to show that

$$l_2^T \mathbf{y} = 0 \implies ((H^{-1})^T l_1)^T \mathbf{y} = l_1^T H^{-1} \mathbf{y} = 0 \tag{28}$$

Using the given hint we can write

$$l_1^T \mathbf{x} = 0 \implies l_1^T H^{-1} H\mathbf{x} \sim l_1^T H^{-1} \mathbf{y} = 0 \tag{29}$$

4.2 Computer Exercise 3

The `filecompEx3.mat` contains the start and end points of a set of lines. Plotting the lines gives the grid in Figure 3

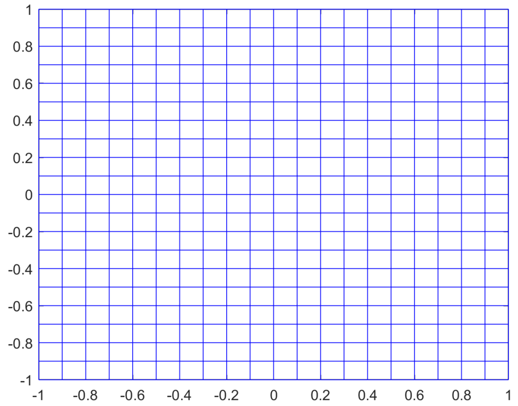


Figure 3: The lines between start and end points in Computer Exercise 3.

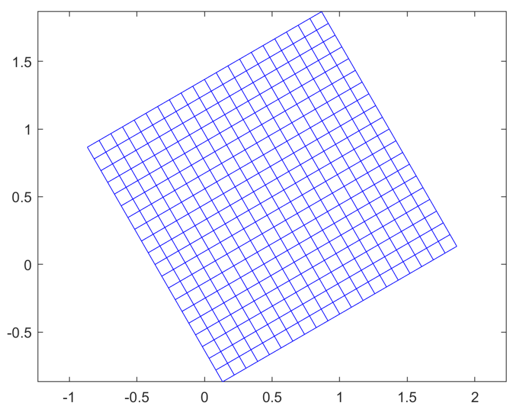
For each of the projective mappings given by the matrices

$$H_1 = \begin{pmatrix} \sqrt{3} & -1 & 1 \\ 1 & \sqrt{3} & 1 \\ 0 & 0 & 2 \end{pmatrix}, \quad H_2 = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{30}$$

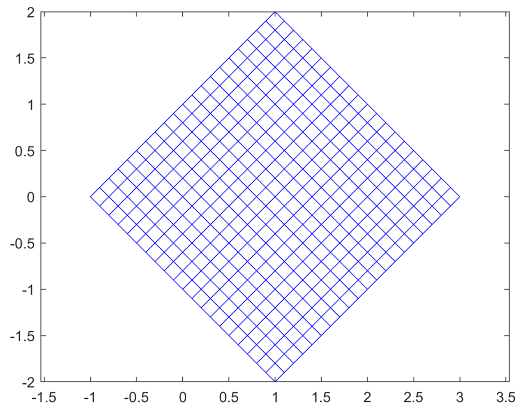
$$H_3 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad H_4 = \begin{pmatrix} \sqrt{3} & -1 & 1 \\ 1 & \sqrt{3} & 1 \\ 1/4 & 1/2 & 2 \end{pmatrix} \tag{31}$$

a) Plotting the lines

As shown in Listing 1, we compute the transformations of the given start and endpoints and plot the lines between them. Then we compute Cartesian coordinates by using the implemented `pflat()` function. The resulting plots are illustrated in Figures 4 and 5.

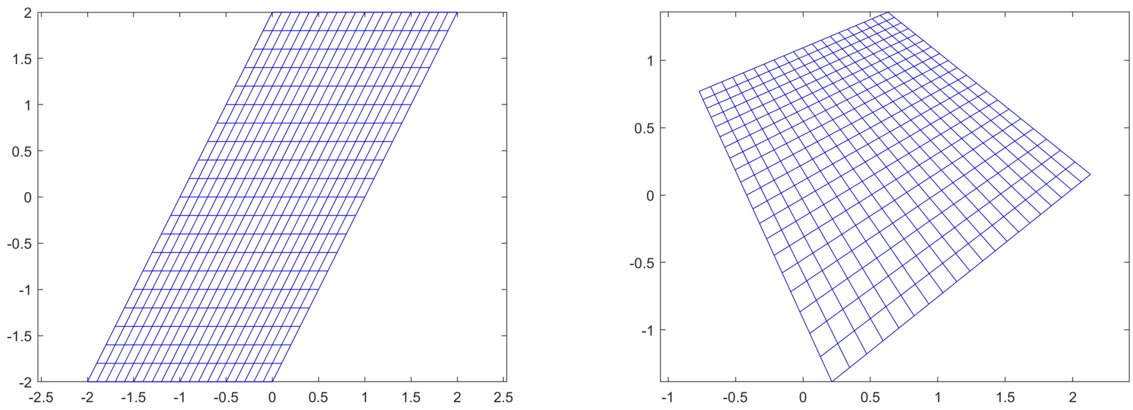


(a)



(b)

*Figure 4: Plot illustrating the lines after computing the transformations of the given start and endpoints with projecting mappings H_1 and H_2. a) **Euclidean**: the separation of two points is unchanged by a Euclidean transformation, translation and rotation. b)**Similarity**: A square tile is imaged as a square. Lines which are parallel or perpendicular have the same relative orientation in the image.*



(a) (b)

Figure 5: Plots illustrating the lines after computing the transformations of the given start and endpoints with projecting mappings H_3 and H_4 respectively. a) ***Affine****: The square is imaged as an parallelogram. Orthogonal world lines are not imaged as orthogonal lines. However, the sides of the square tiles, which are parallel in the world are parallel in the image. b)* ***Projective****: Parallel world lines are imaged as converging lines. Tiles closer to the camera have a larger image than those further away.*

b) Questions: Properties and Classification

In this section we discuss the important specializations of a projective transformation and their **geometric properties**, and then classify the four transformations into euclidean, similarity, affine and projective transformations.

An alternative to describe the transformation algebraically, i.e. as a matrix acting on coordinates of a point, is to describe the transformation in terms of those elements or quantities that are preserved or **invariant**. In this way, we remember that, besides being projective:

- The affine transformation has the special property that parallel lines are mapped to parallel lines.
- The distance between two points is an euclidean, but not similarity invariant.
- The angle between two lines is both a euclidean and a similarity invariant.

Using these properties and looking at the different plots in Figures 4 and 5 we get the resulting classification in Table 1.

Transf. Matrix	Invariant properties	classification
H_1	Length, angles, parallel lines	Euclidean
H_2	Angles, parallel lines	Similarity
H_3	Parallel lines	Affine
H_4	-	Projective

Table 1: Classification and geometric properties invariant to the four transformations.

Listing 1: The code for computing the transformations with matrix H_1.

```

% compute the transformations of the given start and endpoints
startTrans1 = H1 * [startpoints; ones(1, n)];
endTrans1 = H1 * [endpoints; ones(1,n)];

% compute cartesian coordinates by using pflat function
cartStartTrans1 = zeros(m+1,n);
cartEndTrans1 = zeros(m+1,n);
for i = 1 : n
    cartStartTrans1(:,i) = pflat(startTrans1(:,i));
```

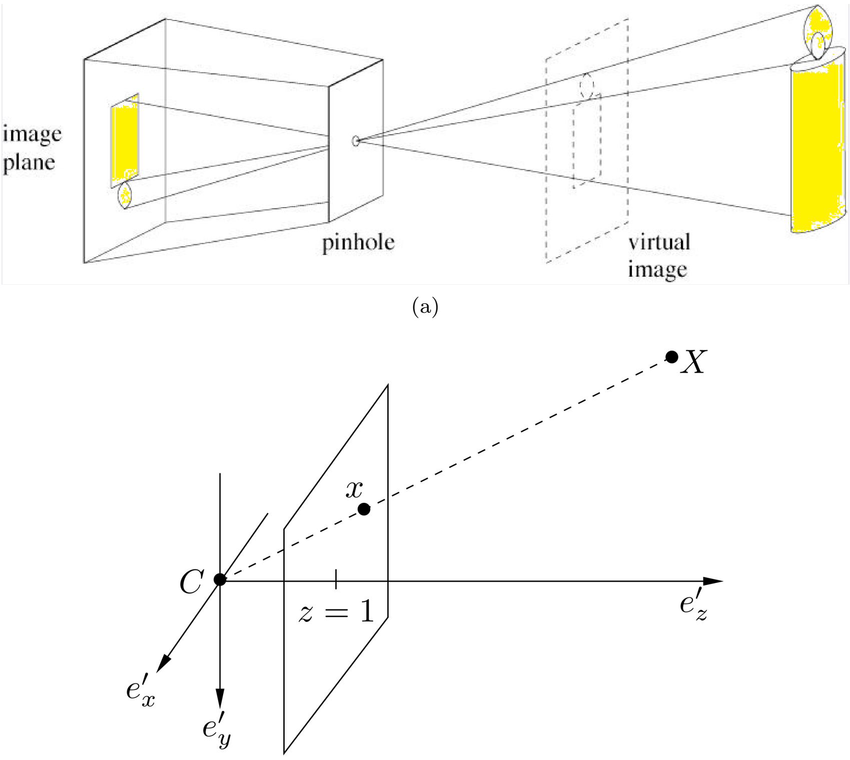
```
    cartEndTrans1(:,i) = pflat(endTrans1(:,i));
end

%plot the lines between start and endpoints
figure
plot([cartStartTrans1(1,:); cartEndTrans1(1,:)], ...
     [cartStartTrans1(2,:); cartEndTrans1(2,:)],'b-');

% use the axis equal command, otherwisethe figures might look distorted
axis equal
```

5 The Pinhole Camera

The most commonly used model, which we will also use in this assignment, is the so called pinhole camera. The model is inspired by the simplest cameras. It has the shape of a box, light from an object enters though a small hole (the pinhole) in the front and produces an image on the back camera wall, as illustrated in Figure 6a.



(a)

(b)

Figure 6: (a) The real Pinhole camera. (b) The camera mathematical model: Note that in contrast to a real pinhole camera in (a) we have placed the image plane in front of the camera center. This has the effect that the image will not appear upside down as in the real model.

5.1 Camera Mathematical Model

To create a *mathematical model* we first select a coordinate system $\{e'_x, e'_y, e'_z\}$, as shown in Figure 6. We will refer to this system as the **camera coordinate system**. The origin $C = (0, 0, 0)$ will represent the so called **camera center** (pinhole).

To generate a *projection* $x = (x_1, x_2, 1)$ of a *scene point* $X = (X'_1, X'_2, X'_3)$ we form the line between X and C and intersect it with the plane $z = 1$. We will refer to this plane as the **image plane** and the line as the **viewing ray** associated with x or X. The plane $z = 1$ has the normal e_z and lies at the distance 1 from the camera center. We will refer to e_z as the **viewing direction**.

Note that in contrast to a real pinhole camera shown in Figure 6a, we have placed the image plane in *front of the camera center.* This has the effect that the image will not appear upside down as in the real model.

5.2 Camera Equation

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

$$\lambda \mathbf{x} = P\mathbf{X} \tag{32}$$

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 plane. 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.*

5.3 Exercise 5

To compute the projections of the 3D points with homogeneous coordinates in the camera with camera matrix P we need to use the camera equations.

$$\mathbf{X}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \end{pmatrix}, \quad \mathbf{X}_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{X}_3 = \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix}, \tag{33}$$

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{34}$$

The interpretation of this equation is that the projection (x_1, x_2) of the scene point with coordinates (X_1, X_2, X_3) can be found by

- first computing $v = P\mathbf{X}$
- then dividing v by its third coordinate.

In this way, we can find the projections

$$\mathbf{x}_1 = \begin{pmatrix} 1/4 \\ 1/2 \\ 1 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 1/2 \\ 1/2 \\ 1 \end{pmatrix}, \quad \mathbf{x}_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \tag{35}$$

We notice that the projection of $\mathbf{X}_3$ has non-zero third coordinate. Thus, $\mathbf{x}_3 = (1, 1, 0)$ can be interpreted as a point infinitely far away in the direction (1,1). This type of point is called a **vanishing point** or a point at infinity.

For computing the camera center C (position) of the camera we use the Matlab function `null()` to solve the equation that fulfils

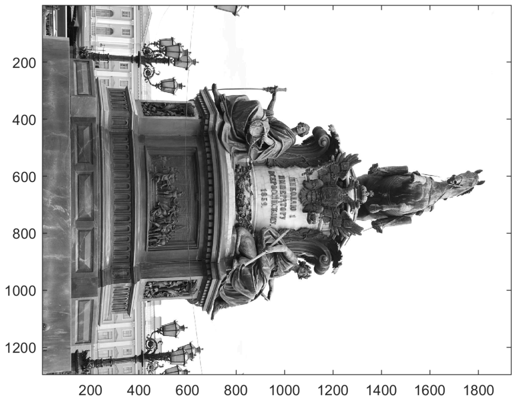
$$0 = PC \implies \mathbf{C} = \begin{pmatrix} 0 \\ 0 \\ -0.7071 \\ 0.7071 \end{pmatrix} \implies c = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{36}$$

For computing the principal axis V (viewing direction) we need to extract the elements P_{31}, P_{32} and P_{33} from the camera matrix P which correspond to the row R_3 in the rotation matrix R where $P = K[R \quad t]$.

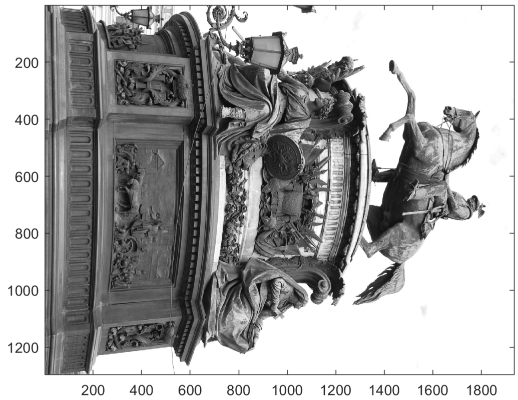
$$V = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

5.4 Computer Exercise 4

In this task, we load and plot the images `compEx4im1.jpg` and `compEx4im2.jpg` as shown in Figure 7. The `filecompEx4.mat` contains the camera matrices P1, P2 and a point model U of the statue.



(a)



(b)

Figure 7: plot of the images compEx4im1.jpg and compEx4im2.jpg

camera centers and principal axes of the cameras

For computing the camera centers and principal axes of the cameras, we use the Matlab function `null()` to solve the equation that fulfils

$$0 = P1 \cdot C1 \implies \mathbf{C1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \implies c_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{37}$$

$$0 = P2 \cdot C2 \implies \mathbf{C2} = \begin{pmatrix} 0.2990 \\ 0.6690 \\ -0.6790 \\ 0.0451 \end{pmatrix} \implies c_2 = \begin{pmatrix} 6.6352 \\ 14.8460 \\ -15.0691 \\ 1 \end{pmatrix} \tag{38}$$

For computing the principal axes V_1 and V_2 (viewing direction) we need to extract the elements P_{31}, P_{32} and P_{33} from the camera matrices $P1$ and $P2$ which correspond to the row R_3 in the rotation matrix R where $P = K[R \quad t]$. Then the resulting principal axes are normalized to length one by dividing by their norm().

$$V1 = \begin{pmatrix} 0.3129 \\ 0.9461 \\ 0.0837 \end{pmatrix} \quad V2 = \begin{pmatrix} 0.0319 \\ 0.3402 \\ 0.9398 \end{pmatrix}$$

Plot of 3D-points in U and the camera centers

By using `pflat()` we make sure that the 4th coordinate of U is one, then we plot the 3D-points in U and the camera centers in the same 3D plot, as illustrated in Figure 8. In addition, we plot a vector in the direction of the principal axes (viewing direction) from the camera center.

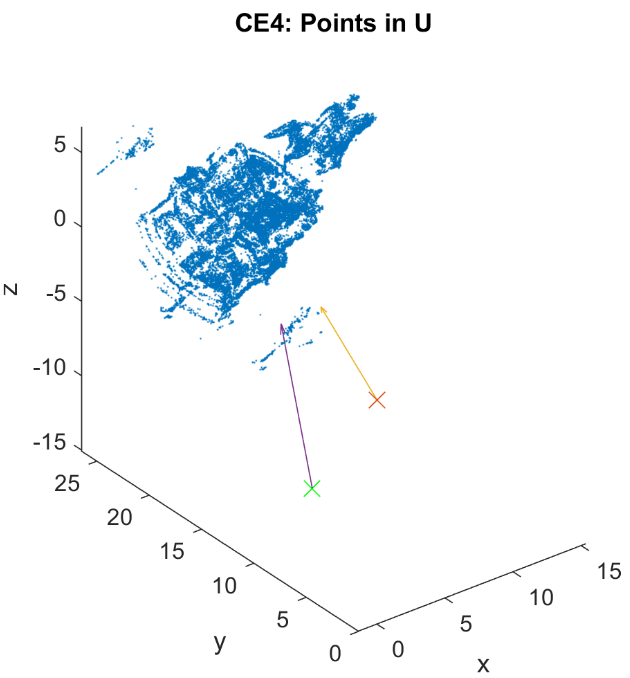
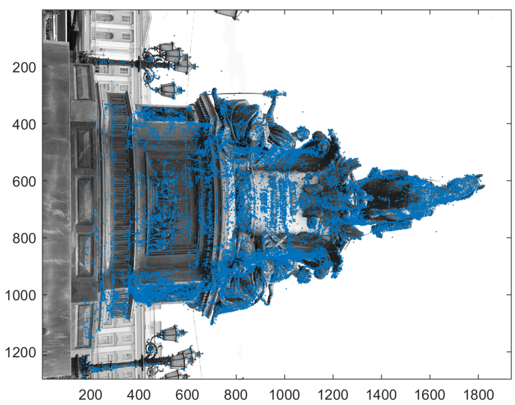


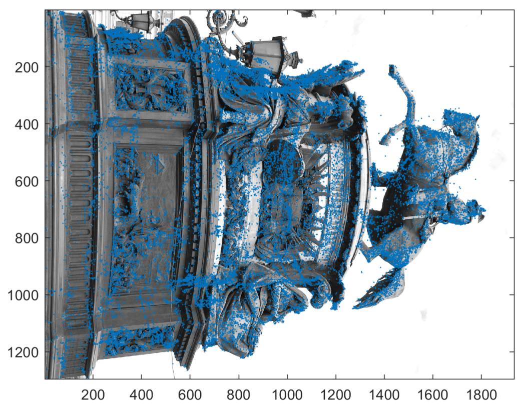
Figure 8: Plot of the 3D-points in U and the camera centers in the same 3D plot. In addition we notice in the plot a vector in the direction of the principal axes (viewing direction) from the camera center.

Projecting the points in U into the cameras P1 and P2

After projecting the points in U into the cameras P1 and P2 and plot the result in the same plots as the images, we notice that the result look reasonable, as shown in Figure 9.



(a)



(b)

Figure 9: After projecting the points in U into the cameras P1 and P2 and plot the result in the same plots as the images compEx4im1.jpg and compEx4im2.jpg

References

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

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