Lecture Notes in Computer Vision

Carl Olsson
carl.olsson@math.lth.se
caols@chalmers.se

January 16, 2022

Contents

Lecture 1:

The Pinhole Camera Model

1.1 Introduction to Structure from Motion

The main goal of Computer Vision is to understand the world from images. One of the classical problems is the so called Structure from Motion problem. Given a collection of images of a scene the goal is to estimate the positions of the cameras that captured the images (the motion) and create a 3D model of the depicted scene (the structure). Figure 1.1 shows an example of such a model. The main goal of this course is to develop methods for solving these types of reconstruction problems.

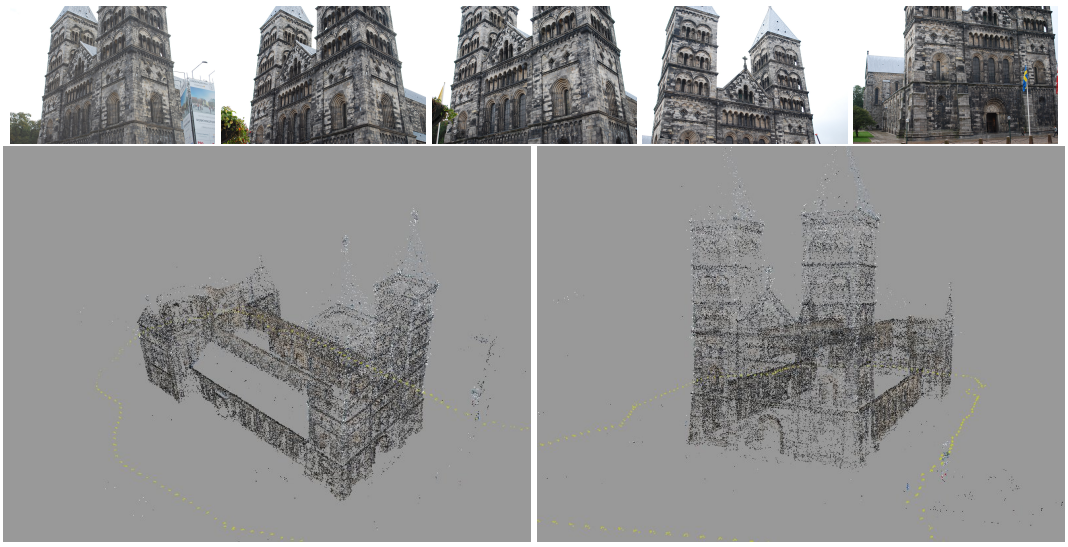


Figure 1.1: An illustration of the Structure from Motion problem. The top row shows 5 images from a collection of 1208 used to reconstruct the scene. The bottom row shows the reconstruction from two different view points.

The scene model in Figure 1.1 is a 3D point cloud consisting of points whose projections are visible in a subset of the images. Figure 1.2 illustrates how these projections are obtained. First point detectors are used to generate a set of 2D points that can potentially be matched to other images (Fig. 1.2(b)). This is followed by a matching step where the local texture in patches around the points is compared to determine which points are projections of the same 3D point. For details on point descriptors and the matching process we refer to the course in image analysis. Here we will assume that the matching process as been completed and we will focus on determining the structure and motion from the obtained projections.

(a) (b) (c)

Figure 1.2: Point detection and matching using the SIFT descriptor.

1.2 A Mathematical Projection Model

The most commonly used camera model, which we will also use in the course, is the so called pinhole camera. The model is inspired by the simplest cameras. It has the shape of a box, light from an object enters though a small hole (the pinhole) in the front and produces an image on the back camera wall (see Figure 1.3).

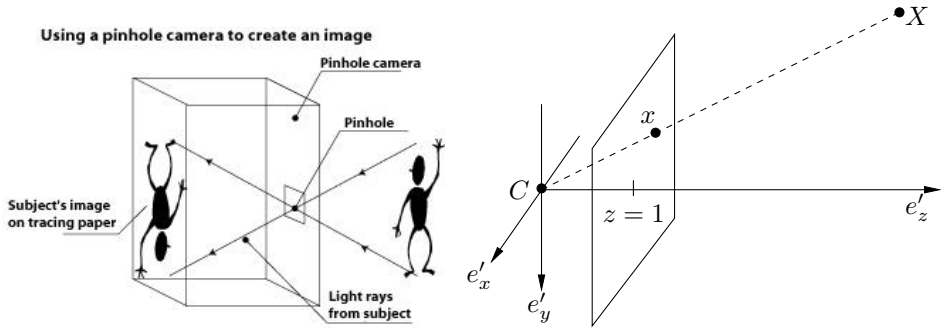


Figure 1.3: The Pinhole camera (left), and a mathematical model (right).

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 1.4) we can parametrize it by the expression

$$C + s(X - C) = sX, \quad s \in \mathbb{R}. \tag{1.1}$$

To find the intersection between this line and the image plane $z = 1$ we need to find and 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

$$x = \begin{pmatrix} X'_1/X'_3 \\ X'_2/X'_3 \\ 1 \end{pmatrix} \tag{1.2}$$

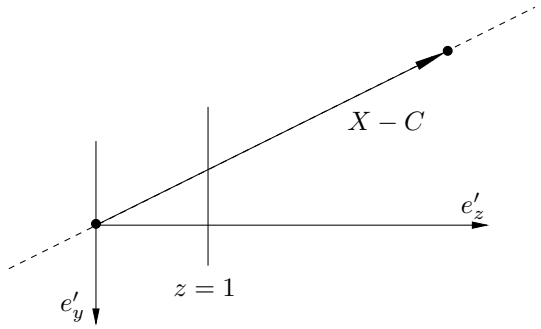


Figure 1.4: The model viewed from the side. (The vector e'_x points out of the figure.)

Exercise 1. Compute the image of the cube with corners in $(\pm 1, \pm 1, 2)$ and $(\pm 1, \pm 1, 4)$, see Figure 1.5.

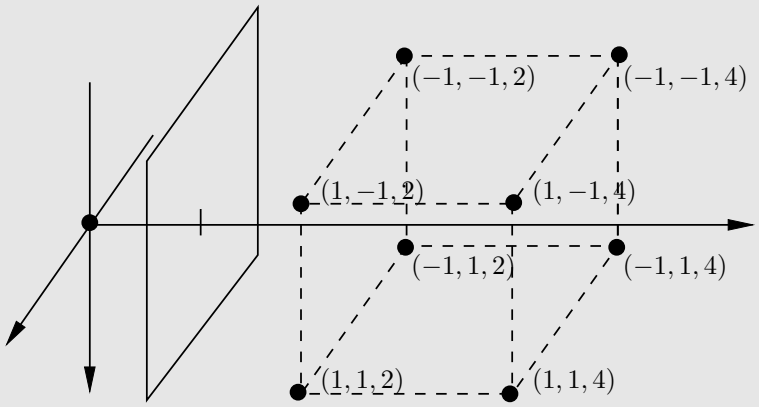


Figure 1.5: The Cube in the camera coordinate system.

1.3 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 1.6. 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

$$\begin{pmatrix} X'_1 \\ X'_2 \\ X'_3 \end{pmatrix} = R \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} + t. \tag{1.3}$$

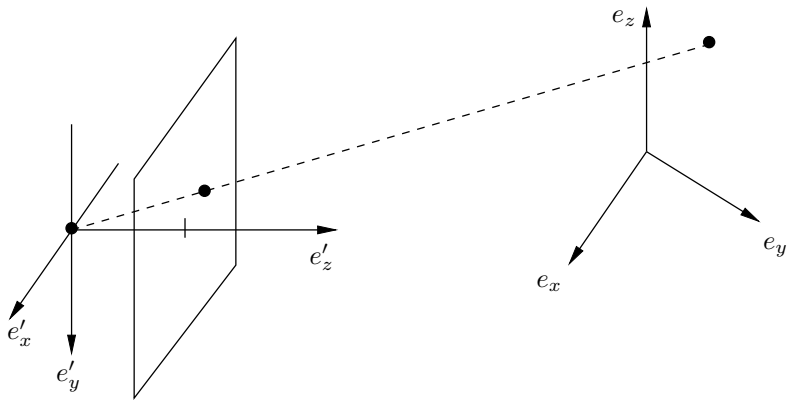


Figure 1.6: New global coordinate system.

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?

If we add an extra 1 to the scene point X we can write (1.3) in matrix form

$$X'_3 \begin{pmatrix} X'_1/X'_3 \\ X'_2/X'_3 \\ 1 \end{pmatrix} = \begin{pmatrix} X'_1 \\ X'_2 \\ X'_3 \end{pmatrix} = [R \quad t] \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}. \tag{1.4}$$

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 (1.2). Therefore we conclude from (1.4) 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 camera

$$P = \frac{1}{3} \begin{pmatrix} 1 & 2 & 2 & 0 \\ 2 & 1 & -2 & 0 \\ -2 & 2 & -1 & 1 \end{pmatrix}. \tag{1.5}$$

What other points give the same projections?

1.4 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 (1.4) the third coordinate is given by

$$X'_3 = [R_3 \quad t_3] \begin{bmatrix} X \\ 1 \end{bmatrix}, \tag{1.6}$$

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 of v is positive.

Exercise 4. Which of the points that project to $(-1, 1)$ in the camera P in Exercise 3 are in front of the camera?

Lecture 2:

Intrinsic Parameters and Projective Geometry

2.1 The Inner Parameters of the Pinhole Camera

In the previous lecture we presented the pinhole camera model. Here the image plane was 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 was be located in $(0, 0, 1)$, and therefore has 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 transform 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 2.1.

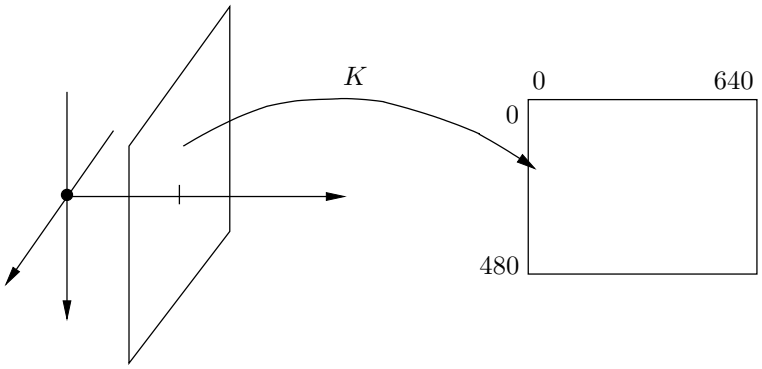


Figure 2.1: The mapping K from the image plane to the real image.

The mapping is represented by an invertible triangular 3×3 matrix K of the form

$$K = \begin{bmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

This matrix contains what is usually referred to as the **inner parameters** of the camera, that is, focal length f, principal point (x_0, y_0) etc. We will discuss this further in Lecture 4, for now we only mention that the diagonal elements of K are all positive implying that its determinant is also positive. The projection (reference coordinate

system to real image) is now given by

$$\lambda \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}}_{=\mathbf{x}} = \underbrace{K \begin{bmatrix} R & t \end{bmatrix}}_{=P} \underbrace{\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}}_{=\mathbf{X}}, \tag{2.2}$$

or in matrix form

$$\lambda \mathbf{x} = P\mathbf{X}. \tag{2.3}$$

Equation (2.3) is usually called the **camera equations** and the 3×4 matrix P is called the **camera matrix**.

Exercise 5. Compute the projection of $X = (0, 0, 1)$ in the cameras

$$P_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 1 \end{pmatrix} \text{ and } P_2 = \begin{pmatrix} -1 & 0 & 1 & 0 \\ 0 & -\sqrt{2} & 0 & 0 \\ -1 & 0 & -1 & -\sqrt{2} \end{pmatrix}. \tag{2.4}$$

What are the camera centers of these cameras?

Remark 1. Since we allways divide with the third coordinate when computing projections rescaling of a camera matrix P makes no difference. In the example above the two camera matrices P_1 and P_2 represent the same camera and give the same projections. The same is true if we rescale the vector $\mathbf{X}$. This observation motivates the use of homogeneous coordinates that are often used in projective geometry.

2.2 Homogeneous Coordinates

As we have seen the interpretation of equation (2.3) 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

$$\mathbf{x} \sim \mathbf{y}. \tag{2.5}$$

The two vectors are said to be representing 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 $(2, 3)$.

Note that in Lecture 1 we interpreted the result of the multiplication

$$v = \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = RX + t, \tag{2.6}$$

as a 3D point in the camera coordinate system. We can now interpret the results as homogeneous coordinates of the projection. To interpret the result as a regular we need to divide with the third coordinate.

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

Similarly the 4×1 vector $\mathbf{X} = \begin{bmatrix} X \\ 1 \end{bmatrix}$ can be seen as a homogeneous representative of a point in $\mathbb{R}^3$, which has regular coordinates X. Any re-scaled version of this vector $\begin{bmatrix} \gamma X \\ \gamma \end{bmatrix}$, $\gamma \neq 0$ represents the same 3D point. In addition

the projection is unaffected by the scale since

$$\begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} \lambda X \\ \gamma \end{bmatrix} = \gamma v \sim v. \tag{2.7}$$

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 $\mathbb{R}^{n+1}$, and if coordinate $n+1$ is not zero then we can interpret them as points of $\mathbb{R}^n$ by dividing with this coordinate.

2.3 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

$$ax + by + c = 0, \tag{2.8}$$

where $(a, b, c) \neq (0, 0, 0)$. For a point $\mathbf{x} \sim (x, y, z)$ in $\mathbb{P}^2$ we instead use the modified formula

$$ax + by + cz = 0. \tag{2.9}$$

If x is a regular point with Cartesian coordinates (x, y), that is, $\mathbf{x} \sim (x, y, 1)$ then it is clear that the two equations (2.8) and (2.9) give the same result. Note that (2.8) 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

$$a\lambda x + b\lambda y + c\lambda = \lambda(ax + by + c) = 0, \tag{2.10}$$

and therefore (2.9) works with and homogeneous representative for $\mathbf{x}$. 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

$$ax + by + cz + d = 0 \tag{2.11}$$

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 6. 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 7. 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.)

2.4 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 2.2.

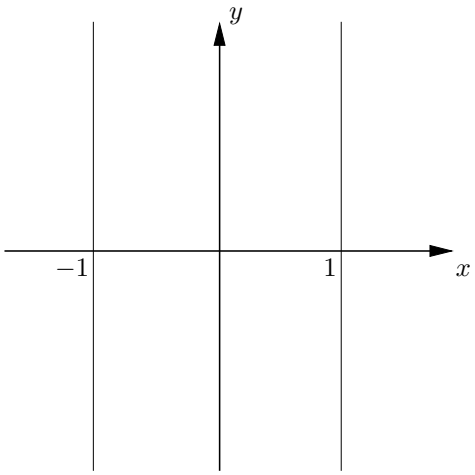


Figure 2.2: The two lines $(-1, 0, 1)$ and $(1, 0, 1)$.

If $\mathbf{x}$ lies on both the lines it is a solution the system of equations

$$\begin{cases} \mathbf{l}_1^T\mathbf{x} &= 0 \\ \mathbf{l}_2^T\mathbf{x} &= 0 \end{cases} \Leftrightarrow \begin{cases} x + z &= 0 \\ -x + z &= 0 \end{cases} . \tag{2.12}$$

Since we have no constraints for y we get

$$\begin{cases} x + z &= 0 \\ 2z &= 0 \\ y &= t \end{cases} \Leftrightarrow \begin{cases} x &= 0 \\ y &= t \\ z &= 0 \end{cases} . \tag{2.13}$$

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.

Lecture 3:

Projective Transformations and Camera Models

3.1 Projective Transformations

In Lecture 2 we introduced homogeneous coordinates and the projective spaces $\mathbb{P}^n$. Each point in $\mathbb{P}^n$ is represented with a vector in $\mathbb{R}^{n+1}\backslash\{0\}$. In addition two vectors $\mathbf{x}$ and $\mathbf{y}$ represent the same point if $\mathbf{x} = \lambda\mathbf{y}$, $\lambda \neq 0$, in which case we say that $\mathbf{x} \sim \mathbf{y}$.

Next we will consider mappings between these spaces that can be represented with matrices. A **projective transformation** is an invertible mapping $\mathbb{P}^n \mapsto \mathbb{P}^n$ defined by

$$\mathbf{x} \sim H\mathbf{y} \tag{3.1}$$

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 8. 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 2 we saw one example of such a mapping, namely the K-matrix. In the camera equation

$$\mathbf{x} \sim K\begin{bmatrix} R & t \end{bmatrix}\mathbf{X} \tag{3.2}$$

the matrix K transforms the point $\begin{bmatrix} R & t \end{bmatrix}\mathbf{X}$ in the image plane to the real image coordinate system (with the unit pixels).

Another example is point transfer via a plane. Figure 3.1 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 (3.1), 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 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

$$\lambda_i\mathbf{x}_i = H\mathbf{y}_i \tag{3.3}$$

(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

$$3n \geq 8 + n \Leftrightarrow 2n \geq 8 \Leftrightarrow n \geq 4 \tag{3.4}$$

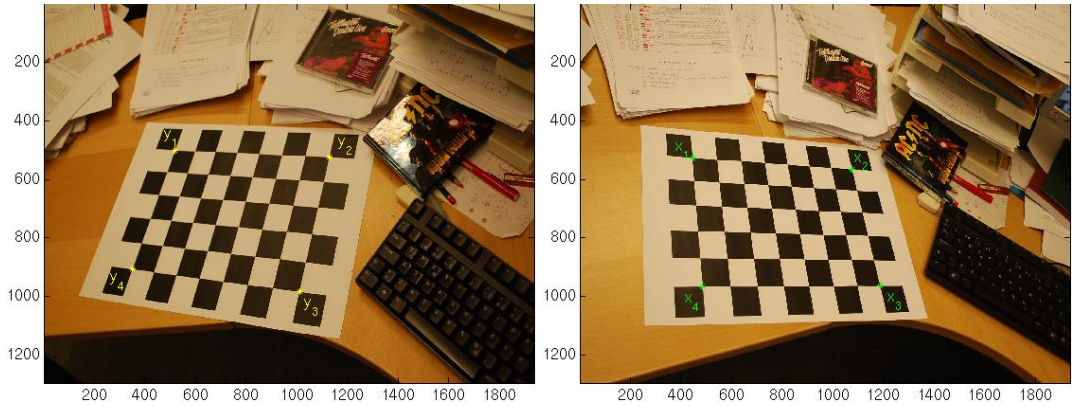


Figure 3.1: Two images of a roughly planar surface with 4 detected point correspondences.

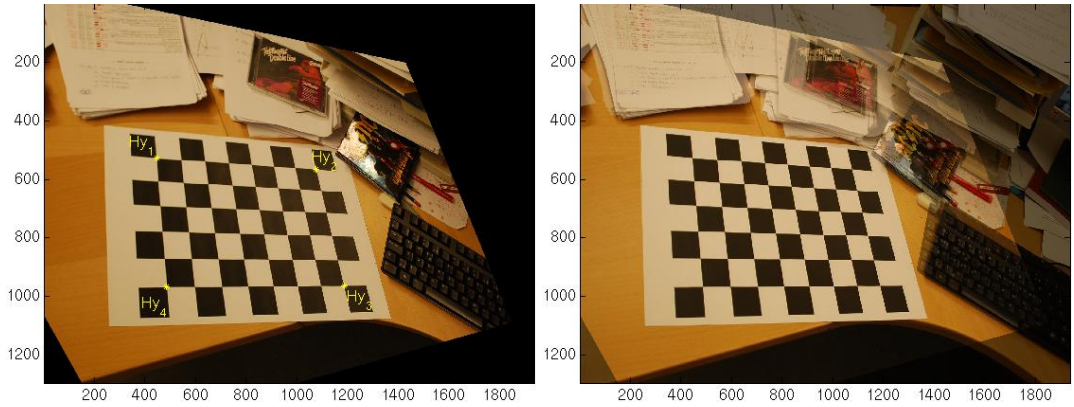


Figure 3.2: Left: the result of appying H to the left image in Figure 3.1. Right: The transformed image overlaid on the right image of Figure 3.1.

point correspondences. Figure 3.1 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.)

When H has been computed we can transform the other points in the image. Figure 3.2 shows the transformation of the left image and the transformed image overlaid on the right image. 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 9. 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 $\hat{\mathbf{l}} = (H^{-1})^T\mathbf{l}$.

3.1.1 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 shape

$$H = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix}, \tag{3.5}$$

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 3.3.

Figure 3.3: The affine transformation preserves parallel lines.

The **similarity transformation** has the form

$$H = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix}, \tag{3.6}$$

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 3.4.

Figure 3.4: The similarity transformation preserves angles between lines.

If $s = 1$ in (3.6) then the transformation is called **Euclidean**. This mapping also preserves distances between points. An example of an affine transformation is shown in Figure 3.5.

Figure 3.5: The Euclidean transformation preserves distances between points.

3.2 Affine Cameras and Parallel Projection

Previously we have derived the pinhole camera model. Here we present a class of cameras that are on one hand less accurate in practical applications but on the other hand yeild simpler solutions to multi-view problems.

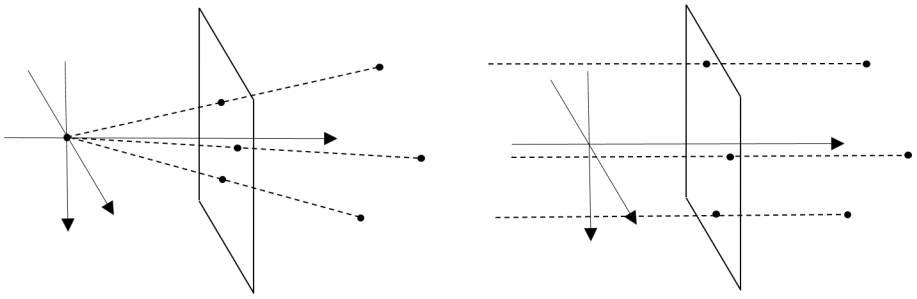


Figure 3.6: *Left* - A regular pinhole camera where all viewing rays go through the camera center. *Right* - The affine camera model where the viewing rays are parallel to the normal of the image ray.

The affine camera model is in a sense a simpler model than the regular pinhole camera since no division by the third coordinate is needed to compute the projection. The geometric interpretation is that the incoming viewing rays are parallel and do not pass through a common camera center as illustrated in Figure 3.6. While regular cameras typically do not work this way it is often a good approximation of the pinhole model if the observed scene points are roughly on the same depth.

Suppose that $\mathbf{X}' \sim \begin{pmatrix} X_1' \\ X_2' \\ X_3' \\ 1 \end{pmatrix}$ in the camera coordinate system. The projection $\mathbf{x} \sim \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$ is obtained by simply discarding the X_3' coordinate. In matrix form this can be written

$$\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_1' \\ X_2' \\ X_3' \\ 1 \end{pmatrix} = \begin{bmatrix} I_{2\times 3} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X' \\ 1 \end{bmatrix}, \tag{3.7}$$

where $I_{2\times 3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ and $X' = \begin{pmatrix} X_1' \\ X_2' \\ X_3' \end{pmatrix}$. The projection of $\mathbf{X} \sim \begin{bmatrix} X \\ 1 \end{bmatrix}$ in world coordinates is therefore

$$\mathbf{x} \sim \begin{bmatrix} I_{2\times 3} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} R_{2\times 3} & t_{2\times 1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}, \tag{3.8}$$

where $R_{2\times 3}$ contains the first two rows of R and $t_{2\times 1}$ the first two elements of t. Finally to obtain the real image in pixels we multiply with a matrix K, which gives

$$\mathbf{x} \sim \underbrace{K \begin{bmatrix} R_{2\times 3} & t_{2\times 1} \\ 0 & 1 \end{bmatrix}}_{=P} \begin{bmatrix} X \\ 1 \end{bmatrix} \tag{3.9}$$

If $K = \begin{bmatrix} K_{2\times 2} & 0 \\ 0 & 1 \end{bmatrix}$ we get

$$P = \begin{bmatrix} K_{2\times 2}R_{2\times 3} & t_{2\times 1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_{2\times 3} & t_{2\times 1} \\ 0 & 1 \end{bmatrix}. \tag{3.10}$$

Any matrix $A_{2\times 3}$ of size 2×3 can be factorized into $K_{2\times 2}R_{2\times 3}$ where $K_{2\times 2}$ is upper triangular and $R_{2\times 3}$ contains two rows from a rotation matrix (using the Gram-Schmidt process, which we will discuss in Lecture 5). Hence this is the most general type of affine camera that only requires that the last row of P is $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$. Similar to an affine transformation the affine camera preserves parallelism in the sense that two lines that are parallel in 3D are projected two lines that are parallel in the image.

If $K = \begin{bmatrix} sI_{2\times 2} & 0 \\ 0 & 1 \end{bmatrix}$ we get

$$P = \begin{bmatrix} sR_{2\times 3} & st_{2\times 1} \\ 0 & 1 \end{bmatrix}. \tag{3.11}$$

This camera type is called scaled orthographic or sometimes weak perspective. The name weak perspective comes from the fact that by changing the scale s one can approximate the perspective effect observed in a pinhole model when a single object moves away or towards the camera.

If $K = I$ we get

$$P = \begin{bmatrix} R_{2\times 3} & t_{2\times 1} \\ 0 & 1 \end{bmatrix}, \tag{3.12}$$

we get the so called orthographic camera. In addition to parallelism this camera also preserves distances that are perpendicular to the viewing direction. Note that when performing projection in any of these models we do not need to divide by the third coordinate. The projection can therefore also be expressed in regular coordinates as

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A_{2x3} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} + t_{2\times 1}. \tag{3.13}$$

Exercise 10. Compute the camera center (null space) of

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.14}$$

3.3 1D Cameras

Navigation for autonomous vehicles is often performed using 1D laser scanners such as the one shown in Figure 3.7. These devices basically sends out a signal that bounces of a reflector and returns to the sender. The result is a measurement of a direction in which the reflection was observed. This direction can be thought of as either an angle or a bearing (unit vector parallel to the ground plane). The projection into a 1D camera can be seen as a

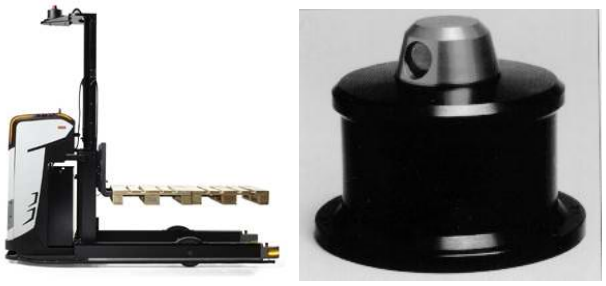


Figure 3.7: Autonomous vehicles (left) can perform navigation using laser scanners (right) that provide angular measurements or bearings (unit vectors in the plane).

mapping $\mathbb{P}^2 \to \mathbb{P}^1$. Similar to the regular camera model we can use homogeneous coordinates for the 2D point $\mathbf{X} \sim \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$ and its 1D projection $\mathbf{x} \sim \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and derive the camera equations

$$\lambda \mathbf{x} = \underbrace{K_{2\times 2} \begin{bmatrix} R_{2\times 2} & t_{2\times 1} \end{bmatrix}}_{P_{2\times 3}} \mathbf{X}, \tag{3.15}$$

where $K_{2\times 2}$ is triangular and $R_{2\times 2}$ is a 2D-rotation matrix. In contrast to the regular camera matrix we usually don't divide by the last coordinate. Instead we typically compute a unit vector, which corresponds to the intersection with a circle as illustrated in Figure 3.8, or the angle ϕ with respect to the e_y' axis. The camera equations can then be written

$$\lambda \begin{pmatrix} \sin\phi \\ \cos\phi \end{pmatrix} = P_{2\times 3}\mathbf{X}. \tag{3.16}$$

Note that from a projective geometry point of view it makes no difference whether we chose to work with unit vectors or with vectors with last element 1. They are both homogeneous coordinates representing the same point in $\mathbb{P}^1$.

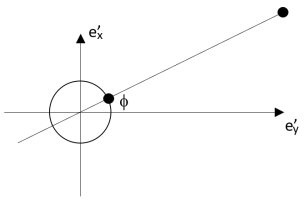


Figure 3.8: The projection into a 1D camera works similarly to a regular camera. However instead of having an image line at $y = 1$ we usually intersect with a circle of radius 1.

Exercise 11. The cameras $P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ and $P_2 = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$ are observing the point $\mathbf{X}$ at angles $\phi_1 = \frac{\pi}{4}$ and $\phi_2 = 0$ respectively. What is the position of the point $\mathbf{X}$?

Lecture 4:

Camera Calibration, DLT, SVD and Depth

4.1 The Inner Parameters

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

$$\lambda \mathbf{x} = P\mathbf{X}, \tag{4.1}$$

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 4.1.

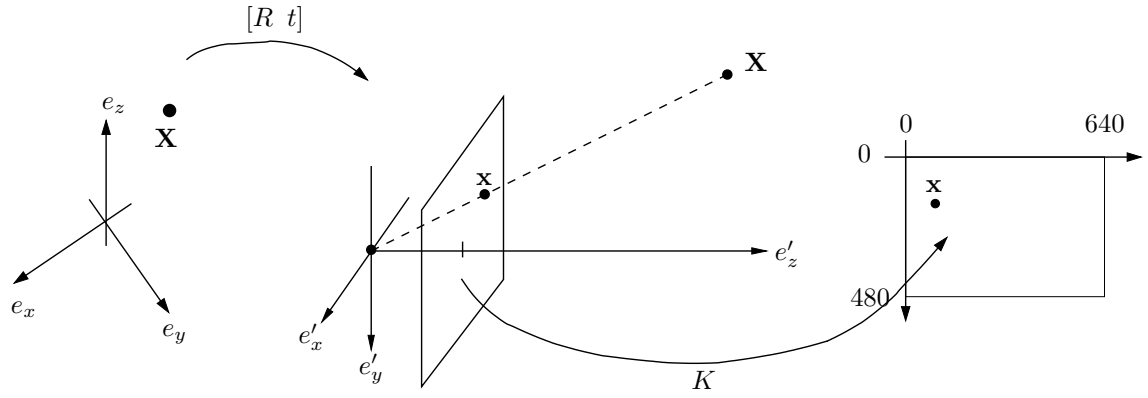


Figure 4.1: The different coordinate systems and mappings.

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

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.2}$$

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

$$\begin{pmatrix} fx + x_0 \\ fy + y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{4.3}$$

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 4.2, and is typically zero.

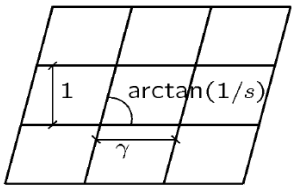


Figure 4.2: The skew parameter s corrects for non-rectangular pixels.

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 (4.1) with K^{-1} from the left. If we let $\tilde{\mathbf{x}} = K^{-1}\mathbf{x}$ we get

$$\lambda\tilde{\mathbf{x}} = K^{-1}K[R \quad t]\mathbf{X} = [R \quad t]\mathbf{X}. \tag{4.4}$$

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 4.3, 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 4.3: Radial distortion can not be handled with the K matrix. This requires a more complicated model.

4.2 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

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j, \quad \forall i, j. \tag{4.5}$$

Note that the scalars λ_{ij} are also unknown and needs to be determined. However, primarily we are interested in the scene points and cameras, the scalars 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

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j. \tag{4.6}$$

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

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j = P_iHH^{-1}\mathbf{X}_j = \tilde{P}_i\tilde{\mathbf{X}}_j. \tag{4.7}$$

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 4.4 a projective reconstruction of the Arch of Triumph in Paris is displayed.

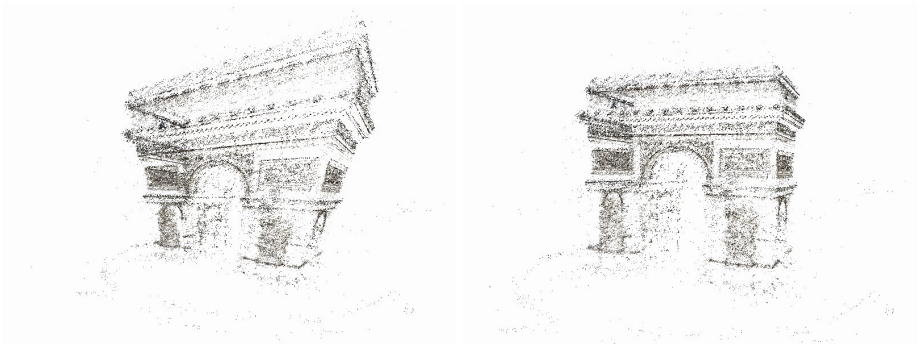


Figure 4.4: Reconstructions of the Arch of Triumph in Paris. *Left:* Projective reconstruction. *Right:* Euclidean reconstruction (known camera calibration). Both reconstructions provide the same projections.

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

$$\lambda_{ij}\tilde{\mathbf{x}}_{ij} = [R_i \quad t_i]\,\mathbf{X}_j, \tag{4.8}$$

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,

$$H = \begin{bmatrix} sQ & v \\ 0 & 1 \end{bmatrix}, \tag{4.9}$$

where Q is a rotation. We then get

$$\frac{\lambda_{ij}}{s}\mathbf{x}_{ij} = [R_i \quad t_i]\begin{bmatrix} Q & \frac{1}{s}v \\ 0 & \frac{1}{s} \end{bmatrix} H^{-1}\mathbf{X}_j = \left[R_iQ \quad \frac{1}{s}(R_iv + t_i)\right]\tilde{\mathbf{X}}_j, \tag{4.10}$$

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 (4.10) 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.)

4.3 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 4.2) 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.

4.3.1 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

$$\lambda_i \mathbf{x}_i = P\mathbf{X}_i, \quad i = 1, ..., n, \tag{4.11}$$

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

$$3n \geq 11 + n \Rightarrow n \geq 6 \tag{4.12}$$

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,

$$P = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} \tag{4.13}$$

then we can write (4.11) as

$$\mathbf{X}_i^T p_1 - \lambda_i x_i = 0 \tag{4.14}$$
$$\mathbf{X}_i^T p_2 - \lambda_i y_i = 0 \tag{4.15}$$
$$\mathbf{X}_i^T p_3 - \lambda_i = 0, \tag{4.16}$$

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

$$\begin{bmatrix} \mathbf{X}_i^T & 0 & 0 & -x_i \\ 0 & \mathbf{X}_i^T & 0 & -y_i \\ 0 & 0 & \mathbf{X}_i^T & -1 \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{4.17}$$

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

$$\underbrace{\begin{bmatrix} \mathbf{X}_1^T & 0 & 0 & -x_1 & 0 & 0 & \dots \\ 0 & \mathbf{X}_1^T & 0 & -y_1 & 0 & 0 & \dots \\ 0 & 0 & \mathbf{X}_1^T & -1 & 0 & 0 & \dots \\ \mathbf{X}_2^T & 0 & 0 & 0 & -x_2 & 0 & \dots \\ 0 & \mathbf{X}_2^T & 0 & 0 & -y_2 & 0 & \dots \\ 0 & 0 & \mathbf{X}_2^T & 0 & -1 & 0 & \dots \\ \mathbf{X}_3^T & 0 & 0 & 0 & 0 & -x_3 & \dots \\ 0 & \mathbf{X}_3^T & 0 & 0 & 0 & -y_3 & \dots \\ 0 & 0 & \mathbf{X}_3^T & 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{=M} \underbrace{\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{pmatrix}}_{=v} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}. \tag{4.18}$$

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

$$\min_{\|v\|^2=1} \|Mv\|^2. \tag{4.19}$$

We refer to this type of problem as a **homogeneous least squares problem**. Note that there are always at least two solutions to (4.19) 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 (4.11) means that the vectors $\mathbf{x}_i$ and $P\mathbf{X}_i$ are be parallel. This can be expressed using the vector product

$$\mathbf{x}_i \times P\mathbf{X}_i = 0, \quad i = 1, ..., n. \tag{4.20}$$

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.

Solving the Homogeneous System

The solution to (4.19) 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

$$\min_{g(v)=1} f(v). \tag{4.21}$$

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

$$\nabla f(v) = \gamma \nabla g(v) \quad \Leftrightarrow \quad 2M^T M v = \gamma 2v \quad \Leftrightarrow \quad M^T M v = \gamma v. \tag{4.22}$$

Therefore the solution of (4.19) 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

$$f(v_*) = v_*^T M^T M v_* = \gamma_* v_*^T v_*. \tag{4.23}$$

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*

$$M = USV^T, \tag{4.24}$$

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

$$S = \begin{bmatrix} diag(\sigma_1, \sigma_2, ..., \sigma_r) & 0 \\ 0 & 0 \end{bmatrix}, \tag{4.25}$$

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

If M has the SVD (4.24) then

$$M^T M = (USV^T)^T USV^T = VS^T U^T USV^T = VS^T SV^T. \tag{4.26}$$

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
$$Mv = 0. \tag{4.27}$$
- Compute the singular value decomposition.
$$M = USV^T. \tag{4.28}$$
- Extract the solution v_* from the last column of V.

Exercise 12. Suppose that $M = USV^T$ where

$$S = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}. \tag{4.29}$$

Find a vector that solves

$$\min_{\|v\|^2=1} \|Mv\|^2, \tag{4.30}$$

and the minimal value.

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

$$\lambda_i \mathbf{x} = P\mathbf{X}_i, \quad i = 1, ..., n \tag{4.31}$$

as outlined in the previous sections.

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

$$N = \begin{bmatrix} s & 0 & -s\bar{x} \\ 0 & s & -s\bar{y} \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.32}$$

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

$$\tilde{\mathbf{x}} = N\mathbf{x} \tag{4.33}$$

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

We can now solve the modified problem

$$\gamma_i \tilde{\mathbf{x}} = \tilde{P}\mathbf{X}_i, \tag{4.34}$$

by forming the system matrix M and computing its singular value decomposition. A solution to the original "un-normalized" problem (4.31) can now easily be found from

$$\gamma_i N\mathbf{x} = \tilde{P}\mathbf{X}_i. \tag{4.35}$$

4.3.2 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

$$P = K\begin{bmatrix} R & t \end{bmatrix}, \tag{4.36}$$

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*

$$A = RQ. \tag{4.37}$$

(If A is invertible and the diagonal elements of R 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

$$K = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}, \quad A = \begin{bmatrix} A_1^T \\ A_2^T \\ A_3^T \end{bmatrix} \text{ and } R = \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix}, \tag{4.38}$$

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

$$\begin{bmatrix} A_1^T \\ A_2^T \\ A_3^T \end{bmatrix} = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix} = \begin{bmatrix} aR_1^T + bR_2^T + cR_3^T \\ dR_2^T + eR_3^T \\ fR_3^T \end{bmatrix} \tag{4.39}$$

From the third row of (4.39) 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

$$f = \|A_3\| \quad \text{and} \quad R_3 = \frac{1}{\|A_3\|} A_3. \tag{4.40}$$

to get a positive coefficient f. When R_3 is known we can proceed to the second row of (4.39). 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

$$e = A_2^T R_3. \tag{4.41}$$

When e is known we can compute R_2 and d from

$$dR_2 = A_2 - eR_3, \tag{4.42}$$

similar to what we did for f and R_3 in (4.40). When R_2 and R_3 is known we use the first row of (4.39)

$$A_1 = aR_1 + bR_2 + cR_3 \tag{4.43}$$

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

$$A_1 - bR_2 - cR_3 = aR_1. \tag{4.44}$$

The resulting matrix K is not necessarily of the form (4.2) 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. In addition there is no guarantee that the determinant of R is 1 and therefore R may not be a rotation. However if $\det(R) = -1$ we switch can simply switch to $-P = \begin{bmatrix} -A & -a \end{bmatrix}$ which gives the factorization $-A = K(-R)$.

Exercise 13. If

$$P = \begin{pmatrix} 3000 & 0 & -1000 & 1 \\ 1000 & 2000\sqrt{2} & 1000 & 2 \\ 2 & 0 & 2 & 3 \end{pmatrix}$$

find the inner parameters of the camera.

4.4 Depth

To determine if a point is in front of a camera we will compute its depth with respect to that camera. Previously we defined the depth of a point $\mathbf{X}$ s the z-coordinate of the of the point in the local camera coordinate system. Given a camera matrix P we therefore need to compute the QR-factorization to extract the extrinsic parameters R and t to find the depth. In this section we derive a direct formula that uses the elements of $P = \begin{bmatrix} A & a \end{bmatrix}$.

First we determine the principal axis. Recall that this if $P = K \begin{bmatrix} R & t \end{bmatrix}$ where R is a rotation and K is upper triangular with positive diagonal elements then the third row R_3 of R is the principal axis. Note that if $\det(A) < 0$ and $A = KR$ then $\det(K)\det(R) = \det(A) < 0$. Since an upper triangular K with positive diagonal elements has positive determinant we must have that $\det(R) = -1$ and therefore R is orthogonal but not a rotation. Since the camera matrix P is only defined up to an unknown scale factor we instead use $P = \text{sign}(\det(A)) \begin{bmatrix} A & a \end{bmatrix}$ to ensure that the RQ factorization gives a rotation. We then get the principal axis

$$R_3 = \frac{\text{sign}(\det(A))}{\|A_3\|} A_3. \tag{4.45}$$

The depth of a point $\mathbf{X} = \begin{bmatrix} X \\ 1 \end{bmatrix}$, where X are the regular Carthesian coordinates is given by

$$\text{depth}(P, \mathbf{X}) = R_3^T (X - C), \tag{4.46}$$

where $C = -A^{-1}a$ are the Carthesian coordinates of the camera center. Since $A_3^T A^{-1} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} a = a_3$, where a_3 is the third element or a we get

$$\text{depth}(P, \mathbf{X}) = \frac{\text{sign}(\det(A))}{\|A_3\|} \begin{bmatrix} A_3^T & a_3 \end{bmatrix} \mathbf{X}. \tag{4.47}$$

The term $\begin{bmatrix} A_3^T & a_3 \end{bmatrix} \mathbf{X}$ is the last element of $P\mathbf{X}$ which we usually denote λ.

For a point $\mathbf{X} = \begin{bmatrix} X \\ \rho \end{bmatrix}$ where the fourth coordinate is $\rho \neq 0$ we first divide by ρ and repeat the above computations. This gives

$$\text{depth}(P, \mathbf{X}) = \frac{\text{sign}(\det(A))\lambda}{\|A_3\|\rho}. \tag{4.48}$$

Note that if P is an affine camera then $A_3 = 0$. Therefore we cannot talk about points being in front or behind an affine camera. Similarly for a vanishing point we have $\rho = 0$ and therefore only finite points can be said to be behind or in front of the camera.

Exercise 14. Which of the 3D points $(1, 0, 1), (0, 1, -2)$ and $(1, 1, 1)$ are in-front of the camera

$$P = \begin{pmatrix} 1 & 2 & 0 & 1 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}?$$

Lecture 5:

Triangulation and Homography Estimation

5.1 Triangulation

The problem of finding an unknown 3D-point position $\mathbf{X}$ from measured projections $\mathbf{x}_i$, $i = 1, ...n$ into known cameras P_i is called **triangulation**. We have previously seen that if $\mathbf{X}$ is to project to the measurements then the camera equations

$$\lambda_i \mathbf{x}_i = P_i \mathbf{X}_i, \quad i = 1, ..., n. \tag{5.1}$$

must hold. Therefore we need to find $\mathbf{X}$ and λ_i that solve these equations. For simplicity we will assume that $\mathbf{X}$ is a regular point, not at infinity, with homogeneous coordinates $\mathbf{X} = \begin{bmatrix} X \\ 1 \end{bmatrix}$, where $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ are the regular Cartesian 3D-coordinates. Each projection gives us 3 equations and for n projections we therefore have $3n$ equations. Each projection also introduces one new unknown, the depth λ_i, and therefore we have $3 + n$ unknowns. To solve the problem we need

$$3n \geq 3 + n \Leftrightarrow n \geq \frac{3}{2}. \tag{5.2}$$

Thus two projections are enough.

If $P_i = \begin{bmatrix} A_i & t_i \end{bmatrix}$, where A_i is 3×3 and invertible the camera equations can be written

$$\lambda_i \mathbf{x}_i = A_i X + t_i \Leftrightarrow X = -A_i^{-1} t_i + \lambda_i A_i^{-1} \mathbf{x}_i. \tag{5.3}$$

Note that $C_i = -A_i^{-1} t_i$ are the 3D-coordinates of the camera center of P_i (see Lecture 1). The geometric interpretation of the above expression is that X should be on a line going though C_i with directional vector $A_i^{-1}\mathbf{x}_i$. When we vary λ_i we get different 3D points on the line and all of them project to $\mathbf{x}_i$. To find the correct X we need to determine the intersection of the lines coming from each camera as illustrated in Figure 5.1.

Exercise 15. Find the 3D-point $\mathbf{X}$ that projects to $\mathbf{x}_1 \sim \left(\frac{1}{2}, \frac{1}{2}, 1\right)$ in $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$ and $\mathbf{x}_2 \sim \left(0, \frac{1}{2}, 1\right)$ in $P_2 = \begin{bmatrix} I & \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \end{bmatrix}$.

Remark 2. Note that the system in (**??**) has three equations but only two unknowns and may therefore not have a solution for other choices of the points $\mathbf{x}_1$ and $\mathbf{x}_2$. Geometrically this means that the viewing lines might not intersect. In real problems with noisy measurements we therefore need to solve the problem approximately in some sense.

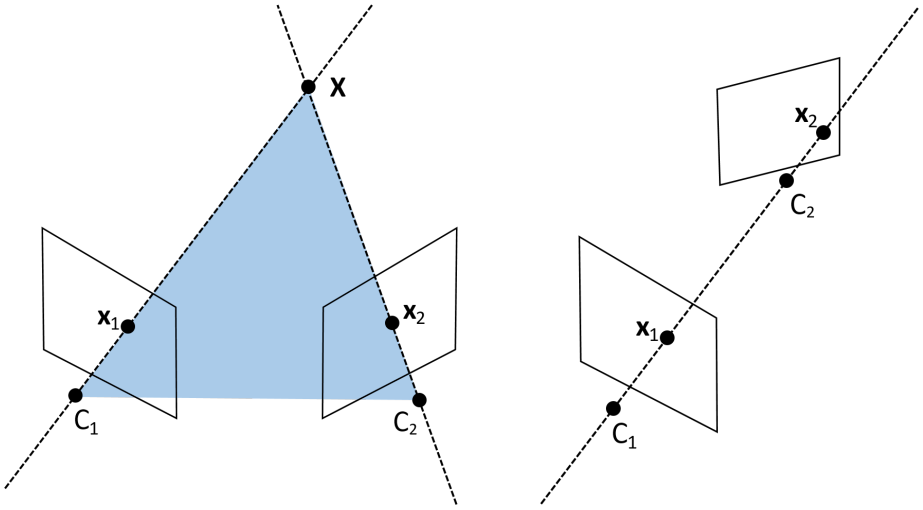


Figure 5.1: Illustration of triangulation. Left: The sought point $\mathbf{X}$ is in the intersection of the two viewing lines. Right: If the camera moves so that the two viewing lines intersect for all points the problem becomes degenerate and we cannot determine $\mathbf{X}$.

In some special cases the problem will have multiple intersections between the lines of (5.3). If the camera centers C_i, $i = 1, ..., n$ and the 3D point X are all located on one line in 3D, the expressions (5.3) become identical and all points on the 3D-line fulfill them. This is called a **degenerate configuration** and is illustrated to the right in Figure 5.1. While it is very unlikely that all camera centers and the 3D point should be exactly on the same line in any real case, degenerate configurations are important since problem instances that are close to degeneracy often exhibit numerics and become very sensitive to noise. Hence in practical cases when the camera centers and the 3D point are close to being on a line we can expect that we will get poor accuracy when we try to recover it.

5.1.1 Noisy Recovery using DLT

In practical cases our measurements will not be exact but will be affected by noise, and therefore we cannot expect the viewing rays to have an exact intersection point, as illustrated in Figure 5.2. Therefore we need to solve the

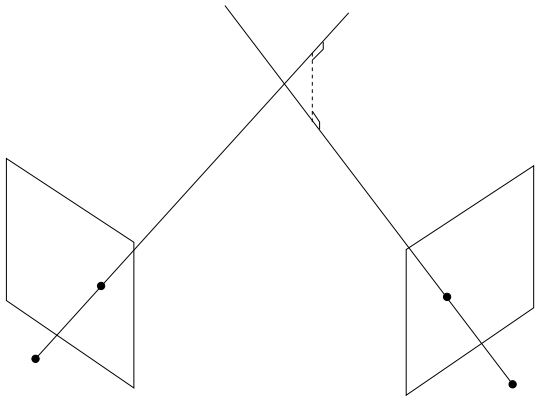


Figure 5.2: In case of noisy measurements we cannot expect the viewing rays to intersect in a point. In such cases we need to solve the camera equations approximately.

camera equations approximately in some sense. One way of doing this is to use the DLT method from Lecture 3. The problem is linear in the unknowns λ_i and X, so we can find the homogeneous least squares solution by

re-formulating the problem as

$$Mv = 0, \tag{5.4}$$

with

$$M = \begin{bmatrix} P_1 & -\mathbf{x}_1 & 0 & \cdots & 0 \\ P_2 & 0 & -\mathbf{x}_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ P_n & 0 & 0 & \cdots & -\mathbf{x}_n \end{bmatrix}, \tag{5.5}$$

and

$$v^T = \begin{bmatrix} X^T & \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}. \tag{5.6}$$

As in Lecture 3 we solve

$$\min_{\|v\|^2=1} \|Mv\|^2 \tag{5.7}$$

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

Figure 5.3 shows three examples of the DLT objective for triangulation. We use 1D-cameras for visualization purposes. Here a camera is a 2×3 matrix which takes a point in the plane $\mathbb{P}^2$ and projects it onto the camera line, essentially providing a direction towards the point. For a given $2D$ point $\mathbf{X} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ we compute

$$f(\mathbf{X}) = \min_{\lambda_1^2+\lambda_2^2+\|\mathbf{X}\|^2\gamma^2=1} \left\| M \begin{bmatrix} \gamma\mathbf{X} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} \right\|^2, \tag{5.8}$$

and plot the result as a function of $\mathbf{X}$. We also plot the cameras and the projection of the optimal 2D point in the same figure. When the cameras are close to each other the level curves of the objective function become thin and elongated. As a result the location of the triangulated point will be more difficult to determine when noise is added to the problem. In the three images in Figure 5.4 perform the same triangulation experiment 100 times with noise added to the image projections. The noise is the same for all the three images only the locations of the cameras differ. The estimated 2D-points are the blue dots. When the cameras are sufficiently far apart all estimations end up close to the true 2D-point. In contrast when the cameras are close to the degenerate case, the result becomes very sensitive to noise and the exact distance from the cameras are difficult to determine accurately.

5.2 Homography Estimation

Projective transformations (or homographies) were introduced in Lecture 2. In practical computer vision problems they occur in a number of common settings. Figure 5.5 shows a plane induced homography. If a set of points from a 3D-plane is projected into two images then there is a projective transformation $\mathbb{P}^2 \to \mathbb{P}^2$ between the two images (see Assignment 1 for a proof). Another example is when two images are captured by cameras that have the same position, only differing by orientation and possibly calibration. This is useful for building panoramas (see Section 5.2.3). When we are solving uncalibrated reconstruction problems all possible reconstructions are related by a projective transformation $\mathbb{P}^3 \to \mathbb{P}^3$ (see Lecture 3).

In homography estimation we want to find a projective transformation from $\mathbb{P}^k$ to $\mathbb{P}^k$. We will describe the problem for $k = 2$ but the procedure is exactly the same for any dimension. Given two sets of points $\mathbf{y}_i$ and $\mathbf{x}_i$ that are related by a homography H we want to solve

$$\lambda_i \mathbf{y}_i = H\mathbf{x}_i, \quad i = 1 \dots n. \tag{5.9}$$

The matrix H has 9 entries, but since its scale is arbitrary we can assume that one of them is fixed. For n points we therefore have $3n$ equations and $8 + n$ unknowns and the problem can therefore be solved if

$$3n \geq 8 + n \Leftrightarrow n \geq 4. \tag{5.10}$$

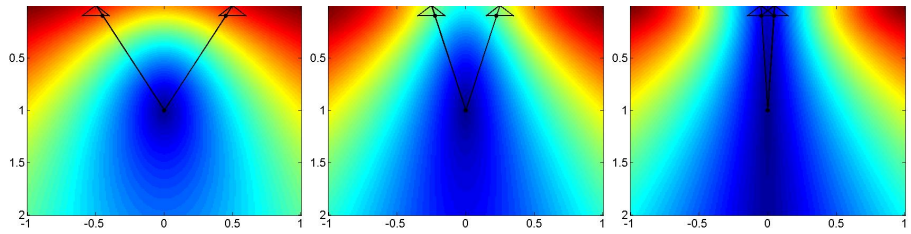


Figure 5.3: Visualization of the DLT objective for 1-dimensional cameras. When the cameras approach the degenerate case (by moving towards each other) the objective is roughly constant along the viewing direction.

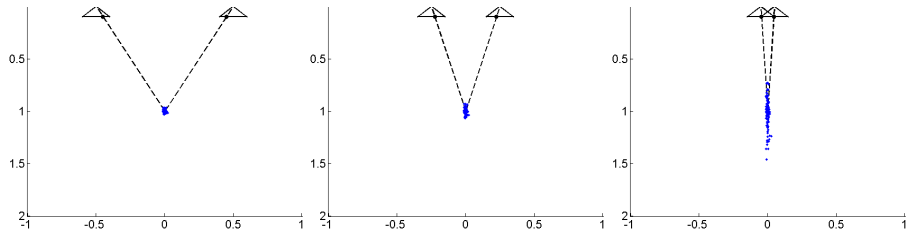


Figure 5.4: Estimation of a 2D point from 1D images using the DLT objective. Close to degenerate cases the depth of the point becomes uncertain.

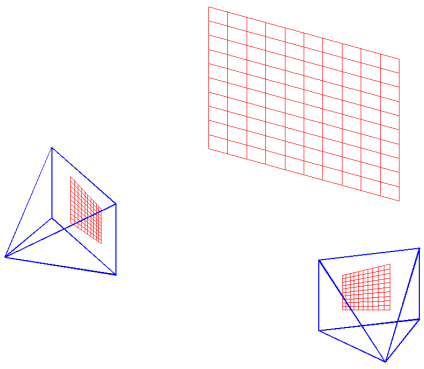


Figure 5.5: Example of a plane induced homography.

Note that in contrast to the triangulation problem, where we obtained an overdetermined system when using two point projections, the problem can be solved exactly if $n = 4$. Before we compute the solution we make one simplification. We assume that

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \ \mathbf{x}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \ \mathbf{x}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \text{ and } \mathbf{x}_4 = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \tag{5.11}$$

where a, b, c are some known numbers. Note that $\mathbf{x}_1$,$\mathbf{x}_2$ are points at infinity. Although we cannot (directly) observe such points in a real image, this assumption simplifies the following derivations. We treat the general case at the end of this section through a change of variables.

Under the assumption (5.11) we have

$$H\mathbf{x}_i = \begin{cases} h_i & i = 1, 2, 3 \\ ah_1 + bh_2 + ch_3 & i = 4 \end{cases}, \tag{5.12}$$

where h_i is column i from H. Therefore H must be of the form

$$H = \begin{bmatrix} \lambda_1 \mathbf{y}_1 & \lambda_2 \mathbf{y}_2 & \lambda_3 \mathbf{y}_3 \end{bmatrix}. \tag{5.13}$$

To determine the unknowns λ_i, $i = 1, 2, 3$ we consider the second case of (5.12) which inserted in (5.9) gives

$$\lambda_4 \mathbf{y}_4 = H\mathbf{x}_4 = \lambda_1 a \mathbf{y}_1 + \lambda_2 b \mathbf{y}_2 + \lambda_3 c \mathbf{y}_3 = \begin{bmatrix} a\mathbf{y}_1 & b\mathbf{y}_2 & c\mathbf{y}_3 \end{bmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix}. \tag{5.14}$$

Note that

$$\begin{bmatrix} a\mathbf{y}_1 & b\mathbf{y}_2 & c\mathbf{y}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 \end{bmatrix}}_{:=Y_{1:3}} \underbrace{\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}}_{:=D_{\mathbf{x}_4}}. \tag{5.15}$$

Since the scale is arbitrary we can assume that $\lambda_4 = 1$. The homography can therefore be uniquely determined when the matrix $Y_{1:3}D_{\mathbf{x}_4}$ is invertible, and is given by the formula

$$\begin{aligned} H &= Y_{1:3} D_{\lambda_{1:3}} && (5.16) \\ \lambda_{1:3} &= D_{\mathbf{x}_4}^{-1} Y_{1:3}^{-1} \mathbf{y}_4, && (5.17) \end{aligned}$$

where

$$\lambda_{1:3} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} \text{ and } D_{\lambda_{1:3}} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}. \tag{5.18}$$

In the next section we will study under what conditions the inverses above exist and what happens when they don't.

We conclude this section by showing how to estimate H with four generally placed points. This can be achieved through a change of coordinates. Let $X_{1:3}$ be the 3×3 matrix with columns $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. If $X_{1:3}$ is invertible then (5.9) can then be written

$$\lambda_i \mathbf{y}_i = H\mathbf{x}_i = \underbrace{HX_{1:3}}_{:=\tilde{H}} \underbrace{X_{1:3}^{-1}\mathbf{x}_i}_{:=\tilde{\mathbf{x}}_i}. \tag{5.19}$$

Since $X_{1:3}^{-1}X_{1:3} = I$ and $\mathbf{x}_i$, $i = 1, 2, 3$ are the columns of $X_{1:3}$ it is now easy to see that in the new coordinates $\tilde{\mathbf{x}}_i$ are of the desired form (5.11). We can therefore instead solve for $\tilde{H}$, as described above, and compute the solution to (5.9) using $H = \tilde{H}X_{1:3}^{-1}$. We will return to the issue of $X_{1:3}$ being invertible in Section 5.2.1.

5.2.1 Uniqueness and Degeneracy

In deriving the formulas for H we have made the assumptions that the inverses $D_{\mathbf{x}_4}^{-1}$, $Y_{1:3}^{-1}$ exist. In addition to make the coordinate change which ensures that (5.11) holds we need that $X_{1:3}^{-1}$ exists. To see when this holds we recall from Linear Algebra that a matrix is invertible if and only if its columns are linearly independent. In $\mathbb{R}^3$ the three vectors are linearly dependent if and only if they lie in a plane (containing the origin). That is, there is a vector l such that $l^T\mathbf{x}_i = 0$ for $i = 1, 2, 3$. In $\mathbb{P}^3$ the interpretation of these equations is that the points represented by $\mathbf{x}_i$ all lie on the same line l (see Lecture 2). Therefore the inverse $X_{1:3}^{-1}$ exists as long as the three points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ are not collinear. Since a homography always maps lines to lines (see Lecture 2) $Y_{1:3}^{-1}$ exists precisely when $X_{1:3}^{-1}$ exist.

In addition we have to ensure that the inverse of $D_{\mathbf{x}_4}$ exists. It is clear that this is true when $a \neq 0$, $b \neq 0$ and $c \neq 0$. Now suppose that for example $c = 0$. Then it is clear that $\mathbf{x}_1$,$\mathbf{x}_2$ and $\mathbf{x}_4$ are linearly dependent (as vectors in $\mathbb{R}^3$). Similar arguments for a and b show that the inverses of $X_{1:3}$, $Y_{1:3}$ and $D_{\mathbf{x}_1}$ all exist if and only if no combination of three points from $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$ are collinear in $\mathbb{P}^2$.

In general a set of points in $\mathbb{P}^n$ are called **projectively independent** if they have homogeneous coordinates that are linearly independent as vectors in $\mathbb{R}^{n+1}$. (Note that since the homogeneous representatives of a point can only differ with a non-zero scaling it does not matter which representatives we choose to test projective independence.) A set of $n + 2$ points in $\mathbb{P}^n$ is called a **projective basis** if no subset of $n + 1$ points is projectively dependent. A

projective transformation $\mathbb{P}^n \to \mathbb{P}^n$ is uniquely determined by the mapping of the $n+2$ points of a projective basis. In the case of $n = 2$ we have already seen that we need four points where no three are on a line. As a second example consider a projective mapping $\mathbb{P}^3 \to \mathbb{P}^3$. If we know how five points are mapped and these form a projective basis we can uniquely determine the mapping. In this case the basis assumption amounts to no subset of four points lie on a plane in $\mathbb{P}^3$.

We now consider what happens in the case of $\mathbb{P}^2$ when there is a subset of 3 points that are on a line. For simplicity let us assume that the first 3 points are not on a line so that $X_{1:3}$ and $Y_{1:3}$ exist. In this case one of the constants a,b or c will be 0. If for example $a = 0$ then (5.15) reduces to

$$\lambda_4 \mathbf{y}_4 = \lambda_2 b \mathbf{y}_2 + \lambda_3 c \mathbf{y}_3. \tag{5.20}$$

It is clear from this expression that λ_1 cannot be determined in this case. However the equations can still be solved since $a = 0$ corresponds to $\mathbf{x}_2$,$\mathbf{x}_3$,$\mathbf{x}_4$ (and therefore $\mathbf{y}_2$,$\mathbf{y}_3$,$\mathbf{y}_4$) being linearly dependent. From Linear Algebra we then know that there are numbers γ_2, γ_3 and γ_4 not all of them zero such that

$$\gamma_2 \mathbf{y}_2 + \gamma_3 \mathbf{y}_3 + \gamma_4 \mathbf{y}_4 = 0. \tag{5.21}$$

Furthermore, we must have $\gamma_4 \neq 0$ since otherwise $\gamma_2 \mathbf{y}_2 = -\gamma_3 \mathbf{y}_3$ which means that $\mathbf{y}_2$ and $\mathbf{y}_3$ are homogeneous representatives of the same point in $\mathbb{P}^2$. Therefore selecting $\lambda_2 = -\frac{\gamma_2}{b}$, $\lambda_3 = -\frac{\gamma_3}{c}$, $\lambda_4 = \gamma_4$ and λ_1 arbitrarily gives a family of solutions, all mapping the 4 points correctly. Figure 5.6 shows an example of a degenerate case where three of the four black points are on a line. Both the estimated transformations map the black points to themselves but differ elsewhere. The first homography maps the blue grid to the green one while the second maps the blue grid to the red one.

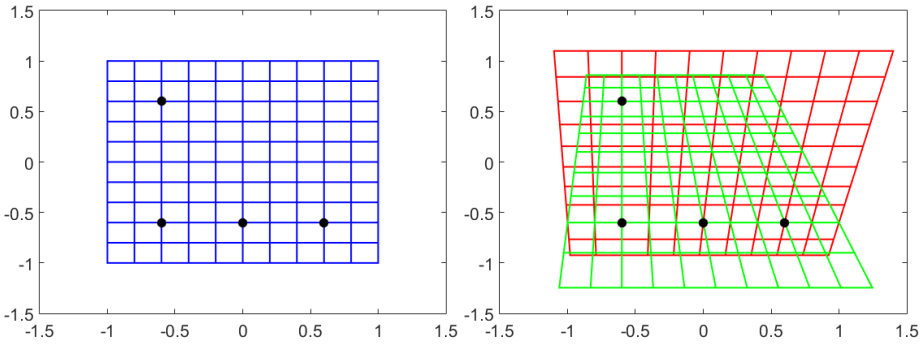


Figure 5.6: Two examples of homographies that map the four black points to themselves but differ for other points. The blue grid (left) is mapped to the green and red grids (right) by the two transformations respectively.

5.2.2 Noisy Recovery using DLT

As in the case of triangulation we often have overdetermined systems with noisy measurements. In such cases we can again apply the DLT approach. Suppose that we want to solve (5.9) with $n \geq 4$. Let h^i be row i of H, that is,

$$H = \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix}, \tag{5.22}$$

and

$$\mathbf{y}_i = \begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix}. \tag{5.23}$$

Here H is a 3×3 matrix so h^1, h^2 and h^3 are 1×3 matrices. By stacking all our unknowns in a vector

$$v^T = \begin{bmatrix} h_1 & h_2 & h_3 & \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}. \tag{5.24}$$

we can write our problem as

$$Mv = 0, \tag{5.25}$$

where

$$M = \begin{bmatrix} \mathbf{x}_1^T & 0 & 0 & -u_1 & 0 & \cdots & 0 \\ 0 & \mathbf{x}_1^T & 0 & -v_1 & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{x}_1^T & -w_1 & 0 & \cdots & 0 \\ \mathbf{x}_2^T & 0 & 0 & 0 & -u_2 & \cdots & 0 \\ 0 & \mathbf{x}_2^T & 0 & 0 & -v_2 & \cdots & 0 \\ 0 & 0 & \mathbf{x}_2^T & 0 & -w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ \mathbf{x}_n^T & 0 & 0 & 0 & 0 & \cdots & -u_n \\ 0 & \mathbf{x}_n^T & 0 & 0 & 0 & \cdots & -v_n \\ 0 & 0 & \mathbf{x}_n^T & 0 & 0 & \cdots & -w_n \end{bmatrix}. \tag{5.26}$$

To account for noise we again solve the solve the homogeneous least squares problem

$$\min_{\|v\|^2=1} \|Mv\|^2 \tag{5.27}$$

using the singular value decomposition of M as described in Lecture 3.

5.2.3 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 (camera center). 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 captured the images in 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 corresponding image points $\mathbf{x}_i$ and $\mathbf{y}_i$ being projections of the 3D point $\mathbf{X}_i = \begin{bmatrix} X_i \\ 1 \end{bmatrix}$ we have the following system of equations

$$\gamma_i \mathbf{x}_i = [R_1 \ 0] \begin{bmatrix} X_i \\ 1 \end{bmatrix}, \tag{5.28}$$

$$\eta_i \mathbf{y}_i = [R_2 \ 0] \begin{bmatrix} X_i \\ 1 \end{bmatrix}, \tag{5.29}$$

where γ_i and η_i are unknown (non-zero) scalars. This gives us

$$\begin{cases} \gamma_i \mathbf{x}_i = R_1 X_i \\ \eta_i \mathbf{y}_i = R_2 X_i \end{cases} \Leftrightarrow \begin{cases} X_i = \gamma_i R_1^T \mathbf{x}_i \\ \eta_i \mathbf{y}_i = R_2 X_i \end{cases} \Leftrightarrow \begin{cases} X_i = \gamma_i R_1^T \mathbf{x}_i \\ \eta_i \mathbf{y}_i = \gamma_i R_2 R_1^T \mathbf{x}_i \end{cases}. \tag{5.30}$$

This means that we can write the last equation as

$$\lambda_i \mathbf{y}_i = H \mathbf{x}_i, \tag{5.31}$$

with $\lambda_i = \frac{\eta_i}{\gamma_i}$ and $H = R_2 R_1^T$. This shows that we can transfer points from the first image plane to the second by the use of a homography. We know from the previous sections that we can estimate this transformation from at least four point correspondences. Once we have estimated the homography, all other points in an image can be transferred using it.

The unnormalized points $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$ are related to the normalized ones by $\mathbf{x}_i = K_1^{-1}\tilde{\mathbf{x}}_i$ and $\mathbf{y}_i = K_2^{-1}\tilde{\mathbf{y}}_i$ respectively. Therefore we have

$$\lambda_i K_2^{-1}\tilde{\mathbf{y}}_i = R_2 R_1^T K_1^{-1}\tilde{\mathbf{x}}_i \Leftrightarrow \lambda_i \tilde{\mathbf{y}}_i = K_2 R_2 R_1^T K_1^{-1}\tilde{\mathbf{x}}_i, \tag{5.32}$$

which means that the unnormalized points are mapped by a homography of the following form

$$H = K_2 R_2 R_1^T K_1^{-1}. \tag{5.33}$$

Figures 5.7-5.9 shows how we can use homographies to build panoramas from multiple images. In this case H_{21}, which transforms points in image 2 to Image 1, is estimated from green matches and H_{32}, which transforms points in image 3 to points in Image 2, is estimated from red matches. Figure 5.8 shows the effects of transforming Image 2 and 3 using the homographies H_{21} and $H_{31} = H_{21}H_{32}$. These transformations show where the pixels would have been placed had images 2 and 3 been captured with camera 1. When these transformations have been applied we can extend Image 1 with pixels from the other images.

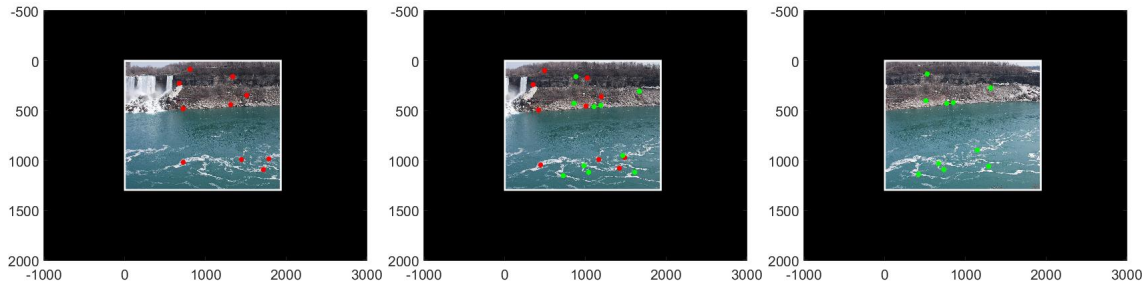


Figure 5.7: Original images with matches between image 1 and 2 (red) and between 2 and 3 (green). The matches can be used to compute two homographies, H_{21} which maps pixels in image 2 to image 1 and H_{32} which maps image 3 to image 2.

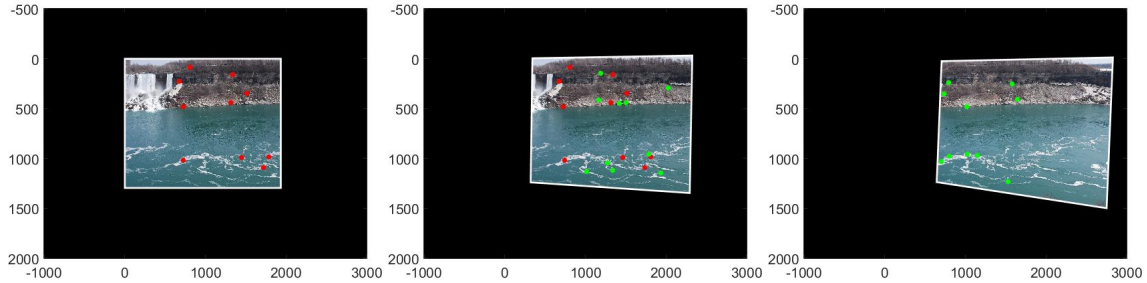


Figure 5.8: The homographies H_{21} and $H_{21}H_{32}$ can be used to transform the pixel of images 2 and 3 respectively, to the coordinate system of the first camera.

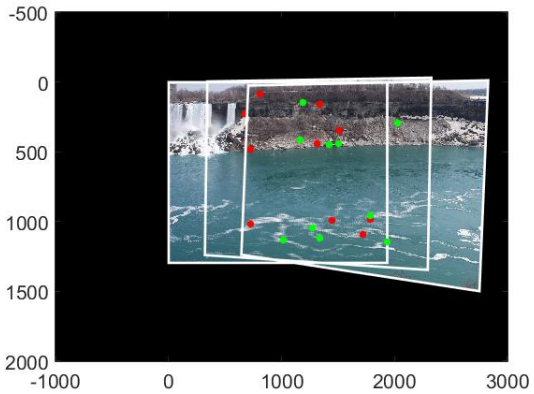


Figure 5.9: When images 2 and 3 has been transformed we can add the images together.

Lecture 6:

Radial Distortion and 1D-cameras

6.1 A Radial Distortion Model

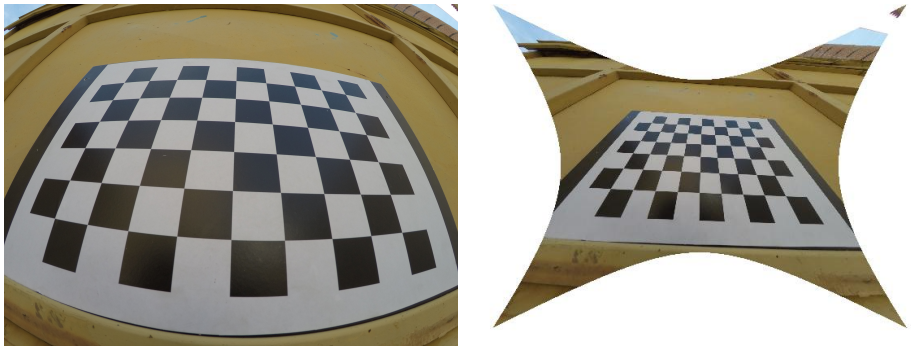


Figure 6.1: Projection of a line in 3D gives a line in the image under the pinhole camera model. Under radial distortion this is not true and the projection of a line often appears as slightly bent as shown in the left image. To the right the image has been undistorted and straight lines appear straight again.

The camera model that we have derived in Lectures 1 and 2 has the property that straight lines are projected onto straight lines. While this is typically enough for many regular cameras, wide field of view lenses and fisheye cameras typically have what is called radial distortion, which is illustrated in Figure 6.1. Directly applying the pinhole camera model to images such as this one usually gives large errors. Therefore we first need an extended model that can handle this kind of images.

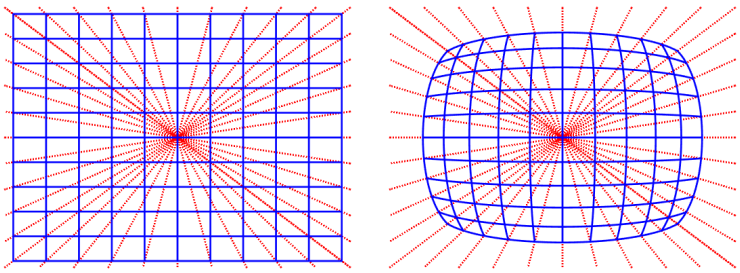


Figure 6.2: Radial distortion is typically modeled by moving points along lines going throught the principal point.

Radial distortion is typically modeled by moving points towards or away from the principal point $(0, 0, 1)$ in the image plane ($z = 1$). The distance the point moves depends on how close it is to the principal point. We let $\mathbf{x}_u \sim (x_u, y_u, 1)$ denote the undistorted point and $\mathbf{x}_d \sim (x_d, y_d, 1)$ the distorted one. Since $\mathbf{x}_d$ is on the line going through $\mathbf{x}_u$ and the principal point we can write it

$$\mathbf{x}_u \sim \begin{pmatrix} d(r_d)x_d \\ d(r_d)y_d \\ 1 \end{pmatrix} = \begin{pmatrix} d(r_d) & 0 & 0 \\ 0 & d(r_d) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix}. \tag{6.1}$$

Here $r_d = \sqrt{x_d^2 + y_d^2}$ is the distance to the principal point and $d(r_d)$ is a function that controls how far the point should be moved. There are several choices for d but usually a low degree polynomial is sufficient. (One may alternatively use $r_u = \sqrt{x_u^2 + y_u^2}$. However the problem of undistorting an image becomes easier if one uses $r_d = \sqrt{x_d^2 + y_d^2}$.)

The full projection consists of the following three steps:

1. Projection into the image plane ($z = 1$):
$$\lambda \mathbf{x}_u = \begin{bmatrix} R & t \end{bmatrix} \mathbf{X}. \tag{6.2}$$
2. Find $\mathbf{x}_d$ such that $\mathbf{x}_u \sim (d(r_d)x_d, d(r_d)y_d, 1)$.
3. Map the image plane to the real image (pixels):
$$\mathbf{x} \sim K\mathbf{x}_d. \tag{6.3}$$

If the inner parameters K and the distortion function $d(r)$ is known we can remove radial distortion using the following process:

1. Normalize the image points with K
$$\mathbf{x}_d \sim K^{-1}\mathbf{x}. \tag{6.4}$$
2. Compute $d(r_d)$, where $r_d = \sqrt{x_d^2 + y_d^2}$ and
$$\mathbf{x}_u \sim \begin{pmatrix} d(r_d)x_d \\ d(r_d)y_d \\ 1 \end{pmatrix} = \begin{pmatrix} d(r_d) & 0 & 0 \\ 0 & d(r_d) & 0 \\ 0 & 0 & 1 \end{pmatrix} K^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{6.5}$$

6.2 1D Radial Cameras

If the camera intrinsic and distortion parameters are known it is possible to undistort the images (as described above) and solve multiple view geometry problems such as triangulation and resectioning as we have discussed in previous sections. Alternatively one can, if the principal point is known, use the so called 1D Radial camera model that is invariant to radial distortion.

Here we assume for simplicity that the principal point is $(x_0, y_0) = (0, 0)$. Note that if $(x_0, y_0) \neq (0, 0)$ we can multiply the image points with the matrix $\begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}$ to move the principal point to $(0, 0)$. See Assignment 2, Exercise 3.

The 1D-radial camera is represented by a 2×4 matrix which is basically the two first rows of a regular 3×4 camera matrix. To derive it we start from the regular camera equations (with radial distortion) which according to the previous section can be written

$$\mathbf{x} \sim \begin{pmatrix} \gamma f & sf & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d(r_d) & 0 & 0 \\ 0 & d(r_d) & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{bmatrix} R & t \end{bmatrix} \mathbf{X} \tag{6.6}$$

Since

$$\begin{pmatrix} d(r_d) & 0 & 0 \\ 0 & d(r_d) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d(r_d) \end{pmatrix} = d(r_d)I \sim I, \tag{6.7}$$

we get

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} \gamma f & sf & 0 \\ 0 & f & 0 \\ 0 & 0 & d(r_d) \end{pmatrix} \begin{bmatrix} R & t \end{bmatrix} \mathbf{X}. \tag{6.8}$$

Only the third coordinate depends on the radial distortion. If we discard it we get

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim \begin{pmatrix} \gamma f & sf & 0 \\ 0 & f & 0 \end{pmatrix} \begin{bmatrix} R & t \end{bmatrix} \mathbf{X}. \tag{6.9}$$

Here we note that last row of $\begin{bmatrix} R & t \end{bmatrix}$ vanishes from this expression. If $\begin{bmatrix} R_{2\times 3} & t_{2x1} \end{bmatrix}$ are the first two rows of $\begin{bmatrix} R & t \end{bmatrix}$ we get

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim \begin{pmatrix} \gamma f & sf \\ 0 & f \end{pmatrix} \begin{bmatrix} R_{2\times 3} & t_{2x1} \end{bmatrix} \mathbf{X} \sim \underbrace{\begin{pmatrix} \gamma & s \\ 0 & 1 \end{pmatrix} \begin{bmatrix} R_{2\times 3} & t_{2x1} \end{bmatrix}}_{P_{2\times 4}} \mathbf{X}. \tag{6.10}$$

Note that the focal length also vanishes from this expression due to the scale ambiguity. Hence, this camera model is invariant to radial distortion and focal length. The above equation can be seen as a projection $\mathbb{P}^3 \to \mathbb{P}^1$. Geometrically we can think of $\begin{pmatrix} x \\ y \end{pmatrix}$ as a directional vector of a line going through the principal point $(0, 0)$ containing the projection of $\mathbf{X}$, regardless of what the radial distortion and focal length are.

The matrix $P_{2\times 4}$ represents the 1D-radial camera. While a projection in such a camera gives less information on the 3D point than what a regular camera does, they are still usefull for multiple view geometry.

Exercise 16. If $\begin{pmatrix} x \\ y \end{pmatrix} \sim P_{2\times 4}\mathbf{X}$ and $P_{2\times 4}$ contains the first two rows of the camera matrix P show that the projection of $\mathbf{X}$ must be on the line $l \sim (y, -x, 0)$.

Exercise 17. If $\mathbf{v} \sim P_{2\times 4}\mathbf{X}$ show that $\mathbf{X}$ must be on the 3D plane $\pi \sim P_{2\times 4}^T \mathbf{v}_\perp$, where $\mathbf{v}_\perp$ is a vector that is perpendicular to $\mathbf{v}$.

For practical problems the measurements that we use are still points in images. However to obtain invariance to distortion (and focal length) we only view them as directional vectors of lines known to contain the undistorted projection. In this way one viewed point constrains the 3D point to lie on a plane that depends on the camera parameters. With sufficiently many such planes we can solve multiple view problems.

6.2.1 Triangulation

In this section we consider the problem corresponding to triangulation, that is, determining a 3D point from line observations in several known 1D radial cameras. Since 3 planes in general have a unique intersection point we can solve the problem exactly when a point is visible in 3 cameras.

Exercise 18. Determine $\mathbf{X}$ if

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{X}, \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{X} \text{ and } \begin{pmatrix} 1 \\ -1 \end{pmatrix} \sim \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbf{X}. \tag{6.11}$$

In practical cases we can solve the problem numerically with more than 3 projections using DLT. Given projections $\mathbf{v}_i \sim (x_i, y_i)$ in cameras $P_{2\times 4}^i$, $i = 1, ..., n$ we want to find the unknown 3D point $\mathbf{X}$. To formulate a suitable

matrix formulation we note that

$$\lambda_i \mathbf{v}_i = P^i_{2\times 4}\mathbf{X} \Leftrightarrow \begin{bmatrix} P^i_{2\times 4} & -\mathbf{v}_i \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda_i \end{bmatrix} = 0. \tag{6.12}$$

Using all the points we obtain

$$\begin{bmatrix} P^1_{2\times 4} & -\mathbf{v}_1 & 0 & 0 & \dots \\ P^2_{2\times 4} & 0 & -\mathbf{v}_2 & 0 & \dots \\ P^3_{2\times 4} & 0 & 0 & -\mathbf{v}_3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = 0. \tag{6.13}$$

In Lecture 4 we mentioned that it is possible to eliminate λ using the vector product. Here it is possible to do the same using a vector $\mathbf{v}_\perp$ that is perpendicular to $\mathbf{v}$. If $\lambda\mathbf{v} = P_{2\times 4}\mathbf{X}$ then $\mathbf{v}_\perp^T P_{2\times 4}\mathbf{X} = \lambda \mathbf{v}_\perp^T \mathbf{v} = 0$. Figure 6.3 shows an example of triangulation with radial distortion.

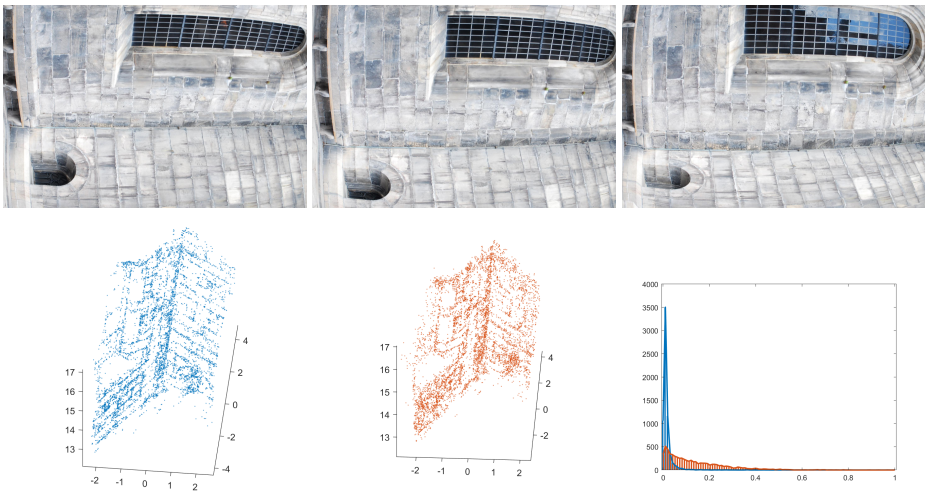


Figure 6.3: Triangulation with radially distorted images. Top row: examples of the images used. Bottom row: Result obtained with the radial camera model (left), result obtained with a regular pinhole camera (middle), histogram over the errors to the ground truth for both approaches (right).

6.2.2 Resection

Next we consider the resection problems for 1D-cameras. Given a number of projections $\mathbf{v}_i \sim (x_i, y_i)$ and corresponding 3D points $\mathbf{X}_i$ we want to find $P_{2\times 4}$ such that

$$\mathbf{v}_i \sim P_{2\times 4}\mathbf{X}. \tag{6.14}$$

Exercise 19. How many degrees of freedom does a 1D-radial camera have?
How may projections are needed to be able to determine a 1D-radial camera?

To derive a DLT formulation we note that

$$\lambda_i \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{bmatrix} P_1^T \\ P_2^T \end{bmatrix} \mathbf{X}_i \Leftrightarrow \begin{bmatrix} \mathbf{X}_i^T & 0 & -x_i \\ 0 & \mathbf{X}_i^T & -y_i \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \lambda_i \end{bmatrix} = 0. \tag{6.15}$$

Using all the points we therefore obtain the homogeneous system

$$\begin{bmatrix} \mathbf{X}_1^T & 0 & -x_1 & 0 & 0 & \dots \\ 0 & \mathbf{X}_1^T & -y_1 & 0 & 0 & \dots \\ \mathbf{X}_2^T & 0 & 0 & -x_2 & 0 & \dots \\ 0 & \mathbf{X}_2^T & 0 & -y_2 & 0 & \dots \\ \mathbf{X}_3^T & 0 & 0 & 0 & -x_3 & \dots \\ 0 & \mathbf{X}_3^T & 0 & 0 & -y_3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{bmatrix} = 0, \tag{6.16}$$

which we can solve with the method presented in Lecture 4. The left image of Figure 6.4 shows the result of running the resection problem on the same data set as in Figure 5.4. Since it is not possible to determine a unique camera center for the 1D radial camera (the nullspace is two dimensional) we can only say that the image was captured somewhere on a line that is parallel to the camera viewing direction. Figure 6.4 shows these lines in purple for every camera.

6.3 Finding K, t_3 and $d(r)$.

When the inner camera and distortion parameters are known it is relatively easy to undistort the image as shown in Figure 6.1. It relies on the fact that using the resection formulation above we can estimate the first 2 rows of the unknown camera matrix independently of radial distortion using DLT. Once this is done the remaining parameters can be found through a linear system of equations. The only assumption we have to make is that the principal point is known, since otherwise we can't solve the resection problem as described above. This is however usually located in the middle of the image and therefore typically known.

From the solution to the resection problem we can extract γ, s, $R_{2\times 3}$ and $t_{2\times 1}$ using RQ-factorization. In addition, when we have the first two rows of a rotation matrix we can compute the third one using the vector product. What remains is now to find f, t_3 and $d(r_d)$. These cannot be determined using the line projections but have to be computed form the regular point projections. Throughout this section we will assume that the extracted 2×4 camera is of the form $P_{2\times 4} = \begin{bmatrix} R_{2\times 3} & t_{2\times 1} \end{bmatrix}$, that is $\gamma = 1, s = 0$ in (6.10). If this is not the case we can multiply the image points with the matrix

$$\begin{pmatrix} \gamma & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{\gamma} & \frac{-s}{\gamma} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.17}$$

Let

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} R & \begin{bmatrix} t_{2\times 2} \\ 0 \end{bmatrix} \end{bmatrix} \mathbf{X}. \tag{6.18}$$

If $t_3 = 0$ these are homogeneous coordinates of x_d. For other values of t_3 the z-coordinate will be $Z + t_3$. We therefore need to find parameters f,$d(r_d)$ and t_3 so that

$$\begin{pmatrix} d(r_d) & 0 & 0 \\ 0 & d(r_d) & 0 \\ 0 & 0 & 1 \end{pmatrix} K^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} X \\ Y \\ Z + t_3 \end{pmatrix}. \tag{6.19}$$

Since $\gamma = 1$ and $s = 0$ we have $K^{-1} = \begin{pmatrix} \frac{1}{f} & 0 & 0 \\ 0 & \frac{1}{f} & 0 \\ 0 & 0 & 1 \end{pmatrix}$ which gives the equations

$$\begin{pmatrix} d(r_d)x/f \\ d(r_d)y/f \\ 1 \end{pmatrix} \sim \begin{pmatrix} X \\ Y \\ Z + t_3 \end{pmatrix} \Leftrightarrow \begin{cases} \frac{d(r_d)}{f} x = \frac{X}{Z+t_3} \\ \frac{d(r_d)}{f} y = \frac{Y}{Z+t_3} \end{cases}. \tag{6.20}$$

Since $\mathbf{x} \sim K\mathbf{x}_d$ we know that $x_d = \frac{x}{f}, y_d = \frac{y}{f}$ and that $r_d = \sqrt{\frac{x^2}{f^2} + \frac{y^2}{f^2}} = \frac{r}{f}$, where $r = \sqrt{x^2 + y^2}$, these are equivalent to

$$\begin{cases} d(\frac{r}{f})\frac{x}{f} = \frac{X}{Z+t_3} \\ d(\frac{r}{f})\frac{y}{f} = \frac{Y}{Z+t_3} \end{cases}. \tag{6.21}$$

Since $\frac{x}{X} = \frac{y}{Y}$ it is not difficult to see that the two equations above are dependent therefore it is enough solving any re-scaled version of one of these. Here we use

$$d(\frac{r}{f})\frac{r}{f} = \frac{\tilde{R}}{Z + t_3}, \tag{6.22}$$

where $\tilde{R} = \sqrt{X^2 + Y^2}$. To proceed further we now need to specify what type of function d is. To obtain a linear system a common choice is $d(r_d) = \frac{1}{p(r_d)}$ where

$$p(r_d) = 1 + k_1 r_d^2 + k_2 r_d^4 + ... + k_n r_d^{2n}. \tag{6.23}$$

Equation (6.22) then implies that

$$(Z + t_3)r = \tilde{R}(f + k_1\frac{r^2}{f} + k_2\frac{r^4}{f^3} + ... + k_n\frac{r^{2n}}{f^{2n-1}}). \tag{6.24}$$

The unknowns are f, t_3 and the coefficients $k_1, ...k_n$. We can collect these in one vector of unknowns by rewriting (6.24) as

$$\begin{pmatrix} \tilde{R} & \tilde{R}r^2 & \tilde{R}r^4 & \dots & \tilde{R}r^{2n} & -r \end{pmatrix} \begin{pmatrix} f \\ k_1/f \\ k_2/f^3 \\ \vdots \\ k_n/f^{2n-1} \\ t_3 \end{pmatrix} = rZ. \tag{6.25}$$

Each point projection gives one such equation. We have $n + 2$ unknowns and we therefore need $n + 2$ point projections to solve the problem.

Once we have solved the problem we can form a complete 3×4 camera for which we can compute a unique camera center. To the right in Figure 6.4 we show the result obtain when applying this approach to the same data as previously. When the function $d(r_d)$ has been determined we can also remove the radial distortion from the image and image points. An example of this is shown in Figure 6.5.

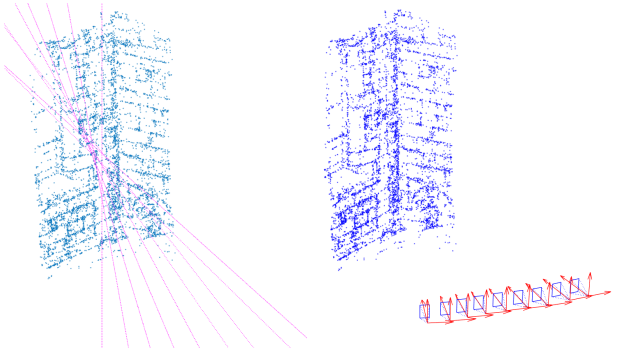


Figure 6.4: Left: Result of running the resection formulation of Section 6.2.2. The radial camera does not have any unique camera center since t_3 is unknown. Varying t_3 gives a line (parallel to the viewing direction) of potential positions where the image could have been captured. Right: The result from the method in Section 6.3.

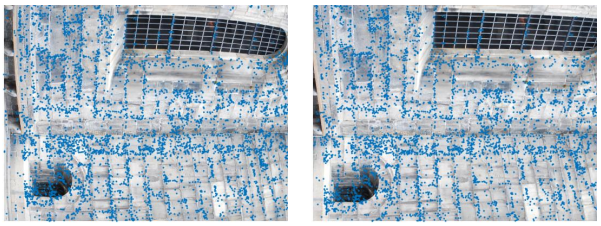


Figure 6.5: After the function $d(r_d)$ has been determined we can remove the radial distortion from the image and image points. The left figure shows the original distorted image with detected points. The right shows the undistributed image.

Lecture 7:

Epipolar Geometry and the Fundamental Matrix

7.1 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

$$\lambda_i \mathbf{x}_i = P_1 \mathbf{X}_i \tag{7.1}$$

$$\bar{\lambda}_i \bar{\mathbf{x}}_i = P_2 \mathbf{X}_i, \tag{7.2}$$

$i = 1, ..., n$, 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.

7.1.1 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 (7.1)-(7.2) 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

$$H = \begin{bmatrix} A_1^{-1} & -A_1^{-1} t_1 \\ 0 & 1 \end{bmatrix} \tag{7.3}$$

to the camera equations (7.1)-(7.2). The camera matrix P_1 is then transformed to

$$P_1 H = \begin{bmatrix} A_1 & t_1 \end{bmatrix} \begin{bmatrix} A_1^{-1} & -A_1^{-1} t_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}. \tag{7.4}$$

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

7.2 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 (7.1)-(7.2).

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 7.1. If we assume that the $\mathbf{X} = \begin{bmatrix} X \\ 1 \end{bmatrix}$, where $X \in \mathbb{R}^3$, are the

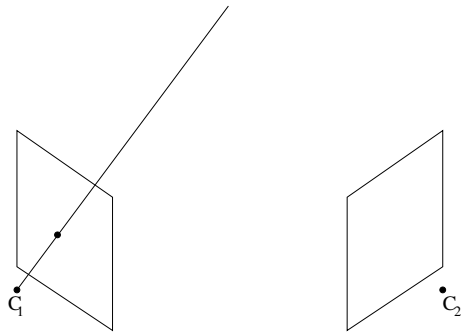


Figure 7.1: All scene points on the line project to the same point in the left camera.

homogeneous coordinates of a scene point projecting to $\mathbf{x}$, then

$$\lambda \mathbf{x} = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = X. \tag{7.5}$$

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

$$\mathbf{X}(\lambda) = \begin{bmatrix} \lambda \mathbf{x} \\ 1 \end{bmatrix}. \tag{7.6}$$

The projection of this line into the second camera is

$$P_2 \mathbf{X}(\lambda) = \begin{bmatrix} A & t \end{bmatrix} \begin{bmatrix} \lambda \mathbf{x} \\ 1 \end{bmatrix} = \lambda A \mathbf{x} + t. \tag{7.7}$$

This is a line in the second image, see Figure 7.2. 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 e_2 of the camera center C_1, see Figure 7.2. The projections of the camera centers e_1 and e_2 are called the **epipoles**.

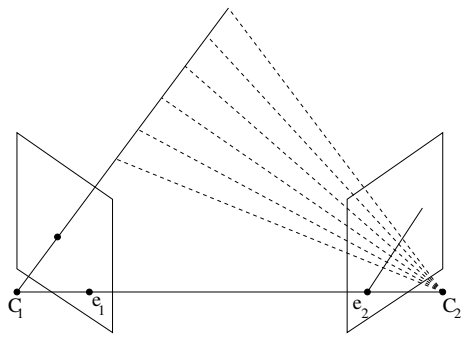


Figure 7.2: The projection of the viewing line into the second camera gives an epipolar line.

Exercise 20. Compute the epipoles for the camera pair $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$ and

$$P_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \tag{7.8}$$

and verify that e_2 lies on the epipolar line of $\mathbf{x} = (0, 1, 1)$.

For a general camera pair $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$, $P_1 = \begin{bmatrix} A & t \end{bmatrix}$, where A invertible we have that $\mathbf{C}_1 = (0, 0, 0, 1)$ and $\mathbf{C}_2 = \begin{bmatrix} -A^{-1}t \\ 1 \end{bmatrix}$ in homogeneous coordinates. Projecting into the two cameras gives $e_2 \sim P_2\mathbf{C}_1 = t$. and $e_1 \sim \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} -A^{-1}t \\ 1 \end{bmatrix} = -A^{-1}t$.

The expression $\lambda A\mathbf{x} + t$ is a parametrization of the epipolar line of $\mathbf{x}$. Note that since $e_2 = t$ we see that e_2 is on the epipolar line by letting $\lambda = 0$. 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

$$\begin{cases} \mathbf{l}^T t = 0 \\ \mathbf{l}^T(A\mathbf{x} + t) = 0 \end{cases}. \tag{7.9}$$

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

$$\mathbf{l} = t \times (A\mathbf{x} + t) = t \times (A\mathbf{x}). \tag{7.10}$$

Since $t \sim e_2$ we can also write this as $e_2 \times (A\mathbf{x})$.

Exercise 21. If $P_1 = [I \quad 0]$ and

$$P_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \tag{7.11}$$

which of the two points $\bar{\mathbf{x}}_1 = (1, 2, 1)$ and $\bar{\mathbf{x}}_2 = (1, 1, 1)$ 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 which we denote $[\mathbf{y}]_\times$. If $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2, y_3)$ then their cross product is

$$\mathbf{y} \times \mathbf{x} = (y_2x_3 - y_3x_2, y_3x_1 - y_1x_3, y_1x_2 - y_2x_1). \tag{7.12}$$

In matrix form we can write this

$$\underbrace{\begin{pmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{pmatrix}}_{=[\mathbf{y}]_\times} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_2x_3 - y_3x_2 \\ y_3x_1 - y_1x_3 \\ y_1x_2 - y_2x_1 \end{pmatrix} \tag{7.13}$$

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

$$\mathbf{l} = e_2 \times (A\mathbf{x}) = [e_2]_\times A\mathbf{x}. \tag{7.14}$$

The matrix $F = [e_2]_\times A$ 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

$$\bar{\mathbf{x}}^T\mathbf{l} = \bar{\mathbf{x}}^T F\mathbf{x} = 0. \tag{7.15}$$

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 22. Let $P_1 = [I \quad 0]$ and

$$P_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}. \tag{7.16}$$

Compute the fundamental matrix F and verify that $Fe_1 = 0$, $e_2^T F = 0$ and $\det(F) = 0$.

7.3 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.

$$\bar{\mathbf{x}}_i^T F \mathbf{x}_i = 0. \tag{7.17}$$

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

$$\begin{aligned} 0 = \bar{\mathbf{x}}_i^T F \mathbf{x}_i = \quad & F_{11}\bar{x}_i x_i + F_{12}\bar{x}_i y_i + F_{13}\bar{x}_i z_i \\ & + F_{21}\bar{y}_i x_i + F_{22}\bar{y}_i y_i + F_{23}\bar{y}_i z_i \\ & + F_{31}\bar{z}_i x_i + F_{32}\bar{z}_i y_i + F_{33}\bar{z}_i z_i. \end{aligned} \tag{7.18}$$

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

$$\underbrace{\begin{pmatrix} \bar{x}_1 x_1 & \bar{x}_1 y_1 & \bar{x}_1 z_1 & \dots & \bar{z}_1 z_1 \\ \bar{x}_2 x_2 & \bar{x}_2 y_2 & \bar{x}_2 z_2 & \dots & \bar{z}_2 z_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{x}_n x_n & \bar{x}_n y_n & \bar{x}_n z_n & \dots & \bar{z}_n z_n \end{pmatrix}}_{M} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ \vdots \\ F_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{7.19}$$

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

$$\min_{\det(F)=0} \|\tilde{F} - F\| \tag{7.20}$$

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

$$USV^T = \tilde{F}, \tag{7.21}$$

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,

$$F = U\text{diag}(\sigma_1, \sigma_2, 0)V^T. \tag{7.22}$$

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 $M^T M$ 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 8:

Extracting Cameras from F

8.1 Computing Cameras From the Fundamental Matrix

In Lecture 7 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

$$\bar{\mathbf{x}}^T F \mathbf{x} = 0 \tag{8.1}$$

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 $P_2 = [A \quad e_2]$ where e_2 is the epipole in the second image. Since we know that $F^T e_2 = 0$ we can find e_2 by computing the null space of F^T. In what follows we will show that $A = [e_2]_\times F$ gives the correct epipolar geometry and therefore solution for the second camera is given by

$$P_2 = [[e_2]_\times F \quad e_2]\,. \tag{8.2}$$

The Fundamental matrix of the camera pair P_1 and P_2 is according to Lecture 5 given by $[t]_\times A = [e_2]_\times [e_2]_\times F$ we need to show that this expression reduces to F. The epipolar line of an arbitrary point $\mathbf{x}$ in image 1 is

$$[e_2]_\times [e_2]_\times F\mathbf{x} = e_2 \times (e_2 \times (F\mathbf{x}))). \tag{8.3}$$

Since e_2 is in the nullspace of F^T it is perpendicular to the columns of F and therefore also the vector $F\mathbf{x}$. This means $\mathbf{v}, e_2, \mathbf{v} \times e_2$, where $\mathbf{v} = \frac{1}{\|F\mathbf{x}\|} F\mathbf{x}$ forms a positive oriented orthonormal basis or $\mathbb{R}^3$. It is now easy to see that

$$e_2 \times (e_2 \times \mathbf{v}) = -e_2 \times (\mathbf{v} \times e_2) = -\mathbf{v}. \tag{8.4}$$

Therefore $e_2 \times (e_2 \times (F\mathbf{x}))) = F\mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^3$, which shows that $[e_2]_\times [e_2]_\times F = F$.

So in conclusion, if we have corresponding points that fulfill the epipolar constraints (8.1) then we can always find 3D points that project to these in the cameras $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$ and $P_2 = \begin{bmatrix} [e_2]_\times F & e_2 \end{bmatrix}$

Exercise 23. Find camera matrices P_1, P_2 such that

$$F = \begin{pmatrix} -1 & 0 & -1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and a 3D point X that projects to $x = (0, 1, 1)$ in P_1 and $x_2 = (1, 1, 1)$ in P_2.

The choice $P_2 = [[e_2]_\times F \quad e_2]$ may seem a little strange since the matrix $[e_2]_\times F$ has the nullspace e_1. Therefore we have

$$0 = [[e_2]_\times F \quad e_2] \begin{bmatrix} e_1 \\ 0 \end{bmatrix}, \tag{8.5}$$

which means that P_2's camera center is a point at infinity. Since there is a projective ambiguity there are however many choices for P_2. Given that $P_1 = [I \quad 0]$ the general formula for P_2 is

$$P_2 = \left[[e_2]_\times F + e_2 v^T \quad \lambda e_2\right], \tag{8.6}$$

where v is some vector in $\mathbb{R}^3$ and λ is a non-zero scalar. It is easy to verify that this camera pair gives the correct fundamental matrix similar to what we did previously.

Exercise 24. If F is as in Ex 1 is there v and λ such that

$$P_2 = \left[[e_2]_\times F + e_2 v^T \quad \lambda e_2\right] = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}?$$

Figure 8.1 shows an example of a reconstruction obtained using the above method. Blue $*$ are the image measurements and red o are the reprojections. While the reprojections appear to be correct the 3D point could still does not look the way we expect it to do. This is because the solution we obtain is only unique up to a unknown projective transformation. Such transformations can severely distort the results. In this case P_2 that has a camera center at infinity and points that are both in front an behind P_1 which makes the results look strange even though the camera equations are solved. To address these issues we will consider how to find a projective transformation that makes all cameras finite and places all the points in front of them.

Figure 8.1: Solution to the two view problem. 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).

8.2 Quasi Affine Upgrade

In this section we will introduce the notion of quasi affine homographies. These are homographies that do not transform any part of the reconstruction to infinity. Figure 8.2 shows an example of a 2D point set $\{\mathbf{x}_i\}$ that is

being transformed by 3 different homographies all of the form

$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & 1 \end{pmatrix}. \tag{8.7}$$

The transformation $H\mathbf{x}_i$ is a point at infinity if its third element is 0, that is, $l^T\mathbf{x}_i = 0$ where $l^T = (l_1, l_2, 1)$. Therefore l is the line of all points that will be mapped to infinity by the transformation. This line is shown in blue in Figure 8.2. Furthermore, the convex hull of the point set (that is, the smallest convex set that contains the point set $\{\mathbf{x}_i\}$) is shown in read. As long as l does not intersect the convex hull this is preserved by the mapping (that is, the points on the boundary of convex hull before transformation are also on the convex hull in the transformed space). However, when the line intersects the convex hull significant distortion is introduced and this is no longer the case. To remove this kind of distortion from reconstructions obtained from the fundamental matrix we will use quasi affine transformations.

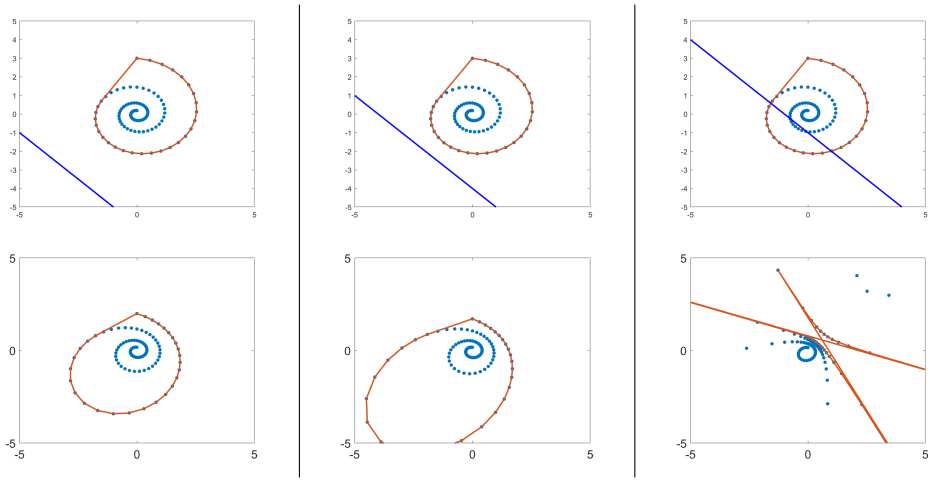


Figure 8.2: Projective transformation of 2D points. First row: original points (blue) and their convex envelope (red). Second row: transformed points (blue) and the transformed convex envelope (red). The two first transformations are quasi affine with respect to the blue points while the third is not.

8.2.1 Quasi Affine Transformations

The examples we showed in the previous section concern 2D projective transformations. However the same principle applies to 3D. When the plane that is mapped to infinity intersects the convex hull of the set of 3D points the resulting geometry will look strange. This is also what has happened in Figure 8.1. In the remainder of this section we will consider sets of 3D. The main purpose is to upgrade the solution from the two view problem to a more reasonably looking one.

A homography H mapping a set of finite points $\mathbf{X}_i = \begin{bmatrix} X_i \\ 1 \end{bmatrix}$ to corresponding points $\mathbf{Y}_i = \begin{bmatrix} Y_i \\ 1 \end{bmatrix}$ is called quasi affine with respect to (Q.A w.r.t) the set $\{\mathbf{X}_i\}$ if

$$H\mathbf{X}_i = \gamma_i \mathbf{Y}_i, \tag{8.8}$$

where γ_i have the same sign for all i.

If a homography is not Q.A w.r.t $\{\mathbf{X}_i\}$ the transformation of the point set generally results in strange geometries. For such a transformation there is always i and j such that

$$H\mathbf{X}_i = \gamma_i \mathbf{Y}_i \tag{8.9}$$
$$H\mathbf{X}_j = \gamma_j \mathbf{Y}_j \tag{8.10}$$

where $\gamma_i < 0$ and $\gamma_j > 0$. Consider the line segment $(1-\lambda)\mathbf{X}_i + (1-\lambda)\mathbf{X}_j$, $0 \leq \lambda \leq 1$ between the two points $\mathbf{X}_i$ and $\mathbf{X}_j$. This is transformed by H to

$$H((1-\lambda)\mathbf{X}_i + (1-\lambda)\mathbf{X}_j) = \lambda\gamma_i \begin{bmatrix} Y_i \\ 1 \end{bmatrix} + (1-\lambda)\gamma_j \begin{bmatrix} Y_j \\ 1 \end{bmatrix}. \tag{8.11}$$

The last coordinate in this expression is $\lambda\gamma_i + (1-\lambda)\gamma_j$. If we let $\lambda = \frac{\gamma_j}{\gamma_j - \gamma_i}$ we get

$$\lambda\gamma_i + (1-\lambda)\gamma_j = \frac{\gamma_j\gamma_i}{\gamma_j - \gamma_i} + (1 - \frac{\gamma_j}{\gamma_j - \gamma_i})\gamma_j = 0. \tag{8.12}$$

Therefore the set of points $H((1-\lambda)\mathbf{X}_i + (1-\lambda)\mathbf{X}_j)$, $0 \leq \lambda \leq 1$ contains a point at infinity. Since all the points $\mathbf{Y}_i \sim H\mathbf{X}_i$ are finite the convex hull of $\{H\mathbf{X}\}$ also consists of finite points, this means that the convex hull of $\{\mathbf{X}_i\}$ is not mapped to the convex hull of $\{H\mathbf{X}_i\}$ by H.

8.2.2 Quasi Affine Reconstructions

Now suppose that we have a camera P and a set of 3D points $\{\mathbf{X}_i\}$ that fulfills

$$\lambda_i \mathbf{x}_i = P\mathbf{X}_i. \tag{8.13}$$

If $\bar{P} \sim PH^{-1}$ and $\bar{\mathbf{X}}_i \sim H\mathbf{X}_i$ when is H Q.A w.r.t the set $\{\mathbf{X}_i\}$? There are non-zero constants c and γ_i such that $\bar{P} = cPH^{-1}$ and $\gamma_i\bar{\mathbf{X}}_i = H\mathbf{X}_i$ therefore we have

$$\lambda_i\mathbf{x}_i = P\mathbf{X}_i = PH^{-1}H\mathbf{X}_i = \frac{\gamma_i}{c}\bar{P}\bar{\mathbf{X}}_i \quad \Leftrightarrow \quad \underbrace{\frac{\lambda_i c}{\gamma_i}}_{:=\bar{\lambda}_i}\mathbf{x}_i = \bar{P}\bar{\mathbf{X}}_i. \tag{8.14}$$

Since c does not depend on i we se that $\bar{\lambda}_i$ has the same sign or the opposite sing as λ_i forall i when $\gamma_i > 0$ for all i. Therefore we have

$$\text{sign}(\lambda_1\bar{\lambda}_1, \lambda_2\bar{\lambda}_2, ..., \lambda_n\bar{\lambda}_n) = \pm(1, 1, ..., 1), \tag{8.15}$$

when H is Q.A. w.r.t $\{\mathbf{X}_i\}$. Conversely, when H is a general homography γ_i will not have the same sign for all i, and therefore the signs of $\bar{\lambda}_i$ will vary with i.

Now suppose that we have a "true" metric reconstruction of a scene (with finite scene points). In particular the scene points $\{\mathbf{X}_i\}$ are in front of the cameras. If $\mathbf{X}_i = \begin{bmatrix} X_i \\ 1 \end{bmatrix}$ and one of cameras is $P = \begin{bmatrix} A & a \end{bmatrix}$ then we have

$$0 < \text{depth}(P, \mathbf{X}_i) = \frac{\det(A)\lambda_i}{\|A_3\|} \tag{8.16}$$

Since $\frac{\det(A)}{\|A_3\|}$ is independent of i it is clear that λ_i, $i = 1, ..., n$ all have the same sign. Then any other projective reconstruction with $\bar{P} \sim PH^{-1}$, $\bar{\mathbf{X}}_i \sim H\mathbf{X}_i$ and $\bar{\lambda}_i\mathbf{x}_i = \bar{P}\bar{\mathbf{X}}_i$, that has same sign for all λ_i is related to the true solution via a quasi-affine transformation (w.r.t $\mathbf{X}_i$). We call such a solution a quasi-affine reconstruction.

Note that if all points are visible in all cameras it is enough to check the signs of one camera to determine if H is quasi-affine. Since the depths of the points are in front of the cameras it is easy to see that the sign condition also will hold in the other cameras. Furthermore, a quasi affine reconstruction does not necessarily have points in front of the cameras since we do not know how $\text{sign}(\det(A))$ is changed by H. The only thing that is guaranteed is that no point in the convex envelope of $\{\mathbf{X}\}$ is transformed to infinity.

8.2.3 A Quasi Affine Solution to The Two View Problem

In the case of the two view problem we have seen in Section 8.1 how to obtain a projective solutions where $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$ and $P_2 = \begin{bmatrix} [e_2]_\times F & e_2 \end{bmatrix}$. In this case it is easy to obtain a quasi affine reconstruction (w.r.t a "true" metric point cloud) using the following three steps:

- Switch the third and fourth column of every camera.
- Switch the third and fourth coordinates of every point.
- Divide every 3D-point by its fourth coordinate.

Note that the projections are not changed by the first two step since these correspond to transforming the reconstruction with the homograpy

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{8.17}$$

Neither does the third step since this only rescales the homogeneous coordinates of the 3D points. After the transformation the first camera is

$$P_1 T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8.18}$$

Since the scene points all have last coordinate 1 it is clear that multiplication with this camera matrix gives all $\lambda_i = 1$. Therefore the resulting projective reconstruction is related to the true metric solution via some quasi affine transformation H. Figure 8.3 shows the resulting structure after applying the above algorithm to the reconstruction in Figure 8.1. Since $P1$ is has its camera center at a vanishing point we only plot the resulting structure.

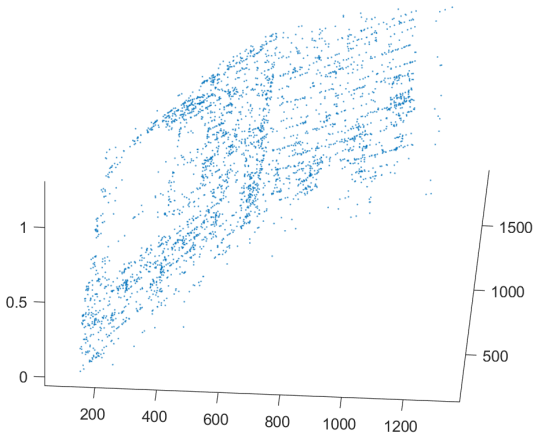


Figure 8.3: New structure after quasi affine upgrade. Note that axes scales are varied. This should be expected since the reconstruction is not metrically accurate. The only thing that is guaranteed by a quasi affine reconstruction is that the convex hull of the points is preserved.

8.2.4 Finite Cameras and Positive Depth

While the structure of the above reconstruction typically looks more reasonable than a general projective solution it has some unreasonable features. Firstly, the camera (8.18) is affine and therefore has a camera center at infinity. For such cameras we can not talk about depth of a point since the length $\|(p_{31}, p_{32}, p_{33})\| = 0$. Secondly, there is no guarantee that the second camera $P_2 T^{-1}$ has points in front of it. In this section we address these two issues under the assumption that the determinant of the first 3 columns of $P_2 T^{-1}$ is non-zero.

To address the second issue one may simply change the signs of the third column of the camera and the third

coordinate of every point or equivalently applying the transformation

$$\bar{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.19}$$

to the reconstruction. This changes sign of the determinant of the first 3×3 part of the second camera. At the same time the multiplication of $P_2 T^{-1} \bar{T}^{-1} \bar{T} T \mathbf{X}_i = P_2 \mathbf{X}_i$ which means that the third coordinate of the multiplication is unchanged resulting in new depths with opposite signs.

Note that $P_1 T^{-1} \bar{T}^{-1} = P_1 T^{-1}$ is still at infinity. To fix this we apply a final transformation

$$\tilde{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\epsilon & 1 \end{bmatrix}, \tag{8.20}$$

with $\epsilon > 0$ which gives

$$P_1 T^{-1} \bar{T}^{-1} \tilde{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \epsilon & 1 \end{bmatrix}. \tag{8.21}$$

This is a regular camera with camera center $(0, 0, -\frac{1}{\epsilon}, 1)$. Note that the determinant of the first 3×3 camera is $\epsilon > 0$. For the second camera we have

$$P_2 T^{-1} \bar{T}^{-1} \tilde{T}^{-1} = P_2 T^{-1} \bar{T}^{-1} (I + \epsilon E_{43}), \tag{8.22}$$

where E_{43} is a 4×4 matrix with a one at element $(4, 3)$ and zeros everywhere else. Since the determinant of the first 3×3 block of $P_2 T^{-1} \bar{T}^{-1}$ is positive the same will be true for $P_2 T^{-1} \bar{T}^{-1} \tilde{T}^{-1}$ when ϵ is small enough. Similarly one can check that the other requirements for a positive depth is fulfilled for small enough ϵ. Figure 8.4 shows the result of applying the above procedure to the reconstruction in Figure 8.3. Note that the structure still appears distorted since there is still a projective ambiguity. However without additional information we cannot resolve this. Ensuring that the image projections are correct and that the points are in front of the cameras is the best we can do if we don't know anything further about either the cameras or the scene. In Figure 8.4 we also show the obtained cameras. Read arrows show the local coordinate systems of each camera. Blue lines show the image borders transformed into the image plane using K^{-1}. Note that none of the images are centred on the principal point which typically the case for regular cameras. Hence introducing constraints on the inner parameters of the cameras can potentially resolve the remaining ambiguities.

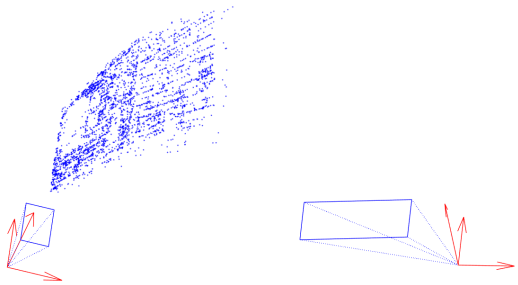


Figure 8.4: Quasi affine reconstruction with finite cameras and points in front of cameras. Read arrows show the local coordinate systems of the cameras. Blue lines show the image borders transformed into the image plane using K^{-1}. Note that for the camera to the right the principal point is not in the image.

8.3 Autocalibration

In general the inner parameters of a camera cannot be extracted from an uncalibrated reconstruction alone. Without any additional knowledge than the image projections the best we can do is a quasi affine reconstruction with points in front of the camera. However by adding some extra information it may be possible to extract the true camera parameters. This is called Autocalibration.

We suppose that we have a projective reconstruction $P, \bar{P}, \{\mathbf{X}_i\}$ of a two view problem and want to find an upgrading homography H that gives new cameras

$$K \begin{bmatrix} R & t \end{bmatrix} \sim PH \tag{8.23}$$

$$\bar{K} \begin{bmatrix} \bar{R} & \bar{t} \end{bmatrix} \sim \bar{P}H, \tag{8.24}$$

where K and $\bar{K}$ has some desirable properties. The approach relies on the fact that KK^T can be directly extracted (without the need to compute an RQ-factorization) using the observation that

$$K \begin{bmatrix} R & 0 \end{bmatrix} \sim PHD, \tag{8.25}$$

where

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{8.26}$$

Therefore

$$P \underbrace{HDH^T}_{:=\Omega} P^T \sim K \begin{bmatrix} R & 0 \end{bmatrix} \begin{bmatrix} R^T \\ t^T \end{bmatrix} K^T = KRR^TK^T = KK^T. \tag{8.27}$$

Hence KK^T (and similarly $\bar{K}\bar{K}^T$) is linear in the elements of the matrix Ω up to an unknown scaling and can be determined from P and Ω without the need for RQ factorization. The matrix Ω is symmetric and positive semi definite with rank 3. The main idea is to add a few reasonable constraints on KK^T and $\bar{K}\bar{K}^T$ and solve for Ω. Once Ω been determined we can compute H to upgrade P,$\bar{P}$ and $\{\mathbf{X}_i\}$. We have

$$KK^T = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \gamma f & 0 & 0 \\ sf & f & 0 \\ x_0 & y_0 & 1 \end{pmatrix} = \begin{pmatrix} (\gamma^2+s^2)f^2 & s^2f^2 + x_0y_0 & x_0 \\ s^2f^2 + x_0y_0 & f^2 + y_0^2 & y_0 \\ x_0 & y_0 & 1 \end{pmatrix}. \tag{8.28}$$

Here we will assume that $x_0 = y_0 = s = 0$, $\gamma = 1$. Note that $s = 0$ and $\gamma = 1$ are typically reasonable assumptions whereas (x_0, y_0) is typically the middle of the image. Therefore a coordinate change of the image coordinates using

$$N = \begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{8.29}$$

is required to make this assumption valid. When all the assumptions above are enforced we get

$$KK^T = \begin{pmatrix} f^2 & 0 & 0 \\ 0 & f^2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \sim P\Omega P^T, \tag{8.30}$$

and similarly for the other camera. This gives us 4 constraints on the matrix $w := P\Omega P^T$, namely, $w_{11} = w_{22}$, $w_{12} = w_{13} = w_{23} = 0$. (Note that w is symmetric by construction and therefore the constraints $w_{12} = w_{13} = w_{23} = 0$ do not give any additional information.) Similarly we get 4 constraints for $\bar{w} = \bar{P}\Omega\bar{P}^T$. The matrix Ω is 4×4 symmetric and therefore has 10 unknown elements, but since the scale is arbitrary it only has 9 DOF. With the constraint $\det(\Omega) = 0$ we can therefore find Ω. To avoid the trivial solution $\Omega = 0$ we can fix the scale, for example by setting $w_{33} = 1$.

8.3.1 Finding Ω

To solve the problem we stack the unknowns of Ω in a vector ω and obtain a system of the form

$$A\omega = b \tag{8.31}$$

$$\det(\Omega) = 0, \tag{8.32}$$

where A has size 9×10. The solutions to (8.31) are of the form

$$\omega_p + \lambda\omega_h, \tag{8.33}$$

where ω_p is one solution to (8.31) and ω_h is in the nullspace of A. (In MATLAB ω_p and ω_h can be found by taking `A\b` and `null(A)`.) The matrix expression $\Omega_p + \lambda\Omega_h$ corresponding to (8.33) should now fulfil $\det(\Omega_p + \lambda\Omega_h) = 0$. This corresponds to a generalized eigenvalue problem

$$\Omega_p v = -\lambda\Omega_h v \tag{8.34}$$

which can be solved using `eig(Omega_p,-Omega_h)`. There are four solutions to the eigenvalue problem however typically only two of these result in solutions with rank 3. The expression $\Omega_p + \lambda\Omega_h$ can be seen as a line that intersects the convex set of positive semi definite matrices. It is easy to see that there has to be either $0, 1, 2$ or infinitely many intersection points at the boundary of the convex set, which consists of matrices with determinant 0. In general there will be two distinct intersection points that correspond to two possible solutions for Ω.

The above approach is sensitive to noise and therefore normalization is important to achieve reasonable results. In addition to moving the principal point it is therefore a good idea to rescale the image coordinates using a matrix of the type $\begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}$ after applying (8.29). Note however that the particular normalization we choose must not invalidate the constraints used to form A.

8.3.2 Finding H

After Ω has been determined we also need to find H such that $\Omega = HDH^T$. This can be done by taking the eigenvalue decomposition of Ω. If $\Omega = V\Lambda V^T$, with

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{8.35}$$

and $\lambda_i > 0$, then

$$H = V \left(\sqrt{\Lambda} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d \end{pmatrix} \right), \tag{8.36}$$

is a valid solution for any $d \neq 0$. Note that the second term vanishes when multiplying with D. However for H to be invertible it needs to be non-zero. Its size effects the scale of the scale of the reconstruction and cannot be determined uniquely. The sign of d is more important as this effects whether points are in front of the camera or not. From the two solutions for Ω and the two possible signs for each solution we get four possible solutions. It can be shown that one of these combinations give points in front of both cameras. We will discuss these solutions further in the context of the Essential matrix. Figure 8.5 shows the results obtained when applying the above process to the reconstruction in Figure 8.1.

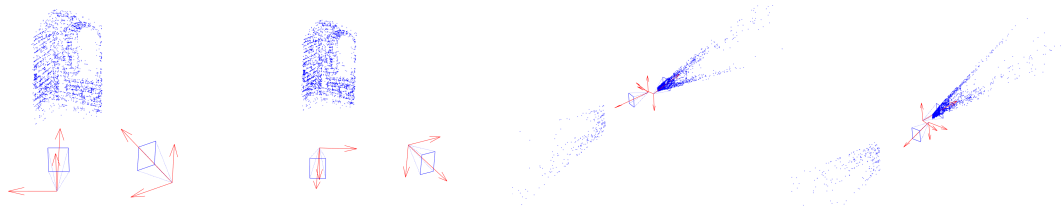


Figure 8.5: The four solutions obtained from the autocalibration process. Note that all cameras seem to have principle points in the middle of the image, but only one have points in front of the camera.

Lecture 9:

The Essential Matrix

9.1 Relative Orientation: The Calibrated Case

When solving the relative orientation problem without camera calibration there is, as we has seen in previous lectures, 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. To remove this ambiguity one has to add additional knowledge about the solution to the problem. In Lecture 8 we added some constraints on the inner parameters and used auto calibration to find an upgrading projective transformation. Alternatively, 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.

If the inner parameters K_1 and K_2 are completely 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

$$\mathbf{x}_i \sim K_1[R_1 \quad t_1]\mathbf{X}_i \tag{9.1}$$

$$\bar{\mathbf{x}}_i \sim K_2[R_2 \quad t_2]\mathbf{X}_i, \tag{9.2}$$

and R_1,R_2 are rotation matrices.

We can make two simplifications to the problem. First we normalize the cameras by multiplying equations (9.1) and (9.2) with K_1^{-1} and K_2^{-1} respectively. Furthermore, we apply the euclidean transformation

$$H = \begin{bmatrix} R_1^T & -R_1^T t_1 \\ 0 & 1 \end{bmatrix} \tag{9.3}$$

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

$$P_1 H = \begin{bmatrix} R_1 & t_1 \end{bmatrix} \begin{bmatrix} R_1^T & -R_1^T t_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix} \tag{9.4}$$

$$P_2 H = \begin{bmatrix} R_2 & t_2 \end{bmatrix} \begin{bmatrix} R_1^T & -R_1^T t_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \underbrace{R_2 R_1^T}_{=R} & \underbrace{-R_2 R_1^T t_1 - t_2}_{=t} \end{bmatrix}. \tag{9.5}$$

Therefore we search for a solution to the equations

$$\mathbf{y}_i \sim [I \quad 0]\mathbf{X}_i \tag{9.6}$$

$$\bar{\mathbf{y}}_i \sim [R \quad t]\mathbf{X}_i, \tag{9.7}$$

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. The requirement that R is a rotation now limits the type of projective ambiguity that can occur. Basically, to retain the rotation constraint any projective transformation acting on the scene has to be a similarly therefore we will not observe the same kind of strange deformations that we have seen in the uncalibrated case.

9.1.1 The Essential Matrix

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

$$E = [t]_\times R, \tag{9.8}$$

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

$$E = U\text{diag}([1 \quad 1 \quad 0])V^T. \tag{9.9}$$

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 (9.9) 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

$$-E = U\text{diag}([1 \quad 1 \quad 0])(-V)^T. \tag{9.10}$$

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

$$\min_{\|v\|^2=1} \|Mv\|^2 \tag{9.11}$$

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).

9.2 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. For this purpose we will begin to

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

find a decomposition of $\text{diag}(\begin{bmatrix}1 & 1 & 0\end{bmatrix}) = ZW$, where Z is skew symmetric and W is a rotation. Since W is orthogonal we have $\text{diag}(\begin{bmatrix}1 & 1 & 0\end{bmatrix}) = ZW \Leftrightarrow \text{diag}(\begin{bmatrix}1 & 1 & 0\end{bmatrix})W^T = Z \Leftrightarrow$

$$\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}w_{11} & w_{21} & w_{31}\\ w_{12} & w_{22} & w_{32}\\ w_{13} & w_{23} & w_{33}\end{pmatrix} = \begin{pmatrix}w_{11} & w_{21} & w_{31}\\ w_{12} & w_{22} & w_{32}\\ 0 & 0 & 0\end{pmatrix} = \begin{pmatrix}0 & -z_3 & z_2\\ z_3 & 0 & -z_1\\ -z_2 & z_1 & 0\end{pmatrix}. \tag{9.12}$$

By inspecting the individual elements we see that $w_{11} = w_{22} = 0$, $w_{31} = z_2 = 0$, $w_{32} = -z_1 = 0$, and $w_{12} = z_3 = -w_{21}$. Since W is a rotation (with columns of length 1) it is clear that $w_{12} = \pm 1$. Choose $w_{12} = Z_3 = -1$ give $w_{21} = 1$ and because the third column of W is the vector product of the first two we get the solution

$$W = \begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}. \tag{9.13}$$

Similarly if we chose $w_{12} = 1$, we see that second solution is given by W^T and Z^T.

To decompose E we now note that

$$E = U\text{diag}([1 \quad 1 \quad 0])V^T = UZWV^T = \underbrace{UZU^T}_{:=S_1}\underbrace{UWV^T}_{:=R_1} \tag{9.14}$$

and similarly

$$E = U\text{diag}([1 \quad 1 \quad 0])V^T = UZ^TW^TV^T = \underbrace{UZ^TU^T}_{:=S_2}\underbrace{UW^TV^T}_{:=R_2} \tag{9.15}$$

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

$$R_1^TR_1 = (UW^TV^T)^TUW^TV^T = VWU^TUW^TV^T = I \tag{9.16}$$

R_1 is orthogonal. Furthermore,

$$\det(R_1) = \det(UW^TV^T) = \det(U)\det(W^T)\det(V^T) = \det(W)\det(UV^T) = 1, \tag{9.17}$$

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

$$-S_1^T = (UZU^T)^T = UZ^TU^T = -UZU^T = S_1, \tag{9.18}$$

and therefore S_1R_1 is a valid decomposition. $E = S_2R_2$ can be verified similarly.

9.2.1 The Twisted Pair

To create the camera matrices corresponding to the solutions S_1R_1 and S_2R_2 we need to extract a translation vectors from the skew symmetric matrices S_1 and S_2. We note that since

$$S_1 = UZU^T = -UZ^TU^T = -S_2, \tag{9.19}$$

these are the same up to a scaling and it is therefore enough to determine t from S_1. Since $[t]_\times t = 0$ the vector t must be in the nullspace of S_1 which is the third column u_3 of U. We therefore obtain the solutions $P_1 = \begin{bmatrix}I & 0\end{bmatrix}$

$$P_2 = \begin{bmatrix}UWV^T & u_3\end{bmatrix} \text{ or } P_2' = \begin{bmatrix}UW^TV^T & u_3\end{bmatrix}. \tag{9.20}$$

The two cameras P_2 and P_2' are called the twisted pair. The relative rotation between P_2 and P_2' is the rotation $R_1^TR_2 = VW^TU^TUW^TV^T = VW^TW^TV^T$. It is not hard to verify that if v_3 is the third column of V then $VW^TW^TV^Tv_3 = v_3$ and therefore v_3 is the rotation axis of $R_1^TR_2$. The rotation angle is given by

$$\cos(\phi) = \frac{\text{tr}(R_2^TR_1) - 1}{2} = \frac{\text{tr}(W^TW^T) - 1}{2} = -1, \tag{9.21}$$

which yields $\phi = \pi$. The camera centers of the two cameras P_2 and P_2' are

$$-R_1^T u_3 = -VWU^T u_3 = -VW \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = -V \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = -v_3 \tag{9.22}$$

and

$$-R_2^T u_3 = -VW^T U^T u_3 = -VW \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = -V \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = -v_3 \tag{9.23}$$

respectively. Hence in both solutions we are moving from the center of P_1 which is the origin in the direction $-v_3$. While P_2' is rotated 180° around this direction with respect to P_2.

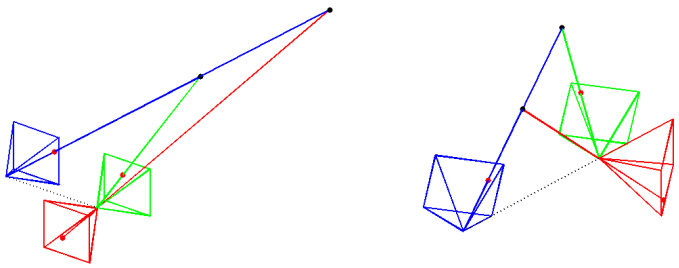


Figure 9.1: Two examples of twisted pair solution. Note that one of P_2 (green) and P_2' (red) has the reconstructed 3D behind itself.

9.2.2 Scale Ambiguity

Note that if $E = [t]_\times R$ then $\lambda E = [\lambda t]_\times R$ is also a valid solution. 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, see Figure 9.2. To make sure that we can find a solution where the points are in front of both the cameras we therefore test $\lambda = \pm 1$ and the twisted solution.

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

$$P_2 = \quad [UWV^T \quad u_3] \quad \text{or} \quad [UW^TV^T \quad u_3] \qquad (\text{from } \lambda = 1) \tag{9.24}$$

$$\text{or} \quad [UWV^T \quad -u_3] \quad \text{or} \quad [UW^TV^T \quad -u_3] \qquad (\text{from } \lambda = -1) \tag{9.25}$$

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 9.3 shows the four calibrated reconstructions of a scene. Only one of them has all the points in front of both the cameras.

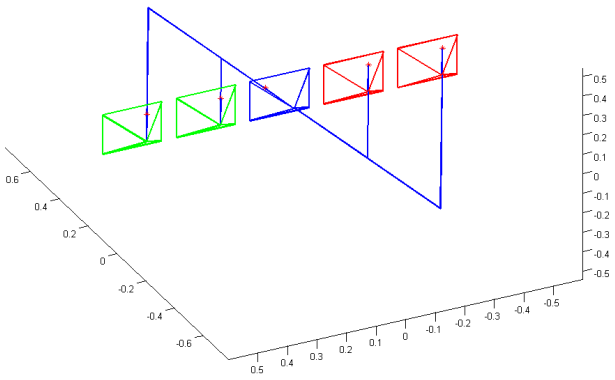


Figure 9.2: Reconstruction with different values of λ. Note that changing sign of λ moves the reconstructed points that are front of the camera to the rear of it and vice versa.

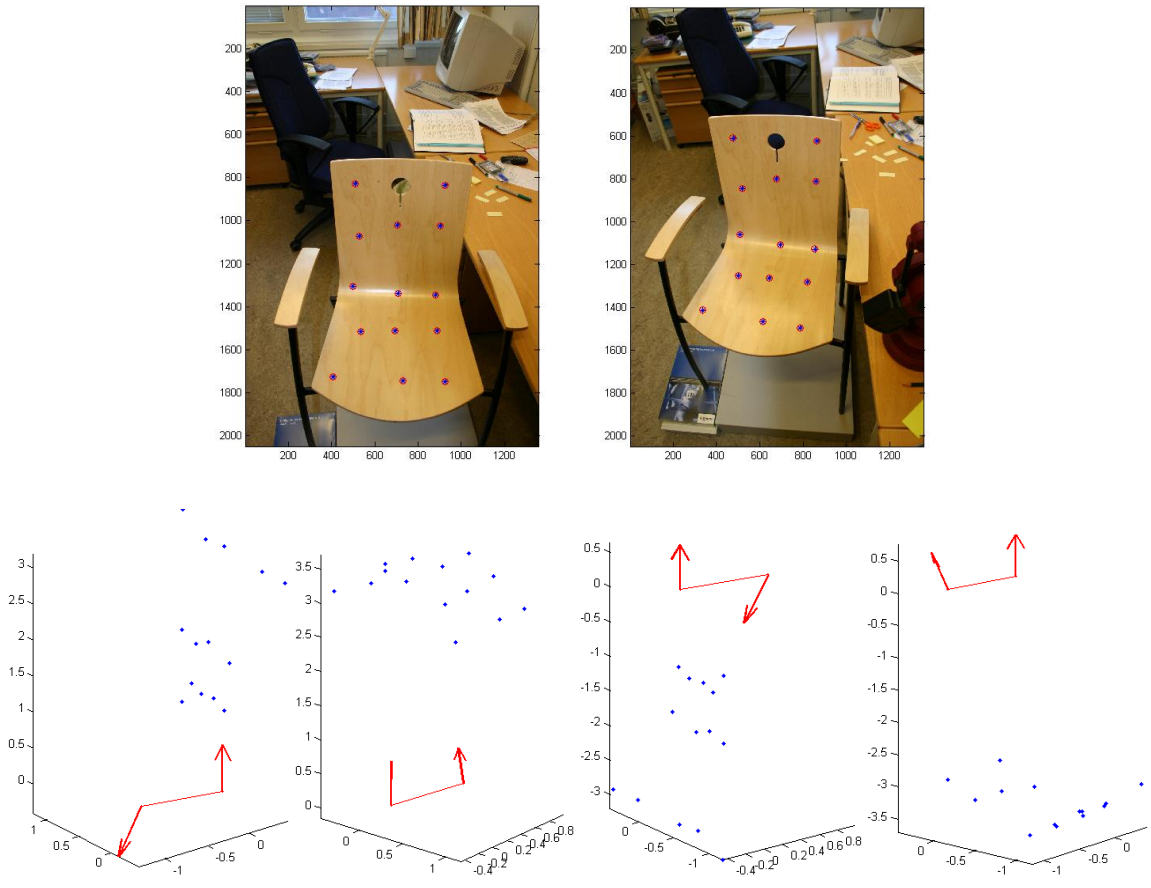


Figure 9.3: The 4 solutions when solving calibrated structure from motion for the chair image images in the first row. Only the second one has positive depths.

Lecture 10:

Model Fitting

10.1 Noise Models

In previous lectures we have addressed various problems using linear formulations that approximately solve the governing algebraic equations. While this is an easy approach we can in general not expect that these equations can be solved exactly. Since the appearance of a patch changes when the viewpoint changes, exact positioning of corresponding points is not possible, see Figure 10.1. Additionally, since matching is done automatically we have to expect that some matches are incorrect causing large deviations from the model. Therefore, our point measurements will in practice always be corrupted by noise of various forms and levels and in general approximate solutions based on DLT will not give the "best" possible fit to the observed 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.

Figure 10.1: 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.

10.2 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.

10.2.1 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,

$$y_i = \tilde{y}_i + \epsilon_i \tag{10.1}$$

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

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}} e^{-\epsilon_i^2/2}. \tag{10.2}$$

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

$$p(\epsilon) = \prod_{i=1}^{n} p(\epsilon_i), \tag{10.3}$$

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

$$p(\epsilon) = \prod_{i=1}^{n} p(\epsilon_i) = \prod_{i=1}^{n} p(y_i - \tilde{y}_i) = \prod_{i=1}^{n} p(y_i - (ax_i + b)) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-(y_i - (ax_i + b))^2/2}. \tag{10.4}$$

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

$$\log\left(\prod_{i=1}^{n} p(\epsilon_i)\right) = -\sum_{i=1}^{n} \frac{(y_i - (ax_i + b))^2}{2} + \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}}\right). \tag{10.5}$$

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

$$\sum_{i=1}^{n} (y_i - (ax_i + b))^2. \tag{10.6}$$

In matrix form we can write this as

$$\left\| \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{=A} \begin{pmatrix} a \\ b \end{pmatrix} - \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{=B} \right\|^2. \tag{10.7}$$

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

$$\begin{pmatrix} a \\ b \end{pmatrix} = (A^T A)^{-1} A^T B, \tag{10.8}$$

which we will derive in Lecture 9. The geometric interpretation of (10.6) is that under this noise model the vertical distance between the line and the measurement should be minimized, see Figure 10.2.

10.2.2 Total Linear Least Squares

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

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} + \delta_i, \tag{10.9}$$

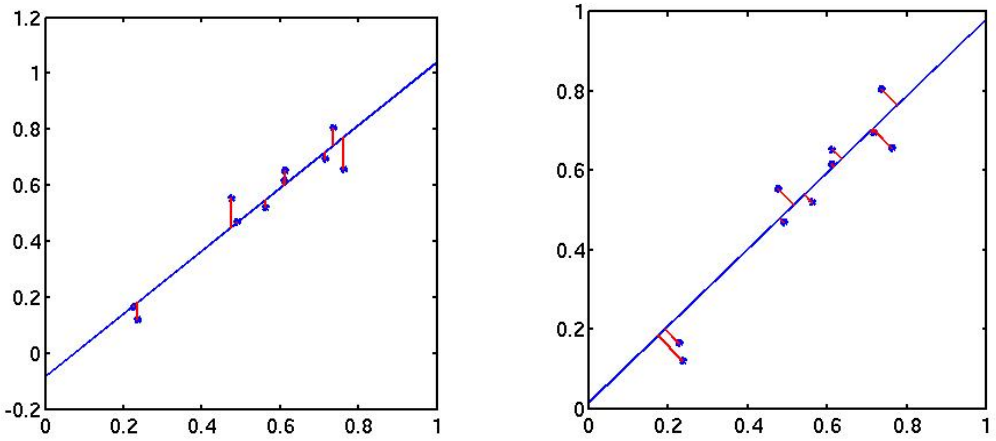


Figure 10.2: Left: The vertical distances between the line and the measured points are minimized in (10.6). In contrast, the minimal distances between the line and the measured points are minimized in (10.12).

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

$$p(\delta_i) = \frac{1}{2\pi} e^{-\|\delta_i\|^2/2}. \tag{10.10}$$

The log likelihood function is

$$\sum_{i=1}^{n} \log(p(\delta_i)) = -\sum_{i=1}^{n} \frac{(x_i - \tilde{x}_i)^2 + (y_i - \tilde{y}_i)^2}{2} + \sum_{i=1}^{n} \log(\frac{1}{2\pi}). \tag{10.11}$$

Therefore, to maximize the likelihood we need to minimize

$$\sum_{i=1}^{n} ((x_i - \tilde{x}_i)^2 + (y_i - \tilde{y}_i)^2), \tag{10.12}$$

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 (10.12) we can restrict it to be the closest point on the line. The expression (10.12) then becomes the distance between (x_i, y_i) and the line. This distance can be expressed using the distance formula as

$$\frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}}. \tag{10.13}$$

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

$$\min \quad \sum_{i=1}^{n} (ax_i + by_i + c)^2 \tag{10.14}$$

$$s.t. \quad a^2 + b^2 = 1. \tag{10.15}$$

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

Solving the Total Least Squares Problem

To solve (10.14),(10.15) we first take derivatives with respect to c. This shows that the the optimal solution must fulfill

$$c = -(a\bar{x} + b\bar{y}), \tag{10.16}$$

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

$$(\bar{x}, \bar{y}) = \frac{1}{n}\sum_{i=1}^{n}(x_i, y_i). \tag{10.17}$$

Substituting into (10.14) we get

$$\min \quad \sum_{i=1}^{m}(a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 \tag{10.18}$$

$$\text{such that} \quad 1 - (a^2 + b^2) = 0. \tag{10.19}$$

By forming the matrix

$$M = \sum_{i=1}^{m} \begin{pmatrix} (x_i - \bar{x})^2 & (x_i - \bar{x})(y_i - \bar{y}) \\ (x_i - \bar{x})(y_i - \bar{y}) & (y_i - \bar{y})^2 \end{pmatrix}, \tag{10.20}$$

we can write this problem as

$$\min \quad t^T M t \tag{10.21}$$

$$\text{such that} \quad 1 - t^T t = 0, \tag{10.22}$$

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

$$\min \quad f(t) \tag{10.23}$$

$$\text{such that} \quad g(t) = 0. \tag{10.24}$$

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

$$\nabla f(t) + \lambda \nabla g(t) = 0. \tag{10.25}$$

Therefore the solution of (10.21)-(10.22) must fulfill

$$2Mt + \lambda(-2t) = 0 \Leftrightarrow Mt = \lambda t. \tag{10.26}$$

That is, the solution t has to be an eigenvector of the matrix M. Furthermore, inserting into (10.21), and using that $t^T t = 1$ we see that it has to be the eigenvector corresponding to the smallest eigenvalue.

10.2.3 Outliers and Robust Loss-Functions

In structure from motion problems we typically have two types of noise. Appearance changes that makes it hard to exactly localize corresponding points giving rise to displacements that are typically in the order of at most a few pixels. This type of noise is usually modeled as Gaussian. In addition to this we often have mismatches that give rise to very large errors. When minimizing the least squares objective such measurements can severely distort the results. Figure 10.3 shows a line fitting example with one outlier point. Because of the quadratic growth of the least squares criterion (10.6) points that are far away from the estimated line will have a proportionally larger effect on the estimate than points that are close to it. This gives a poor estimate shown in the middle of Figure 10.3. By using a robust loss-function that essentially removes the effect of the outlier it is possible obtain a better estimate of the line as shown to the right in Figure 10.3.

To find a suitable robust loss-function we can replace the assumption of normalized noise with a more general distribution with density function

$$p(\epsilon_i) = ce^{-\rho(\epsilon_i^2)}. \tag{10.27}$$

Here ρ is a function that should be less sensitive to outliers and c is some constant which ensures that the density function integrates to 1. As in Section 10.2.1 we assume that we have 2D measurements (x_i, y_i) of points belonging to a line $y = ax + b$ with noise in the y-coordinate $y_i = \tilde{y}_i + \epsilon_i$. To obtain the maximum likelihood estimate we should then minimize

$$\sum_{i=1}^{n} \rho\left((ax_i + b - y_i)^2\right). \tag{10.28}$$

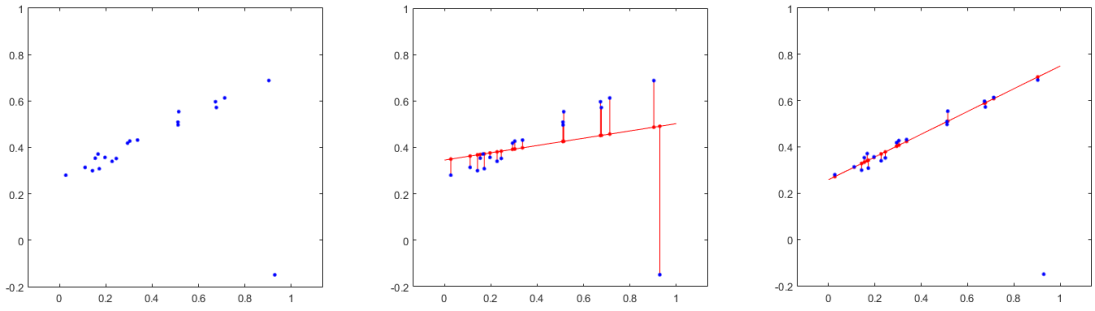


Figure 10.3: Line fitting with one outlier. *Left:* Measurements. *Middle:* Obtained model fit using the least squares objective (10.6). *Right:* Obtained model fit using a robust loss-function.

Taking derivatives with respect to a and b we get

$$\sum_{i=1}^{n} \rho'\left((ax_i + b - y_i)^2\right) 2(ax_i + b - y_i)x_i = 0 \tag{10.29}$$

$$\sum_{i=1}^{n} \rho'\left((ax_i + b - y_i)^2\right) 2(ax_i + b - y_i) = 0 \tag{10.30}$$

respectively. In general these equations lack closed form solutions and have to be solved iteratively. None the less, we can still draw some simple conclusions from these expressions. In matrix form we can also write these two equations $M \begin{pmatrix} a \\ b \end{pmatrix} = m$, where

$$M = \sum_{i=1}^{n} \rho'(\epsilon_i^2) \begin{pmatrix} x_i^2 & x_i \\ x_i & 1 \end{pmatrix} \quad \text{and} \quad m = \sum_{i=1}^{n} \rho'(\epsilon_i^2) \begin{pmatrix} x_i y_i \\ x_i \end{pmatrix}. \tag{10.31}$$

Assuming that we somehow know what ϵ_i is at the optimal values of (a, b) then we have $\begin{pmatrix} a \\ b \end{pmatrix} = M^{-1}m$. We note that M and m can be seen as weighted sums of the matrices $\begin{pmatrix} x_i^2 & x_i \\ x_i & 1 \end{pmatrix}$ and $\begin{pmatrix} x_i y_i \\ x_i \end{pmatrix}$ respectively. If we use $\rho(t) = t$ then (10.28) reduces to the least squares solution and $\rho'(\epsilon_i^2) = 1$. By modifying the weight $\rho'(\epsilon_i^2)$ we can increase or reduce the impact of measurement i on the solution. For example selecting ρ so that its derivative is decreasing will reduce the effects of measurements with large errors. A common approach is to use a threshold τ. If we let $\rho_2(t) = \min(t, \tau^2)$ for some value τ then $\rho_2'(\epsilon_i^2)$ will be zero if $|\epsilon_i|$ larger than τ and therefore this residuals completely disappears from (10.31). In contrast if $|\epsilon_i|$ is smaller than τ then $\rho_2'(\epsilon_i^2) = 1$. (Note that this is the derivative of $\rho_2(t)$ with respect to t where we have inserted $t = \epsilon_i^2$). Therefore this option will essentially remove large residuals and compute the least squares solution of remaining ones, which is what is illustrated to the right in Figure 10.3. Figure 10.4 shows the truncated loss function $\rho_2(\epsilon_i^2)$.

Strictly speaking the function $\min(\epsilon_i^2, \tau^2)$ will not be differentiable at $\epsilon_i = \tau$. However, we can always find approximations that are close to $\min(\epsilon_i^2, \tau^2)$ while still being differentiable. In Figure 10.4 we show $\rho_3(\epsilon_i^2)$ which is one such commonly used approximation.

As we noted above using decreasing derivatives generally makes the loss-function more robust to outliers. On the other hand optimization then becomes more difficult since this often leads to non-convex problems with local minimizers. If we for example use $\sum_{i=1}^{n} \rho_2\left((ax_i + b - y_i)^2\right)$ the derivatives will all be zero for any choice of a and b that makes all error residuals larger than τ. Thus an iterative algorithm, making use of the derivatives, will immediately get stuck if the starting point is not close enough to the global minimizer. Selecting ρ so that the derivatives are decreasing but never reach zero can help to achieve better convergence. The function $\rho_4(\epsilon_i^2)$ in Figure 10.4 shows one such example. However, without convexity there is in general no guarantee that we will

reach the global minimizer. Initialization is therefore an important issue (which we will address in Lecture 8) for these methods.

The last example $h(\epsilon_i) := \rho_5(\epsilon_i^2)$ of Figure 10.4 is the Huber function. If we consider h as a function of ϵ_i it has the derivative

$$h'(\epsilon_i) = \begin{cases} 2\epsilon_i & |\epsilon_i| < b \\ 2b\text{sign}(\epsilon_i) & |\epsilon_i| \geq b \end{cases}. \tag{10.32}$$

Since h' is non-decreasing h is convex. Furthermore, since a sum of a set of convex functions is also convex the line fitting problem defined by (10.28) is convex. Therefore local optimization staring from any initialization is guaranteed to converge to the globally optimal solution. On the other hand the weights $\rho'(\epsilon_i^2)$ are decreasing (and tend to 0 as $\epsilon_i \to \infty$). In that sense this is a good trade off between robustness and convexity. We will study global optimization further in Lecture 10.

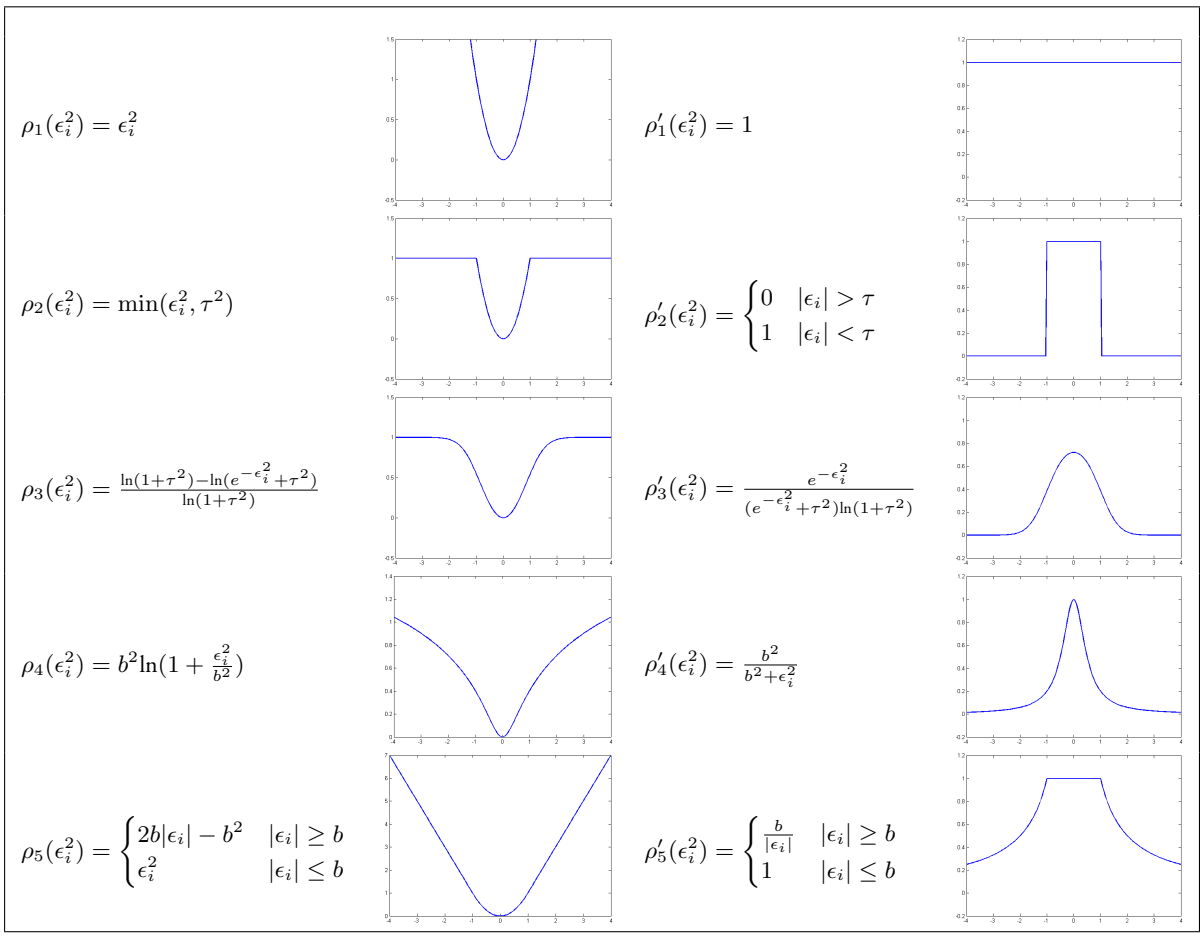


Figure 10.4: Some examples of loss-functions and the resulting weights. (Note that the derivative $\rho_j'(\epsilon_i^2)$ is obtained by differentiating $\rho_j(t)$ with respect to t and inserting $t = \epsilon_i^2$).

10.3 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

$$\left(\frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j}, \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j}\right), \tag{10.33}$$

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,

$$(x_{ij}^1, x_{ij}^2) = \left(\frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j}, \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j}\right) + \delta_{ij}, \tag{10.34}$$

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

$$p(\delta_{ij}) = \frac{1}{2\pi} e^{-\frac{1}{2}||\delta_{ij}||^2}. \tag{10.35}$$

Similarly to Section 10.2.2 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

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} \left\| \left(x_{ij}^1 - \frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j},\ x_{ij}^2 - \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j} \right) \right\|^2. \tag{10.36}$$

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 10.5. 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.

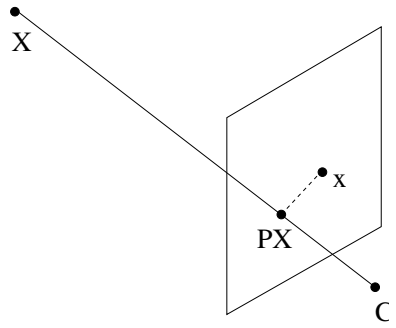


Figure 10.5: Geometric interpretation of the maximum likelihood estimate for projection problems. The dashed distance should be minimized.

10.3.1 Optimal 2-view Triangulation

The objective function (10.36) is in very hard to minimize and in general we are limited to local refinement from a suitably selected starting point. In this section we will study the special case of 2-view triangulation where it can be shown that the problem can be solved by finding the roots of a 6th degree polynomial. We assume that the two cameras P and $\bar{P}$ are known and we are looking for the 3D point $\mathbf{X}$ that projects as close to the measured points $\mathbf{x}$ and $\bar{\mathbf{x}}$ as possible by minimizing the objective

$$\min_X \left\| \left(x^1 - \frac{P^1\mathbf{X}}{P^3\mathbf{X}},\ x^2 - \frac{P^2\mathbf{X}_j}{P^3\mathbf{X}} \right) \right\|^2 + \left\| \left(\bar{x}^1 - \frac{\bar{P}^1\mathbf{X}}{\bar{P}^3\mathbf{X}},\ \bar{x}^2 - \frac{\bar{P}^2\mathbf{X}}{\bar{P}^3\mathbf{X}} \right) \right\|^2. \tag{10.37}$$

To simplify notation we will let $\mathbf{y}$ and $\bar{\mathbf{y}}$ be the exact projections of $\mathbf{X}$, that is $\lambda\mathbf{y} = P\mathbf{X}$ and $\bar{\lambda}\bar{\mathbf{y}} = \bar{P}\mathbf{X}$. From Lecture 5 we know that there is a 3D point that projects to $\mathbf{y}$ and $\bar{\mathbf{y}}$ if and only if the epipolar constraint $\bar{\mathbf{y}}^T F\mathbf{y} = 0$ holds. Furthermore, if we let $d(\cdot,\cdot)$ denote the distance between two points then (10.37) can be written

$$\min_{\bar{\mathbf{y}}^T F\mathbf{y}=0} d(\mathbf{x},\mathbf{y})^2 + d(\bar{\mathbf{x}},\bar{\mathbf{y}})^2. \tag{10.38}$$

The epipolar constraint says that $\mathbf{y}$ has to lie on the epipolar line $\mathbf{l} = F^T\bar{\mathbf{y}}$ in image 1. Similarly $\bar{\mathbf{y}}$ has to lie on the epipolar line $\bar{\mathbf{l}} = F\mathbf{y}$ in image 2. It turns out that all points $\mathbf{p}$ lying on the epipolar line $\mathbf{l}$ in image 1 gives the same epipolar line $F\mathbf{p}$ in the second image. To see this we note that since the epipole in image 1 e is on $\mathbf{l}$ the points of $\mathbf{l}$ can be written

$$\mathbf{p} = e + s\mathbf{v}, \tag{10.39}$$

where $\mathbf{v}$ is a direction and $t \in \mathbb{R}$. Therefore we have $F\mathbf{p} = Fe + tF\mathbf{v} = tF\mathbf{v} \sim F\mathbf{v}$, which shows that all these points give the same epipolar line. Similarly one sees that all points on $\bar{\mathbf{l}}$ give rise to the same epipolar line in image 1. There is therefore a one-to-one correspondence of epipolar lines in the two images and as long as $\mathbf{y}$ and $\bar{\mathbf{y}}$ lies on corresponding epipolar lines there is a 3D point $\mathbf{X}$ that projects to them.

Figure 10.6 gives a geometric interpretation of corresponding epipolar lines. The camera centers $\mathbf{C}$,$\bar{\mathbf{C}}$ and the 3D point $\mathbf{X}$ are coplanar and forms an epipolar plane. The epipolar lines are the intersections between this plane and the image planes. A 3D plane has 3 degrees of freedom. Any epipolar plane contains the two camera centers (and the two epipoles) which fixes two of these degrees. The family of epipolar planes, and consequently the family of corresponding epipolar lines, can therefore be parametrized by a single parameter s.

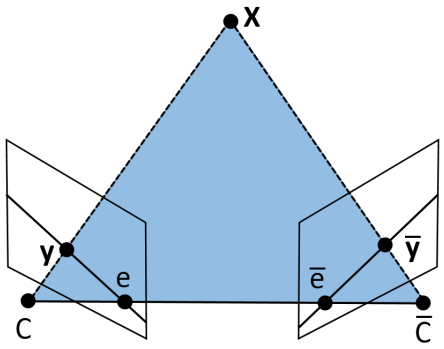


Figure 10.6: Visualization of corresponding epipolar lines. The camera centers $\mathbf{C}$ and $\bar{\mathbf{C}}$ and the 3D-point $\mathbf{X}$ forms a plane. The intersections with the image planes form a pair of corresponding epipolar lines.

Now suppose that we search for the best points over a given pair of corresponding epipolar lines $\mathbf{l}$ and $\bar{\mathbf{l}}$. It is then clear that the points that minimize the distance to $\mathbf{x}$ and $\bar{\mathbf{x}}$ respectively are given by the ortogonal projections of $\mathbf{x}$ and $\bar{\mathbf{x}}$. To solve (10.37) we can therefore equivalently search for the corresponding epipolar lines $\mathbf{l}$ and $\bar{\mathbf{l}}$ that minimizes

$$d(\mathbf{x}, \mathbf{l})^2 + d(\bar{\mathbf{x}}, \bar{\mathbf{l}})^2 \tag{10.40}$$

where $d(\mathbf{x}, \mathbf{l})$ denotes the (orthogonal) distance between the measured point $\mathbf{x}$ and the line $\mathbf{l}$.

To parametrize the family of corresponding lines we note that $\mathbf{l}$ goes through e_2 and specify its direction $\mathbf{v}$. If we let $\mathbf{v} = \begin{pmatrix} s \\ 1 \\ 0 \end{pmatrix}$ the line can take any 2D-direction (except $(1, 0)$ which corresponds to $s \to \infty$ which has to be checked separately). The expression for the corresponding lines are now given by

$$\mathbf{l} = e_1 \times v = \begin{bmatrix} e_1 \end{bmatrix}_\times \mathbf{v} \tag{10.41}$$

$$\bar{\mathbf{l}} = F(e_1 + t\mathbf{v}) \sim F\mathbf{v} \tag{10.42}$$

Note that the elements of these two vectors are affine functions (1st degree polynomials).

Exercise 25. Compute all epipolar line correspondences for the cameras

$$P = \begin{bmatrix} I & 0 \end{bmatrix} \text{ and } \bar{P} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{10.43}$$

The squared distance between $\mathbf{l}$ and $\mathbf{x}$ is given by

$$\frac{(l_1x_1 + l_2x_2 + l_3)^2}{(l_1^2 + l_2^2)}. \tag{10.44}$$

Since $\mathbf{x}$ is independent of s this is a quotient of quadratic functions of the type

$$\frac{as^2 + bs + c}{ds^2 + es + f}. \tag{10.45}$$

Since the same holds for the squared distance between $\bar{\mathbf{x}}$ and $\bar{\mathbf{l}}$ the objective function (10.40) is a sum of two such expressions. To find its minimizer we compute its stationary points by differentiating. The derivative of (10.45) is

$$\frac{(ae - bd)s^2 + 2(af - cd)s + bf - ce}{(ds^2 + es + f)^2} \tag{10.46}$$

which is a quotient of a degree two and a degree four polynomial. Setting the derivative of (10.40) to zero and multiplying with the two denominators thus gives a polynomial of at most degree 6 with roots contain the optimal value of s.

Exercise 26. Compute the corresponding epipolar lines that minimize $d(\mathbf{x}, \mathbf{l})^2 + d(\bar{\mathbf{x}}, \bar{\mathbf{l}})^2$ for the cameras in Exercise 25.

Figure 10.7 compares the optimal triangulation objective to the DLT objective from Lecture 4.

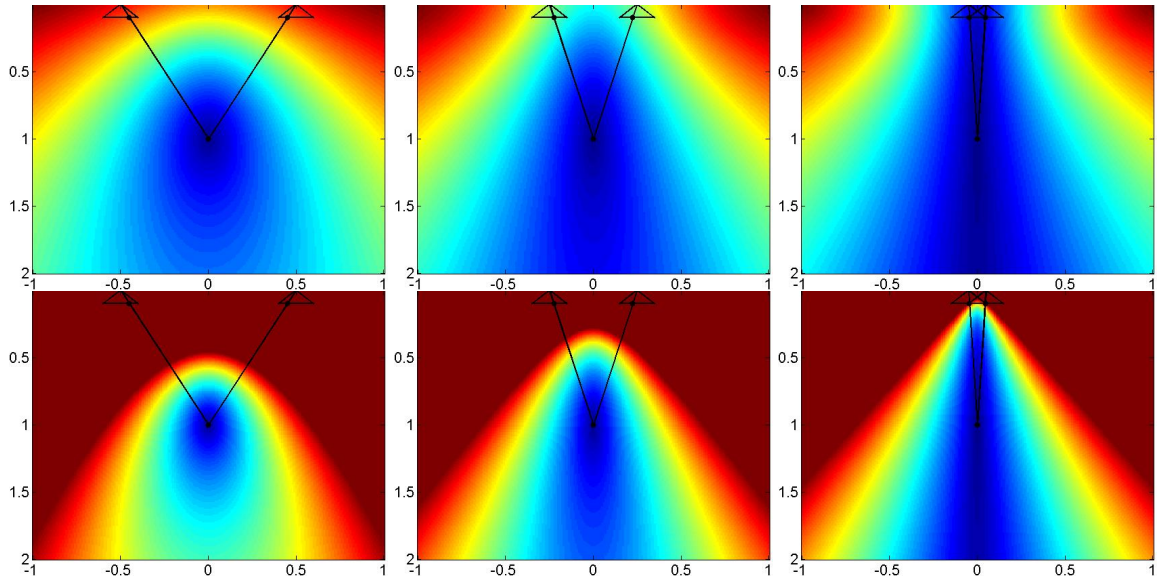


Figure 10.7: Comparison between the DLT- and the ML objectives for triangulation. First row DLT, second row ML objective (with the same colormap).

10.3.2 Affine Cameras

In general the maximum likelihood estimator (10.36) 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

$$P = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix}, \tag{10.47}$$

where A is a 2×3 matrix and t is a 2×1 vector. Geometrically this means that viewing rays are assumed to be perpendicular to the image plane, see Figure 10.8. If the observed scene points are roughly at the same distance from the camera this model often works well as an approximation of the pinhole camera model.

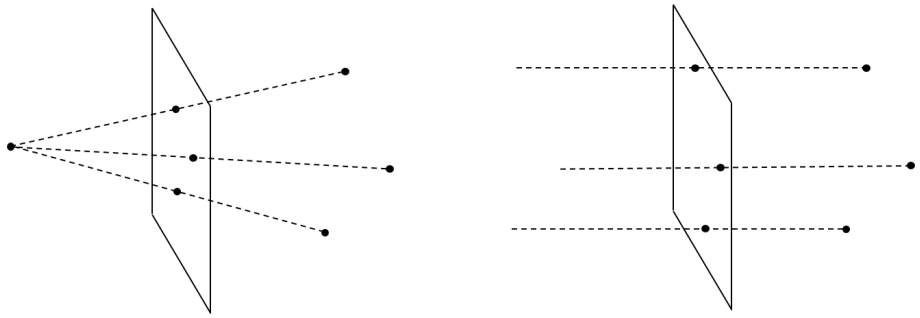


Figure 10.8: *Left* - A regular pinhole camera where all viewing rays go through the camera center. *Right* - The affine camera model where the viewing rays are parallel to the normal of the image ray.

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

$$x_{ij} = A_i X_j + t_i. \tag{10.48}$$

To find he maximum likelihood estimate we therefore need to solve

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \|x_{ij} - A_i X_j + t_i\|^2. \tag{10.49}$$

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

$$\min \sum_{ij} ||\tilde{x}_{ij} - A_i \tilde{X}_j||^2. \tag{10.50}$$

In matrix form we can write this as

$$\min \left\| \underbrace{\begin{bmatrix} \tilde{x}_{11} & \tilde{x}_{12} & \dots & \tilde{x}_{1m} \\ \tilde{x}_{21} & \tilde{x}_{22} & \dots & \tilde{x}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{x}_{n1} & \tilde{x}_{n2} & \dots & \tilde{x}_{nm} \end{bmatrix}}_{M} - \underbrace{\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} \begin{bmatrix} \tilde{X}_1 & \tilde{X}_2 & \dots & \tilde{X}_m \end{bmatrix}}_{\text{rank 3 matrix}} \right\|^2. \tag{10.51}$$

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:
$$M = USV^T. \tag{10.52}$$
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.

Lecture 11:

RANSAC and Minimal Solvers

11.1 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 11.1 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**.

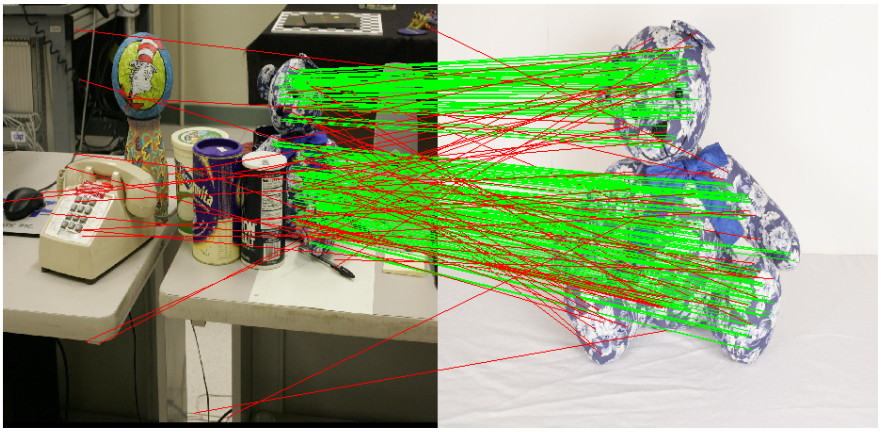


Figure 11.1: The Outlier Problem. When automatically detecting correspondences using descriptors such as SIFT there will always be a portion of incorrect matches. Green lines in the figure correspond to correct matches and red lines correspond to outliers.

As we saw in Lecture 7 the outliers typically do not fulfill the Gaussian noise assumption for the particular problem that we are trying to solve, and they can severely degrade the quality of the estimation. To address this issue we can use robust loss-functions. However, these can be sensitive to initialization and can often only handle a relatively small portion of outliers. Therefore we need a method for removing outliers as well as providing reliable starting solutions that can be locally refined.

11.2 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 27. 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 28. 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.

11.3 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.

$$\begin{cases} x^2 - y - 3 &= 0 \\ xy - x &= 0. \end{cases} \tag{11.1}$$

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

$$c_p = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \quad \text{and} \quad m(x, y) = \begin{pmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{pmatrix}. \tag{11.2}$$

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.

11.3.1 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

$$1 \mapsto x \tag{11.3}$$
$$x \mapsto x^2 \tag{11.4}$$
$$y \mapsto xy. \tag{11.5}$$

Now let us assume that (x_0, y_0) is a solution to (11.1). The result of applying T_x to the above monomials can then be simplified if we insert (x_0, y_0),

$$1 \mapsto x_0 \tag{11.6}$$
$$x_0 \mapsto x_0^2 = y_0 + 3 \tag{11.7}$$
$$y_0 \mapsto x_0 y_0 = x_0. \tag{11.8}$$

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 (11.6)-(11.8) we get

$$q(x_0, y_0) = c_p^1 x_0 + c_p^2(y_0 + 3) + c_p^3 x_0 = 3c_p^2 1 + (c_p^1 + c_p^3)x_0 + c_p^2 y_0. \tag{11.9}$$

We see that if (x_0, y_0) solves (11.1) 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 (11.9),

$$\begin{pmatrix} c_q^1 \\ c_q^2 \\ c_q^3 \end{pmatrix} = \begin{pmatrix} 3c_p^2 \\ c_p^1 + c_p^3 \\ c_q^3 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 3 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{=M_x} \begin{pmatrix} c_p^1 \\ c_p^2 \\ c_p^3 \end{pmatrix}. \tag{11.10}$$

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 $x_0 p(x_0, y_0)$ provided that (x_0, y_0) solves the system (11.1). Under certain conditions it is possible to compute the roots of the system from this matrix.

11.3.2 Finding the Roots.

Next we will show that the roots of the system (11.1) are eigenvalues to the action matrix M_x. For any polynomial p we have

$$x_0 p(x_0, y_0) = x_0 c_p^T m(x_0, y_0). \tag{11.11}$$

Furthermore,

$$x_0 p(x_0, y_0) = q(x_0, y_0) = c_q^T m(x_0, y_0) = (M_x c_p)^T m(x_0, y_0) = c_p^T M_x^T m(x_0, y_0). \tag{11.12}$$

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

$$x_0 m(x_0, y_0) = M_x^T m(x_0, y_0). \tag{11.13}$$

Therefore we can conclude that if (x_0, y_0) is a root of (11.1) then $m(x_0, y_0)$ is an eigenvector of M_x^T with eigenvalue x_0.

Exercise 29. Verify that $m(x_0, y_0)$ is an eigenvector of

$$M_x^T = \begin{pmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{11.14}$$

for all the three roots $(0, -3)$, $(2, 1)$ and $(-2, 1)$ of the system (11.1).

11.3.3 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 11.3.2 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.

What degree of polynomials do we need?

In the above example we only considered monomials of degree 1 in (11.6)-(11.8). 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 (11.1) we

can consider all second degree polynomials. For example we can use the reductions:

$$\begin{aligned} 1 &\mapsto x_0 && (11.15)\\ x_0 &\mapsto x_0^2 && (11.16)\\ y_0 &\mapsto x_0 y_0 && (11.17)\\ x_0^2 &\mapsto x_0^3 = x_0(y_0+3) = x_0 y_0 + 3x_0 && (11.18)\\ x_0 y_0 &\mapsto x_0^2 y_0 = x_0^2 && (11.19)\\ y_0^2 &\mapsto x_0 y_0^2 = x_0 y_0. && (11.20) \end{aligned}$$

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 11.3.2 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

$$M_x = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{11.21}$$

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.

Using other mappings than T_x.

In section 11.3.1 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 (11.6)-(11.8). For example suppose that we use T_y instead. We get

$$\begin{aligned} 1 &\mapsto y_0 && (11.22)\\ x_0 &\mapsto x_0 y_0 && (11.23)\\ y_0 &\mapsto y_0^2 && (11.24)\\ x_0^2 &\mapsto x_0^2 y_0 = x_0^2 && (11.25)\\ x_0 y_0 &\mapsto x_0 y_0^2 = x_0 y_0 && (11.26)\\ y_0^2 &\mapsto y_0^3 = y_0^2(x_0^2 - 3) = x_0^2 - 3y_0^2. && (11.27) \end{aligned}$$

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 (11.1). (It is however possible to generate new equations from (11.1) 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

$$M_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -3 \end{pmatrix}. \tag{11.28}$$

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 (11.21) and (11.28) respectively) has five distinct eigenvalues.

Another trick that modifies the eigenspace is to drop some of the monomials. In (11.28) 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

$$M_y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -3 \end{pmatrix}. \tag{11.29}$$

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.

11.4 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:

$$\bar{\mathbf{x}}_i^T E \mathbf{x}_i = 0, \qquad i = 1, ..., 5. \tag{11.30}$$

$$\det(E) = 0, \tag{11.31}$$

$$2EE^TE - \text{trace}(EE^T)E = 0. \tag{11.32}$$

The third constraint (11.32) actually consists of 9 polynomial equations since it is a matrix expression. Any matrix E that has a singular value decomposition of the form

$$E = U \underbrace{\begin{pmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{=S} V^T \tag{11.33}$$

will fulfill this constraint. This can be seen by inserting (11.33) into (11.32). Furthermore, it can be shown that any matrix that fulfills (11.32) must have a singular value decomposition as in (11.33).

To find the solutions of (11.30)-(11.32) we will first use (11.30) 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

$$M(a_1 v_1 + a_2 v_2 + a_3 v_3 + a_4 v_4) = 0, \tag{11.34}$$

for any choice of coefficients $a_1, ..., a_4$. Reshaping these vectors into matrices we get

$$\bar{\mathbf{x}}_i^T (a_1 E_1 + a_2 E_2 + a_3 E_3 + a_4 E_4) \mathbf{x}_i = 0, \quad i = 1, ..., 5. \tag{11.35}$$

What remains is to find the coefficients $a_1, ..., a_4$ such that (11.31) and (11.32) are fulfilled. Note that since the scale of the essential matrix is arbitrary, we can assume that (for example) $a_1 = 1$.

To determine the coefficients $a_2, ..., a_4$ we will use the method presented in the previous section. Equation (11.32) consists of 9 third order polynomials in a_2, a_3, a_4. In addition we have (11.31) 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 (11.32)

$$2EE^TE - \text{trace}(EE^T)E = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} a_i a_j a_k \left(2E_i E_j^T E_k - \text{trace}(E_i E_j^T)E_k\right). \tag{11.36}$$

For each of the 9 constraints in (11.32) we can extract coefficients for the monomial $a_i a_j a_k$ 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 $a_i a_j a_k = a_1^2 a_2 = a_2$. 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

$$\det(E) = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} a_i a_j a_k \big(e_{11}^i e_{22}^j e_{33}^k + e_{12}^i e_{23}^j e_{31}^k + e_{13}^i e_{21}^j e_{32}^k \\ - e_{11}^i e_{23}^j e_{32}^k - e_{12}^i e_{21}^j e_{33}^k - e_{13}^i e_{22}^j e_{31}^k\big), \tag{11.37}$$

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:

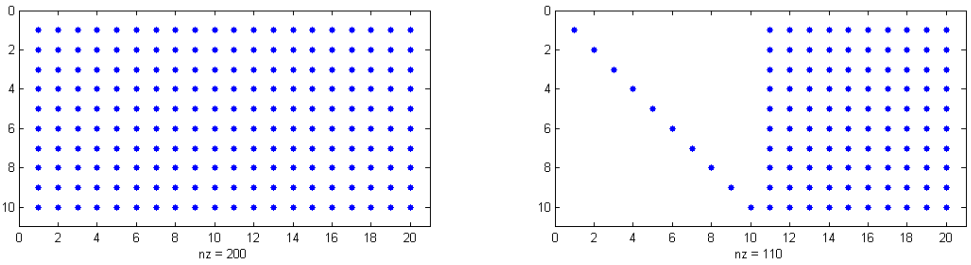


Figure 11.2: Shape of the 10×20 coefficient matrix before (left) and after elimination (right). A '*' means the element is non-zero.

$$\{a_4^3, a_3a_4^2, a_3^2a_4, a_3^3, a_2a_4^2, a_2a_3a_4, a_2a_3^2, a_2^2a_4, a_2^2a_3, a_2^3, a_4^2, a_3a_4, a_3^2, a_2a_4, a_2a_3, a_2^2, a_4, a_3, a_2, 1\}. \tag{11.38}$$

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 11.2. For example, after elimination, the first row of the matrix only contains a_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_{a_2}, T_{a_3} or T_{a_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 (11.36) and (11.37) compute coefficients for all monomials and construct the 10×20 coefficient matrix. (The monomial order does not have to be the same as (11.38), 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_{a_2}, T_{a_3} or T_{a_4} using the reductions available in the modified coefficient matrix.

Figure 11.3 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.

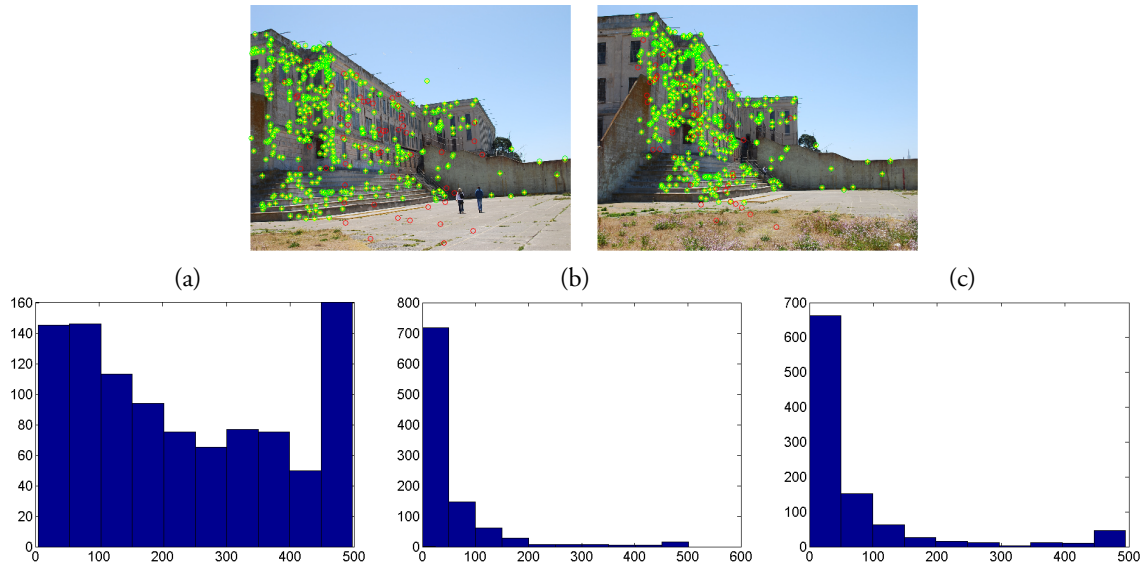


Figure 11.3: First row: The best solution obtained from the 5-point solver. Yellow '*' is an image point, green 'o' is the reprojection of an inlier and red 'o' is a reprojection of an outlier. Second row: Histogram over the size of the consensus set in each iteration of a 1000-iteration-RANSAC, using (a) - 5 points, (b) - 8 points and (c) - 10 points.

Lecture 12:

Local Optimization

12.1 The Maximal Likelihood Estimator

In the previous lecture we derived the maximum likelihood estimator for our class of projection problems. If $x_{ij} = (x_{ij}^1, x_{ij}^2)$ is the projection in regular Cartesian coordinates of the 3D-point $\mathbf{X}_j$ in camera P_i then the model parameters that make the measurements x_{ij} most likely is found by minimizing

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\left(x_{ij}^1 - \frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j},\ x_{ij}^2 - \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j}\right)\right\|^2. \tag{12.1}$$

This problem is however difficult to solve since there is in general no closed form solution. In practice we are limited to locally improving the objective function from some suitable selected starting point. The starting point is typically generated using algebraic solvers like the ones we have presented previously in the course. In this lecture we will present some simple methods for local optimization.

12.2 Background

We first recall some basic concepts from optimization.

Definition 1. A point v is called a stationary point of f if

$$\nabla f(v) = 0. \tag{12.2}$$

Definition 2. A point v is called a local minimizer of f if

$$f(v) \leq f(s) \tag{12.3}$$

for all s in a neighborhood around v. (More technically, for all s such that $\|v - s\| \leq \delta$ for some $\delta > 0$).

Theorem 3. *Any local minimizer of a differentiable function is also a stationary point.*

Definition 3. A matrix M is positive definite ($M \succ 0$) if $v^T M v > 0$ for all $v \neq 0$. It is called positive semi definite ($M \succeq 0$) if $v^T M v \geq 0$ for all v.

Theorem 4. *If v is a stationary point of f ($\nabla f(v) = 0$) and the hessian of f is positive definite at t ($\nabla^2 f(v) \succ 0$) then v is also a local minima of f.*

12.3 Linear Least Squares

Minimizing (12.1) is an example of a non-linear least squares problem. In this section we will show how linear least squares problems can be minimized. In in the following sections we will use this approach to tackle the more difficult non-linear versions.

In the linear least squares problem we want to fit a set of linear functions $A_i v$ to a set of measurements b_i, $i = 1, .., n$. Here A_i^T is a vector of the same size as v representing a linear function of v. The least squares problem is now

$$\min_v \sum_{i=1}^{n} \|A_i v - b_i\|^2. \tag{12.4}$$

If we concatenate all the A_i into one matrix A and all the b_i into one vector b we can write the problem in matrix form as

$$\min_v \|Av - b\|^2. \tag{12.5}$$

Expanding the norm gives

$$\|Av - b\|^2 = (Av - b)^T(Av - b) = \underbrace{v^T A^T Av - 2b^T Av + b^T b}_{=f(v)}. \tag{12.6}$$

To find the minimum of this function we compute its stationary points.

$$\nabla f(v) = 0 \Leftrightarrow 2A^T Av - 2A^T b = 0 \Leftrightarrow A^T Av = A^T b. \tag{12.7}$$

These equations are called the normal equations, and assuming that $A^T A$ is invertible they give the stationary point

$$v = (A^T A)^{-1} A^T b. \tag{12.8}$$

To see that this is a local minimizer we look at the Hessian of f

$$\nabla^2 f(v) = A^T A. \tag{12.9}$$

This matrix is positive semi definite since

$$v^T A^T Av = \|Av\|^2 \geq 0. \tag{12.10}$$

Furthermore, it has to be positive definite since if there is $v \neq 0$ such that $\|Av\| = 0$ then, by multiplying with A^T, we see that $A^T Av = 0$ which would imply that $A^T A$ is not invertible (which we have assumed above). Therefore, (12.8) is a local minimum.

It is in fact also a global minimum which can be seen by changing coordinates. Let us call the stationary point s and let $v = \delta + s$. We then get

$$\|A(\delta + s) + b\|^2 = (\delta + s)^T A^T A(\delta + s) - 2b^T A(\delta + s) + b^T b. \tag{12.11}$$

Since s is the stationary point it fulfills (12.7) and therefore the right hand side can be re-written as

$$(\delta + s)^T A^T A(\delta + s) - 2s^T A^T A(\delta + s) + b^T b = \underbrace{\delta^T A^T A\delta}_{\text{depends on } \delta} - \underbrace{s^T A^T As + b^T b}_{\text{const.}}. \tag{12.12}$$

The result of this simplification is a term which depends on δ and one that is constant. Since $A^T A$ is positive definite any choice of δ that is not zero will result in the first term being positive. Therefore the optimal choice is $\delta = 0$ and thereby $v = s$.

12.4 Non-Linear Least Squares

Next we turn to the problem of solving the non-linear least squares formulation. In the general formulation we have a set of residuals $r_i(v)$ which we want to minimize in a least squares sense

$$\min_v \sum_i r_i(v)^2. \tag{12.13}$$

If we stack all the residuals $r_i(v)$ in a vector $r(v)$ then we can write the problem

$$\min_v \|r(v)\|^2. \tag{12.14}$$

In this section we will present three simple procedures that from a starting point improves the objective value by locally searching for better values.

12.4.1 Steepest Descent

The perhaps simplest possible strategy is the **steepest descent** search. Given a starting point v_0 this method finds the direction in which the function decreases most rapidly, and takes a step in this direction. For the function $f(v)$ the directional derivatives at a point v_0 are given by

$$f'_d(v_0) = \nabla f(v_0)^T d. \tag{12.15}$$

Here d is a direction (vector of unit length). To find the d for which $f'_d(v_0)$ is maximally negative we should select $d = -\nabla f(v_0)/|\nabla f(v_0)|$ since this choice would give

$$f'_d = -\nabla f(v_0)^T \frac{\nabla f(v_0)}{|\nabla f(v_0)|} = -|\nabla f(v_0)|. \tag{12.16}$$

For each term $r_i(v)^2$ of the function $f(v) = \|r(v)\|^2 = r_1(v)^2 + r_2(v)^2 + ... + r_n(v)^2$ we have

$$\nabla \left(r_i(v)^2\right) = 2r_i(v)\nabla r_i(v), \tag{12.17}$$

and therefore

$$\nabla f(v) = 2\sum_i r_i(v)\nabla r_i(v). \tag{12.18}$$

Once the descent direction d has been computed the algorithm picks a new point v_1 by searching along this direction, that is $v_1 = v_0 + \lambda d$, where lambda is selected such that $f(v_1) < f(v_0)$. It is always possible to find such a λ (buy choosing λ small) unless $\nabla f(v_0) = 0$, that is v_0 is already a stationary point.

Then the whole process is repeated for the point v_1 and so on.

12.4.2 Gauss-Newton

While very cheap to compute the gradient does not contain any information about step-length. Therefore in the steepest descent approach we are forced to select a suitable λ by other means.

An alternative approach is the Gauss-Newton method. In this method we first approximate the residuals $r_i(v)$ with linear functions and then solve the resulting approximation using the approach from Section 12.3. The Taylor approximation of $r(v)$ at v_0 is

$$r(v) = \begin{pmatrix} r_1(v) \\ r_2(v) \\ \vdots \\ r_n(v) \end{pmatrix} \approx \underbrace{\begin{pmatrix} r_1(v_0) \\ r_2(v_0) \\ \vdots \\ r_n(v_0) \end{pmatrix}}_{r(v_0)} + \underbrace{\begin{pmatrix} \nabla r_1(v_0)^T \\ \nabla r_2(v_0)^T \\ \vdots \\ \nabla r_n(v_0)^T \end{pmatrix}}_{J(v_0)} \underbrace{(v - v_0)}_{d} \tag{12.19}$$

The matrix $J(v_0)$ is the Jacobian of the vector r. Its rows contain the gradients of each entry r_i in r. The approximating linear least squares problem is therefore

$$\min_d \|r(v_0) + J(v_0)d\|^2, \tag{12.20}$$

which according to (12.8) has the solution

$$d = -(J(v_0)^T J(v_0))^{-1} J(v_0)^T r(v_0). \tag{12.21}$$

The Gauss-Newton update is now $v_1 = v_0 + d$.

Remark 3. Note that with the notation in this section the steepest descent direction can (if we ignore the normalization) be written

$$d = -J(v_0)^T r(v_0). \tag{12.22}$$

12.4.3 Levenberg-Marquardt

While convergence of the Gauss-Newton method is very rapid close to the minimum it can be unstable at points not sufficiently close to the minimum. The reason is that the approach may generate large steps even though the approximation is only valid in a local neighborhood around v_0. Therefore we add a penalty for large steps and solve

$$\min_d \|r(v_0) + J(v_0)d\|^2 + \lambda\|d\|^2. \tag{12.23}$$

To find the minimum of this problem we can again compute the stationary point. Expanding the function gives

$$\|r(v_0) + J(v_0)d\|^2 + \lambda\|d\|^2 = d^T \left(J(v_0)^T J(v_0) + \lambda I\right) d + 2r(v_0)^T J(v_0)d + r(v_0)^T r(v_0). \tag{12.24}$$

Differentiating with respect to d gives

$$2\left(J(v_0)^T J(v_0) + \lambda I\right) d + 2J(v_0)^T r(v_0) = 0, \tag{12.25}$$

and therefore

$$d = -\left(J(v_0)^T J(v_0) + \lambda I\right)^{-1} J(v_0)^T r(v_0). \tag{12.26}$$

This approach is called the Levenberg-Marquardt update and is often much more stable than the original Gauss-Newton formulation. If λ is selected large enough the update $v_1 = v_0 + d$ is guaranteed to give a better objective value. On the other hand for a very large λ (12.26) is almost the same as (12.22) (multiplied with $1/\lambda$). A common strategy is to start with a large λ and gradually reducing it to increase convergence speed when approaching the minimum value.

12.4.4 Bundle Adjustment

We now return to the structure from motion problem. We will consider the calibrated version, that is, we are assuming that the image points x_{ij} have already been normalized and that we are searching for camera matrices of the form $P_i = [R_i \quad t_i]$ and scene points or the form $\mathbf{X}_j = \begin{bmatrix} X_j \\ 1 \end{bmatrix}$. In Computer Vision literature this problem is often called bundle adjustment since the bundle of viewing rays are locally adjusted to reduce the reprojection errors. The objective function (12.1) can then be written

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left\| \left(x_{ij}^1 - \frac{R_i^1 X_j + t_i^1}{R_i^3 X_j + t_i^3},\; x_{ij}^2 - \frac{R_i^2 X_j + t_i^2}{R_i^3 X_j + t_i^3} \right) \right\|^2, \tag{12.27}$$

where R_i^1, R_i^2, R_i^3 are the rows of the rotation R_i and t_i^1, t_i^2, t_i^3 are the entries of the translation t_i. Note that we have assumed in (12.27) that all points are visible in all cameras. However, this is no restriction. If for example, point 2 is not visible in camera 3 then we simply remove the term $(i, j) = (3, 2)$ from the sum.

To be able to apply the Gauss-Newton method we need to linearize the expressions of the type

$$\underbrace{\frac{R^1X+t^1}{R^3X+t^3}}_{=f_1} \text{ and } \underbrace{\frac{R^2X+t^2}{R^3X+t^3}}_{=f_2}. \tag{12.28}$$

(For ease of notation we drop the indexes i, j for now.) There are two difficulties with this that we need to handle: First, the functions are nonlinear since they contain both fractions and quadratic terms. Second, the rotation R depends non-linearely on a set of parameters.

One way to parametrize a rotation is through the exponential map

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!}A^k = I + A + \frac{1}{2}A^2 + \frac{1}{6}A^3 + ... \tag{12.29}$$

If R_0 is our current rotation estimate, then any other rotation R can be written as R_0 multiplied with a the exponential map of a skew symmetric matrix

$$R = \exp\begin{pmatrix} 0 & -a_1 & -a_2 \\ a_1 & 0 & -a_3 \\ a_2 & a_3 & 0 \end{pmatrix} R_0. \tag{12.30}$$

Since we are interested in linearizing (12.28) we need only consider the first order terms of (12.29). If we let

$$S_1 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad S_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \tag{12.31}$$

then close to R_0 we have

$$R \approx (I + (a_1S_1 + a_2S_2 + a_3S_3))\, R_0. \tag{12.32}$$

This gives us a linear local parametrization of the rotation in terms of the variables a_1, a_2, a_3. To determine the linearization of (12.28) we need to compute derivatives with respect to all the variables $a_1, a_2, a_3, t^1, t^2, t^3, X_1, X_2$ and X_3 in the current parameter estimates R_0, t_0, X_0. This is straight forward (although tedious) and we only give a few of the derivatives here:

$$\frac{\partial f_1}{\partial a_1} = \frac{S_1^1 R_0 X_0}{R_0^3 X_0 + t_0^3} - \frac{R_0^1 X_0 + t_0^1}{(R_0^3 X_0 + t_0^3)^2} S_1^3 R_0 X_0 \tag{12.33}$$

$$\frac{\partial f_1}{\partial X_0^1} = \frac{R_0^{11}}{R_0^3 X_0 + t_0^3} - \frac{R_0^1 X_0 + t_0^1}{(R_0^3 X_0 + t_0^3)^2} R_0^{31} \tag{12.34}$$

$$\frac{\partial f_1}{\partial t^1} = \frac{1}{R_0^3 X_0 + t_0^3} \tag{12.35}$$

$$\frac{\partial f_1}{\partial t^2} = 0 \tag{12.36}$$

$$\frac{\partial f_1}{\partial t^3} = -\frac{R_0^1 X_0 + t_0^1}{(R_0^3 X_0 + t_0^3)^2}, \tag{12.37}$$

where S_1^i is the i'th row of S_1, $R_0^{i,j}$ is element (i, j) in R_0. Using these (and the rest of the) derivatives we can now form the Jacobian $J(v_0)$ in (12.19). The entries of $r(v_0)$ is obtained by computing the residual values at the current parameter estimate. Note that v_0 contain the parameter values of $a_1, a_2, a_3, t^1, t^2, t^3, X_1, X_2$ and X_3 for all the cameras and all the points of the problem. Thus if there are n cameras, m scene points and all the points are visible in all the cameras then v_0 is of size $(6n + 3m) \times 1$, $r(v_0)$ is $mn \times 1$ and $J(v_0)$ is $(6n + 3m) \times mn$.

Since the reconstruction is only uniquely determined up to an unknown similarity transformation we can reduce the number of variables slightly. For example we can assume that the first camera is $[I \quad 0]$. This fixes six of the seven degrees of freedom of the similarity transformation. To also fix the scale we can for example select the first coordinate of the first point.

Lecture 13:
Global Optimization

13.1 Projective Least Squares

In Lecture 9 we studied local optimization methods for multiple view geometry problems. Under the assumption of Gaussian image noise we optimized the maximum likelihood function

$$\sum_{i,j}(r_{ij}(v))^2 = \sum_{ij}\left\|\left(x_{ij}^1 - \frac{R_i^1 X_j + t_i^1}{R_i^3 X_j + t_i^3}, x_{ij}^2 - \frac{R_i^2 X_j + t_i^2}{R_i^3 X_j + t_i^3}\right)\right\|^2. \tag{13.1}$$

In general this problem is difficult to solve since the involved terms are quotients of quadratic functions. In this lecture we will study some special cases that we can optimize globally by making a slight modification to the problem. Specifically, we will assume that the residuals are of the form

$$r_i(v) = \left\|\left(\frac{a_i^T v + \tilde{a}_i}{c_i^T v + \tilde{c}_i}, \frac{b_i^T v + \tilde{b}_i}{c_i^T v + \tilde{c}_i}\right)\right\|, \tag{13.2}$$

where $c_i^T v + \tilde{c}_i > 0$, that is, each coordinate is a quotient of affine functions. Instead of solving the least squares formulation we will consider the optimization problem

$$\begin{aligned} \min_v \max_i \quad & r_i(v) && (13.3)\\ \text{s.t.} \quad & c_i^T v + \tilde{c}_i > 0, && (13.4) \end{aligned}$$

which can be solved globally optimally because of some convexity properties.

Below we give two examples of problems where the residuals are of the type (13.2).

Triangulation In this problem we know the camera matrices $P_i = [A_i \quad t_i]$ and image points x_i and want to find the scene point X. The residuals are of the form

$$r_i(X) = \left\|\left(x_i^1 - \frac{A_i^1 X + t_i^1}{A_i^3 X + t_i^3}, x_i^2 - \frac{A_i^2 X + t_i^2}{A_i^3 X + t_i^3}\right)\right\|. \tag{13.5}$$

To see that this expression is of the correct type (13.2) we can re write it as

$$\left\|\left(\frac{(x_i^1 A_i^3 - A_i^1)X + x_i^1 t_i^3 - t_i^1}{A_i^3 X + t_i^3}, \frac{(x_i^2 A_i^3 - A_i^2)X + x_i^2 t_i^3 - t_i^2}{A_i^3 X + t_i^3}\right)\right\|. \tag{13.6}$$

The constraint $A_i^3 X + t_i^3 > 0$ means that the scene point should be in front of the camera.

Resection In the resection problem we want to estimate the camera parameters A and t from scene points X_i and their projections (x_i^1, x_i^2). The residuals of this problem are

$$r_i(A,t) = \left\| \left(x_i^1 - \frac{A^1 X_i + t^1}{A^3 X_i + t^3}, x_i^2 - \frac{A^2 X_i + t^2}{A^3 X_i + t^3} \right) \right\|. \tag{13.7}$$

Structure from Motion with Known Camera Orientations If the camera orientations of all the cameras are known (in practice computed with some other method such as the one described in Section 13.4) then we can solve for both the 3D points and the camera positions simultaneously. In this case the unknowns are X and t which gives residuals of the form

$$r_i(X,t) = \left\| \left(x_i^1 - \frac{A^1 X_i + t^1}{A^3 X_i + t^3}, x_i^2 - \frac{A^2 X_i + t^2}{A^3 X_i + t^3} \right) \right\|. \tag{13.8}$$

13.2 Convex Optimization

In this section we review some properties of convex functions and sets that will be useful for solving (13.3)-(13.4).

13.2.1 Convex Sets

A set $C \in \mathbb{R}^n$ is called convex if the line segment joining any two points in C is contained in C. That is, if $x, y \in C$ then $\lambda x + (1-\lambda)y \in C$ for all λ with $0 \leq \lambda \leq 1$.

We call a point x of the form

$$x = \sum_{i=1}^{n} \lambda_i x_i, \tag{13.9}$$

where $\sum_{i=1}^n \lambda_i = 1$, $0 \leq \lambda_i \leq 1$ and $x_i \in C$, a convex combination of the points $x_1, ..., x_n$. A convex set always contains every convex combination of its points. Furthermore, it can be shown that a set is convex only if it contains all its convex combinations. Figure 13.1 shows simple examples of the notions introduced.

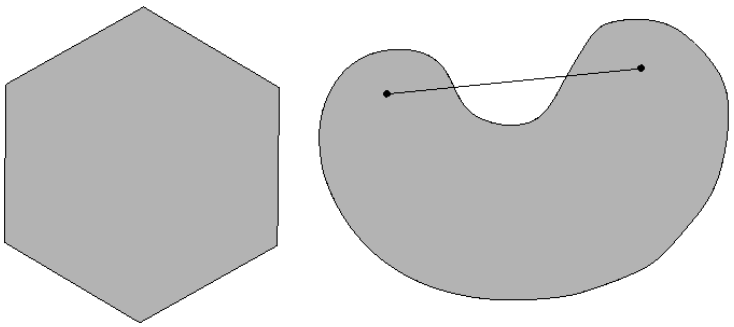


Figure 13.1: Left: A convex set. Right: A non-convex set.

Next we will state two special cases of convex sets that will be useful to us.

The halfspace. A halfspace is a set of the form

$$\{x \in \mathbb{R}^n; a^T x \leq b\}, \tag{13.10}$$

where $a \neq 0$, i.e., it is the solution set of a nontrivial affine inequality. The boundary of the half space is the hyperplane $\{x \in \mathbb{R}^n; a^T x = b\}$. It is straight forward to verify that these sets are convex.

The second order cone. The second order cone in $\mathbb{R}^{n+1}$ is the set

$$\{(x,t) \in \mathbb{R}^{n+1}; ||x|| \leq t\}. \tag{13.11}$$

From the general properties of norms it follows that the second order cone is a convex set in $\mathbb{R}^{n+1}$.

If $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ is an affine mapping then the set $C' = \{x; f(x) \in C\}$ is convex in $\mathbb{R}^m$ if C is convex in $\mathbb{R}^n$. That is, convexity is preserved under affine mappings. When applied to the second order cone we get sets of the type

$$\{x; ||Ax + b||_2 \leq c^T x + d\}. \tag{13.12}$$

Convexity is also preserved under intersection. Thus a set C that is given by several of the constraints above (half spaces and cone-constraints) is a convex set.

13.2.2 Convex Functions

A function $f : C \mapsto \mathbb{R}$ is called convex if C is a convex set, and for all $x, y \in C$ and $0 \leq \lambda \leq 1$, we have

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y). \tag{13.13}$$

The geometric interpretation of this definition is that the line segment between the points $(x, f(x))$ and $(y, f(y))$ should lie above the graph of f. Figure 13.2 shows the geometric interpretation of the definition.

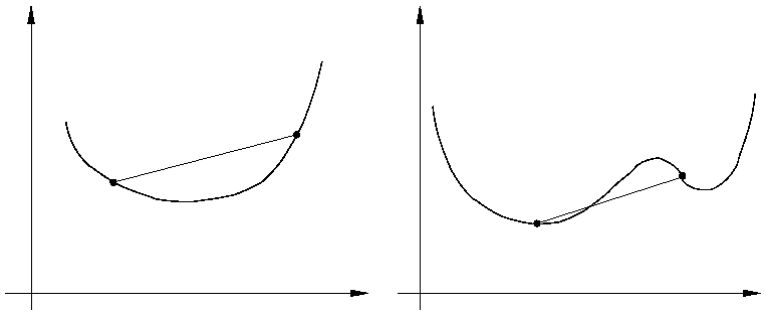


Figure 13.2: Left: Graph of a convex function. The line segment joining two points $(x, f(x))$ and $(y, f(y))$ lies above the graph. Right: Graph of a non-convex function.

Theorem 5. *If f is convex on a convex set C then any local minimum is also global.*

Proof. Asume that x is a local minimizer. Then there is a local neighborhood around x such that

$$f(x) \leq f(y) \tag{13.14}$$

for all y such that $\|x - y\| \leq \delta$. Suppose that x is not the global minimizer. Then there is an x^* such that

$$f(x^*) < f(x). \tag{13.15}$$

Since C is convex we can form the line segment between x^* and x and look at the values of f.

$$f(\lambda x^* + (1-\lambda)x) \leq \lambda f(x) + (1-\lambda)f(x^*) < \lambda f(x) + (1-\lambda)f(x) = f(x), \tag{13.16}$$

if $0 < \lambda \leq 1$. Now if we choose λ small enough such that $y = \lambda x^* + (1-\lambda)x$ fulfills $\|x - y\| \leq \delta$ we see that x cannot be a local minimizer since

$$f(x) > f(y). \tag{13.17}$$

□

13.3 Solving the Min-Max Problem

We now return to the min-max problem (13.3)-(13.4). By adding an extra variable γ we can rewrite the problem as

$$\begin{aligned} \min_{\gamma,v} \quad & \gamma & (13.18)\\ \text{s.t.} \quad & r_i(v) \leq \gamma, \quad \forall i. & (13.19) \end{aligned}$$

Since we minimize γ and $\gamma \geq r_i(v)$ for all i, γ has to take the same value as the largest residual $\max_i r_i(v)$. Therefore the two formulations are equivalent. Since $c_i^T v + \tilde{c}_i > 0$ and we can write the problem as

$$\begin{aligned} \min_{\gamma,v} \quad & \gamma & (13.20)\\ \text{s.t.} \quad & \left\| \left(a_i^T v + \tilde{a}_i, b_i^T v + \tilde{b}_i \right) \right\| \leq \gamma (c_i^T v + \tilde{c}_i), \quad \forall i. & (13.21) \end{aligned}$$

For a fixed γ the constraint (13.21) of the type (13.12) and therefore convex. Furthermore the intersection of all these constraints is also convex. This makes it possible to determine if there is a set of variables v that fulfill all the constraints for a given γ. Specifically, we can solve the convex program

$$\begin{aligned} \min_{s,v} \quad & s & (13.22)\\ \text{s.t.} \quad & \left\| \left(a_i^T v + \tilde{a}_i, b_i^T v + \tilde{b}_i \right) \right\| \leq \gamma (c_i^T v + \tilde{c}_i) + s, \quad \forall i. & (13.23) \end{aligned}$$

If the optimal $s > 0$ then it is not possible to find a v that fulfills all the constraints at the same time. In this case we know that the current γ is smaller than $\min_v \max_i r_i(v)$. Otherwise if the optimal $s \leq 0$ we know that the current γ is larger than $\min_v \max_i r_i(v)$. This makes it possible to search for the $\min_v \max_i r_i(v)$ by solving a sequence of convex problems.

Below we outline a simple procedure, called bisection, that finds the optimal solution.

1. Let γ_l and γ_u be lower and upper bounds on the optimal error.
2. Check if there is a solution such that
$$r_i(v) \leq \frac{\gamma_u + \gamma_l}{2}, \quad \forall i$$
(convex optimization problem).
3. If there is set $\gamma_u = \frac{\gamma_u + \gamma_l}{2}$, otherwise set $\gamma_l = \frac{\gamma_u + \gamma_l}{2}$.
4. If $\gamma_u - \gamma_l > tol$ (some predefined tolerance) goto 2.

The result is an interval $[\gamma_l, \gamma_u]$ that is guaranteed to contain the optimal value.

13.4 Rotation Averaging

The methods that we described in the previous section can be used to solve the structure from motion problem if we have estimates for the camera rotations by optimizing over camera positions and 3D points. For this to be useful we need a method that can accurately estimate camera rotations which we will describe in this section.

For each pair of cameras that have a sufficient number of matches we can solve the (calibrated) relative pose problem (see Lecture 6). This gives us two cameras $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$ and $P_2 = \begin{bmatrix} R_{12} & t_{12} \end{bmatrix}$. The rotation R_{12} essentially tells us how we should rotate camera 1 to get camera 2. Similarly, if we solve the relative pose between cameras 2 and 3 we get the solution $P_2 = \begin{bmatrix} I & 0 \end{bmatrix}$ and $P_3 = \begin{bmatrix} R_{23} & t_{23} \end{bmatrix}$. Note that the solution to the relative pose problem is only defined up to a similarity transformation. That is, if we have one solution we may rotate translate and scale

the solution to obtain a new one. Now let's say that the camera orientations are R_1,R_2 and R_3 in some common coordinate system. Then, by applying the rigid transformation $\begin{bmatrix} R_1^T & 0 \\ 0 & 1 \end{bmatrix}$ to the first solution we get

$$\begin{bmatrix} I & 0 \end{bmatrix} = \begin{bmatrix} R_1 & 0 \end{bmatrix} \begin{bmatrix} R_1^T & 0 \\ 0 & 1 \end{bmatrix} \tag{13.24}$$

and

$$\begin{bmatrix} R_{12} & t_{12} \end{bmatrix} = \begin{bmatrix} R_{12}R_1 & t_{12} \end{bmatrix} \begin{bmatrix} R_1^T & 0 \\ 0 & 1 \end{bmatrix} \tag{13.25}$$

That is $R_2 = R_{12}R_1$. Similarly by applying $\begin{bmatrix} R_2^T & 0 \\ 0 & 1 \end{bmatrix}$ to the second camera pair we get $R_3 = R_{23}R_2 = R_{23}R_{12}R_1$, which gives us estimates for the three camera orientations in the same coordinate system.

Now suppose that we also solve for the relative orientation between cameras 3 and 1 giving us the additional equation $R_1 = R_{31}R_3$. The system is now over-determined and due to noise it will in general not be possible to find an exact solution to all these equations at once. Least squares rotation averaging attempts to find the rotations that minimizes the problem

$$\min_{R_1,R_2,\dots,R_n} \quad \|R_{ij}R_i - R_j\|^2 \tag{13.26}$$

$$\text{such that} \quad R_i^T R_i = I \quad \forall i = 1,\dots,n. \tag{13.27}$$

This is a quadratic objective function with quadratic equallity constraints which in general results in a non-convex problem. In the following section we will describe how such problems can be addressed using Lagrangian duality.

13.5 Duality

Suppose that we have an a quadratic objective function $q(x) = x^T\tilde{A}x + 2b^Tx + c$, where the variable $x \in \mathbb{R}^n$. By extending x to $\mathbb{R}^{n+1}$ and letting the extra coordinate be one we can write the objective as $q(x) = x^TAx$ where

$$A = \begin{bmatrix} \tilde{A} & b \\ b^T & c \end{bmatrix}. \tag{13.28}$$

Similarly suppose that we have quadratic functions $q_i(x) = x^TA_ix - 1$ and we want to solve

$$\min_x \quad q(x) \tag{13.29}$$

$$\text{such that} \quad q_i(x) - 1 = 0. \tag{13.30}$$

We will call this problem the primal problem. To eliminate the constraint we form the Lagrangian $L(x,\lambda) = q(x) + \sum_i \lambda_i(q_i(x) - 1)$ and consider

$$\min_x L(x,\lambda) \tag{13.31}$$

for a given value of λ. Let x^* be the minimizer of of (13.31). If x^* fulfills the constraints $q_i(x^*) = 0$ then we have found the minimizer of the original problem since $L(x,\lambda) = q(x)$ for all x that fulfill $q_i(x) = 0$. If the minimizer does not fulfill the constraints then we found a solution to that was better than all other points fulfilling the constraints. In both cases we have that $L(x^*,\lambda) < \min_x q(x)$ such that $q_i(x) = 0$. That is for each value of λ we get a lower bound on the optimal value of the primal problem. Lagrangian duality consists in trying to find the largest lower bound by solving

$$\max_\lambda \min_x L(x,\lambda). \tag{13.32}$$

We will refer to this as the dual problem. When both the objective and the constraints are quadratic it is possible to solve the inner minimization in closed from since $L(x,\lambda) = x^T(A + \sum_{i=1}^n \lambda_i A_i)x - \sum_{i=1}^n \lambda_i$. The minimum of this function is

$$L(x^*,\lambda) = \begin{cases} -\sum_{i=1}^n & \text{if } A + \sum_{i=1}^n \lambda_i A_i \succeq 0 \\ -\infty & \text{otherwise} \end{cases} \tag{13.33}$$

Since the dual problem maximizes with respect to λ we do not need to consider the second case. The dual problem is then

$$\max_{\lambda} \quad -\sum_{i=1}^{n} \lambda_i \tag{13.34}$$

$$\text{such that} \quad A + \sum_{i=1}^{n} \lambda_i A_i \succeq 0. \tag{13.35}$$

This problem has a linear objective function and a convex constraint and is therefore convex and can be reliably solved with standard solvers.

For rotation averaging the dual problem has the from

$$\max_{\Lambda} \quad -\text{trace}(\Lambda) \tag{13.36}$$

$$\text{such that} \quad \Lambda - M \succeq 0, \tag{13.37}$$

where

$$M = \begin{bmatrix} 0 & R_{12} & \dots & R_{1n} \\ R_{21} & 0 & \dots & R_{2n} \\ \vdots & & \ddots & \vdots \\ R_{n1} & R_{n2} & \dots & 0 \end{bmatrix} \quad \text{and} \quad \Lambda = \begin{bmatrix} \Lambda_1 & 0 & 0 & \dots \\ 0 & \Lambda_2 & 0 & \dots \\ 0 & 0 & \Lambda_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{13.38}$$

While the dual problem is in general only a lower bound on the primal problem it can be shown that for the rotation averaging problem it is tight if the solution has small enough error residuals. Specifically, if the rotation angle of the rotation $R_j^T R_{ij} R_i$ is smaller than 42.9° then the lower bound is tight.

Lecture 14:

Factorization and Dimensionality Reduction

14.1 Dynamic Scenes and Linear Basis Models

Previously we have considered image data generated by viewing a rigid scene in a moving camera. In this section we will consider images depicting more general deforming objects. Solving for both camera and object motions is an ill posed problem if points are allowed to move arbitrarily. However, in real scenes point motions are typically highly correlated. Figure 14.1 shows for images from a dataset consisting of points tracked through a sequence off hand. While all the points are allowed to move only a small subset of simple motions give rise to reasonable hand shapes. Restricting the set of shapes and motions regularizes the problem and makes it solvable.

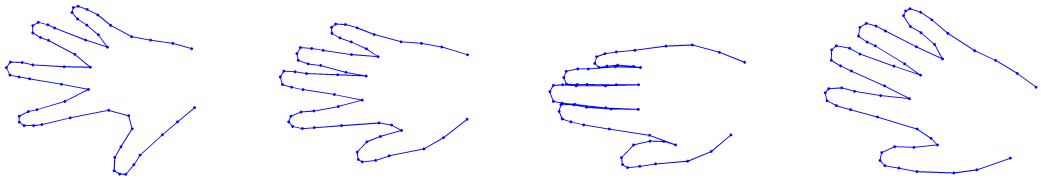


Figure 14.1: Four (out of 40) images of a deformable model with 56 tracked point. By concatenating x- and y-coordinates from all images each track can be seen as a data point in an 80 dimensional space.

From a mathematical point of view we can see the coordinates from each track as a data point in a high dimensional space. Let (x_{ij}, y_{ij}) be the coordinates of point j in image i. We form the measurement matrix M by

$$M = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ y_{11} & y_{12} & y_{13} & \dots & y_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \\ y_{m1} & y_{m2} & y_{m3} & \dots & y_{mn} \end{pmatrix}. \tag{14.1}$$

Here each column corresponds to the coordinates of a point track and every two consecutive rows correspond to an image. The columns of M can be seen as data points in a $2m$-dimensional space. To model dependence between points a common approach is to assume that the data points can be written as a linear combination of a few basis columns, or equivalently, to assume that the columns of M belong to a low dimensional subspace.

In linear algebra we have the following useful concepts:

- The **column space** of M consists of all linear combinations of columns in M.
- The **row space** of M consists of all linear combinations of rows in M.
- The **rank** of M is the dimension of the row and column spaces.

Exercise 30. Show that the columns $B_1 = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$ and $B_2 = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$ form a basis for the column space of

$$M = \begin{pmatrix} 1 & 2 & 2 & 0 \\ 2 & 3 & 2 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \tag{14.2}$$

and determine its rank.

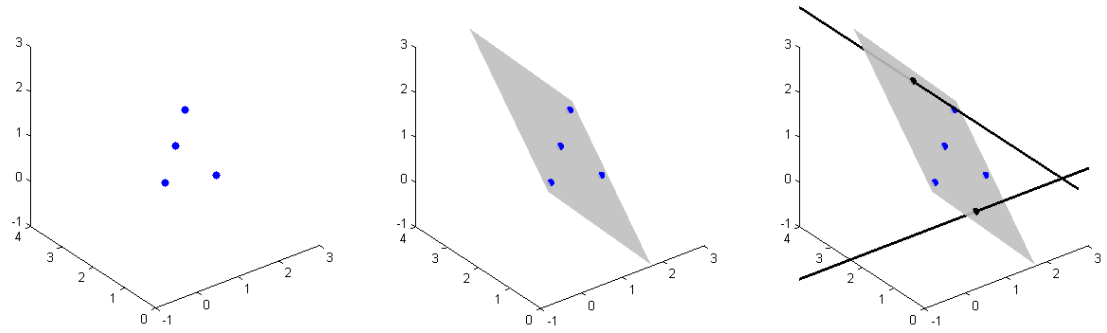


Figure 14.2: *Left* - The columns of M seen as points in $\mathbb{R}^3$. *Middle* - The factorization 2D basis B_1 and B_2 spanning a plane that all points belong to. *Right* - When a coordinate of a new data point is missing it can be recovered by enforcing that the points lie on the detected plane.

Since the columns of M have 3 entries they can be interpreted as points in $\mathbb{R}^3$, see Figure 14.2. The fact that we can write all the columns of M as a linear product of the basis vectors B_1 and B_2 shows that they all belong to a 2-dimensional subspace spanned by B_1 and B_2.

Exercise 31. Find a 3×2 matrix B and a 2×4 matrix C such that $M = BC^T$ and determine a basis for the row space of M.

As we have seen previously we can interpret the columns of M as data points in $\mathbb{R}^3$ and B as a basis for the 2-dimensional column space. Alternatively we can consider the rows of M as data points in $\mathbb{R}^4$ and C as a basis for the row space. In this case the matrix B contains the coefficients used to form the data points in M from the basis in C. The dimension of these two spaces is the same as the rank of the matrix. For the data shown in Figure 14.1 the row space will consist of a set of possible hand-shapes whereas the column space consists of a set of point trajectories.

The expression $M = BC^T$ is called a **factorization** of M and B and C are the **factors**. Without any additional constraints there are many possible factorizations. For example if we let $\tilde{B} = BH$ and $\tilde{C} = CH^{-1^T} = CH^{-T}$ we get

$$\tilde{B}\tilde{C}^T = BHH^{-1}C^T = BC^T = M, \tag{14.3}$$

for any invertible matrix H.

14.2 Low Rank Approximation

When using real data our measurements are usually corrupted by noise. Therefore the measurement matrix is normally of full rank and it is not possible to find any subspace containing the data points unless the whole space is used. In order to reduce the dimensionality of the data we therefore have to remove noise. Assuming the we add Gaussian noise to a matrix with rank r to get the measurement matrix M the maximum likelihood estimator is

$$\min_{\text{rank}(X)=r} \|X - M\|_F^2, \tag{14.4}$$

that is, we want to find the matrix X that is closest to M in a least squares sense and has $\text{rank}(X) = r$.

The solution to this problem can be obtained from the SVD of M.

Theorem 6 (Ekhart-Young 1936)**.** *If rank$(M) = k > r$ and M has the singular value decomposition $M = USV^T$ with*

$$S = \begin{bmatrix} \textit{diag}(\sigma_1, \sigma_2, ..., \sigma_k) & 0 \\ 0 & 0 \end{bmatrix}, \tag{14.5}$$

then the solution to (14.4) *is given by $X = US_rV^T$ where*

$$S_r = \begin{bmatrix} \textit{diag}(\sigma_1, \sigma_2, ..., \sigma_r, 0, 0, ...) & 0 \\ 0 & 0 \end{bmatrix}. \tag{14.6}$$

To find the best rank r approximation of M we thus take its SVD and set all but the first r singular values to zero. We can directly obtain a factorization $X = BC^T$ from the SVD $X = US_rV^T$. Let U' and V' be the first r columns of U and V and S'_r be the top $r \times r$ block of S_r. Since all but the first r columns of U vanish in the multiplication US_r, and similarly all but the first r rows of V^T vanish in S_rV^T, it is clear that

$$X = US_rV^T = U'S'_rV'^T. \tag{14.7}$$

Thus we can for example let $B = U'S'_r$ and $C = V'$ to obtain $X = BC^T$.

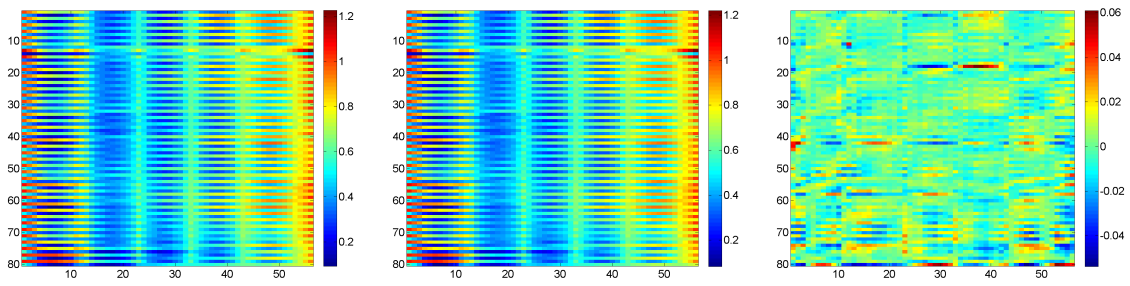


Figure 14.3: *Left* - The measurement matrix for the hand data set in Figure 14.1. *Middle* - A rank 5 approximation of the measurement matrix. *Right* - The difference between the measurement matrix and the rank 5 approximation.

Figure 14.1 shows the measurement matrix for the hand data set from Figure 14.1. The left image shows the original noisy measurement matrix and the middle image shows the low rank approximation obtained using SVD as described above. The difference is shown in the right image, note the different scales. While the two matrices are very similar the first one has $80 \cdot 56 = 4480$ elements while the second one can be represented with $(80 + 56) \cdot 5 - 5^2 = 665$ numbers.

If the assumption that all hand shapes can be written as a linear combination of five basis shapes is accurate, we can use the obtained factorization to compute new hand shapes. To generate a new image we need to know the x- and y- coordinates of all the points. According to our assumption these should be linear combinations of the rows of C^T. We therefore should have

$$\begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{pmatrix} = B_{\text{new}}C^T, \tag{14.8}$$

where $(x_1, y_1), (x_2, y_2), ...$ are the unknown point coordinates and B_{new} is a 2×5 matrix of parameters that specify the point coordinates. Thus we have a linear function $f(B_{\text{new}}) := B_{\text{new}}C^T$ that can be used to generate new images. It can be difficult to find reasonable values for the parameters in B_{new} since there are many possible choices of C^T that can be obtained from the factorization. One way to generate reasonable values is to instead specify the coordinates of a few (in this case 5) points and determine the values of B_{new} from these. If we for example specify the first 5 point positions we get

$$\begin{pmatrix} x_1 & x_2 & \dots & x_5 \\ y_1 & y_2 & \dots & y_5 \end{pmatrix} = B_{\text{new}} C_{5\times 5}^T, \tag{14.9}$$

where $C_{5\times 5}$ are the first five rows of C. Assuming that $C_{5\times 5}$ is invertible we get

$$B_{\text{new}} = \begin{pmatrix} x_1 & x_2 & \dots & x_5 \\ y_1 & y_2 & \dots & y_5 \end{pmatrix} C_{5\times 5}^{-T}. \tag{14.10}$$

Inserting (14.10) in (14.8) gives a new linear function that takes 5 point positions and returns the coordinates for all the points 56.

The choice of using the first five points to construct the function above is arbitrary and we may use any collection of 5 points. For numerical reasons it is often beneficial to use points that are evenly distributed in the scene. In Figure 14.4 we selected the five marked with a red dot and varied the coordinates of one of them. The blue curves show the positions obtained for all points by using (14.10) and (14.8). As long as we stay in the vicinity of the previously observed shapes the silhouettes look reasonable, however if we move too far away from previously observed shapes the result may start to look unreasonable, as in the right example of Figure 14.4).

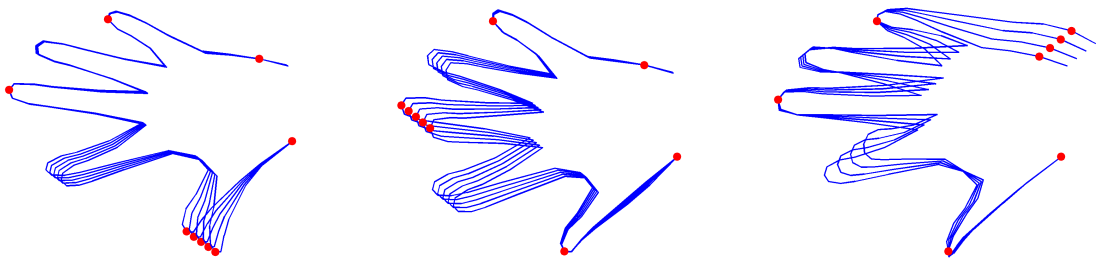


Figure 14.4: Modifying the point coordinates of 5 of the observed points to generate new shapes using the computed row space of C^T.

14.3 The Missing Data Problem

Now lets consider the degrees of freedom (DOF) of the set of matrices that can be written BC^T if B is of size $m \times r$ and C is $n \times r$. The two factors have $(m + n)r$ elements in total, however since the product (14.3) is independent of the $r \times r$ matrix H we get $(m + n)r - r^2$. In contrast a general $m \times n$ matrix has mn elements. Since $mn = (m + n)n - n^2 = (m + n)m - m^2$ it is clear that BC^T has fewer degrees of freedom if $r < m$ or $r < n$. In the example in Exercise 30 we have $mn = 12$ and $(m + n)r - r^2 = 10$. Thus BC^T can be seen as a compressed representation where we can specify the content of M using fewer values than its total number of elements.

The fact that BC^T has fewer DOF than a general matrix of size $m \times n$ makes it possible to recover M even if only a subset of the elements of M is known.

Exercise 32. Find the elements m_{15} and m_{26} of

$$M = \begin{pmatrix} 1 & 2 & 2 & 0 & m_{15} & 1 \\ 2 & 3 & 2 & 1 & 1 & m_{26} \\ 1 & 1 & 0 & 1 & 0 & 2 \end{pmatrix}, \tag{14.11}$$

such that $\text{rank}(M) = 2$.

To the right Figure 14.2 we give a geometric interpretation of the above exercise. Then varying m_{15} and m_{26} we obtain the points of two lines. If the unknown points are assumed to belong to the same subspace as the rest of the columns of M they have to lie in the intersections between the lines and the plane.

When some elements of M are unknown we cannot use the SVD to recover a low rank approximation of M. If W is a matrix that has $w_{ij} = 1$ if m_{ij} is known and $w_{ij} = 0$ otherwise we seek a solution to

$$\min_{B,C} \|W \odot (BC^T - M)\|_F^2, \tag{14.12}$$

where $\odot$ denotes element-wise multiplication. This problem has to be solved for relatively high levels of missing data using iterative local methods. Figure 14.5 shows a solution recovered from a measurement matrix containing roughly 50% of the data from the hand data set.

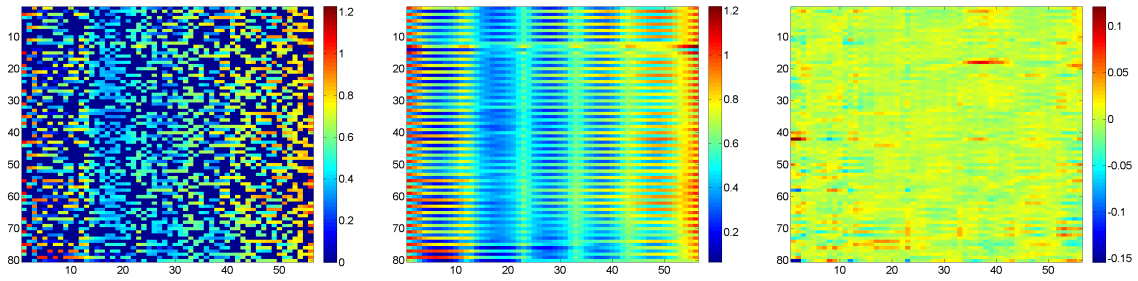


Figure 14.5: *Left* - The measurement matrix with roughly 50% missing entries for the hand data set in Figure 14.1. *Middle* - A rank 5 approximation obtained using local optimization. *Right* - The difference between the true measurement matrix (without missing data) and the obtained rank 5 approximation.

Lecture 15:

Stereo and Surface Estimation

When camera positions have been determined, using structure from motion, we would like to compute a dense surface model of the scene. In this lecture we will study the so called Stereo Problem, where the goal is to estimate the depth of each pixel in an image. This requires computing a match for each pixel in the image even if the texture is ambiguous.

Figure 15.1: Left and Middle: Two images used for computing depth estimates for every pixel in the left image. Right: The depth estimate color coded.

15.1 Rectified Cameras, Disparity vs. Depth

Dense depth estimation requires matching of each pixel to a corresponding pixel in neighboring images. Because of ambiguous texture this problem is difficult to solve. Since cameras are known we can however use the epipolar lines to limit the search.

We will start by assuming that we have a pair of so called rectified cameras, $P_1 = K[I \quad 0]$ and

$$P_2 = K \begin{bmatrix} I & \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix} \end{bmatrix}. \tag{15.1}$$

Geometrically this means that both cameras have the same orientation and that the second camera position is a translation in the x-direction of the first camera, see Figure 15.2. In general image pairs taken with regular cameras do not fulfill these assumptions, however they can be modified to do so. The the line segment joining the two camera centers is called the baseline.

There are several ways of rectifying two cameras. A simple approach is to first select the orientation of axes of the new camera coordinate system in one of the cameras and then rotate both the cameras to this new coordinate

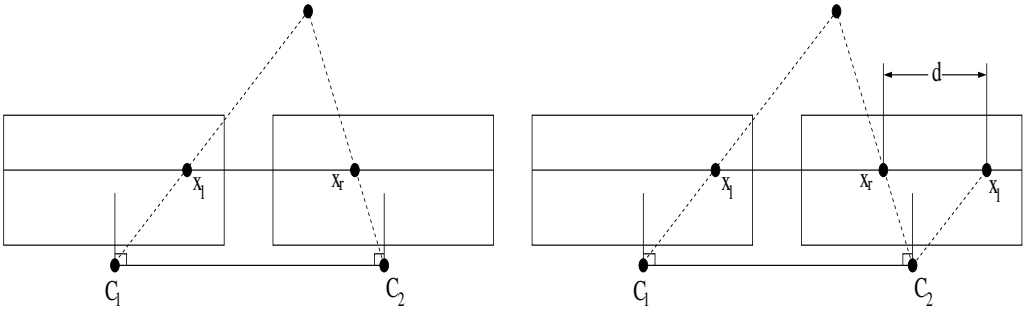


Figure 15.2: Stereo with rectified cameras. The cameras have the same orientation and the image plane normals are perpendicular to the baseline. Left: The epipolar lines are parallel to the x-axis. Right: The disparity.

system. The new x-axis should be parallel to the baseline and the new y and z-axes should be perpendicular to it. To keep the changes that occur when transforming the image small, we should select the new coordinate system to be as similar to the old one as possible. We can select the new x-axis by projecting the old x-axis onto the baseline and normalizing it. When the x-axis has been determined we can chose the new-z axis as the projection of the old z-axis onto the plane going through the camera center with the new x-axis as normal. When both the x and z axes are determined we can find the y axis by using the cross product. Once we known the new camera orientations we rotate the cameras. Since the rectified cameras and old cameras are related through pure rotations the rectified images are obtained by transforming the images using a homography.

Next we will show that for this setup the epipolar lines are parallel to the x-axis of the image. Suppose that

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{15.2}$$

Then the projection of a scene point $X = (X, Y, Z, 1)$ in the cameras P_1 and P_2 is given by

$$x_l = K \begin{bmatrix} I & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \gamma f X + sfY + x_0 Z \\ fY + y_0 Z \\ Z \end{pmatrix} \tag{15.3}$$

$$x_R = K \begin{bmatrix} I & \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \gamma f (X + b) + sfY + x_0 Z \\ fY + y_0 Z \\ Z \end{pmatrix} \tag{15.4}$$

In regular coordinates this gives us the projections

$$(x_L, y_L) = \left(\tfrac{fX + sfY + x_0 Z}{Z}, \tfrac{fY + y_0 Z}{Z} \right) \tag{15.5}$$

$$(x_R, y_R) = \left(\tfrac{f(X+b) + sfY + x_0 Z}{Z}, \tfrac{fY + y_0 Z}{Z} \right). \tag{15.6}$$

Therefore $y_l = y_R$ for any two corresponding points, which means that the epipolar lines are horizontal. The difference between the x-coordinates

$$d = x_R - x_L = \frac{\gamma f b}{Z} \tag{15.7}$$

is called the disparity. The matching problem can be seen as the problem of assigning a disparity to each point in the image. The disparity is inversely proportional to the depth Z, see Figure 15.3. If we assume that disparities can only be determined up to integer values (no-subpixel accuracy), then it can be seen from Figure 15.3 that accurate (high resolution) depth estimation can only be achieved when Z is relatively small (with respect to the baseline b).

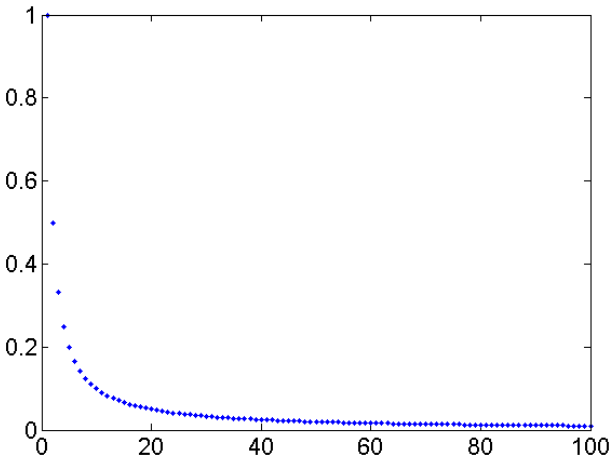


Figure 15.3: Depth as a function of disparity ($\gamma fb = 1$). Large disparities gives higher depth resolution.

15.2 Matching Criteria

Given a pixel and a candidate depth/disparity we need to have a criteria for evaluating if this is a good match or not. Previously in the course we have used SIFT features. However, this type of features have various invariance properties that we do not want here, since the camera positions are already known. A simple commonly used criteria is the normalized cross correlation. The cross correlation compares a patch around the pixel to a patch around the potential match. If I_1 and I_2 are the two patches then their correlation is

$$NCC(I_1, I_2) = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(I_1(x_i) - \bar{I}_1)(I_2(x_i) - \bar{I}_2)}{\sigma(I_1)\sigma(I_2)}, \tag{15.8}$$

where the sequence x_i are the pixels being compared, $\bar{I}_1$, $\bar{I}_2$, $\sigma(I_1)$ and $\sigma(I_2)$ are the mean values and standard deviations of of the two patches. The result is a number between -1 and 1 where 1 means that the patches similar.

The normalized cross correlation is invariant to translation and rescaling of the image intensities, which is very useful if the two images are captured under different lighting conditions. Figure 15.4 shows the evaluation of normalized cross correlation along an epipolar line.

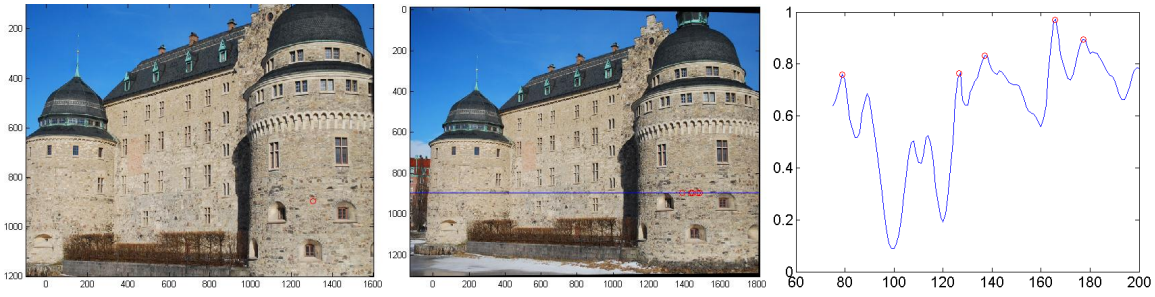


Figure 15.4: Evaluation of the normalized cross correlation along an epipolar line. Left: Left image and the image point of interest. Middle: Corresponding epipolar line and a few local maxima of the NCC. Right: NCC for a range of disparities. Red rings are the same local maxima as in the middle image.

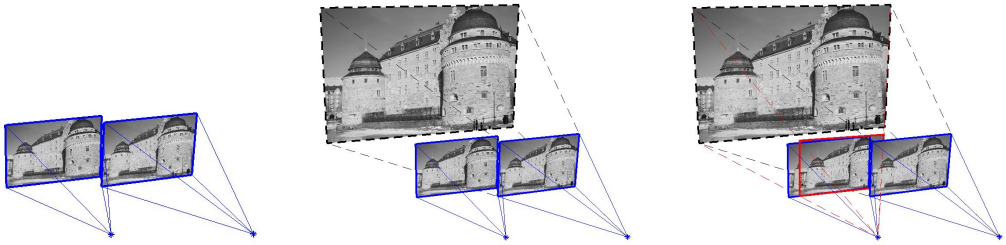
15.3 Plane Sweep Algorithms

When we use more than two cameras it might not be possible to rectify all the images simultaneously. In this case an alternative is to employ a plane sweep algorithm. The basic idea is that if all the pixels have the same depth then all the scene points are located on a plane in 3D. Since projections in two images from points on a plane are related by a homography, we can hypothesize a depth, transform the pixels of one camera into the second and compute cross correlation. Sweeping the plane through a series of hypotheses gives a cost for assigning pixels to the hypothesized depths.

If the cameras are $P_1 = K[I \quad 0]$ and $P_2 = K[R \quad t]$ then the plane Π with scene points at the depth Z is given by $\Pi = (0, 0, -1/Z, 1)$. The homography H_Z for the depth Z is then given by the formula

$$H_Z = K\left(R + t\begin{pmatrix}0 & 0 & \frac{1}{Z}\end{pmatrix}\right)K^{-1}. \tag{15.9}$$

(See Assignment 1, Exercise 6 for a derivation.)



(a) (b) (c)

Figure 15.5: Plane sweep algorithms. Given two cameras (a), hypothesize a depth for all the pixels (b), project into the second camera and compare pixel values (c). Sweeping the scene plane over different depths gives costs of assigning pixels to these hypothesized depths.

Compared to rectified cameras where we can simply compare patches along the epipolar line, transforming the image using the homography is more computationally expensive. On the other hand this approach is very convenient in that it works with general camera configurations. Furthermore, since the image is re-sampled during the transformation we do not need to limit ourselves to depths that are are inverse values of integer valued disparities. Therefore sup-pixel accuracy is naturally achieved with this approach.

15.4 Regularization/Energy Minimization

The result of the plane sweep algorithm is a function for each pixel that specifies a cost of assigning that pixel a certain depth. By selecting the smallest cost for each pixel we obtain a dense depth estimate. The resulting surface will often be noisy. This is because individual pixel estimates can be unreliable due to ambiguous texture and self occlusion.

To improve the estimated surface and reduce the noise a common approach is to seek an assignment where neighboring pixels have similar depths. Typically one minimizes an energy functional of the form

$$\sum_i \sum_{j \in \mathcal{N}(i)} E_{ij}(Z_i, Z_j) + \sum_i E_i(Z_i). \tag{15.10}$$

The second term $E_i(Z_i)$ is the cost of assigning the depth Z_i to pixel i. This term is typically referred to as the data term since it is based on image data. The first term $E_{ij}(Z_i, Z_j)$ penalizes differences in depth between pixel i and a pixel j that is in the neighborhood $\mathcal{N}(i)$ of pixel i. This term is usually called the smoothness term since it favors solutions with smooth surfaces.

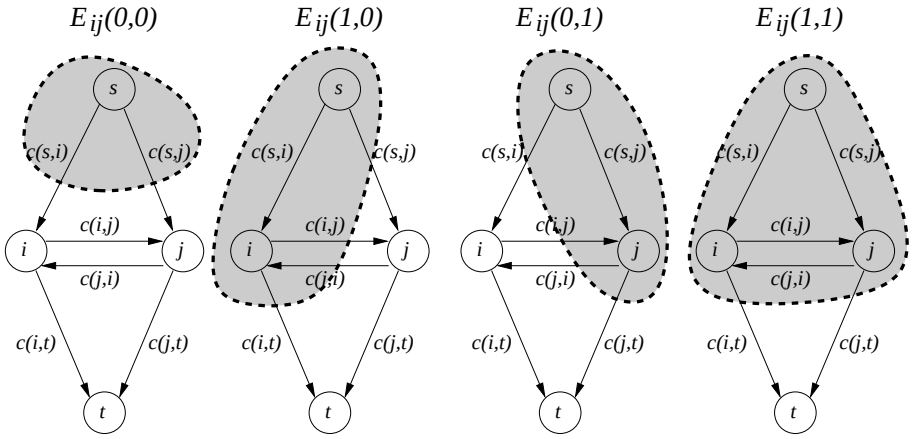


Figure 15.6: The four cuts corresponding to the values of the energy term in (15.12)-(15.15).

Optimizing this type of energy is challenging due to the numerous local minima of the individual data terms $E_i(Z_i)$, see Figure 15.4. The most successful methods for doing this are based on discrete graph methods. If Z_i is binary, then the energy (15.10) can be seen as the minimum cut on a graph where the nodes correspond to the pixels and the edges correspond to the neighborhood structure. Finding the minimum cut in a graph can be solved efficiently (in polynomial time complexity) by computing the maximal flow on the same graph.

Let us first consider a two pixel case with the energy functional

$$E(Z_i, Z_j) = E_{ij}(Z_i, Z_j) + E_{ji}(Z_j, Z_i) + E_i(Z_i) + E_j(Z_j). \tag{15.11}$$

The corresponding graph has two nodes representing the pixels and two special nodes; the source s and the sink t. The graph has directed edges from s to i and j, from i and j to t and between i and j. In addition each edge has a (non-negative) capacity $c(i, j)$. A cut in this graph is a partitioning of the nodes into two disjoint sets S and T where S are the nodes connected to the source and T the nodes connected to the sink. The value of the cut is the sum of all the edge capacities of the edges going form S to T.

The energy (15.11) can be solved as a minimal cut problem on this graph under certain conditions. First of all we see that $E(Z_i, Z_j) \geq 0$ since graph edges have non-negative capacities. Note that if the energy is not positive we can always add a constant to $E(Z_i, Z_j)$ without changing the minimizer and thereby obtain an equivalent positive energy. If we let $i \in S$ mean that $Z_i = 1$ then from Figure 15.6 we see that

$$\begin{aligned} E(0,0) &= c(s,i) + c(s,j) && (15.12)\\ E(1,0) &= c(i,j) + c(s,j) + c(i,t) && (15.13)\\ E(0,1) &= c(j,i) + c(s,i) + c(j,t) && (15.14)\\ E(1,1) &= c(i,t) + c(j,t). && (15.15) \end{aligned}$$

Since the capacities are all positive we see from these equations that in order to be able to represent this energy with a graph, then

$$E(0,0) + E(1,1) \leq E(1,0) + E(0,1) \tag{15.16}$$

must hold. Energies with pairwise smoothing terms that fulfill (15.16) are called submodular. It can be shown that the condition (15.16) actually ensures that the energy can be represented by a graph. Furthermore, it is easy to see that the sum of submodular energies is also submodular. This means that if the terms E_{ij} of the energy (15.10) are all submodular then we can minimize this energy by solving a max-flow/min-cut problem.

When Z_i is not binary (as in stereo) we typically employ move making algorithms. The goal of these algorithms is to modify as many pixels as possible simultaneously in order to avoid getting stuck in poor local minima. One of the most popular approaches is that of α-expansion. Suppose that we have a current assignment of depths. Then each pixel is given the option to switch from the current depth to a new depth α. Since each pixel has two choices

(move to α or retain the old value) this can be formulated as a binary problem. If this new binary problem is submodular it can be solved efficiently using max-flow/min-cut algorithms.

Suppose that in the current assignment $Z_i = \beta$ and $Z_j = \gamma$ and consider the term $E_{ij}(Z_i, Z_j)$ when we let the pixels switch to α. We define a new binary energy term $E_B(x_i, x_j)$ by letting $x_i = 0$ when Z_i retains its current assignment, $x_i = 1$ when Z_i switches to α (and x_j similarly). Then

$$E_B(0,0) = E_{ij}(\beta, \gamma) \tag{15.17}$$
$$E_B(0,1) = E_{ij}(\beta, \alpha) \tag{15.18}$$
$$E_B(1,0) = E_{ij}(\alpha, \gamma) \tag{15.19}$$
$$E_B(1,1) = E_{ij}(\alpha, \alpha). \tag{15.20}$$

Therefore, if the smoothnes terms of the energy (15.10) fulfills

$$E_{ij}(\beta, \gamma) + E_{ij}(\alpha, \alpha) \leq E_{ij}(\beta, \alpha) + E_{ij}(\alpha, \gamma) \tag{15.21}$$

we can solve the α-expansion efficiently using max-flow/min-cut algorithms.

In many cases we can assume that the energy is non-negative, symmetric $E_{i,j}(\beta, \gamma) = E(\gamma, \beta)$ and that $E(\alpha, \alpha) = 0$. In these cases it can be seen that E_{ij} has to be a metric distance since (15.21) reduces to the triangle inequality. A commonly used smoothness term that fulfills the metric conditions is the truncated absolute distance

$$E_{ij}(\beta, \gamma) = \min(|\beta - \gamma|, t). \tag{15.22}$$

This term favors assignments where neighboring depths have small disparity differences. In real images we should however also allow for discontinuous depth assignments due to object boundaries (transitions between smooth surfaces). Therefore the threshold t is added to the energy to prevent over-smoothing at discontinuities.