Lecture 1: The Pinhole Camera Model

1 Mathematical Model

The most commonly used model, which we will also use in the course, is the so called pinhole camera. The model is inspired by the simplest cameras. The camera has the shape of a box, light from an object enters through a small hole (the pinhole) in the front and produces an image on the back camera wall (see Figure 1).

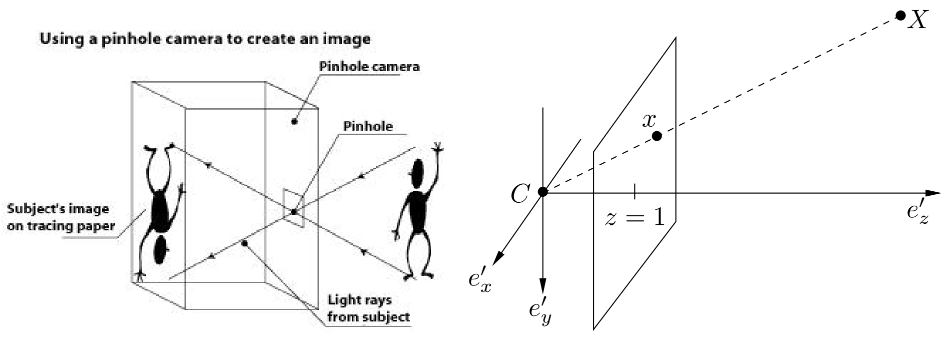


Figure 1: The Pinhole camera (left), and a mathematical model (right).

To create a mathematical model we first select a coordinate system $\{e'_x, e'_y, e'_z\}$.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 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.

Since $X - C$ is a direction vector of the viewing ray (see Figure 2) we can parametrize it by the expression

$$C + s(X - C) = sX, \quad s \in \mathbb{R}. \tag{1}$$

To find the intersection between this line and the image plane $z = 1$ we need to find an s such that the third coordinate sX'_3 of sX fulfills $sX'_3 = 1$. Therefore, assuming $X'_3 \neq 0$, we get $s = 1/X'_3$ and the projection

$$x = \begin{pmatrix} X'_1/X'_3 \\ X'_2/X'_3 \\ 1 \end{pmatrix} \tag{2}$$

Exercise 1. Compute the image of the cube with corners in $(\pm1, \pm1, 2)$ and $(\pm1, \pm1, 4)$ (see Figure 3).

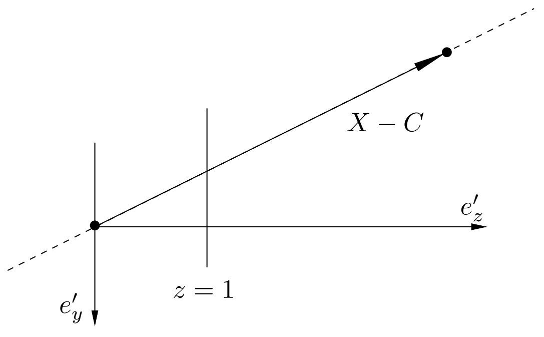


Figure 2: The model viewed from the side. (The vector e'_x points out of the figure.)

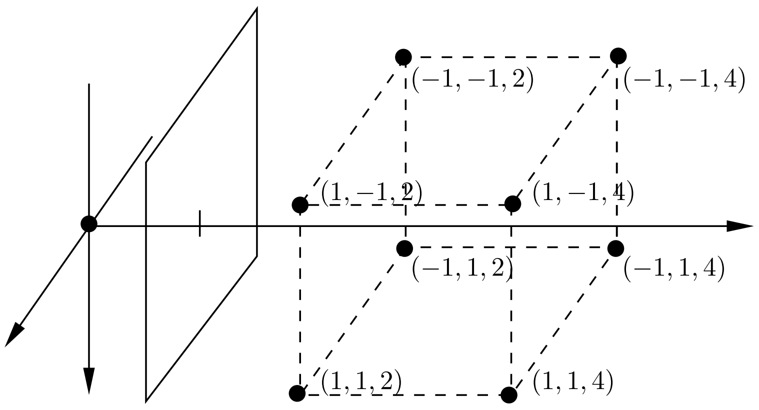


Figure 3: The Cube in the camera coordinate system.

2 Moving Cameras

In our applications we will frequently have cameras that have been capturing images from different viewpoints. Therefore we need to have a way of modeling camera movements. A camera can undergo translation and rotation. We will represent the translation with a vector $t \in \mathbb{R}^3$ and the rotation with a 3×3 matrix R. Since R is a rotation matrix it has to fulfill $R^T R = I$ and $\det(R) = 1$.

To encode camera movements we introduce a new reference coordinate system $\{e_x, e_y, e_z\}$, see Figure 4. We will refer to this coordinate system as the global coordinate system, since all the camera movements will be related to this system. Typically all the scene point coordinates are also specified in this coordinate system. Let's assume that a scene point X has coordinates (X'_1, X'_2, X'_3) in the camera coordinate system and (X_1, X_2, X_3) in the global coordinate system. Since the camera can be rotated and translated there is a rotation matrix R and translation vector t that relates the two coordinate systems via

$$\begin{pmatrix} X'_1 \\ X'_2 \\ X'_3 \end{pmatrix} = R \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} + t. \tag{3}$$

Exercise 2. What is the position of the camera (the coordinates of the camera center) in the global coordinate system? What is the viewing direction in the global coordinate system?

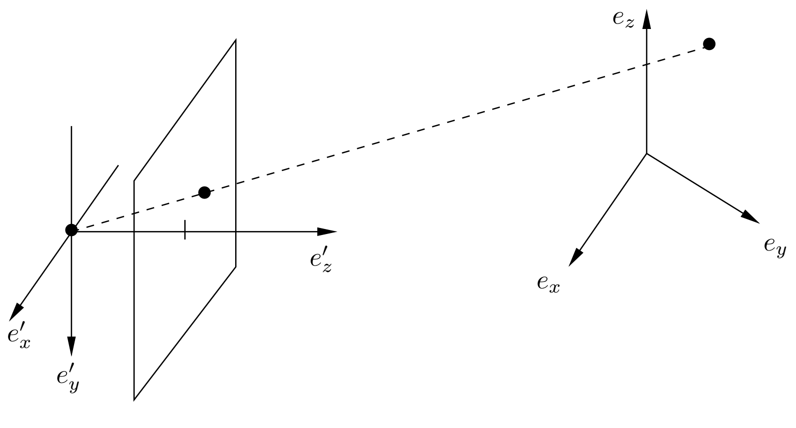


Figure 4: New global coordinate system.

If we add an extra 1 to the scene point X we can write (3) in matrix form

$$X'_3 \begin{pmatrix} X'_1/X'_3 \\ X'_2/X'_3 \\ 1 \end{pmatrix} = \begin{pmatrix} X'_1 \\ X'_2 \\ X'_3 \end{pmatrix} = [R \quad t] \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}. \tag{4}$$

Here $[R \quad t]$ is the 3×4 matrix where the first 3×3 block is R and the last column is t. As we saw previously, the projection of X is given by (2). Therefore we conclude from (4) that the projection x of a scene point X (with coordinates given in the global coordinate system) is obtained by computing the vector $v = [R \quad t] \begin{bmatrix} X \\ 1 \end{bmatrix}$ and dividing the elements of v by the third coordinate of v. From here on we will always assume that scene point coordinates are given in the global coordinate system, if nothing else is stated.

Exercise 3. Compute the projection of $X = (0, 0, 1)$ in the cameras

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & \sqrt{2} & 0 & 0 \\ 1 & 0 & 1 & \sqrt{2} \end{pmatrix}. \tag{5}$$

3 Depth of a Point

We say that a scene point is in front of the camera (or has positive depth) if its third coordinate is positive in the camera coordinate system. According to (4) the third coordinate is given by

$$X'_3 = [R_3 \quad t_3] \begin{bmatrix} X \\ 1 \end{bmatrix}, \tag{6}$$

where R_3 is the third row R and t_3 is the third coordinate of t. To determine whether a point is in front of the camera we therefore compute $v = [R \quad t] \begin{bmatrix} X \\ 1 \end{bmatrix}$ and check if the third coordinate is positive.

4 The Inner Parameters

In the pinhole camera model the image plane is embedded in $\mathbb{R}^3$. That is, image projections are given in the length unit of $\mathbb{R}^3$ (e.g. meters). Furthermore, the center of the image will be located in $(0, 0, 1)$, and will therefore have image coordinate $(0,0)$. For real cameras we typically obtain images where the coordinates are measured in pixels with $(0,0)$ in the upper left corner. To be able to do geometrically meaningful computations we need to translate pixel coordinates into the length unit of $\mathbb{R}^3$. We do this by adding a mapping from the image plane embedded in $\mathbb{R}^3$ to the real image, see Figure 5.

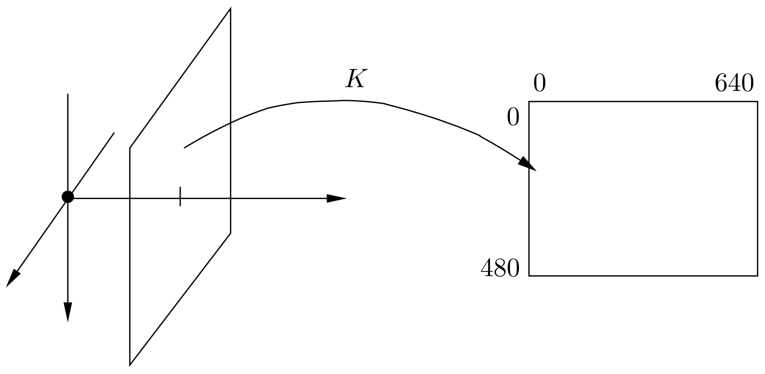


Figure 5: The mapping K from the image plane to the real image.

The mapping is represented by an invertible triangular 3×3 matrix K. This matrix contains what is usually referred to as the inner parameters of the camera, that is, focal length, principal point etc. (We will discuss this more in Lecture 3.) The projection (reference coordinate system to real image) is now given by

$$\lambda \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}}_{=\mathbf{x}} = \underbrace{K \left[R \quad t \right]}_{=P} \underbrace{\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}}_{=\mathbf{X}}, \tag{7}$$

or in matrix form

$$\lambda \mathbf{x} = P\mathbf{X} \tag{8}$$

Equation (8) is usually called the camera equations and the matrix P is called the camera matrix.

Exercise 4. What is the position (camera center) and viewing direction of the camera $P = K \left[R \quad t \right]$?

Lecture 2: Homogeneous Coordinates, Lines and Conics

1 Homogeneous Coordinates

In Lecture 1 we derived 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 position and orientation of the camera and K contains the inner parameters.

The interpretation of these 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 $v = P\mathbf{X}$ and then dividing v by its third coordinate. There are several vectors v that give the same projection. For example, $v = (3, 2, 1)$ gives the projection $(3, 2)$ and $v = (6, 4, 2)$ gives $\left(\frac{6}{2}, \frac{4}{2}\right) = (3, 2)$.

Formally, we will say that two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ are equivalent if there is a number $\lambda \neq 0$ such that $\mathbf{x} = \lambda\mathbf{y}$, and write

$$\mathbf{x} \sim \mathbf{y}. \tag{2}$$

The two vectors are said to represent the same element of the so called two dimensional projective space $\mathbb{P}^2$. The space consists of all the elements that can be represented by vectors in $\mathbb{R}^3$ (with the exception of the zero vector). For example, the two vectors $(6, 9, 3)$ and $(4, 6, 2)$ are equivalent and therefore both represent the same element of $\mathbb{P}^2$. Furthermore, by dividing with the third coordinate we can interpret this element as a point in $\mathbb{R}^2$, namely $(2, 3)$. The vectors $(6, 9, 3)$ and $(4, 6, 2)$ are called homogeneous coordinates of $(2, 3)$.

There are elements in $\mathbb{P}^2$ that can not be interpreted as points in $\mathbb{R}^2$. Specifically, if the third coordinate is zero we can not divide by it. We will soon see that these points also have a simple geometric interpretation.

The projective space of n dimensions $\mathbb{P}^n$ is defined similarly as $\mathbb{P}^2$. In general the homogeneous coordinates for representing $\mathbb{P}^n$ are the vectors of $\mathbb{R}^{n+1}$ (with the exception of the vector with all coordinates equal to zero), and if coordinate $n + 1$ is not zero then we can interpret them as points of $\mathbb{R}^n$ by dividing with this coordinate.

2 Lines and Points in $\mathbb{P}^2$

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

$$ax + by + c = 0, \tag{3}$$

where $(a, b, c) \neq (0, 0, 0)$. If we think of $\mathbf{x} \sim (x, y, 1)$ as a point in $\mathbb{P}^2$ then $\mathbf{x}$ belongs to the line $\mathbf{l} = (a, b, c)$ if (3) holds. Note that (3) can be seen as the scalar product of the vectors $(x, y, 1)$ and (a, b, c). If we use $(\lambda x, \lambda y, \lambda)$ to represent $\mathbf{x}$ instead of $(x, y, 1)$ we get the scalar product

$$a\lambda x + b\lambda y + c\lambda = \lambda(ax + by + c) = 0. \tag{4}$$

Therefore we see that it does not matter which representative we use for $\mathbf{x}$. They all fulfill (4). Similarly if $c \neq 0$ we can represent $\mathbf{l}$ with $\left(\frac{a}{c}, \frac{b}{c}, 1\right)$ instead of (a, b, c).

Note that lines and points behave in the same way here. They are both represented with 3-vectors. As a consequence of this we say that points are dual to lines in $\mathbb{P}^2$. That is, whenever we have a theorem involving lines and points in $\mathbb{P}^2$ we can always exchange points for lines and get a dual statement. For example, the statement "Two lines intersect each other in one point" is dual to "For any two points there is one line going through them both". (Note that the first statement is not true in $\mathbb{R}^2$ if the lines are parallel.)

In three dimensions the equation

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

represents a plane. Therefore, for the space $\mathbb{P}^3$ (the space consisting of all elements that can be represented by vectors in $\mathbb{R}^4$) we have duality between points and planes instead.

Exercise 1. Compute the point of intersection $\mathbf{x} \in \mathbb{P}^2$ ($\mathbf{x} \sim (x, y, z)$) of the two lines $\mathbf{l}_1 \sim (-1, 0, 1)$ and $\mathbf{l}_2 \sim (0, -1, 1)$.

Exercise 2. Compute the line $\mathbf{l} \sim (a, b, c)$ passing through the points $\mathbf{x}_1 \sim (-1, 0, 1)$ and $\mathbf{x}_2 \sim (0, -1, 1)$. (Hint: look at the previous exercise.)

3 Vanishing Points

In the space $\mathbb{P}^2$ every pair of lines have a common intersection point, even parallel ones. Consider for example the two lines $\mathbf{l}_1 = (-1, 0, 1)$ and $\mathbf{l}_2 = (1, 0, 1)$, see Figure 1.

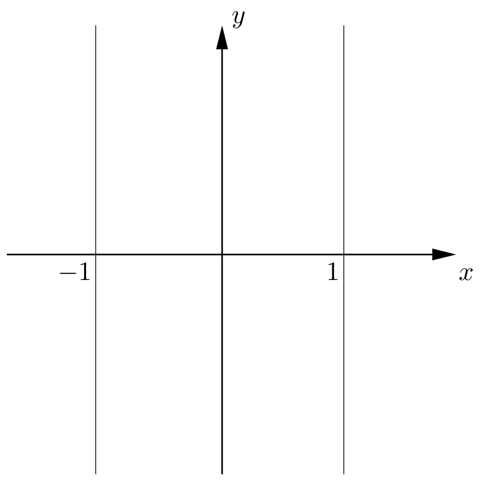


Figure 1: The two lines $(-1, 0, 1)$ and $(1, 0, 1)$.

Since $\mathbf{x}$ lies on both the lines we have to solve the system of equations

$$\begin{cases} \mathbf{l}_1^T\mathbf{x} & = 0 \\ \mathbf{l}_2^T\mathbf{x} & = 0 \end{cases} \Leftrightarrow \begin{cases} x + z & = 0 \\ -x + z & = 0 \end{cases}. \tag{6}$$

Since we have no constraints for y we get

$$\begin{cases} x + z & = 0 \\ 2z & = 0 \\ y & = t \end{cases} \Leftrightarrow \begin{cases} x & = 0 \\ y & = t \\ z & = 0 \end{cases}. \tag{7}$$

Hence, for example $(0, 1, 0)$ is a representative of our intersection point. In this case we cannot interpret the result by dividing with the third coordinate since this one is zero, which makes sense since the lines are parallel and therefore do not intersect in $\mathbb{R}^2$. To interpret $(0,1,0)$ geometrically we look at $(0,1,\epsilon)$, where ϵ is a small positive number. This point has non-zero third coordinate and is equivalent to $\left(0,\frac{1}{\epsilon},1\right)$, that is, it is a point with x-coordinate zero and a very large y-coordinate. Making ϵ smaller we see that $(0,1,0)$ can be interpreted as a point infinitely far away in the direction $(0,1)$. We call this type of point a vanishing point or a point at infinity.

Note that if we instead assume that ϵ is a small negative number we get a point far away in the direction $(0,-1)$. We therefore do not differ between these points, and in addition $(0,1,0) \sim (0,-1,0)$.

The line $z = 0$ is called the vanishing line or the line at infinity since it only contains points that has third coordinate 0.

4 Conics

The conics are all second order curves of the form

$$\mathbf{x}^T C \mathbf{x} = 0 \tag{8}$$

where C is a symmetric matrix. For example the circle of radius 1 can be written

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = x^2 + y^2 - 1 = 0. \tag{9}$$

The line $\mathbf{l} = Cx$ is a tangent line to to the conic at the point x. The dual conic is the set of all lines that tangent to the conic. If the matrix C is invertible then the expression for the dual conic can be computed by noting that

$$0 = x^T C x = x^T C C^{-1} C x = (Cx)^T C^{-1} C x = \mathbf{l}^T C^{-1} \mathbf{l}. \tag{10}$$

Therefore any line that is tangent to the conic has to fulfill $l^T C^{-1} l = 0$ which is a conic in the space of all lines. If C is not invertible one can use the pseudo inverse instead.

5 Projective Transformations

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

$$\mathbf{x} \sim H\mathbf{y} \tag{11}$$

where $\mathbf{x} \in \mathbb{R}^{n+1}$ and $\mathbf{y} \in \mathbb{R}^{n+1}$ 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.

Exercise 3. Show that it does not matter what representative we choose, the result will be the same. (Hint: $\mathbf{y}$ and $\lambda\mathbf{y}$ are two representatives of the same point.)

Projective mappings occur often when working with images. In Lecture 1 we saw one example of such a mapping, namely the K-matrix. In the camera equation

$$\mathbf{x} \sim K \left[R \quad t\right] \mathbf{X} \tag{12}$$

the matrix K transforms the point $\left[R \quad t\right] \mathbf{X}$ in the image plane to the real image coordinate system (with the unit pixels).

Another example is point transfer via a plane. Figure 2 shows two images of a desk with a roughly planar surface. With this setup the there is a homography that transforms the points of one image to the other given by (11), where H is a 3×3 matrix. To find the transformation we need to determine the elements of H. There are 9 elements but since thee scale does not matter (H and λH represents

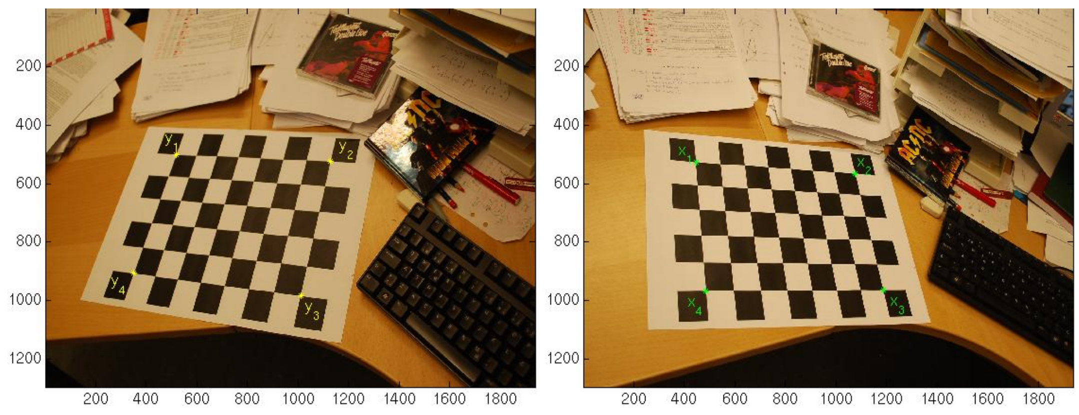


Figure 2: Two images of a roughly planar surface with 4 detected point correspondences.

the same transformation) there are only 8 degrees of freedom. Now suppose that we have n points $\mathbf{y}_i$, $i = 1, ..., n$ that we know are transformed to n corresponding points $\mathbf{x}_i$, $i = 1, ..., n$ in the second image. Each point pair gives us 3 equations

$$\lambda_i \mathbf{x}_i = H\mathbf{y}_i \tag{13}$$

(recall that $\mathbf{y}_i$ and $H\mathbf{x}_i$ are vectors of size 3) but one new unknown λ_i is introduced. We now have $3n$ equations and $8 + n$ degrees of freedom. To be able to find H we therefore need

$$3n \geq 8 + n \Leftrightarrow 2n \geq 8 \Leftrightarrow n \geq 4 \tag{14}$$

point correspondences. Figure 2 shows 4 point correspondences that can be used to compute H. (An approach for doing this, the so called DLT method, will be presented in Lecture 4.)

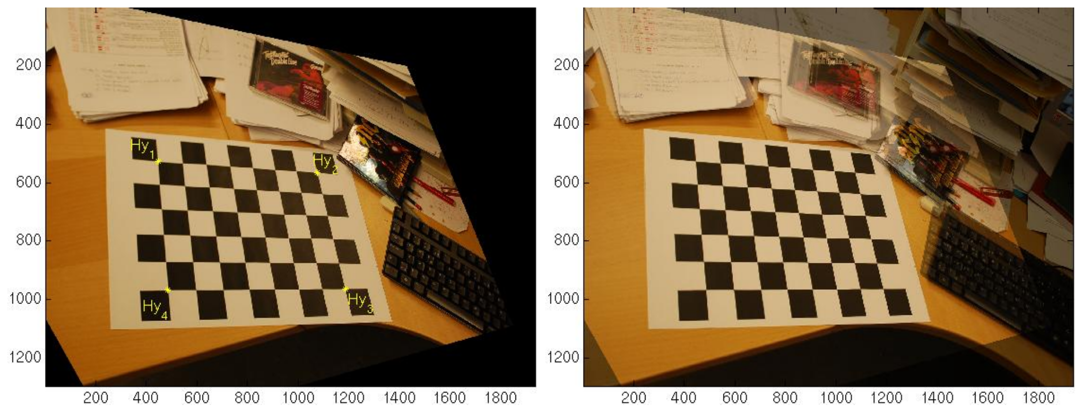


Figure 3: Left: the result of appying H to the left image in Figure 2. Right: The transformed image overlaid on the right image of Figure 2.

When H has been computed we can transform the other points in the image. Figure 3 shows the transformation of the left image and the transformed image overlaid on the right image. It can be ween that the images agree well where the scene is roughly planar.

When looking carefully at the checkerboard pattern it is evident that this transformation preserves lines, that is, points on a line in the original image is mapped to another line in the new figure. This is always the case when we have projective transformations.

Exercise 4. Assume that $\mathbf{y}$ lies on the line $\mathbf{l}$, that is, $\mathbf{l}^T\mathbf{y} = 0$, and that $\mathbf{x} \sim H\mathbf{y}$. Show that $\mathbf{x}$ lies on the line $\hat{\mathbf{l}} = (H^{-1})^T\mathbf{l}$.

5.1 Special Cases of Transformations

A special case of projective transformation $\mathbb{P}^n \mapsto \mathbb{P}^n$ is the affine transformation. For this type of mapping the matrix H has the special shape

$$H = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix}, \tag{15}$$

where A is an invertible $n \times n$ matrix, t is an $n \times 1$ vector and 0 is a $1 \times n$ vector of all zeros. Besides being projective, the affine transformation has the special property that parallel lines are mapped to parallel lines. Furthermore, it preserves the line at infinity, that is, vanishing points are mapped to vanishing points and regular points to regular points. If we only consider points in $\mathbb{R}^2$ (with regular Cartesian coordinates) then the transformation can be written $x = Ay + t$. An example of an affine transformation is shown in Figure 4.

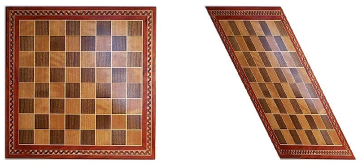


Figure 4: The affine transformation preserves parallel lines.

The similarity transformation has the form

$$H = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix}, \tag{16}$$

where R is an $n \times n$ rotation and s is a positive number. This mapping also preserves angles between lines. An example of a similarity transformation is shown in Figure 5.

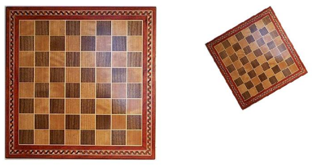


Figure 5: The similarity transformation preserves angles between lines.

If $s = 1$ in (16) then the transformation is called Euclidean. This mapping also preserves distances between points. An example of an affine transformation is shown in Figure 6.

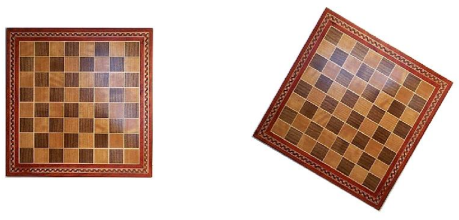


Figure 6: The Euclidean transformation preserves distances between points.

Lecture 3: Camera Calibration, DLT, SVD

1 The Inner Parameters

In this section we will introduce the inner parameters of the cameras. Recall from the camera equations

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

where $P = K[R \quad t]$, K is a 3×3 matrix R is a 3×3 rotation matrix and t is a 3×1 vector. The 3×4 matrix $[R \quad t]$ encodes the orientation and position of the camera with respect to a reference coordinate system. Given a 3D point in homogeneous coordinates $\mathbf{X}$ the product $[R \quad t]\,\mathbf{X}$ can be interpreted as the 3D coordinates of the scene point in the camera coordinate system. Note that alternatively we can interpret the result as the homogeneous coordinates of the projection of $\mathbf{X}$ into the image plane embedded in $\mathbb{R}^3$, since the projection in the camera coordinate system is computed by division with the third coordinate.

The 3×3 matrix K transforms the image plane in $\mathbb{R}^3$ to the real image coordinate system (with unit pixels), see Figure 1.

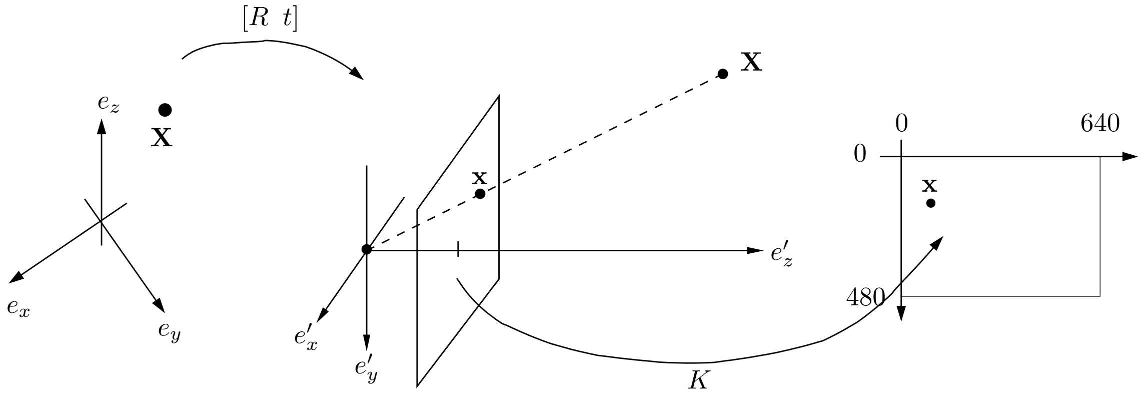


Figure 1: The different coordinate systems and mappings.

The matrix K is an upper triangular matrix with the following shape:

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

The parameter f is called the focal length. This parameter re-scales the image coordinates into pixels. The point (x_0, y_0) is called the principal point. For many cameras it is enough to use the focal length and principal point. In this case the K matrix transforms the image points according to

$$\begin{pmatrix} fx + x_0 \\ fy + y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{3}$$

that is, the coordinates are scaled by the focal length and translated by the principal point. Note that the center point $(0, 0, 1)$ of the image in $\mathbb{R}^3$ is transformed to the principal point (x_0, y_0).

The parameter γ is called the aspect ratio. For cameras where the pixels are not square the re-scaling needs to be done differently in the x-direction and the y-direction. In such cases the aspect ratio γ will take a value different from 1.

The final parameter s is called the skew. This parameter corrects for tilted pixels, see Figure 2, and is typically zero.

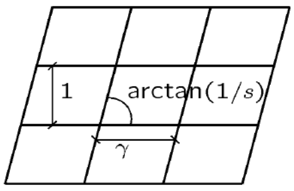


Figure 2: The skew parameter s corrects for non-rectangular pixels.

A camera $P = K [R \quad t]$ is called calibrated if the inner parameters K are known. For such cameras we can eliminate the K matrix from the camera equations by multiplying both sides of (1) with K^{-1} from the left. If we let $\tilde{\mathbf{x}} = K^{-1}\mathbf{x}$ we get

$$\lambda\tilde{\mathbf{x}} = K^{-1}K [R \quad t] \mathbf{X} = [R \quad t] \mathbf{X}. \tag{4}$$

The new camera matrix $[R \quad t]$ is called the normalized (calibrated) camera and the new image points $\tilde{\mathbf{x}}$ are called the normalized image points. (Note that later in this lecture there is a different concept of normalization that is used for improving stability of computations. However in the context of calibrated cameras, normalization always means multiplication with K^{-1}.)

The calibration model presented in this section is limited in the sense that the normalization is carried out by applying the homography K^{-1} to the image coordinates. For some cameras this is not sufficient. For example, in the image displayed in Figure 3, lines that are straight in 3D do not appear as straight lines in the image. Such distortion is common in cameras with wide field of view and can not be removed with a homograpy.

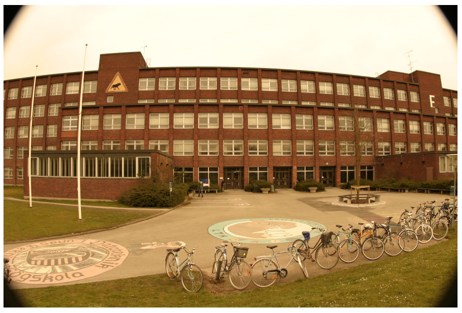


Figure 3: Radial distortion can not be handled with the K matrix. This requires a more complicated model.

2 Projective vs. Euclidean Reconstruction

The main problem of interest in this course is the Structure from Motion problem, that is, given image projections $\mathbf{x}_{ij}$ (of scene point j in image i) determine both 3D point coordinates $\mathbf{X}_j$ and camera matrices P_i such that

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j, \quad \forall i,j. \tag{5}$$

Note that the depths λ_{ij} are also unknown and need to be determined. However, primarily we are interested in the scene points and cameras, the depths are bi-products of this formulation.

If the calibration is unknown, that is P_i can be any non-zero 3×4 matrix then the solution to this problem is called a projective reconstruction. Such a solution can only be uniquely determined up to a projective transformation. To see this suppose that we have found cameras P_i and 3D-points $\mathbf{X}_j$ such that

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j. \tag{6}$$

To construct a different solution we can take an unknown projective transformation H ($\mathbb{P}^3 \mapsto \mathbb{P}^3$) and let $\tilde{P}_i = P_iH$ and $\tilde{\mathbf{X}}_j = H^{-1}\mathbf{X}_j$. The new cameras and scene points also solve the problem since

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j = P_iHH^{-1}\mathbf{X}_j = \tilde{P}_i\tilde{\mathbf{X}}_j. \tag{7}$$

This means that given a solution we can apply any projective transformation to the 3D points and obtain a new solution. Since projective transformations do not necessarily preserve angles or parallel lines projective reconstructions can look distorted even though the projections they give match the measured image points. To the left in Figure 4 a projective reconstruction of the Arch of Triumph in Paris is displayed.

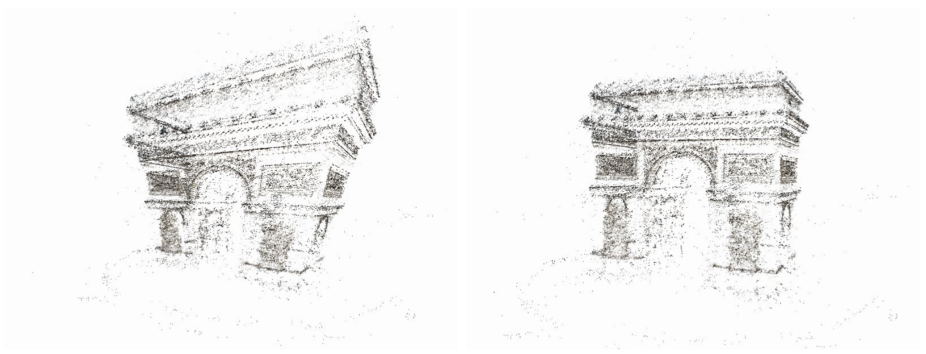


Figure 4: Reconstructions of the Arch of Triumph in Paris. Left: Projective reconstruction. Right: Euclidean reconstruction (known camera calibration). Both reconstructions provide the same projections.

One way to remove the projective ambiguity is to use calibrated cameras. If we normalize the image coordinates using $\tilde{\mathbf{x}} = K^{-1}\mathbf{x}$ then the structure form motion problem becomes that of finding normalized (calibrated) cameras $[R_i \quad t_i]$ and scene points $\mathbf{X}_j$ such that

$$\lambda_{ij}\tilde{\mathbf{x}}_{ij} = [R_i \quad t_i]\,\mathbf{X}_j, \tag{8}$$

where the first 3×3 block R_i is a rotation matrix. The solution of this problem is called a Euclidean Reconstruction. Given a solution $[R_i \quad t_i]$ and $\mathbf{X}_j$ we can try to do the same trick as in the projective case. However when multiplying $[R_i \quad t_i]$ with H the result does not necessarily have a rotation matrix in the first 3×3 block. To achieve a valid solution we need H to be a similarity transformation,

$$H = \begin{bmatrix} sQ & v \\ 0 & 1 \end{bmatrix}, \tag{9}$$

where Q is a rotation. We then get

$$\frac{\lambda_{ij}}{s}\mathbf{x}_{ij} = [R_i \quad t_i]\begin{bmatrix} Q & \frac{1}{s}v \\ 0 & \frac{1}{s} \end{bmatrix} H^{-1}\mathbf{X}_j = \begin{bmatrix} R_iQ & \frac{1}{s}(R_iv + t_i) \end{bmatrix}\tilde{\mathbf{X}}_j, \tag{10}$$

which is a valid solution since R_iQ is a rotation. Hence, in the case of Euclidean reconstruction we do not have the same distortion since similarity transformations preserve angles and parallel lines. Note that there is still an ambiguity here. The entire reconstruction can be re-scaled, rotated and translated without changing the image projections.

(Strictly speaking, Equation (10) only shows that we can apply a similarity transformation without violating the rotational constraints. It does not show that if H is not a similarity the constraints are violated. This can however be seen by considering the effects of adding H to all camera equations.)

3 Finding the Inner Parameters

In this section we will present a simple method for finding the camera parameters. We will do it in two steps:

1. First, we will compute a camera matrix P. To make sure that there is not projective ambiguity present (as in Section 2) we will assume that the scene point coordinates are known. This can for example be achieved by using an image of a known object where we have measured all the points by hand.
2. Secondly, once the camera matrix P is known we can factorize it into $K[R \quad t]$, where K is triangular and R is a rotation. This can be done using the so called RQ-factorization.

3.1 Finding P: The Resection Problem

In this section we will outline a method for finding the camera matrix P. We are assuming that the scene points $\mathbf{X}_i$ and their projections $\mathbf{x}_i$ are known. The goal is to solve the equations

$$\lambda_i \mathbf{x}_i = P\mathbf{X}_i, \quad i = 1, .., N, \tag{11}$$

where the λ_i and P are the unknowns. This problem, determining the camera matrix from know scene points and projections is called the resection problem. The 3×4 matrix P has 12 elements, but the scale is arbitrary and therefore it only has 11 degrees of freedom. There are $3N$ equations (3 for each point projection), but each new projection introduces one additional unknown λ_i. Therefore we need

$$3N \geq 11 + N \Rightarrow N \geq 6 \tag{12}$$

points in order for the problem to be well defined. To solve the problem we will use a simple approach 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. If we let p_i, $i = 1, 2, 3$ be 4×1 vectors containing the rows of P, that is,

$$P = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} \tag{13}$$

then we can write (11) as

$$\mathbf{X}_i^T p_1 - \lambda_i x_i = 0 \tag{14}$$
$$\mathbf{X}_i^T p_2 - \lambda_i y_i = 0 \tag{15}$$
$$\mathbf{X}_i^T p_3 - \lambda_i = 0, \tag{16}$$

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

$$\begin{bmatrix} \mathbf{X}_i^T & 0 & 0 & -x_i \\ 0 & \mathbf{X}_i^T & 0 & -y_i \\ 0 & 0 & \mathbf{X}_i^T & -1 \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{17}$$

Note that since $\mathbf{X}_i$ is a 4×1 vector each 0 on the left hand side actually represents a 1×4 block of zeros. Thus the left hand side is a 3×13 matrix multiplied with a 13×1 vector. If we include all the projection equations in one matrix we get a system of the form

$$\underbrace{\begin{bmatrix} \mathbf{X}_1^T & 0 & 0 & -x_1 & 0 & 0 & \dots \\ 0 & \mathbf{X}_1^T & 0 & -y_1 & 0 & 0 & \dots \\ 0 & 0 & \mathbf{X}_1^T & -1 & 0 & 0 & \dots \\ \mathbf{X}_2^T & 0 & 0 & 0 & -x_2 & 0 & \dots \\ 0 & \mathbf{X}_2^T & 0 & 0 & -y_2 & 0 & \dots \\ 0 & 0 & \mathbf{X}_2^T & 0 & -1 & 0 & \dots \\ \mathbf{X}_3^T & 0 & 0 & 0 & 0 & -x_3 & \dots \\ 0 & \mathbf{X}_3^T & 0 & 0 & 0 & -y_3 & \dots \\ 0 & 0 & \mathbf{X}_3^T & 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{=M} \underbrace{\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{pmatrix}}_{=v} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}. \tag{18}$$

Here we are interested in finding a non-zero vector in the nullspace of M. Since the scale is arbitrary we can add the constraint $\|v\|^2 = 1$. 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

$$\min_{\|v\|^2=1} \|Mv\|^2. \tag{19}$$

We refer to this type of problem as a homogeneous least squares problem. Note that there are always at least two solutions to (19) since $\|Mv\| = \|M(-v)\|$ and $\|v\| = \|-v\|$. These solutions give the same projections, however for one of them the camera faces away from the scene points thereby giving negative depths. If the homogeneous representative for the scene points have positive fourth coordinate then we should select the solution where the λ_i are all positive.

An alternative formulation with only the p-variables can be found by noting that (11) means that the vectors $\mathbf{x}_i$ and $P\mathbf{X}_i$ are be parallel. This can be expressed using the vector product

$$\mathbf{x}_i \times P\mathbf{X}_i = 0, \quad i = 1, ..., N. \tag{20}$$

These equations are also linear in p_1, p_2, p_3 and we can therefore set up a similar homogeneous least squares system but without the λ_i.

3.1.1 Solving the Homogeneous System

The solution to (19) can be found by eigenvalue computations. If we let $f(v) = v^T M^T M v$ and $g(v) = v^T v$ we can write the problem as

$$\min_{g(v)=1} f(v). \tag{21}$$

From basic courses in Calculus (e.g. Person-Böiers Fler-dim.) we know that the solution must fulfill

$$\nabla f(v) = \gamma \nabla g(v) \quad \Leftrightarrow \quad 2M^T M v = \gamma 2v \quad \Leftrightarrow \quad M^T M v = \gamma v. \tag{22}$$

Therefore the solution of (19) has to be an eigenvector of the matrix $M^T M$. Suppose v_* is an eigenvector with eigenvalue γ_*. If we insert into the objective function we get

$$f(v_*) = v_*^T M^T M v_* = \gamma_* v_*^T v_*. \tag{23}$$

Since $\|v_*\| = 1$ we see that in order to minimize f we should select the eigenvector with the smallest eigenvalue.

Because of the special shape of $M^T M$ we compute the eigenvectors efficiently using the so called Singular Value Decomposition (SVD).

Theorem 1. Each $m \times n$ matrix M (with real coefficients) can be factorized into

$$M = USV^T, \tag{24}$$

where U and V are orthogonal ($m \times m$ and $n \times n$ respectively),

$$S = \begin{bmatrix} \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_r) & 0 \\ 0 & 0 \end{bmatrix}, \tag{25}$$

$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ and r is the rank of the matrix.

If M has the SVD (24) then

$$M^T M = (USV^T)^T USV^T = VS^T U^T USV^T = VS^T SV^T. \tag{26}$$

Since $S^T S$ is a diagonal matrix this means that V diagonalizes $M^T M$ and therefore $S^T S$ contains the eigenvalues and V the eigenvectors of $M^T M$. The diagonal elements of $S^T S$ are ordered decreasingly $\sigma_1^2, \sigma_2^2, ..., \sigma_r^2, 0, ..., 0$. Thus, to find an eigenvector corresponding to the smallest eigenvalue we should select the last column of V. Note that if $r < n$, that is, the matrix M does not have full rank, then the eigenvalue we select will be zero which means that there is an exact nonzero solution to $Mv = 0$. In most cases however $r = n$ due to noise.

We summarize the steps of the algorithm here:

- Set up the linear homogeneous system
$$Mv = 0. \tag{27}$$
- Compute the singular value decomposition.
$$M = USV^T. \tag{28}$$
- Extract the solution v_* from the last column of V.

3.1.2 Normalization.

The matrix M will contain entries x_i, y_j and ones. Since the x_i and y_i are measured in pixels the values can be in the thousands. In contrast the third homogeneous coordinate is 1 and therefore the matrix M contains coefficient of highly varying magnitude. This can make the matrix $M^T M$ poorly conditioned resulting buildup of numerical errors.

The numerics can often be greatly improved by translating the coordinates such that their “center of mass” is zero and then rescaling the coordinates to be roughly 1.

Suppose that we want to solve

$$\lambda_i \mathbf{x} = P\mathbf{X}_i, \quad i = 1, ..., N \tag{29}$$

as outlined in the previous sections.

We can change the coordinates of the image points by applying the normalization mapping

$$N = \begin{bmatrix} s & 0 & -s\bar{x} \\ 0 & s & -s\bar{y} \\ 0 & 0 & 1 \end{bmatrix}. \tag{30}$$

This mapping will first translate the coordinates by $(-\bar{x}, -\bar{y})$ and then re-scale the result with the factor s. If for example $(\bar{x}, \bar{y})$ is the mean point then the transformation

$$\tilde{\mathbf{x}} = N\mathbf{x} \tag{31}$$

gives re-scaled coordinates with "center of mass" in the origin.

We can now solve the modified problem

$$\gamma_i \tilde{\mathbf{x}} = \tilde{P}\mathbf{X}_i, \tag{32}$$

by forming the system matrix M and computing its singular value decomposition. A solution to the original "un-normalized" problem (29) can now easily be found from

$$\gamma_i N\mathbf{x} = \tilde{P}\mathbf{X}_i. \tag{33}$$

3.2 Computing the Inner Parameters from P

When the camera matrix has been computed we want to find the inner parameters K by factorizing P into

$$P = K\left[R \quad t\right], \tag{34}$$

where K is a right triangular and R is a rotation matrix. We this can be done using the *RQ*-factorization.

Theorem 2. If A is an $n \times n$ matrix then there is an orthogonal matrix Q and a right triangular matrix R such that

$$A = RQ. \tag{35}$$

(If A is invertible and the diagonal elements are chosen positive then the factorization is unique.)

In order to be consistent with the notation in the rest of the lecture we will 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 $A = KR$. If

$$K = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}, \quad A = \begin{bmatrix} A_1^T \\ A_2^T \\ A_3^T \end{bmatrix} \text{ and } R = \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix}, \tag{36}$$

that is, 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, then we get

$$\begin{bmatrix} A_1^T \\ A_2^T \\ A_3^T \end{bmatrix} = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix} = \begin{bmatrix} aR_1^T + bR_2^T + cR_3^T \\ dR_2^T + eR_3^T \\ fR_3^T \end{bmatrix} \tag{37}$$

From the third row of (37) we see that $A_3 = fR_3$. Since the matrix R is orthogonal R_3 has to have the length 1. We therefore see that need to select

$$f = \|A_3\| \quad \text{and} \quad R_3 = \frac{1}{\|A_3\|}A_3. \tag{38}$$

to get a positive coefficient f. When R_3 is known we can proceed to the second row of (37). The equation $A_2 = dR_2 + eR_3$ 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

$$e = A_2^T R_3. \tag{39}$$

When e is known we can compute R_2 and d from

$$dR_2 = A_2 - eR_3, \tag{40}$$

similar to what we did for f and R_3 in (38). When R_2 and R_3 is known we use the first row of (37)

$$A_1 = aR_1 + bR_2 + cR_3 \tag{41}$$

to compute b and c. Finally we can compute a and R_1 from

$$A_1 - bR_2 - cR_3 = aR_1. \tag{42}$$

The resulting matrix K is not necessarily of the form (2) 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 $(3,3)$. Note however, that this does not modify the camera in any way since the scale is arbitrary.

Lecture 4: Autocorrelation, Triangulation and Homography Estimation

1 The Autocorrelation function

If we want to investigate how the structure of an image I varies we can look at the following energy function

$$E_{AC}(\tilde{u},t) = \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})(I(u+t)-I(u))^2 du, \tag{1}$$

where $w(u)$ are weights that are typically zero outside some small window. It is usually chosen as a Gaussian function with some width σ. This function describes how much the image changes at the points $\tilde{u}=(x,y)$ by translating the image by a vector t, taking the difference, and intergrate over some neighbourhood around $\tilde{u}$. We can further investigate this function by linearizing the image around the point $\tilde{u}$,

$$I(u+t) \approx I(u) + \nabla I(u)^T t \Leftrightarrow I(u+t) - I(u) \approx \nabla I(u)^T t, \tag{2}$$

so that

$$E_{AC}(\tilde{u},t) \approx \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})(\nabla I(u)^T t)^2 du = \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u}) t^T \nabla I(u) \nabla I(u)^T t du. \tag{3}$$

Here

$$\nabla I(u)\nabla I(u)^T = \begin{bmatrix} \frac{\partial I}{\partial x}^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \frac{\partial I}{\partial y}^2 \end{bmatrix}. \tag{4}$$

The integration doesn't depend on t so we can write our energy as

$$E_{AC}(\tilde{u},t) \approx t^T \begin{bmatrix} \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial x}^2 du & \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} du \\ \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} du & \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial y}^2 du \end{bmatrix} t. \tag{5}$$

We call

$$A(\tilde{u}) = \begin{bmatrix} \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial x}^2 du & \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} du \\ \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} du & \iint\limits_{u\in\mathbb{R}^2} w(u-\tilde{u})\frac{\partial I}{\partial y}^2 du \end{bmatrix}, \tag{6}$$

The autocorrelation function. It is also known as the structure tensor or the orientation tensor. It is a symmetric positive semidefinite matrix, and hence its Eigenvalues are real and non-negative. The Eigenvalues and Eigenvectors of this matrix tells a lot about the local intensity structure around the point $\tilde{u}$. At smooth areas A will have two small Eigenvalues, at edgelike structures it will have one large and one small Eigenvalue, and at cornerlike structures it will have two large Eigenvalues. The Eigenvectors of A will be directed along the dominant directions of the local structure. Many feature detectors are based on the functions on A for instance the classic Harris corner detector. The Harris corners are chosen finding the local maxima of

$$\det(A(\tilde{u})) - \alpha \text{trace}(A(\tilde{u}))^2 = \lambda_0\lambda_1 - \alpha(\lambda_0+\lambda_1)^2, \tag{7}$$

where λ_j are the Eigenvalues of $A(\tilde{u})$. In Figure 1 the result of applying the Harris corner on an image is shown.

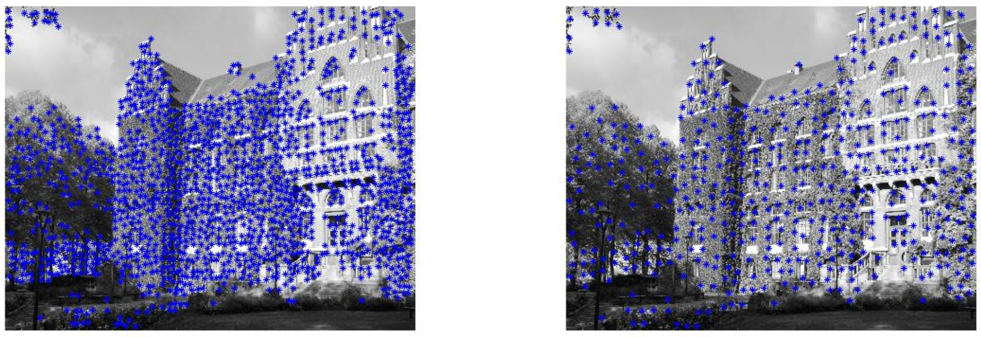


Figure 1: The result of the Harris corner detector. To the right only the strongest local maxima within some fixed radius.

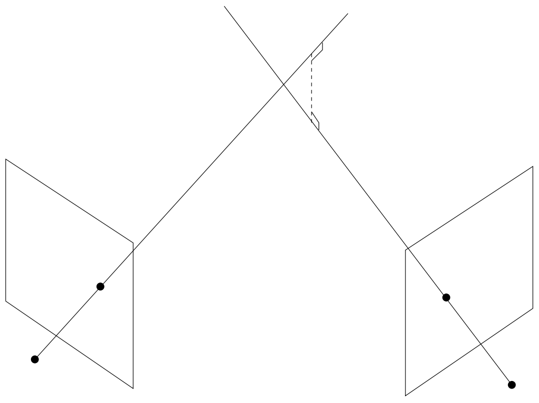


Figure 2: Geometrically triangulation is finding the 3D point closest to the image rays of the corresponding image points.

2 Triangulation

We will in this section describe a method for finding the position of a 3D point given that the projections of this point in a number of images is known, as well as the camera matrices. This problem is known as triangulation. 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, see Figure 2. It is clear that we at least need two lines, and so at least projections of the 3D point in two images. We can decribe our problem as

$$\lambda_i x_i = P_i X, i = 1 \dots n. \tag{8}$$

For n image points we have $3n$ equations and $n+3$ unknowns, so $3n \geq n+3$ or $n \geq \frac{3}{2}$. So this is consistent with our geometric argument. The problem is linear in the unknown λ_i and X, so we can find the least squares solution to this problem by formulating it as

$$Mv = 0, \tag{9}$$

with

$$M = \begin{bmatrix} P_1 & -x_1 & 0 & \cdots & 0 \\ P_2 & 0 & -x_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ P_n & 0 & 0 & \cdots & -x_n \end{bmatrix}, \tag{10}$$

and

$$v^T = \begin{bmatrix} X^T & \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}. \tag{11}$$

As before we can find the solution using the singular value decomposition of M.

3 Homography Estimation

In homography estimation we want to find a projective transformation from $\mathbb{P}^k$ to $\mathbb{P}^k$, i.e. a homography. Usually $k = 2$ or $k = 3$. We will describe the problem for $k = 2$ but the procedure is exactly the same for any dimension. So given two sets of points u_i and v_i that are related by a homography H we want to solve

$$\lambda_i u_i = H v_i, i = 1 \ldots n. \tag{12}$$

We write

$$H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, \tag{13}$$

and

$$u_i^T = [x_i \; y_i \; 1]. \tag{14}$$

Here H is a 3×3 matrix so h_1, h_2 and h_3 are 1×3 matrices. We can now write our problem as

$$Mv = 0, \tag{15}$$

with

$$M = \begin{bmatrix} v_1^T & 0 & 0 & -x_1 & 0 & \cdots & 0 \\ 0 & v_1^T & 0 & -y_1 & 0 & \cdots & 0 \\ 0 & 0 & v_1^T & -1 & 0 & \cdots & 0 \\ v_2^T & 0 & 0 & 0 & -x_2 & \cdots & 0 \\ 0 & v_2^T & 0 & 0 & -y_2 & \cdots & 0 \\ 0 & 0 & v_2^T & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ v_n^T & 0 & 0 & 0 & 0 & \cdots & -x_n \\ 0 & v_n^T & 0 & 0 & 0 & \cdots & -y_n \\ 0 & 0 & v_n^T & 0 & 0 & \cdots & -1 \end{bmatrix}, \tag{16}$$

and

$$v^T = \begin{bmatrix} h_1 & h_2 & h_3 & \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}. \tag{17}$$

And this can again be solved in a least squares sense using the singular value decomposition of M. For n points we have $3n$ equations and $8 + n$ unknown (remember that H is only determined up to scale), so we need at least 4 point correspondences to estimate a two-dimensional projective transformation. For higher dimensions we need a larger number of correspondences, but the procedure to estimate the homography will be the same.

3.1 Panoramic stitching

As an example of homography estimation we will show how we can stitch together a number of images taken from the same position. So we assume that we have taken a number of images from the same location, and only rotated the camera between the images. For ease of notation we will assume that we have calibrated cameras and that the image coordinates are normalized using the inverse of the calibration matrices. Since we have taken the images from the same point and we are free to choose a global coordinate system, we will assume that the camera center is at the origin. This means that for two cameras and two corresponding image points we have the following system

$$\lambda_1 x_1 = [R_1 \; 0] \begin{bmatrix} X \\ 1 \end{bmatrix}, \tag{18}$$

$$\lambda_2 x_2 = [R_2 \; 0] \begin{bmatrix} X \\ 1 \end{bmatrix}. \tag{19}$$

This gives us

$$\left\{ \begin{array}{l} \lambda_1 x_1 = R_1 X \\ \lambda_2 x_2 = R_2 X \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} X = \lambda_1 R_1^T x_1 \\ \lambda_2 x_2 = R_2 X \end{array} \right. \Leftrightarrow \left\{ \begin{array}{c} X = \lambda_1 R_1^T x_1 \\ \lambda_2 x_2 = \lambda_1 R_2 R_1^T x_1 \end{array} \right. . \tag{20}$$

This means that we can write the last equation as

$$\lambda x_2 = H x_1, \tag{21}$$

with $\lambda = \frac{\lambda_2}{\lambda_1}$ and $H = R_2 R_1^T$. We know see that we can transfer points from the first image plane to the second by the use of a homography. We know from the previous discussion that we can estimate this homgraphy from at least four point correspondences. Once we have estimated the homgraphy, all points in an image can be transferred using this homography. If we have uncalibrated cameras the only difference is that the homography will be of the following form

$$H = K_2 R_2 R_1^T K_1^{-1}. \tag{22}$$

Lecture 5: Epipolar Geometry and the Fundamental Matrix

1 Two-View Structure from Motion

In this lecture we will consider the two-view structure from motion problem. That is, given two images we want to compute both the camera matrices and the scene points such that the camera equations

$$\lambda_i \mathbf{x}_i = P_1 \mathbf{X}_i \tag{1}$$
$$\bar{\lambda}_i \bar{\mathbf{x}}_i = P_2 \mathbf{X}_i, \tag{2}$$

$i = 1, ..., n$, are fulfilled. In previous lectures we have considered sub-problems where either the camera matrices are known (the triangulation problem) or the scene points are known (the resection problem). Since the camera equations become linear in these two cases we could solve these directly by applying DLT. The situation becomes more complicated when both the scene points and camera matrices are unknown. The approach we will take in this lecture formulates a set of algebraic constraints that involve only the image points and the cameras, thereby eliminating the scene points. The resulting equations are linear and can be solved using SVD. Once the cameras are known the 3D points can be computed using triangulation.

1.1 Problem Formulation

Given two sets of corresponding points $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$, $i = 1, .., n$ our goal is to find camera matrices P_1 and P_2 such that (1)-(2) are solved. As we observed in lecture 3, if the cameras are uncalibrated the reconstruction can only be determined uniquely up to an unknown projective transformation. If the cameras are $P_1 = [A_1 \quad t_1]$ and $P_2 = [A_2 \quad t_2]$ then we can apply the transformation

$$H = \begin{bmatrix} A_1^{-1} & -A_1^{-1}t_1 \\ 0 & 1 \end{bmatrix} \tag{3}$$

to the camera equations (1)-(2). The camera matrix P_1 is then transformed to

$$P_1 H = \begin{bmatrix} A_1 & t_1 \end{bmatrix} \begin{bmatrix} A_1^{-1} & -A_1^{-1}t_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}. \tag{4}$$

Therefore, we can always assume that there is a solution where $P_1 = [I \quad 0]$ and $P_2 = [A \quad t]$.

2 Epipolar Geometry

In this section we will derive the so called epipolar constraints. In the following sections we will use these constraints to find camera matrices that solve (1)-(2).

We first consider a point $\mathbf{x}$ in the first image. The scene points that can project to this image point all lie on a line (the viewing ray of $\mathbf{x}$) in 3D space, see Figure 1. If we assume that the $\mathbf{X} = \begin{bmatrix} X \\ 1 \end{bmatrix}$, where $X \in \mathbb{R}^3$, are the homogeneous coordinates of a scene point projecting to $\mathbf{x}$, then

$$\lambda \mathbf{x} = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = X. \tag{5}$$

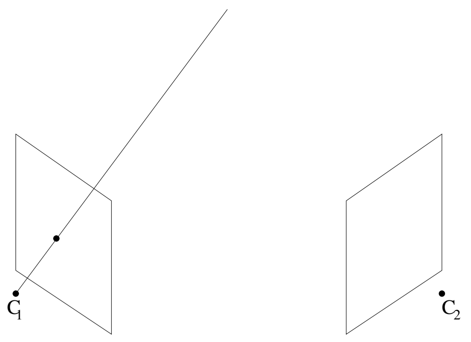


Figure 1: All scene points on the line project to the same point in the left camera.

Therefore the viewing ray of $\mathbf{x}$ can be parametrized by

$$\mathbf{X}(\lambda) = \begin{bmatrix} \lambda\mathbf{x} \\ 1 \end{bmatrix}. \tag{6}$$

The projection of this line into the second camera is

$$P_2\mathbf{X}(\lambda) = \begin{bmatrix} A & t \end{bmatrix} \begin{bmatrix} \lambda\mathbf{x} \\ 1 \end{bmatrix} = \lambda A\mathbf{x} + t. \tag{7}$$

This is a line in the second image, see Figure 2. We refer to this line as the epipolar line of $\mathbf{x}$. Since all scene points that can project to $\mathbf{x}$ are on the viewing ray, all points in the second image that can correspond $\mathbf{x}$ have to be on the epipolar line. This condition is called the epipolar constraint. For different points $\mathbf{x}$ in the first image we get different viewing rays that project to different epipolar lines. Since the viewing rays all pass through the camera center C_1 of the first camera the epipolar lines will all intersect each other in the projection $\mathbf{e}_2$ of the camera center C_1, see Figure 2. The projections of the camera centers $\mathbf{e}_1$ and $\mathbf{e}_2$ are called the epipoles.

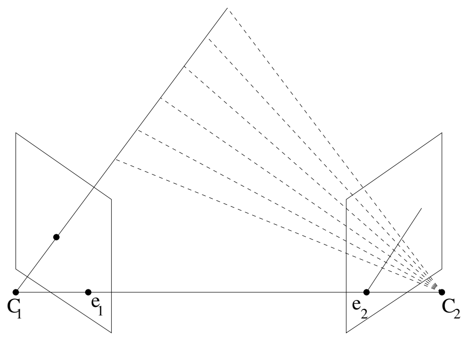


Figure 2: The projection of the viewing line into the second camera gives an epipolar line.

Exercise 1. Consider the two cameras $P = [I \quad 0]$ and $P_2 = [A \quad t]$, where A is invertible. Compute the epipoles $\mathbf{e}_1 \sim P_1C_2$, $\mathbf{e}_2 \sim P_2C_1$ and show that the line $\lambda A\mathbf{x} + t$ contains the $\mathbf{e}_2$.

The expression $\lambda A\mathbf{x} + t$ is a parametrization of the epipolar line of $\mathbf{x}$. We know that a line in $\mathbb{P}^2$ can also be represented by a line equation $\mathbf{l}^T\mathbf{x} = 0$. To find the vector $\mathbf{l}$ we pick two points on the line, e.g. t and $A\mathbf{x} + t$ and insert into the line equation

$$\begin{cases} \mathbf{l}^T t = 0 \\ \mathbf{l}^T(A\mathbf{x} + t) = 0 \end{cases}. \tag{8}$$

These equations tells us that $\mathbf{l}$ has to be perpendicular to both t and $A\mathbf{x}+t$. We can find such an $\mathbf{l}$ using the vector product

$$\mathbf{l} = t \times (A\mathbf{x}+t) = t \times (A\mathbf{x}). \tag{9}$$

Since $t \sim \mathbf{e}_2$ we can also write this as $\mathbf{e}_2 \times (A\mathbf{x})$.

Exercise 2. If $P_1 = [I \quad 0]$ and

$$P_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \tag{10}$$

which of the two points $\bar{\mathbf{x}}_1 = (0,1,2)$ and $\bar{\mathbf{x}}_2 = (1,2,3)$ in image 2 could correspond to $\mathbf{x} = (0,1,1)$ in image 1?

A cross pruduct $\mathbf{y} \times \mathbf{x}$ is a linear operation on $\mathbf{x}$ and can therefore be represented by a matrix $[\mathbf{y}]_\times$. If $\mathbf{x} = (x_1,x_2,x_3)$ and $\mathbf{y} = (y_1,y_2,y_3)$ then their cross product is

$$\mathbf{y} \times \mathbf{x} = (y_2x_3 - y_3x_2, y_3x_1 - y_1x_3, y_1x_2 - y_2x_1). \tag{11}$$

In matrix form we can write this

$$\underbrace{\begin{pmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{pmatrix}}_{=[\mathbf{y}]_\times} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_2x_3 - y_3x_2 \\ y_3x_1 - y_1x_3 \\ y_1x_2 - y_2x_1 \end{pmatrix} \tag{12}$$

The matrix $[\mathbf{y}]_\times$ is skew symmetric, that is, $[\mathbf{y}]_\times = -[\mathbf{y}]_\times^T$. It is easy to see that for any 3×3 skew symmetric matrix S there is a vector $\mathbf{y}$ such that $S = [\mathbf{y}]_\times$.

With this notation the epipolar line can be written

$$\mathbf{l} = \mathbf{e}_2 \times (A\mathbf{x}) = [\mathbf{e}_2]_\times A\mathbf{x}. \tag{13}$$

The matrix $F = [\mathbf{e}_2]_\times A$ is called the fundamental matrix. It maps points in image 1 to lines in image 2. If $\bar{\mathbf{x}}$ corresponds to $\mathbf{x}$ then the epipolar constraint can be written

$$\bar{\mathbf{x}}^T\mathbf{l} = \bar{\mathbf{x}}^T F\mathbf{x} = 0. \tag{14}$$

Note that F only depends on the cameras and therefore the epipolar constraints only involves the image points and the camera P_2. We will use these constraints to compute F from a number of image correspondences. Once F has been determined the camera P_2 can be extracted.

Exercise 3. Show that if F is a fundamental matrix then $F^T\mathbf{e}_2 = 0$ and $\det(F) = 0$.

3 Finding F: The Eight Point Algorithm

Recall that the objective of the two-view structure from motion problem is to find the scene points and the camera P_2. We will see in the next lecture that if the Fundamental matrix is known then P_2 can be extracted from it. We now present a simple algorithm for estimating F.

As we saw in the previous section, for each point correspondence $\mathbf{x}_i$,$\bar{\mathbf{x}}_i$ we get one epipolar constraint.

$$\bar{\mathbf{x}}_i^T F\mathbf{x}_i = 0. \tag{15}$$

If $\mathbf{x}_i \sim (x_i,y_i,z_i)$ and $\bar{\mathbf{x}} \sim (\bar{x}_i,\bar{y}_i,\bar{z}_i)$ then we can write this as

$$\begin{aligned} 0 = \bar{\mathbf{x}}_i^T F\mathbf{x}_i = \quad & F_{11}\bar{x}_ix_i + F_{12}\bar{x}_iy_i + F_{13}\bar{x}_iz_i \\ & +F_{21}\bar{y}_ix_i + F_{22}\bar{y}_iy_i + F_{23}\bar{y}_iz_i \\ & +F_{31}\bar{z}_ix_i + F_{32}\bar{z}_iy_i + F_{33}\bar{z}_iz_i. \end{aligned} \tag{16}$$

Therefore each correspondence gives one linear constraint on the entries of F. In matrix form we can write the resulting system as

$$\underbrace{\begin{pmatrix} \bar{x}_1 x_1 & \bar{x}_1 y_1 & \bar{x}_1 z_1 & \dots & \bar{z}_1 z_1 \\ \bar{x}_2 x_2 & \bar{x}_2 y_2 & \bar{x}_2 z_2 & \dots & \bar{z}_2 z_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{x}_n x_n & \bar{x}_n y_n & \bar{x}_n z_n & \dots & \bar{z}_n z_n \end{pmatrix}}_{M} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ \vdots \\ F_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{17}$$

This is a linear homogeneous system which we can solve using SVD as in Lecture 3. The matrix F has 9 entries but the scale is arbitrary and the system therefore has 8 degrees of freedom. Each correspondence gives one new constraint on F and we therefore need 8 correspondences to solve this problem.

Note that it is in fact possible to solve the problem with only 7 point correspondences since we also have the constraint $\det(F) = 0$. However, this constraint is a polynomial of third order and we cannot use SVD to solve the resulting system. Therefore we use at least 8 correspondences.

Because of noise the resulting estimation $\tilde{F}$ of the fundamental matrix is not likely to have zero determinant. Therefore given this estimation we chose the matrix F that solves

$$\min_{\det(F)=0} \|\tilde{F} - F\| \tag{18}$$

(where the norm is the Frobenious/sum-of-squares norm). The solution to this problem is given by the SVD of $\tilde{F}$. If

$$USV^T = \tilde{F}, \tag{19}$$

where $S = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$. Then F can be found by setting the smallest singular value $\sigma_3 = 0$, that is,

$$F = U\text{diag}(\sigma_1, \sigma_2, 0)V^T. \tag{20}$$

As was the case with the resection problem, normalization is important for numerical stability. If for example x_1 and $\bar{x}_1$ are both in the order of a 1000 pixels then $x_1\bar{x}_1 \approx 10^6$ while $z_1\bar{z}_1 = 1$. This makes the matrix M^TM very poorly conditioned. To improve the numerics we can use the same normalization as in Lecture 3 (for both the cameras).

We summarize the different steps of the algorithm here:

- Extract at least 8 point correspondences.
- Normalize the coordinates (see Lecture 3).
- Form M and solve
$$\min_{||v||^2=1} ||Mv||^2,$$
using svd.
- Form the matrix F (and ensure that $\det(F) = 0$).
- Transform back to the original (un-normalized) coordinates.
- Compute P_2 from F (next lecture).
- Compute the scene points using triangulation (see Lecture 4).

Lecture 6: Camera Computation and the Essential Matrix

1 Computing Cameras From the Fundamental Matrix

In Lecture 5 we considered the two-view structure from motion problem, that is, given a number of measured points in two images we want to compute both camera matrices and 3D points such that they project to the measurements. We showed that the 3D points can be eliminated from the problem by considering the fundamental matrix F. If $\mathbf{x}$ is an image point belonging to the fist image and $\bar{\mathbf{x}}$ belongs to the second then there is a 3D point that projects to to these if and only if the epipolar constraint

$$\bar{\mathbf{x}}^T F\mathbf{x} = 0 \tag{1}$$

is fulfilled. Using the projections of 8-scene points we can compute the fundamental matrix by solving a homogeneous least squares problem (the 8-point algorithm). What remains in order to find a solution to the two-view structure from motion camera is to compute cameras from F and finally compute the 3D-points.

In general we may assume (see Lecture 5) that the cameras are of the form $P_1 = [I \quad 0]$ and $P_2 = [A \quad \mathbf{e}_2]$ where $\mathbf{e}_2$ is the epipole in the second image. Since we know that $F^T\mathbf{e}_2 = 0$ we can find $\mathbf{e}_2$ by computing the null space of F^T. A solution for the second camera is then given by

$$P_2 = [[\mathbf{e}_2]_\times F \quad \mathbf{e}_2]\,. \tag{2}$$

To see that these cameras have the fundamental matrix F we verify that their projections fulfill the epipolar constraint. If $\mathbf{X} = \begin{bmatrix} X \\ \rho \end{bmatrix}$, where $X \in \mathbb{R}^3$ and $\rho \in \mathbb{R}$ then the projections in the two cameras are given by

$$\begin{aligned} \mathbf{x} &\sim \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} X \\ \rho \end{bmatrix} = X, && (3) \\ \bar{\mathbf{x}} &\sim \begin{bmatrix} [\mathbf{e}_2]_\times F & \mathbf{e}_2 \end{bmatrix} \begin{bmatrix} X \\ \rho \end{bmatrix} = [\mathbf{e}_2]_\times FX + \mathbf{e}_2\rho = \mathbf{e}_2 \times (FX) + \mathbf{e}_2\rho. && (4) \end{aligned}$$

Therefore

$$\bar{\mathbf{x}}^T F\mathbf{x} \sim (\mathbf{e}_2 \times (FX) + \mathbf{e}_2\rho)^T FX = (\mathbf{e}_2 \times (FX))^T FX + \rho\mathbf{e}_2^T FX. \tag{5}$$

Since $e_2 \times (FX)$ is a vector that is perpendicular to FX the term $(\mathbf{e}_2 \times (FX))^T FX = 0$. Furthermore, since $\mathbf{e}_2$ is in the null space of F^T the term $\rho\mathbf{e}_2^T FX = \rho(F^T\mathbf{e}_2)^T X = 0$. Therefore the projections of these camera matrices will fulfill the epipolar constraints.

Exercise 1. What is the camera center of $P_2 = [[\mathbf{e}_2]_\times F \quad \mathbf{e}_2]$? (Hint: Recall that $F\mathbf{e}_1 = 0$.)

Since there is a projective ambiguity there are many choices for P_2. Given that $P_1 = [I \quad 0]$ the general formula for P_2 is

$$P_2 = \begin{bmatrix} [\mathbf{e}_2]_\times F + \mathbf{e}_2 v^T & \lambda\mathbf{e}_2 \end{bmatrix}, \tag{6}$$

where v is some vector in $\mathbb{R}^3$ and λ is a non-zero scalar.

Exercise 2. Verify that the projections in $P_1 = [I \quad 0]$ and P_2 given by (6) fulfill the epipolar constraints for any $v \in \mathbb{R}^3$ and $\lambda \neq 0$.

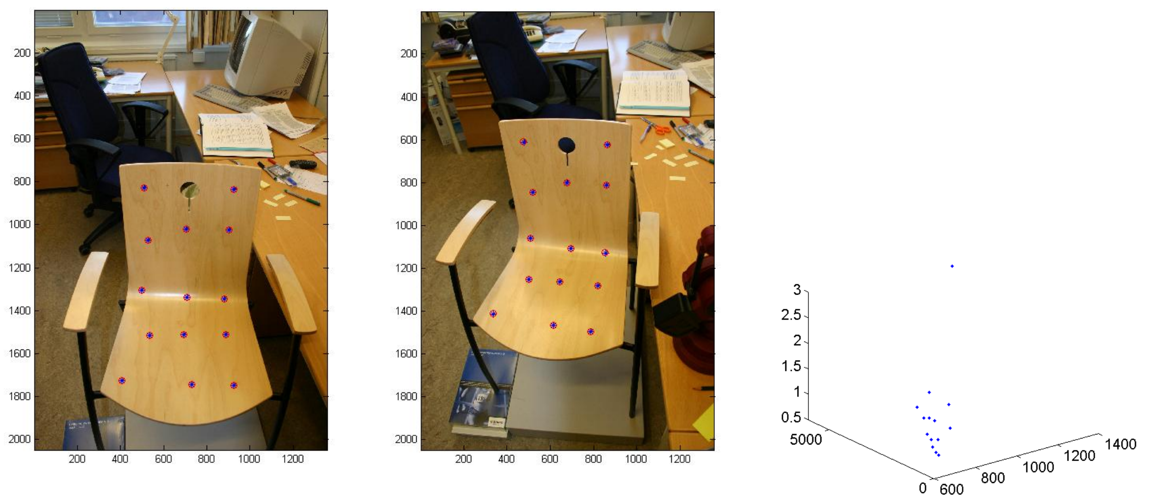


Figure 1: Two images of a chair with 14 known point correspondences. Blue $*$ are the image measurements and red o are the reprojections. The 3D points (to the right) look strange because of the projective ambiguity (note the difference in scale on the axes).

2 Relative Orientation: The Calibrated Case

When solving the relative orientation problem without camera calibration there is, as we saw in Lecture 3, an ambiguity. Basically any projective transformation can be applied to the 3D-points to give a new solution. Therefore the resulting constructions can often look strange even though the reprojections are correct (see Figure 1). To remove this ambiguity one has to add additional knowledge about the solution to the problem. For example, if we have some knowledge about the 3D scene, such as the distance between a few of the points, then we can apply a transform to the solution that make these distances correct.

Alternatively we can add knowledge about the cameras. If the inner parameters K_1 and K_2 are known we consider the calibrated two-view structure from motion problem. Given two sets of corresponding points $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$, $i = 1, ..., n$ and inner parameters K_1 and K_2 our goal is to find $[R_1 \quad t_1]$, $[R_2 \quad t_2]$ and $\mathbf{X}_i$ such that

$$\mathbf{x}_i \sim K_1[R_1 \quad t_1]\mathbf{X}_i \tag{7}$$
$$\bar{\mathbf{x}}_i \sim K_2[R_2 \quad t_2]\mathbf{X}_i, \tag{8}$$

and R_1,R_2 are rotation matrices.

We can make two simplifications to the problem. First we normalize the cameras by multiplying equations (7) and (8) with K_1^{-1} and K_2^{-1} respectively. Furthermore, we apply the euclidean transformation

$$H = \begin{bmatrix} R_1^T & -R_1^T t_1 \\ 0 & 1 \end{bmatrix} \tag{9}$$

to the cameras (and H^{-1} to the 3D points). This gives us the new cameras

$$P_1 H = [R_1 \quad t_1] \begin{bmatrix} R_1^T & -R_1^T t_1 \\ 0 & 1 \end{bmatrix} = [I \quad 0] \tag{10}$$
$$P_2 H = [R_2 \quad t_2] \begin{bmatrix} R_1^T & -R_1^T t_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \underbrace{R_2 R_1^T}_{=R} & \underbrace{-R_2 R_1^T t_1 - t_2}_{=t} \end{bmatrix}. \tag{11}$$

Therefore we search for a solution to the equations

$$\mathbf{y}_i \sim [I \quad 0]\mathbf{X}_i \tag{12}$$
$$\bar{\mathbf{y}}_i \sim [R \quad t]\mathbf{X}_i, \tag{13}$$

where $\mathbf{y}_i = K_1^{-1}\mathbf{x}_i$ and $\bar{\mathbf{y}}_i = K_1^{-1}\bar{\mathbf{x}}_i$ are the normalized image coordinates.

2.1 The Essential Matrix

The fundamental matrix for a pair of cameras of the form $[I \quad 0]$ and $[R \quad t]$ is given by

$$E = [t]_\times R, \tag{14}$$

and is called the Essential matrix. A rotation has 3 degrees of freedom and a translation 3. Since the scale of the essential matrix does not matter it has 5 degrees of freedom. The reduction in freedom compared to F, results in extra constraints on the singular values of E. In addition to having $\det(E) = 0$ the two non-zero singular values have to be equal. Furthermore, since the scale is arbitrary we can assume that these singular values are both 1. Therefore E has the SVD

$$E = U\text{diag}([1 \quad 1 \quad 0])V^T. \tag{15}$$

The decomposition is not unique. We will assume that we have a singular value decomposition where $\det(UV^T) = 1$. It is easy to ensure this; If we have an SVD as in (15) with $\det(UV^T) = -1$ then we can simply switch the sign of the last column of V. Alternatively we can switch to $-E$ which then has the SVD

$$-E = U\text{diag}([1 \quad 1 \quad 0])(-V)^T. \tag{16}$$

with $\det(U(-V)^T) = (-1)^3\det(UV^T) = 1$. Note however that if we recompute the SVD for $-E$ we might get another decomposition since it is not unique.

To find the essential matrix we can use a slightly modified 8-point algorithm. From 8 points correspondences we form the M matrix (see Lecture 6) and solve the homogeneous least squares system

$$\min_{\|v\|^2=1} \|Mv\|^2 \tag{17}$$

using SVD. The resulting vector v can be used to form a matrix $\tilde{E}$ that does not necessarily have the right singular values $1, 1, 0$. We therefore compute the decomposition $\tilde{E} = USV^T$ and construct an essential matrix using $E = U\text{diag}([1 \quad 1 \quad 0])V^T$.[1]

Since the essential matrix has only 5 degrees of freedom it is possible to find it using only 5 correspondences. However as in the case of the fundamental matrix the extra constraints are non-linear which makes estimation more difficult. (We will consider this problem in Lecture 7.)

We summarize the steps of the modified 8-point algorithm here:

- Extract at least 8 point correspondences.
- Normalize the coordinates using K_1^{-1} and K_2^{-1} where K_1 and K_2 are the inner parameters of the cameras.
- Form M and solve
$$\min_{||v||^2=1} ||Mv||^2,$$
using SVD.
- Form the matrix E (and ensure that E has the singular values $1, 1, 0$).
- Compute P_2 from E (next section).
- Compute the scene points using triangulation (see Lecture 4).

[1] Alternatively $E = U\text{diag}([1 \quad 1 \quad 0])(-V)^T$ if $\det(UV^T) = -1$.

3 Computing Cameras from E

Once we have determined the essential matrix E we need extract cameras from it. Basically we want to decompose it into $E = SR$ where S is a skew symmetric matrix and R is a rotation. We will use the two matrices

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{18}$$

The matrix W is a rotation and Z is skew symmetric. Furthermore, for these matrices we have that

$$ZW = \operatorname{diag}([1 \quad 1 \quad 0]) \tag{19}$$

$$ZW^T = -\operatorname{diag}([1 \quad 1 \quad 0]). \tag{20}$$

If E has the SVD from (15) we can now find two solutions; $E = S_1R_1$, where

$$S_1 = -UZU^T, \quad R_1 = UW^TV^T \tag{21}$$

and $E = S_2R_1$, where

$$S_2 = UZU^T, \quad R_2 = UWV^T. \tag{22}$$

To see that these are valid solutions we first verify that R_1 and R_2 are rotations. Since

$$R_1^TR_1 = (UW^TV^T)^TUW^TV^T = VWU^TUW^TV^T = I \tag{23}$$

R_1 is orthogonal. Furthermore,

$$\det(R_1) = \det(UW^TV^T) = \det(U)\det(W^T)\det(V^T) = \det(W)\det(UV^T) = 1, \tag{24}$$

and therefore R_1 is a rotation. (Note that if $\det(UV^T) = -1$ then the R_1 that we obtain is not a rotation but a rotation composed with a reflexion and therefore not a valid solution.) That S_1 is skew symmetric is easy to see since

$$-S_1^T = (UZU^T)^T = UZ^TU^T = -UZU^T = S_1. \tag{25}$$

Finally we see that

$$S_1R_1 = -UZU^TUWV^T = -UZW^TV^T = -U(-\operatorname{diag}([1 \quad 1 \quad 0]))V^T = E, \tag{26}$$

and therefore S_1R_1 is a valid decomposition. $E = S_2R_2$ can be verified similarly. Furthermore, it can be shown that these are the only two possibilities for each given E (see Hartley, Zisserman 2004).

When we have determined a decomposition $E = SR$ we need to compute a translation vector t from S such that $[t]_\times = S$. For such a t we have

$$St = [t]_\times t = t \times t = 0. \tag{27}$$

Therefore the vector t is in the null space of S. The null space of the two matrices S_1 and S_2 are the same and we can find it by looking in the third column of U.

Note that if t is in the null space of S then so is λt. In fact any non-zero λ gives a valid solution since $[\lambda t]_\times R = \lambda[t]_\times R = \lambda E$ which is also a valid essential matrix for the problem. Different λ corresponds to rescaling the solution and since there is a scale ambiguity we cannot determine a "true" value of λ. However the sign of λ is important since it determines whether points are in front of the cameras or not in the final reconstruction. To make sure that we can find a solution where the points are in front of both the cameras we therefore test $\lambda = \pm 1$.

If u_3 is the third column of U we get the four solutions

$$P_2 = \quad [UWV^T \quad u_3] \quad \text{or} \quad [UW^TV^T \quad u_3] \qquad (\text{from } \lambda = 1) \tag{28}$$

$$\text{or} \quad [UWV^T \quad -u_3] \quad \text{or} \quad [UW^TV^T \quad -u_3] \qquad (\text{from } \lambda = -1) \tag{29}$$

When we have computed these four solutions we compute the 3D points using triangulation for all the choices of P_2 and select the one with where points are in front of both P_1 and P_2. Figure 2 shows the four calibrated reconstructions obtained using the images in Figure 1. Only one of them have all the points in front of both the cameras.

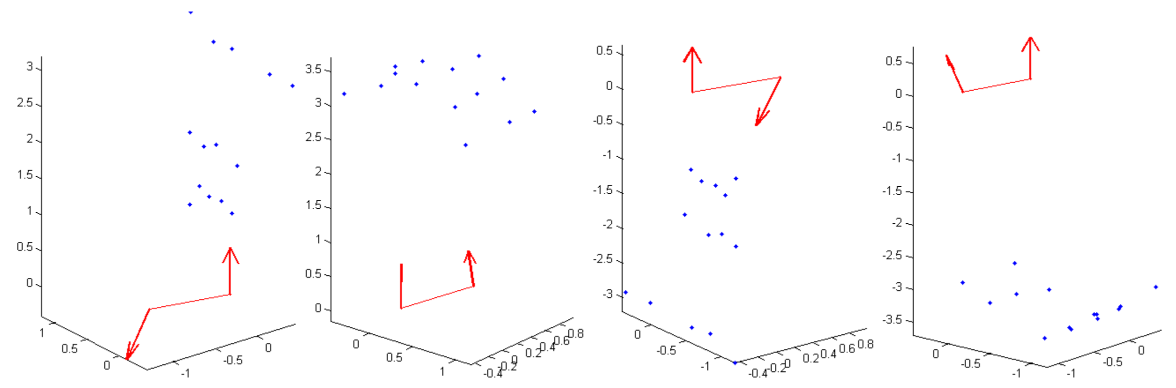


Figure 2: The 4 solutions when solving calibrated structure from motion for the chair image images in Figure 1. Only the second one have positive depths.

Lecture 7: RANSAC and Minimal Solvers

1 The Outlier Problem

In previous lectures we have studied the algebraic equations that govern projective camera systems. Under the assumption that the data given to us in the form of point correspondences is correct, we have derived algorithms for approximately solving these in the presence of moderate noise. However, since correspondences are determined automatically this will not be true in practice. A typical situation is shown in Figure 1 where correspondences between two images have been determined using SIFT descriptors. In practice we have to expect that the data contains (at least) a small portion of incorrect matches. We refer to these as outliers and the rest as inliers.

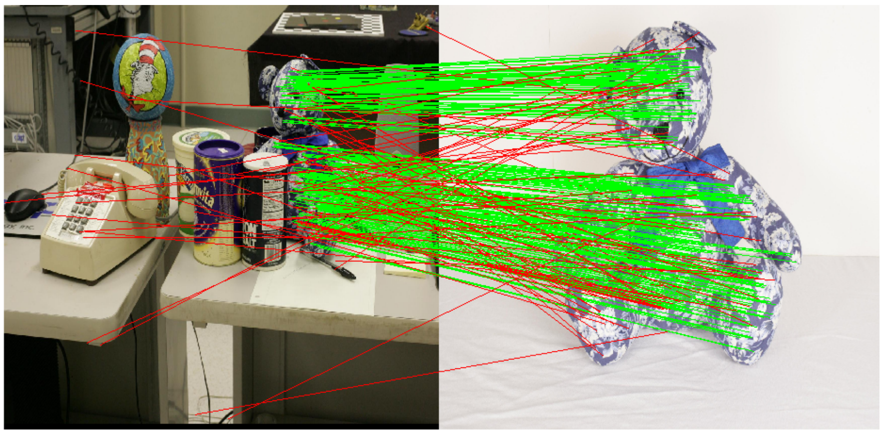


Figure 1: The Outlier Problem. When automatically detecting correspondences using descriptors such as SIFT there will always be a portion of incorrect matches. Green lines in the figure correspond to correct matches and red lines correspond to outliers.

Since the outliers typically do not fulfill the algebraic equations for the particular problem that we are trying to solve, they can severely degrade the quality of the estimation. Therefore we need a method for removing these before solving the equations.

2 RANSAC

Random sample consensus (RANSAC) is a method for removing outliers. The idea is simple; If the number of outliers is small, then if we pick a small subset of the measurements at random, we are likely to pick an outlier free set.

The outline of the algorithm is as follows:

1. Randomly select a small subset of measurements and solve the problem using only these.

2. Evaluate the error residuals for the rest of the measurements under the solution from 1. The Consensus Set for this solution is the set of measurements with error residuals less than some predefined threshold.
3. Repeat a number of times and select the solution that gives the largest consensus set.

The probability of randomly selecting a set of inliers depends on the size of the set and the proportion of inliers.

Exercise 1. Assume that we want to fit a line to a set of points. We randomly select 2 points and fit a line to these in each RANSAC iteration. Suppose that 10% of the points are outliers. How many iterations are required to find at least one set of only inliers with probability $p = 95\%$? You may assume that the set of points is large such that the portion of outliers do not change when removing a point. (Hint: First compute the probability of failure.)

Exercise 2. Same question as before but now we select 8 point correspondences to estimate a Fundamental matrix.

In practice it is a good idea to run more iterations than what is needed since, because of noise, not all inlier sets work equally well for estimating the solution. For example in the case of line estimation; if the two inlier points used to estimate the line are very close to each other then the line estimate can be very poor due to noise. Therefore the estimation may still generate a small consensus set.

3 Minimal Solvers and Solution of Polynomial Equation Systems

The more measurements we use in each RANSAC iteration the more iterations we need to run for finding good inlier sets. Therefore it is essential to use as few measurements as possible. Minimal solvers are a class of algebraic solvers that compute solutions from a minimal amount of data. In Lecture 6 we computed the Essential matrices from 8 or more point correspondences. However the essential matrix has only 5 degrees of freedom and a minimal solver for this problem therefore only uses 5 points. The 8-point algorithm is more general in that it works for any number of correspondences above 8 whereas the minimal solver only works for precisely 5. Still, in the context of a RANSAC algorithm the minimal solver is preferable.

Minimal solvers often need to find solutions to systems of non-linear equations. Next we will present a method for solving polynomial systems of equations. The idea is to transform the problem into an eigenvalue problem.

For simplicity we will first consider a system of two equations in two variables.

$$\begin{cases} x^2 - y - 3 & = 0 \\ xy - x & = 0. \end{cases} \tag{1}$$

By factoring out x from the second equation it can be seen that the system has three roots, namely $(0, -3)$, $(2, 1)$ and $(-2, 1)$.

In the following sections we will present a method for automatically finding these roots, that work under fairly general conditions. A polynomial p of degree n can represented using a monomial vector $m(x, y)$ containing all monomials of degree at most n and a coefficient vector c_p. For example, the polynomial $p(x, y) = 1 + 2x + 3y + 4x^2 + 5xy + 6y^2$ can be represented by $c_p^T m(x, y)$, where

$$c_p = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \quad \text{and} \quad m(x, y) = \begin{pmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{pmatrix}. \tag{2}$$

Using the monomials in $m(x,y)$ and a coefficient vector we can represent any second degree polynomial. The collection of monomials in $m(x,y)$ is called a monomial basis. The approach we will present is based on the observation that if we insert a root (x_0,y_0) in the monomial vector $m(x,y)$ then the resulting vector can be found by computing eigenvectors of a particular matrix.

3.1 The Action Matrix

We define T_x to be an operator that takes a polynomial $p(x,y)$ and multiplies it with x. If we apply T_x to the three monomials $1,x,y$ we get

$$\begin{aligned} 1 &\mapsto x && (3)\\ x &\mapsto x^2 && (4)\\ y &\mapsto xy. && (5)\end{aligned}$$

Now let us assume that (x_0,y_0) is a solution to (1). The result of applying T_x to the above monomials can then be simplified if we insert (x_0,y_0),

$$\begin{aligned} 1 &\mapsto x_0 && (6)\\ x_0 &\mapsto x_0^2 = y_0+3 && (7)\\ y_0 &\mapsto x_0y_0 = x_0. && (8)\end{aligned}$$

Now suppose that a first order polynomial p is given by the coefficient vector $c_p = (c_p^1, c_p^2, c_p^3)$ and monomial vector $m(x,y) = (1,x,y)$. By $q(x,y)$ we denote the result of applying T_x to $p(x,y)$. Because of the reductions (6)-(8) we get

$$q(x_0,y_0) = c_p^1x_0 + c_p^2(y_0+3) + c_p^3x_0 = 3c_p^21 + (c_p^1+c_p^3)x_0 + c_p^2y_0. \tag{9}$$

We see that if (x_0,y_0) solves (1) then because of the reductions we can represent $q(x_0,y_0)$ using the vector $m(x_0,y_0)$ and a coefficient vector c_q. The coefficient vector can be found by identifying the monomials in (9),

$$\begin{pmatrix} c_q^1\\ c_q^2\\ c_q^3\end{pmatrix} = \begin{pmatrix} 3c_p^2\\ c_p^1+c_p^3\\ c_q^3\end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 3 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{pmatrix}}_{=M_x}\begin{pmatrix} c_p^1\\ c_p^2\\ c_p^3\end{pmatrix}. \tag{10}$$

The matrix M_x is called the action matrix for the mapping T_x. Given the coefficients of $p(x,y)$ it computes the coefficients of $x_0p(x_0,y_0)$ provided that (x_0,y_0) solves the system (1). Under certain conditions it is possible to compute the roots of the system from this matrix.

3.2 Finding the Roots.

Next we will show that the roots of the system (1) are eigenvalues to the action matrix M_x. For any polynomial p we have

$$x_0p(x_0,y_0) = x_0c_p^Tm(x_0,y_0). \tag{11}$$

Furthermore,

$$x_0p(x_0,y_0) = q(x_0,y_0) = c_q^Tm(x_0,y_0) = (M_xc_p)^Tm(x_0,y_0) = c_p^TM_x^Tm(x_0,y_0). \tag{12}$$

Since this is true for any degree one polynomial (and therefore any coefficient matrix c_q of size 3×1) we must have that

$$x_0m(x_0,y_0) = M_x^Tm(x_0,y_0). \tag{13}$$

Therefore we can conclude that if (x_0,y_0) is a root of (1) then $m(x_0,y_0)$ is an eigenvector of M_x^T with eigenvalue x_0.

Exercise 3. Verify that $m(x_0, y_0)$ is an eigenvector of

$$M_x^T = \begin{pmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{14}$$

for all the three roots $(0, -3)$, $(2, 1)$ and $(-2, 1)$ of the system (1).

3.3 Algorithm

Based on the above derivations we now give an algorithm for finding the roots of a system of polynomials.

1. Select a basis of monomials.
2. Apply the mapping T_x to the monomial basis and reduce the result until the resulting expressions consists only of monomials from the basis.
3. Construct the action matrix M_x.
4. Compute eigenvalues and eigenvectors of M_x^T.
5. Extract solutions from the eigenvectors.

The theory from Section 3.2 says that the solutions will be among the eigenvectors. It does however not guarantee that there are no other eigenvectors. Therefore we might have to check the extracted solutions by inserting into the system of equations.

Furthermore, if the eigenvalues are not distinct there might be infinitely many eigenvectors to search. For example, if the x-coordinate of two of the roots are the same then M_x^T will have a double eigenvalue. If we cannot find the correct eigenvector then the eigenvalue will still give us some information, since it is not possible to have a root with x-coordinate that is not an eigenvalue M_x^T.

3.3.1 What degree of polynomials do we need?

In the above example we only considered monomials of degree 1 in (6)-(8). This worked since all the equations resulting from multiplication with x could be reduced to monomials of degree 1. Therefore we could represent both $p(x_0, y_0)$ and $x_0 p(x_0, y_0)$ with the monomial basis $1, x_0, y_0$. If this is possible or not depends on the system of equations. In general the basis has to be selected large enough so that all the reductions result in terms that are present in the basis.

The theory still holds even if we should not select the smallest possible monomial basis. For the system (1) we can consider all second degree polynomials. For example we can use the reductions:

$$\begin{aligned} 1 &\mapsto x_0 && (15)\\ x_0 &\mapsto x_0^2 && (16)\\ y_0 &\mapsto x_0 y_0 && (17)\\ x_0^2 &\mapsto x_0^3 = x_0(y_0 + 3) = x_0 y_0 + 3x_0 && (18)\\ x_0 y_0 &\mapsto x_0^2 y_0 = x_0^2 && (19)\\ y_0^2 &\mapsto x_0 y_0^2 = x_0 y_0. && (20) \end{aligned}$$

Here we made reductions so that all the terms on the right hand side have degree 2 or less. Since all the monomials on the right hand side are also present on the left hand side we can construct an action matrix from these reductions. Note that some of these terms can be reduced further. However, the theory from Section 3.2 holds regardless if we do this or not. Further reduction would result in a different action matrix.

The resulting action matrix is in this case the 6×6 matrix

$$M_x = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{21}$$

The transpose of this matrix has eigenvalues $\lambda = -2, 0, 2$ which agrees with our roots. The eigenvalue 0 does however have multiplicity four and therefore there is no unique eigenvector to this value. Hence we might not be able to find the solution $(0, -3)$ using the eigenvectors of M_x.

3.3.2 Using other mappings than T_x.

In section 3.1 we chose to construct the action matrix for multiplication with x. However, in principle any mapping $T_{q(x,y)}$ could be used. The choice of q does however affect the reductions (6)-(8). For example suppose that we use T_y instead. We get

$$1 \mapsto y_0 \tag{22}$$
$$x_0 \mapsto x_0 y_0 \tag{23}$$
$$y_0 \mapsto y_0^2 \tag{24}$$
$$x_0^2 \mapsto x_0^2 y_0 = x_0^2 \tag{25}$$
$$x_0 y_0 \mapsto x_0 y_0^2 = x_0 y_0 \tag{26}$$
$$y_0^2 \mapsto y_0^3 = y_0^2(x_0^2 - 3) = x_0^2 - 3y_0^2. \tag{27}$$

Here it does not seem possible to use only 1st order monomials since the degree of y_0^2 can not be reduced further using the equations in (1). (It is however possible to generate new equations from (1) that can be used for further reduction. However this is more complicated and we do not pursue this further.)

The resulting action matrix is in this case the 6×6 matrix

$$M_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -3 \end{pmatrix}. \tag{28}$$

The transpose of this matrix has eigenvalues $-3, 1, 0$. Since there are two roots with y coordinate 1 the eigenvalue 1 will have at least multiplicity two. Therefore we can only extract the solution to $(0, -3)$ from this eigenspace.

A simple heuristic for generating action matrices with more distinct eigenvalues is to use a mapping $T_{q(x,y)}$ where $q(x, y)$ is a random combination of x and y. For example $0.5M_x^T + 0.5M_y^T$ (with M_x and M_y from (21) and (28) respectively) has five distinct eigenvalues.

Another trick that modifies the eigenspace is to drop some of the monomials. In (28) rows 1 and 2 are all zeros. This means that 1 and x_0 do not occur in any of the expressions after the reductions. Therefore we can remove these from the system. The new action matrix is

$$M_y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -3 \end{pmatrix}. \tag{29}$$

Note however, that the monomial vector is now $m(x, y) = (y, x^2, xy, y^2)$ and therefore the eigenvector will not contain the value of x_0.

4 The 5-point solver

In this section we will construct a minimal solver for the problem of finding an Essential matrix. Given 5 point correspondences we will use the following equations:

$$\bar{\mathbf{x}}_i^T E \mathbf{x}_i = 0, \qquad i = 1, ..., 5. \tag{30}$$

$$\det(E) = 0, \tag{31}$$

$$2EE^TE - \text{trace}(EE^T)E = 0. \tag{32}$$

The third constraint (32) actually consists of 9 polynomial equations since it is a matrix expression. Any matrix E that has a singular value decomposition of the form

$$E = U \underbrace{\begin{pmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{=S} V^T \tag{33}$$

will fulfill this constraint. This can be seen by inserting (33) into (32). Furthermore, it can be shown that any matrix that fulfills (32) must have a singular value decomposition as in (33).

To find the solutions of (30)-(32) we will first use (30) to reduce the number of variables. We construct an M matrix of size 5×9 from the 5 epipolar constraints, similar to what we did in Lecture 6. In contrast to the eight point algorithm, the M matrix is in itself not enough to determine all the 8 parameters of the essential matrix since it only represents 5 equations. Since the dimension of the nullspace of M is 4 we can find 4 linearly independent vectors v_i, $i = 1, ..., 4$ such that

$$M(\alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + \alpha_4 v_4) = 0, \tag{34}$$

for any choice of coefficients $\alpha_1, ..., \alpha_4$. Reshaping these vectors into matrices we get

$$\bar{\mathbf{x}}_i^T(\alpha_1 E_1 + \alpha_2 E_2 + \alpha_3 E_3 + \alpha_4 E_4)\mathbf{x}_i = 0, \quad i = 1, ..., 5. \tag{35}$$

What remains is to find the coefficients $\alpha_1, ..., \alpha_4$ such that (31) and (32) are fulfilled. Note that since the scale of the essential matrix is arbitrary, we can assume that (for example) $\alpha_1 = 1$.

To determine the coefficients $\alpha_2, ..., \alpha_4$ we will use the method presented in the previous section. Equation (32) consists of 9 third order polynomials in $\alpha_2, \alpha_3, \alpha_4$. In addition we have (31) which consists of 1 third degree polynomial. To construct the action matrix we first need to compute the coefficients of these polynomials. These can easily be determined by rewriting (32)

$$2EE^TE - \text{trace}(EE^T)E = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} \alpha_i\alpha_j\alpha_k \left(2E_iE_j^TE_k - \text{trace}(E_iE_j^T)E_k\right). \tag{36}$$

For each of the 9 constraints in (32) we can extract coefficients for the monomial $\alpha_i\alpha_j\alpha_k$ using this expression. Note that the monomials occur several times in this sum. For example, $(i, j, k) = (1, 1, 2)$ and $(i, j, k) = (1, 2, 1)$ both yield $\alpha_i\alpha_j\alpha_k = \alpha_1^2\alpha_2 = \alpha_2$. There are 64 terms in the sum but only 20 distinct monomials. The determinant constraint can be handled in the same way using the expression

$$\begin{aligned}\det(E) = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} \alpha_i\alpha_j\alpha_k \big(& e_{11}^i e_{22}^j e_{33}^k + e_{12}^i e_{23}^j e_{31}^k + e_{13}^i e_{21}^j e_{32}^k \\ & - e_{11}^i e_{23}^j e_{32}^k - e_{12}^i e_{21}^j e_{33}^k - e_{13}^i e_{22}^j e_{31}^k\big),\end{aligned} \tag{37}$$

where e_{ab}^i is element (a, b) in matrix E_i. In summary we construct a 10×20 matrix that contains the coefficients of all the 10 polynomials. The columns of this matrix correspond to the coefficients of the monomials:

$$\{\alpha_4^3, \alpha_3\alpha_4^2, \alpha_3^2\alpha_4, \alpha_3^3, \alpha_2\alpha_4^2, \alpha_2\alpha_3\alpha_4, \alpha_2\alpha_3^2, \alpha_2^2\alpha_4, \alpha_2^2\alpha_3, \alpha_2^3, \alpha_4^2, \alpha_3\alpha_4, \alpha_3^2, \alpha_2\alpha_4, \alpha_2\alpha_3, \alpha_2^2, \alpha_4, \alpha_3, \alpha_2, 1\}. \tag{38}$$

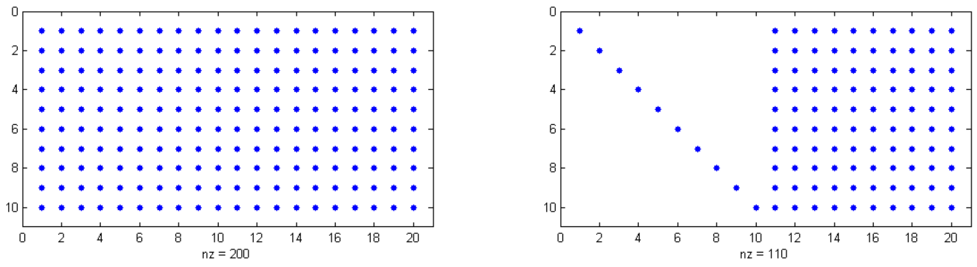


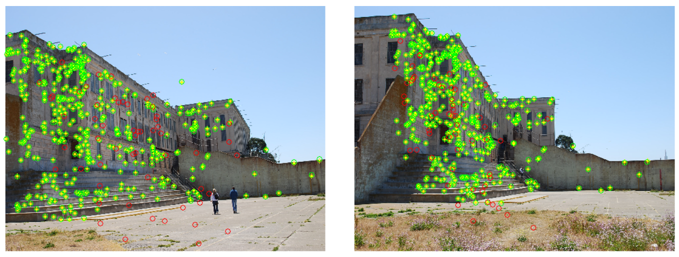
Figure 2: Shape of the 10×20 coefficient matrix before (left) and after elimination (right). A '*' means the element is non-zero.

The 10 first monomials are of degree 3 and the rest are of lower degree. If we modify the coefficient matrix by performing Gaussian elimination we can determine reductions for each of these terms, see Figure 2. For example, after elimination, the first row of the matrix only contains α_4^3 from the third order monomials and can therefore be used to replace this term with lower order terms. To create an action matrix we use the 10 equations represented by the new coefficient matrix to compute reductions of all the third order monomials. In this case it does not matter if we use T_{α_2}, T_{α_3} or T_{α_4}, since reductions to second order or less for all third order monomials are availible in the modified coefficient matrix.

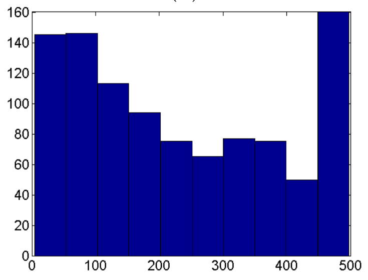
In summary the 5-point solver consists of the following steps:

1. Construct the 5×9 matrix M from the 5 point correspondences. (Note that the image points should be normalized as in Lecture 6.)
2. Compute the 4 vectors that span the nullspace of M and reshape them to the matrices E_1, E_2, E_3, E_4.
3. Using the expressions (36) and (37) compute coefficients for all monomials and construct the 10×20 coefficient matrix. (The monomial order does not have to be the same as (38), however the first 10 terms needs to be the 3rd order monomials.)
4. Perform Gaussian elimination on the coefficient matrix.
5. Construct the action matrix for either T_{α_2}, T_{α_3} or T_{α_4} using the reductions available in the modified coefficient matrix.

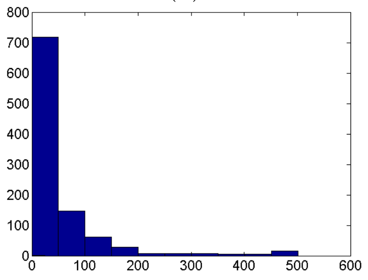
Figure 3 shows a comparison between the 5-point solver and the 8-point solver. For the pair of images depicted in the first row of the figure we apply a 1000 iterations of RANSAC. In the histograms on the second row we plot the size of the consensus set in each iteration. It can be seen that the 5-point solver generally finds consensus sets with a larger number of inliers.



(a)



(b)



(c)

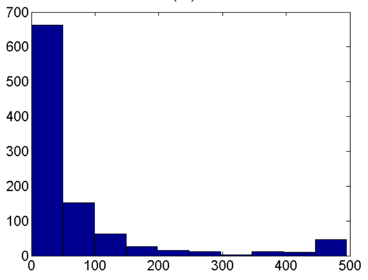


Figure 3: First row: The best solution obtained from the 5-point solver. Yellow '*' is an image point, green 'o' is the reprojection of an inlier and red 'o' is a reprojection of an outlier. Second row: Histogram over the size of the consensus set in each iteration of a 1000-iteration-RANSAC, using (a) - 5 points, (b) - 8 points and (c) - 10 points.

Lecture 8: Model Fitting

1 Noise Models

In Lecture 7 we studied the RANSAC approach for removing incorrect point correspondences. Once these outliers have been removed we still have to deal with noise in the measurements. Since the appearance of a patch changes when the viewpoint changes, exact positioning of corresponding points is not possible, see Figure 1,. Therefore, our point measurements will always be corrupted by noise of various forms and levels.

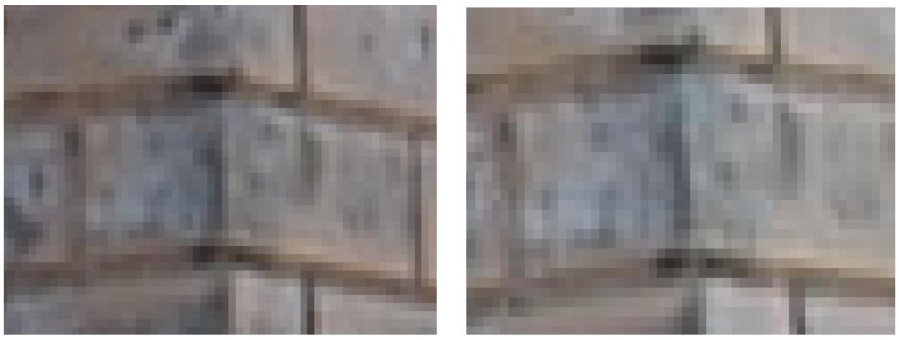


Figure 1: Two patches extracted from images with slightly different viewpoint. Exact localization of corresponding points is made difficult because of slight appearance differences and limited image resolution.

In Lectures 3,4,5 and 6 we solved various problems using linear formulations for approximately solving the governing algebraic equations. While this is an easy approach it does in general not give the "best" possible fit to the data. In this lecture we will derive formulations that gives the "best" fit under the assumption of Gaussian noise. The resulting problems are in general more difficult to solve than the formulations that we have used previously. In many cases they can only be locally optimized. Therefore the linear approaches are still very useful since they provide an easy way of creating a starting solution.

2 Line Fitting

What is meant by the "best" fit depends on the particular noise model. In this section we will consider two different noise models and show that they lead to different optimization criteria. For simplicity we will consider the problem of line fitting since this leads to closed form solutions.

2.1 Linear Least Squares

Suppose that (x_i, y_i) are measurements of 2D-points belonging to a line $y = ax + b$. Furthermore, we assume that y_i is corrupted by Gaussian noise, that is,

$$y_i = \tilde{y}_i + \epsilon_i \tag{1}$$

where $\epsilon_i \in \mathcal{N}(0,1)$ (Gaussian noise with mean 0 and standard deviation 1) and $\tilde{y}_i$ is the true y-coordinate. Our goal is to estimate the line parameters a and b for which the measurements y_i are most likely. Since $\epsilon_i \in \mathcal{N}(0,1)$, its probability density function is

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}} e^{-\epsilon_i^2/2}. \tag{2}$$

Furthermore, if we assume that the ϵ_i, $i = 1, ..., n$ are independent of each other then their joint distribution is

$$p(\epsilon) = \prod_{i=1}^{n} p(\epsilon_i), \tag{3}$$

where $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_n)$. Since $\epsilon_i = y_i - \tilde{y}_i$ we can compute the likelihood of the measurements by

$$p(\epsilon) = \prod_{i=1}^{n} p(\epsilon_i) = \prod_{i=1}^{n} p(y_i - \tilde{y}_i) = \prod_{i=1}^{n} p(y_i - (ax_i + b)) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-(y_i - (ax_i + b))^2/2}. \tag{4}$$

We now want to find the the line parameters a and b that make these measurements most likely. To simplify the maximization we maximize the logarithm of the likelihood

$$\log\left(\prod_{i=1}^{n} p(\epsilon_i)\right) = -\sum_{i=1}^{n} \frac{(y_i - (ax_i + b))^2}{2} + \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}}\right). \tag{5}$$

Since the second term does not depend on a of b this is the same as minimizing

$$\sum_{i=1}^{n} (y_i - (ax_i + b))^2. \tag{6}$$

In matrix form we can write this as

$$\left\| \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{=A} \begin{pmatrix} a \\ b \end{pmatrix} - \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{=B} \right\|^2 . \tag{7}$$

The minimum of this expression can be computed using the normal equations

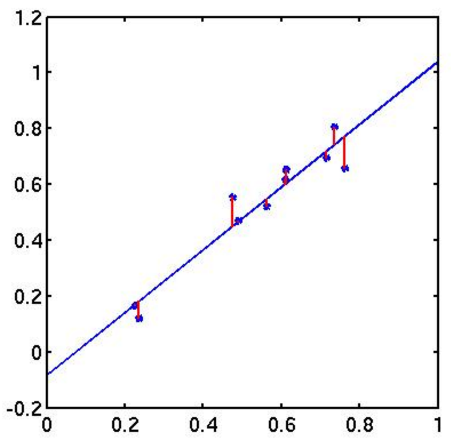
$$\begin{pmatrix} a \\ b \end{pmatrix} = (A^T A)^{-1} A^T B, \tag{8}$$

which we will derive in Lecture 9. The geometric interpretation of (6) is that under this noise model the vertical distance between the line and the measurement should be minimized, see Figure 2.

2.2 Total Linear Least Squares

Next we will assume that we have noise in both coordinates, that is,

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} + \delta_i, \tag{9}$$



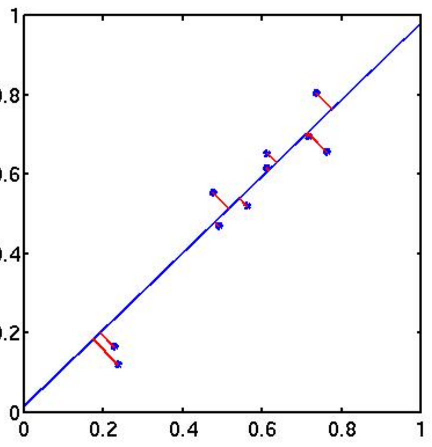


Figure 2: Left: The vertical distances between the line and the measured points are minimized in (6). In contrast, the minimal distances between the line and the measured points are minimized in (12).

where $\delta_i \in \mathcal{N}(0, I)$ and $a\tilde{x}_i + b\tilde{y}_i = c$. The δ_i now belong to a two dimensional normal distribution with probability density function

$$p(\delta_i) = \frac{1}{2\pi} e^{-\|\delta_i\|^2/2}. \tag{10}$$

The log likelihood function is

$$\sum_{i=1}^{n} \log(p(\delta_i)) = -\sum_{i=1}^{n} \frac{(x_i - \tilde{x}_i)^2 + (y_i - \tilde{y}_i)^2}{2} + \sum_{i=1}^{n} \log(\frac{1}{2\pi}). \tag{11}$$

Therefore, to maximize the likelihood we need to minimize

$$\sum_{i=1}^{n} ((x_i - \tilde{x}_i)^2 + (y_i - \tilde{y}_i)^2), \tag{12}$$

where $a\tilde{x}_i + b\tilde{y}_i = c$. The point $(\tilde{x}_i, \tilde{y}_i)$ can be any point on the line, however since we are minimizing (12) we can restrict it to be the closest point on the line. The expression (12) then becomes the distance between (x_i, y_i) and the line. This distance can be expressed using the distance formula as

$$\frac{|ax_i + by_i + c|}{a^2 + b^2}. \tag{13}$$

Without loss of generality we ca assume that $a^2 + b^2 = 1$, and therefore we need to solve

$$\min \quad \sum_{i=1}^{n} (ax_i + by_i + c)^2 \tag{14}$$

$$s.t. \quad a^2 + b^2 = 1. \tag{15}$$

This problem is often referred to as the total linear least squares problem.

2.2.1 Solving the Total Least Squares Problem

To solve (14),(15) we first take derivatives with respect to c. This shows that the the optimal solution must fulfill

$$c = -(a\bar{x} + b\bar{y}), \tag{16}$$

where $\bar{x}$ and $\bar{y}$ are the mean values

$$(\bar{x}, \bar{y}) = \frac{1}{n}\sum_{i=1}^{n}(x_i, y_i). \tag{17}$$

Substituting into (14) we get

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}(a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 & (18)\\ \text{such that} \quad & 1 - (a^2 + b^2) = 0. & (19)\end{aligned}$$

By forming the matrix

$$M = \sum_{i=1}^{m}\begin{pmatrix} (x_i - \bar{x})^2 & (x_i - \bar{x})(y_i - \bar{y}) \\ (x_i - \bar{x})(y_i - \bar{y}) & (y_i - \bar{y})^2 \end{pmatrix}, \tag{20}$$

we can write this problem as

$$\begin{aligned} \min \quad & t^T M t & (21)\\ \text{such that} \quad & 1 - t^T t = 0, & (22)\end{aligned}$$

where t is a 2×1 vector containing a and b. This is a constrained optimization problem of the type

$$\begin{aligned} \min \quad & f(t) & (23)\\ \text{such that} \quad & g(t) = 0. & (24)\end{aligned}$$

According to Persson-Böiers, "Analys i flera variabler" and the method of Lagrange multipliers the solution of such a system has to fulfill

$$\nabla f(t) + \lambda \nabla g(t) = 0. \tag{25}$$

Therefore the solution of (21)-(22) must fulfill

$$2Mt + \lambda(-2t) = 0 \Leftrightarrow Mt = \lambda t. \tag{26}$$

That is, the solution t has to be an eigenvector of the matrix M. Furthermore, inserting into (21), and using that $t^T t = 1$ we see that it has to be the eigenvector corresponding to the smallest eigenvalue.

3 The Maximum Likelihood Solution for Camera Systems

In this section we derive the maximum likelihood estimator for our class projection problems. Suppose the 2D-point $x_{ij} = (x_{ij}^1, x_{ij}^2)$ is a projection in regular Cartesian coordinates of the 3D-point $\mathbf{X}_j$ in camera P_i. The projection in regular coordinates can be written

$$\left(\frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j}, \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j}\right), \tag{27}$$

where P_i^1, P_i^2, P_i^3 are the rows of the camera matrix P_i. Also we assume that the observations are corrupted by Gaussian noise, that is,

$$(x_{ij}^1, x_{ij}^2) = \left(\frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j}, \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j}\right) + \delta_{ij}, \tag{28}$$

and δ_{ij} is normally distributed with covariance I. The probability density function is then

$$p(\delta_{ij}) = \frac{1}{2\pi}e^{-\frac{1}{2}||\delta_{ij}||^2}. \tag{29}$$

Similarly to Section 2.2 we now see that the model configuration that maximizes the likelihood of the obtaining the observations $x_{ij} = (x_{ij}^1, x_{ij}^2)$ is obtained by solving

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \left(x_{ij}^1 - \frac{P_i^1 \mathbf{X}_j}{P_i^3 \mathbf{X}_j},\ x_{ij}^2 - \frac{P_i^2 \mathbf{X}_j}{P_i^3 \mathbf{X}_j} \right) \right\|^2 . \tag{30}$$

where n is the number of cameras and m is the number of scene points. The geometric interpretation of the above expression is that the distance between the projection and the measured point in the image should be minimized, see Figure 3. Note that it does not matter which of the variables P_i and X_i we consider as unknowns, it is always the reprojection error that should be minimized.

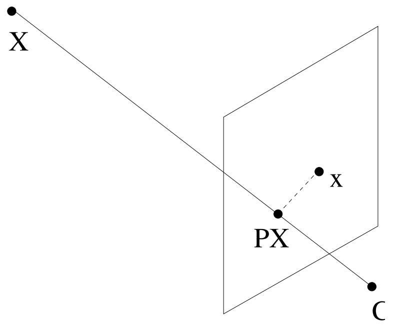


Figure 3: Geometric interpretation of the maximum likelihood estimate for projection problems. The dashed distance should be minimized.

3.1 Affine Cameras

In general the maximum likelihood estimator (30) can only be solved using local iterative methods. However in the special case of affine cameras there is a closed form solution. An affine camera is a camera where the third row of P, $P^3 = [0\ 0\ 0\ t_3]$. Since the scale of the camera matrix is arbitrary we may assume that $t_3 = 1$, and therefore the camera matrix has the form

$$P = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix}, \tag{31}$$

where A is a 2×3 matrix and t is a 2×1 vector. If we use regular Cartesian coordinates for both image points and scene points the camera equations can be simplified. If x_{ij} is the projection of the scene point X_j in the affine cameras P_i then the projection can be written

$$x_{ij} = A_i X_j + t_i. \tag{32}$$

To find he maximum likelihood estimate we therefore need to solve

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \| x_{ij} - A_i X_j + t_i \|^2. \tag{33}$$

Dy differentiating with respect to t_i it can be seen that the optimal t_i is given by

$$t_i = \bar{x}_i - A_i \bar{X},$$

where $\bar{X} = \frac{1}{m} \sum_j X_j$ and $\bar{x}_i = \frac{1}{m} \sum_j x_{ij}$. To simplify the problem we therefore change coordinates so that all these mean values are zero by translating all image points and scene points. Using $\tilde{x}_{ij} = x_{ij} - \bar{x}_i$ and $\tilde{X}_i = X_i - \bar{X}$, gives the simplified problem

$$\min \sum_{ij} ||\tilde{x}_{ij} - A_i \tilde{X}_j||^2. \tag{34}$$

In matrix form we can write this as

$$\min \left\| \underbrace{\begin{bmatrix} \tilde{x}_{11} & \tilde{x}_{12} & \dots & \tilde{x}_{1m} \\ \tilde{x}_{21} & \tilde{x}_{22} & \dots & \tilde{x}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{x}_{n1} & \tilde{x}_{n2} & \dots & \tilde{x}_{nm} \end{bmatrix}}_{M} - \underbrace{\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} \begin{bmatrix} \tilde{X}_1 & \tilde{X}_2 & \dots & \tilde{X}_m \end{bmatrix}}_{\text{rank 3 matrix}} \right\|^2 . \tag{35}$$

Since the A_i has only 3 columns the second term will be rank 3 matrix. Thus our problem is to find the matrix of rank 3 that best approximates M. The best approximating matrix can be found by computing the SVD of M and setting all but the first 3 singular values to zero.

We summarize the algorithm for affine cameras here:

1. Re-center all images so that the center of mass of the image points is zero in each image.
2. Form the measurement matrix M.
3. Compute the SVD:
$$M = USV^T. \tag{36}$$
4. A solution can be found by extracting the cameras from $U(:, 1:3)$ and the structure from $S(1:3, 1:3) * V(:, 1:3)'$.
5. Transform back the solution to the original image coordinates.

Note that the approach only works when all points are visible in all images. Furthermore, the camera model is affine, which is a simplification. This is often a good approximation when the scene point have roughly the same depth.