Some lecture notes on
Geometric Computer Vision

Peter Sturm
INRIA Grenoble – Rhône-Alpes
peter.sturm@inria.fr

Contents

0 Background

0.1 General Bibliography

Recommended:

- R.I. Hartley et A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000 (2nd edition 2004).

- T. Moons, L. Van Gool and M. Vergauwen, *3D Reconstruction from Multiple Images: Part 1 Principles*, Foundations and Trends in Computer Graphics and Vision, Volume 4, Issue 4, 2009.

0.2 Notations

Matrices are written as P, vectors as $\mathbf{a}$, and scalars as s. Coefficients of matrices or vectors are written as scalars with appropriate indices, for example P_{ij} or a_i.

The identity matrix of size $n \times n$ is written as $\mathtt{I}_n$ (the index n may be omitted if the size is clear from the context). The vector of length n consisting of zeroes, is denoted $\mathbf{0}_n$ (or $\mathbf{0}$ if the length is clear from the context).

Coordinate vectors associated with 3D points are named by capital letters, those of 2D points, by lower case letters: $\mathbf{Q}$ respectively $\mathbf{q}$.

The transpose respectively inverse of a matrix are denoted by P^T respectively P^{-1}. The transpose of the inverse of a matrix is $\mathsf{P}^{-\mathsf{T}}$.

Vectors are implicitly considered as column vectors (or, matrices consisting of a single column). The transpose of a vector, denoted by $\mathbf{a}^\mathsf{T}$, is thus interpreted as a row vector or, a matrix consisting of a single row.

The scalar product of two vectors $\mathbf{a}$ and $\mathbf{b}$ of length n is given by $\sum_{i=1}^{n} a_i b_i$. It can also be written as $\mathbf{a}^\mathsf{T}\mathbf{b}$.

The symbol $\times$ stands for the cross-product of two vectors of length 3. The vector $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ is orthogonal to $\mathbf{a}$ and $\mathbf{b}$. The cross-product can be written as a matrix-vector multiplication:

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \mathbf{b} \ .$$

The matrix in this equation – the "matrix associated with the cross-product of vector $\mathbf{a}$ " – is skew-symmetric. It is denoted as $[\mathbf{a}]_\times$.

The symbol $\sim$ expresses equality up to scale, of two vectors or two matrices: $\mathbf{a} \sim \mathbf{b}$ means that there exists $s \neq 0$ with $\mathbf{a} = s\mathbf{b}$. The notion of equality up to scale is important when homogeneous coordinates are used.

0.3 Matrix decompositions

Bibliography:

- G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd edition, The John Hopkins University Press, 1996.
- W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C*, 2nd edition, Cambridge University Press, 1992.

0.3.1 QR decomposition

The QR decomposition of an $m \times n$ matrix A is given by:

$$\mathsf{A} = \mathsf{QR} \ ,$$

where Q is an orthonormal matrix of size $m \times m$ and R an upper triangular matrix of size $m \times n$.
The above is the standard definition of the QR decomposition. Another variant is required in the lecture; we write it down for square matrices:

$$\mathsf{A}_{m\times m} = \mathsf{B}_{m\times m}\mathsf{C}_{m\times m} \ ,$$

where B is upper triangular:

$$\mathsf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ 0 & B_{22} & \cdots & B_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & B_{mm} \end{pmatrix}$$

and C is orthonormal.

0.3.2 Cholesky decomposition

Let A be a symmetric and positive definite matrix of size $m \times m$. It can be decomposed as:

$$\mathsf{A} = \mathsf{B}\mathsf{B}^{\mathsf{T}} \ ,$$

where B is upper triangular[1] of size $m \times m$.

[1]The standard definition of Cholesky decomposition is for B being lower triangular.

1 Camera modeling

In order to carry out numerical computations and geometric reasoning using images, we require a model that describes how a camera projects the 3D world onto a 2D image. There are many such models which describe in more or less detail and accuracy the characteristics of a camera (optical, electronic, mechanical). The most used model in computer vision represents a good compromise between simplicity and closeness to the "true" projection realized by most cameras: the so-called **pinhole model**, which effectively represents a perspective projection.

1.1 Geometric model

Geometrically, the pinhole model can be described by an **optical center** (or, projection center) and an **image plane** (or, retina). A 3D point gets projected along the **line of sight** that connects it with the optical center: its **image point** is the intersection of this line, with the image plane (see figure 1). The line of sight is also sometimes called camera ray or optical ray. The 3D-to-2D projection realized by the above operation, is a perspective projection.

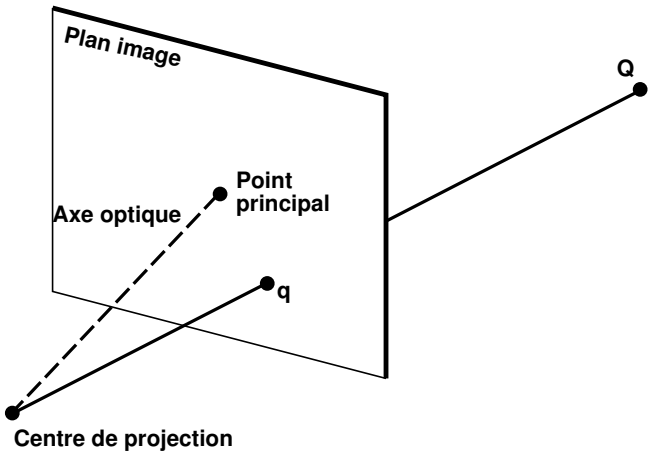


Figure 1: The pinhole model.

Two important elements of the pinhole model are (cf. figure 1. The line passing through the optical center and that is perpendicular to the image plane, is called **optical axis**. The intersection of the optical axis and the image plane, is the **principal point**.

1.2 Algebraic model

We just described a geometric camera model. In order to perform computations with it, we need to derive an algebraic description of it. In the following, we do this, based on defining physical camera parameters. In order to carry out computations with points, we need coordinates and thus coordinate systems. For much of this lecture, homogeneous coordinates shall be used.

1.2.1 Stage 1 – 3D-to-2D projection and the focal length

We will derive projection equations using 4 coordinate systems (cf. figure 2). We start with a 3D coordinate system that is attached to the camera – the **camera system**. As its origin we choose the optical center and as Z-axis, the optical axis. The X and Y axes are chosen as being parallel to the image plane and perpendicular to one another. In addition, they are chosen such as to be "aligned" with the directions of the pixel grid.

Next, we define a 2D coordinate system for the image plane – the **image system**. Its origin is the principal point and its x and y axes are parallel to the X and Y axes of the camera system. In simple terms, we may consider the image system as the orthographic projection of the camera system onto the image plane.

The **focal length** f is the distance between the optical center and the image plane (hence, the distance between the optical center and the principal point).

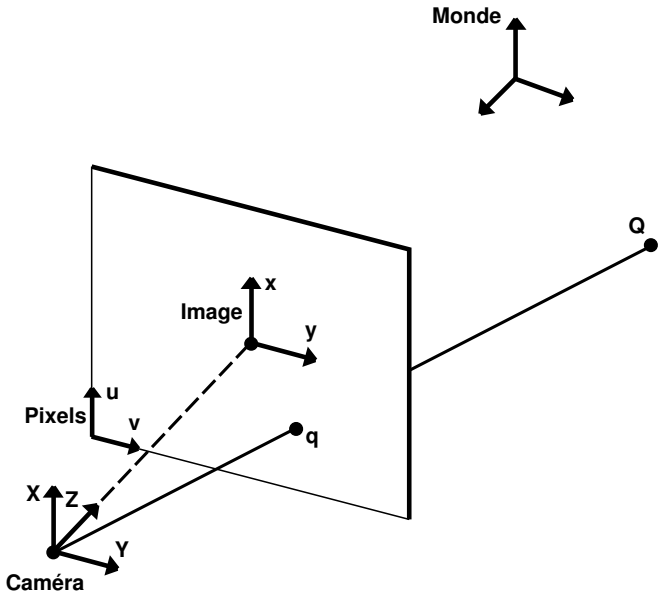


Figure 2: The coordinate systems used to define projection equations.

We can now derive the projection equations for a 3D point $\mathbf{Q}$, whose coordinates are given in the camera system (hence the exponent ^{c}):

$$\mathbf{Q}^c = \begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} \quad .$$

The x and y coordinates of the image point $\mathbf{q}$ can be computed using similar triangles (see

lecture):

$$x = f\frac{X^c}{Z^c} \quad (1)$$

$$y = f\frac{Y^c}{Z^c} \ . \quad (2)$$

In homogeneous coordinates, these equations can be written as a matrix-vector product:

$$\mathbf{q} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} \ . \quad (3)$$

Mind the symbol $\sim$ in this equation: the equality between the vectors represented by the left and right hande side, is up to scale only (this is the price to pay for using homogeneous coordinates).

We may verify that equation (3) is equivalent to equations (1) and (2). To do so, consider the right hand side of equation (3):

$$\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} = \begin{pmatrix} fX^c \\ fY^c \\ Z^c \end{pmatrix} \ .$$

This is a vector of homogeneous coordinates. If we transform this into non homogeneous coordinates (by dividing the entire vector by its last coordinate), then we obtain:

$$\begin{pmatrix} fX^c \\ fY^c \\ Z^c \end{pmatrix} \sim \begin{pmatrix} fX^c/Z^c \\ fY^c/Z^c \\ 1 \end{pmatrix} \ .$$

It is easy to see that this corresponds to equations (1) and (2).

1.2.2 Stage 2 – Taking into account pixels

In this lecture, we only consider digital images. A digital image is a matrix of numbers (e.g. greylevels of RGB color values) – its cells corresponding to the **pixels**. For image processing operations for example, pixels are addressed by their coordinates, which are effectively obtained by counting pixels in horizontal and vertical direction starting from a corner of the image area. This means that we cannot directly use the 2D coordinates defined in the previous section (i.e. those of the image coordinate system). Rather, we have to operate a change of coordinate system.

Different image display or processing softwares use sometimes different coordinate systems for pixels. The origin is usually either the upper left or the lower left corner of the image

area. The first coordinate axis is sometimes horizontal, sometimes vertical. Here, in order to simplify expressions, the **pixel system** is defined as shown in figure 2. Pixel coordinates are denoted by u and v.

The transformation between image and pixel system requires a translation. In addition, one has to operate a change of unit: the *image* system is metric – without saying it so far, it is usually based on a metric unit, like mm – whereas the unitof the *pixel* system is "number of pixels", which is not a metric unit of course. Especially for older cameras, pixels are not equally spaced in horizontal and vertical direction (due to historic TV standards). This implies that the change of unit may be different in these two directions.

Let $-x_0$ and $-y_0$ be the coordinates of the lower left corner of the image area, with respect to the image system. Let k_u be the density of pixels in the u direction and k_v that in v direction (expressed in number of pixels per mm for example).

Consider now the point $\mathbf{q}$ with coordinates x and y in the image system. Its coordinates in the pixel system are obtained by carrying out a translation, followed by the changes of unit described above. In homogeneous coordinates:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad . \tag{4}$$

We can now combine equations (3) and (4) to model the 3D-to-2D projection from the camera into the pixel system:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} \quad . \tag{5}$$

Remarks. In order to give some orders of magnitude: the focal length f typically ranges between several to several hundreds of mm (above, it is usually considered as infinite). The photosensitive area of a camera is typically a rectangle with several mm side length and the pixel density is usually of the order of one or several hundreds of pixels per mm.

For cameras with well mounted optics, the principal point is usually very close to the center of the photosensitive area.

1.2.3 Stage 3 – Taking into account camera displacements

Until now, we have represented 3D points in a coordinate system attached to the camera. In order to take into account camera displacements, we have to choose a "static" coordinate system, attached to the 3D **scene**. We thus introduce a **world system** (cf. figure 2). It can be chosen arbitrarily but once chosen, one needs to stick to it, for the duration of an application. The position of 3D points **and** of cameras, is thus given with respect to this world system.

In the following, we model the pose of a camera. This subsumes the **position** as such – the position of the optical center – we well as the **orientation** of the camera. Let us represent

the position by a 3-vector $\mathbf{t}$ such that

$$\begin{pmatrix} \mathbf{t} \\ 1 \end{pmatrix}$$

are the homogeneous coordinates of the optical center. The orientation of the camera can be expressed by a rotation; we represent this by a **rotation matrix**[2] of size 3×3, denoted by R. At the end of this section, we briefly remind some characteristics of rotation matrices.

Let X^m, Y^m, Z^m be the coordinates of a 3D point $\mathbf{Q}$, relative to the world system. The change of coordinates to the camera system, can be written as:

$$\begin{pmatrix} X^c \\ Y^c \\ Z^c \end{pmatrix} = \mathsf{R} \left(\begin{pmatrix} X^m \\ Y^m \\ Z^m \end{pmatrix} - \mathbf{t} \right) = \mathsf{R} \begin{pmatrix} X^m \\ Y^m \\ Z^m \end{pmatrix} - \mathsf{R}\mathbf{t} \ .$$

In homogeneous coordinates:

$$\begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} = \begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} X^m \\ Y^m \\ Z^m \\ 1 \end{pmatrix} \ . \tag{6}$$

Let us verify that the optical center is indeed the origin of the camera system:

$$\begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{t} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathsf{R}\mathbf{t} - \mathsf{R}\mathbf{t} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \ .$$

Remark. Above, we have expressed the position and orientation of a camera. **Displacements** can then be modeled by sequences of rotation matriced and translation vectors.

Reminders on rotation matrices. The matrices that serve, for our purposes, the representation of rotations, are *orthonormal* matrices: their columns (respectively rows) are mutually orthogonal 3-vectors (i.e. their scalar products vanish) of norm 1. This implies that their determinant equals ± 1. For a rotation matrix, the determinant must equal $+1$ (if it equals -1, the matrix represents a *reflection*). The above implies that the inverse of a rotation matrix is its transpose: $\mathsf{R}\mathsf{R}^\mathsf{T} = \mathtt{I}$.

Each rotation can be decomposed into three base rotations, around axes of the current coordinate system. Among the different permutations that are possible, one that is often used, is:

$$\mathsf{R} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \ .$$

The angles α, β and γ are associated with the X, Y and Z axes respectively. They are also called **Euler angles**.

[2] Other representations exist, such as the quaternions.

1.2.4 Complete model

We simply combine the results of the previous sections, i.e. equations (5) and (6) in order to obtain the complete algebraic camera model:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} X^m \\ Y^m \\ Z^m \\ 1 \end{pmatrix} \quad . \tag{7}$$

Let us identify the following 3×3 matrix K:

$$\mathsf{K} = \begin{pmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{pmatrix} \quad .$$

With this notation, the matrices in equation (7) can be written as:

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{K} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}$$

which is the same as:

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{KR} & -\mathsf{KR}\mathbf{t} \end{pmatrix} \quad . \tag{8}$$

and still the same as:

$$\mathsf{P} \sim \mathsf{KR} \begin{pmatrix} \mathtt{I} & -\mathbf{t} \end{pmatrix} \quad .$$

The 3×4 matrix P in equation (8) is called **projection matrix**. It maps 3D points to 2D image points, all expressed in homogeneous coordinates.

1.2.5 Extrinsic and intrinsic parameters

We can group the parameters defining the 3D-to-2D projection carried out by a camera, into two sets: the **extrinsic parameters** – the rotation matrix R and the translation vector $\mathbf{t}$ – representing position and orientation of the camera with respect to the "world". And the **intrinsic parameters**, which explain what happens "inside" the camera; these are the parameters f, k_u, k_v, x_0 and y_0 defined in sections 1.2.1 and 1.2.2. They are contained in the upper triangular co-called **calibration matrix** K defined above:

$$\mathsf{K} = \begin{pmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{pmatrix} \quad .$$

We can see that these 5 parameters define only 4 coefficients of the calibration matrix. Hence, we may define the following 4 intermediate parameters (which also have the advantage to be

unit-free):

$$\begin{aligned} \alpha_u &= k_u f \\ \alpha_v &= k_v f \\ u_0 &= k_u x_0 \\ v_0 &= k_v y_0 \quad . \end{aligned}$$

It is these 4 parameters that will be considered in the following as the **intrinsic parameters** of a camera. Their meaning is:

- α_u and α_v express the focal length, in numbers of pixels (once in horizontal, once in vertical direction respectively pixel density).
- u_0 and v_0 are the coordinates of the principal point, in the pixel system.

The calibration matrix thus becomes:

$$\mathsf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{9}$$

Remark. The ratio of the pixels' side lengths, $\tau = k_u/k_v$, is an important intrinsic parameter. It is also called a camera's **aspect ratio**. For modern cameras, it normally equals 1.

Remark. Sometimes, an additional intrinsic parameter is introduced that allows to model u and v axes that are not orthogonal, the so-called **skew parameter** s:

$$\mathsf{K} = \begin{pmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad .$$

This parameter is only considered due to historical reasons (some older cameras may have slightly unsynchronized pixel-readout from the CCD). We neglect it in the rest of this lecture.

2 Camera calibration

The main goal of camera calibration is to determine the camera's intrinsic parameters. However, it is not straightforward to dissociate intrinsic from extrinsic parameters, which is why the standard calibration process usually starts by first computing both sets of parameters (intrinsic and extrinsic). The usual calibration process consists of taking one or several images of a **calibration object** whose 3D structure is perfectly known. For the object shown in figure 3 for example, the positions of the white markers are known to a high precision, in a coordinate system attached to the object. In images, the markers are easy to extract due to the high white/black contrast. After extraction, one needs to establish **correspondence**: for each marker extracted in an image, one needs to determine the original marker whose image it is. This process can be completely automatized, based on the particular spatial layout of the object and its markers.

Figure 3: An example of a calibration object.

We thus have a set of n 3D points whose coordinates are known, and the corresponding image points, whose coordinates are known as well. For each such correspondence, we can establish an equation of type (7) and aim at using this to determine the unknown camera parametes.

Equation (7) has the drawback that the unknowns are multiplied with one another, i.e. equations are non-linear and thus difficult to solve. We will thus not determine intrinsic and extrinsic parameters directly, but will proceed in two stages. First, we will compute the 12 elements of the projection matrix P (cf. equation (8)). Second, once this is done, the intrinsic and extrinsic parameters will be extracted from P. These two stages are explained in the following.

2.1 Computation of the projection matrix

As explained above, we have n equations of type:

$$\mathbf{q}_p = \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} \sim \mathsf{P}\mathbf{Q}_p \ , \tag{10}$$

where $\mathbf{q}_p$ and $\mathbf{Q}_p$, for $p = 1 \dots n$ are the homogeneous coordinate vectors of 2D and 3D points respectively and where P is the unknown projection matrix, of size 3×4.

Equation (10), being defined up to scale only (cf. the $\sim$ symbol), cannot be used directly. However, upon dividing the vector on each side of the equation by its last coordinate, we get actual equations:

$$\begin{aligned} u_p &= \frac{(\mathsf{P}\mathbf{Q}_p)_1}{(\mathsf{P}\mathbf{Q}_p)_3} \\ v_p &= \frac{(\mathsf{P}\mathbf{Q}_p)_2}{(\mathsf{P}\mathbf{Q}_p)_3} \ . \end{aligned}$$

Finally, multiplying these equations by the denominator of the right-hand side, we obtain:

$$\begin{aligned} u_p \, (\mathsf{P}\mathbf{Q}_p)_3 &= (\mathsf{P}\mathbf{Q}_p)_1 \\ v_p \, (\mathsf{P}\mathbf{Q}_p)_3 &= (\mathsf{P}\mathbf{Q}_p)_2 \ , \end{aligned}$$

or, in full detail:

$$\begin{aligned} u_p \, (P_{31}Q_{p,1} + P_{32}Q_{p,2} + P_{33}Q_{p,3} + P_{34}Q_{p,4}) &= P_{11}Q_{p,1} + P_{12}Q_{p,2} + P_{13}Q_{p,3} + P_{14}Q_{p,4} \\ v_p \, (P_{31}Q_{p,1} + P_{32}Q_{p,2} + P_{33}Q_{p,3} + P_{34}Q_{p,4}) &= P_{21}Q_{p,1} + P_{22}Q_{p,2} + P_{23}Q_{p,3} + P_{24}Q_{p,4} \ . \end{aligned}$$

We can observe that these equations are linear in the coefficients P_{ij} of the projection matrix P, thus "easy" to solve. The above equations, for all n 2D-to-3D correspondences, can be gathered in an equation system of the following form:

$$\left(\begin{array}{ccc|ccc|ccc} Q_{1,1} & \cdots & Q_{1,4} & 0 & \cdots & 0 & -u_1Q_{1,1} & \cdots & -u_1Q_{1,4} \\ 0 & \cdots & 0 & Q_{1,1} & \cdots & Q_{1,4} & -v_1Q_{1,1} & \cdots & -v_1Q_{1,4} \\ \hline & \vdots & & & \vdots & & & \vdots & \\ & \vdots & & & \vdots & & & \vdots & \\ \hline Q_{n,1} & \cdots & Q_{n,4} & 0 & \cdots & 0 & -u_nQ_{n,1} & \cdots & -u_nQ_{n,4} \\ 0 & \cdots & 0 & Q_{n,1} & \cdots & Q_{n,4} & -v_nQ_{n,1} & \cdots & -v_nQ_{n,4} \end{array}\right) \left(\begin{array}{c} P_{11} \\ \vdots \\ P_{14} \\ \hline P_{21} \\ \vdots \\ P_{24} \\ \hline P_{31} \\ \vdots \\ P_{34} \end{array}\right) = \mathbf{0} \ ,$$

or, in short:

$$\mathsf{A}_{2n\times 12} \ \mathbf{x}_{12} = \mathbf{0}_{2n} \ .$$

The matrix A can be fully computed from the coordinates of the 2D and 3D points and the vector $\mathbf{x}$ contains all the unknowns – the 12 elements of P. Each point correspondence gives 2 equations. Hence, 6 correspondences are in general sufficient to compute P.

In practice, the data, i.e. the point coordinates (especially of the 2D points) are affected by "noise" (i.e. are inaccurate): the 3D points $\mathbf{Q}_p$ are only known up to a finite precision and the extraction of the image points $\mathbf{q}_p$ cannot be done without error. Noise on the data means that there is no exact solution $\mathbf{x}$ to the above equation system. Hence, one needs to compute a solution $\hat{\mathbf{x}}$ that is the best possible, according to some criterion. The usual criterion is to solve for $\mathbf{x}$ in the least squares sense:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{i=1}^{2n} \left((\mathsf{A}\mathbf{x})_i \right)^2 \quad .$$

Since the equation system is homogeneous (the right-hand side is a vector of zeroes), one needs to avoid the trivial solution $\mathbf{x} = \mathbf{0}$. This can be done by adding a constraint on the norm, e.g.: $\|\mathbf{x}\| = 1$. Thus, the problem to be solved becomes:

$$\begin{aligned} \hat{\mathbf{x}} \quad = \quad & \arg\min_{\mathbf{x}} \sum_{i=1}^{2n} \left((\mathsf{A}\mathbf{x})_i \right)^2 \\ & \text{such that } \|\mathbf{x}\| = 1 \quad . \end{aligned}$$

This is a standard problem in numerical analysis. It can for example be solved using the *singular value decomposition* (SVD) of matrix A.

We have mentioned above that 6 correspondences are sufficient to obtain a solution. However, in order to reduce the impact of noise in the data, it is advised to use as many correspondences as possible in practice (usually, of the order of one or several hundreds).

2.2 Extracting intrinsic and extrinsic parameters

Once the projection matrix P is determined, we can extract the parameters we're actually interested in – the calibration matrix K (i.e. the intrinsic parameters), the rotation matrix R and the camera position $\mathbf{t}$ – noting that R and $\mathbf{t}$ will be given with respect to the coordinate system of the calibration object. From (8), we have:

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{KR} & -\mathsf{KR}\mathbf{t} \end{pmatrix} \quad .$$

2.2.1 Extracting the intrinsic parameters

Let $\bar{\mathsf{P}}$ be the 3×3 sub-matrix of P composed of its first 3 columns. We have:

$$\bar{\mathsf{P}} \sim \mathsf{KR} \quad .$$

Let us multiply each side of the equation by its own transpose:

$$\bar{\mathsf{P}}\bar{\mathsf{P}}^{\mathsf{T}} \sim \mathsf{KR}\mathsf{R}^{\mathsf{T}}\mathsf{K}^{\mathsf{T}} \quad .$$

Since R is orthonormal, we have $\mathsf{R}\mathsf{R}^\mathsf{T} = \mathtt{I}$. Hence:

$$\bar{\mathsf{P}}\bar{\mathsf{P}}^\mathsf{T} \sim \mathsf{K}\mathsf{K}^\mathsf{T} \ . \tag{11}$$

On the left-hand side, we have a positive definite symmetric matrix [3]. Its Cholesky decomposition (see §0.3.2) gives an upper triangular matrix B with:

$$\bar{\mathsf{P}}\bar{\mathsf{P}}^\mathsf{T} \sim \mathsf{B}\mathsf{B}^\mathsf{T} \ .$$

If we compare this equation with (11) and remember that K is upper triangular by construction, we observe that B can be nothing else than the calibration matrix K we're looking for!

Nearly at least: the element $(3,3)$ of K equals 1 (cf. equation (9)). We thus should divide all elements of B by B_{33}. Once this is done, we can directly read off K from the resulting matrix and extract the individual intrinsic parameters using (9) :

$$\mathsf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \ .$$

2.2.2 Extracting the extrinsic parameters

Once the intrinsic parameters are known, the computation of the extrinsic parameters is not very difficult. From (8), we have:

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{K}\mathsf{R} & -\mathsf{K}\mathsf{R}\mathbf{t} \end{pmatrix} \ .$$

Hence:

$$\mathsf{K}^{-1}\mathsf{P} \sim \begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \end{pmatrix} \ .$$

Let A be the 3×3 sub-matrix consisting of the first three columns of $\mathsf{K}^{-1}\mathsf{P}$. Thus, we have: $\mathsf{A} \sim \mathsf{R}$. We can turn this into an exact equation (not only up to scale) by multiplying A with the appropriate scalar λ:

$$\lambda\mathsf{A} = \mathsf{R} \ .$$

How to choose λ? The equality of the above two matrices implies the equality of their respective determinants, hence:

$$\det(\lambda\mathsf{A}) = \lambda^3 \det\mathsf{A} = \det\mathsf{R} = +1 \ .$$

We can thus compute λ as:

$$\lambda = \sqrt[3]{1/\det\mathsf{A}} \ .$$

Now we have:

$$\lambda\mathsf{K}^{-1}\mathsf{P} = \begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \end{pmatrix} \ .$$

(note the exact equality, i.e. $=$ instead of $\sim$).

The first three columns of the matrix on the left-hand side directly give the rotation matrix R. Once R is computed, $\mathbf{t}$ can be determined trivially, by multiplying the fourth column of the left-hand side's matrix with $-\mathsf{R}^\mathsf{T}$.

[3] The product of a non singular matrix with its own transpose is positive definite and symmetric.

2.3 Bibliography

- R.I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.
- J. Kannala, J. Heikkilä, and S.S. Brandt, *Geometric camera calibration*, Wiley Encyclopedia of Computer Science and Engineering, 2008.
- P. Sturm et S.J. Maybank, *On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications,* IEEE International Conference on Computer Vision and Pattern Recognition, pp. 432-437, 1999.

3 Image mosaics

We consider the case where two or more images are taken by the same camera, **from the same viewpoint** (but with different orientations). It is not possible to infer a 3D model of the scene (without doing object recognition for example). However, one may stitch the images together in order to generate a larger and more complete image of the scene. Often, images are taken over $360°$; stitched together, this will result in a **panoramic image**; such images may for example be used for virtual visits of museums. A panoramic image, or **mosaic**, allows to generate novel intermediate images, that do not correspond to one of the input images. This way, a user may visualize the scene in "video mode" [4].

In this section, we see how to stitch images together that were taken from the same viewpoint. This comes, at the basis, down to estimating projective transformations between image pairs. At first, we only consider two images; the application of our findings to multiple images is then straightforward.

The goal for now is thus to determine a projective transformation that allows to stitch one image next to another one. Coordinate systems are as defined in the previous system. Since both images are taken from the same viewpoint, we may simplify things by adopting this viewpoint (the optical center) as origin of the world coordinate system. Hence, the camera position is given by $\mathbf{t} = \mathbf{0}$.

Let K be the camera's calibration matrix and R_1 and R_2 the rotation matrices associated with the two images. The two projection matrices are thus, following (8):

$$\begin{aligned} \mathsf{P}_1 &\sim \begin{pmatrix} \mathsf{K}\mathsf{R}_1 & \mathbf{0} \end{pmatrix} \\ \mathsf{P}_2 &\sim \begin{pmatrix} \mathsf{K}\mathsf{R}_2 & \mathbf{0} \end{pmatrix} . \end{aligned}$$

Consider now a 3D point $\mathbf{Q}$ (given in homogeneous coordinates):

$$\mathbf{Q} = \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} .$$

Its images are:

$$\mathbf{q}_1 \sim \mathsf{K}\mathsf{R}_1 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{12}$$

$$\mathbf{q}_2 \sim \mathsf{K}\mathsf{R}_2 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} . \tag{13}$$

On the side, we note that the image of $\mathbf{Q}$ does not depend on its coordinate T, which corresponds to the fact that every point on the line spanned by $\mathbf{Q}$ and the optical center, gets projected to the same image point.

[4] See e.g. Apple's QuickTime VR: `http://www.apple.com/quicktime/qtvr/index.html`

From equations (12) and (13) we obtain:

$$\begin{aligned} \mathsf{R}_1^\mathsf{T}\mathsf{K}^{-1}\mathbf{q}_1 &\sim \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \\ \mathsf{R}_2^\mathsf{T}\mathsf{K}^{-1}\mathbf{q}_2 &\sim \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad . \end{aligned}$$

Hence:

$$\mathsf{R}_2^\mathsf{T}\mathsf{K}^{-1}\mathbf{q}_2 \sim \mathsf{R}_1^\mathsf{T}\mathsf{K}^{-1}\mathbf{q}_1 \quad ,$$

and finally:

$$\mathbf{q}_2 \sim \mathsf{K}\mathsf{R}_2\mathsf{R}_1^\mathsf{T}\mathsf{K}^{-1}\mathbf{q}_1 \quad . \tag{14}$$

This signifies that there exists a projective transformation:

$$\mathsf{H} \sim \mathsf{K}\mathsf{R}_2\mathsf{R}_1^\mathsf{T}\mathsf{K}^{-1} \quad , \tag{15}$$

that links the two projections of the same 3D point.

The definition of H does not depend on the coordinates of the 3D point $\mathbf{Q}$ we started with, which means that the two images of *any* 3D point are linked by one and the same transformation H.

Notation. 2D projective transformations are often called **homographies**.

Two questions arise now:

- How to compute the homography H between two images?
- How to use it to stitch the two images together?

3.1 Estimating the homography

In case it is possible to measure the rotations a camera undergoes (for example by using girometers) and if the camera is calibrated (K is known), then one can directly compute H from equation (15). In practice however, one would like to compute the homography without requiring special equipment and/or without having to calibrate the camera. We will see that it is indeed possible to compute the homography from the images themselves, without any additional information. More precisely, one first determines point correspondences between the two images. This can be done manually, but fully automatic methods are now commonplace. The computation of H from such correspondences is very similar to that of the projection matrix P during calibration (see the previous section), the only difference being that 2D-to-2D correspondences are used to compute a 3×3 matrix, instead of 3D-to-2D correspondences to compute a 3×4 matrix.

Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be two corresponding points:

$$\mathbf{q}_1 = \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \qquad \mathbf{q}_2 = \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \quad .$$

They provide the following constraint on H:

$$\mathbf{q}_2 \sim \mathsf{H}\mathbf{q}_1 \quad .$$

Like in section 2.1, we go from homogeneous to "standard" coordinates:

$$\begin{aligned} u_2 &= \frac{(\mathsf{H}\mathbf{q}_1)_1}{(\mathsf{H}\mathbf{q}_1)_3} \\ v_2 &= \frac{(\mathsf{H}\mathbf{q}_1)_2}{(\mathsf{H}\mathbf{q}_1)_3} \quad . \end{aligned}$$

Then, each equation is multiplied by the denominator of the respective right-hand side:

$$\begin{aligned} u_2 \, (\mathsf{H}\mathbf{q}_1)_3 &= (\mathsf{H}\mathbf{q}_1)_1 \\ v_2 \, (\mathsf{H}\mathbf{q}_1)_3 &= (\mathsf{H}\mathbf{q}_1)_2 \quad . \end{aligned}$$

In full detail:

$$\begin{aligned} u_2 \, (H_{31}u_1 + H_{32}v_1 + H_{33}) &= H_{11}u_1 + H_{12}v_1 + H_{13} \\ v_2 \, (H_{31}u_1 + H_{32}v_1 + H_{33}) &= H_{21}u_1 + H_{22}v_1 + H_{23} \quad . \end{aligned}$$

These equations, for all point correspondences, can be gathered in a single linear equation system:

$$\left(\begin{array}{ccc|ccc|ccc} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u_2 & -v_1u_2 & -u_2 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v_2 & -v_1v_2 & -v_2 \\ \hline \vdots & & & \vdots & & & \vdots & & \\ \vdots & & & \vdots & & & \vdots & & \\ \vdots & & & \vdots & & & \vdots & & \end{array}\right) \left(\begin{array}{c} H_{11} \\ H_{12} \\ H_{13} \\ \hline H_{21} \\ H_{22} \\ H_{23} \\ \hline H_{31} \\ H_{32} \\ H_{33} \end{array}\right) = \mathbf{0} \quad .$$

As for the solution of this system, we refer to the remarks made in §2.1.

3.2 Application of the homography

The homography H effectively indicates how one would have to transform one of the images in order to make it superimpose the other one (within the part that is common to both images). At the same time, it gives us the means to introduce, in that other image, the parts that are only visible in the first image. Figure 5 shows an example of the mosaic generated from three of the images shown in figure 4.

In order to stitch more than two images together, one may "chain" pair-wise homographies together. For example, the homography between views 1 and 3 can be computed from the homographies between views 1 and 2, as well as between 2 and 3:

$$\mathsf{H}_{13} \sim \mathsf{H}_{23}\mathsf{H}_{12} \ .$$

Figure 4: Four images of the quay of Grenoble, taken from the same viewpoint.

Remark. Above, we only considered the geometric part of image stitching, however stitching also requires image processing or computer graphics techniques:

- stitching two images requires to transfer color/intensity information from pixels in one image to pixels in another. In practice, this involves an interpolation stage.
- when acquiring images with a rotating camera, one will usually encounter changes in apparent illumination (for example, when turning progressively towards the sun, images will progressively become underexposed). In order to generate high-quality mosaics, it is thus not sufficient to stitch images together, but a photometric correction (color balancing, matting) is also necessary.

Remark. Generating a mosaic by stitching images on top of one reference image, is not the ideal approach, since this does not allow to handle a total field of view approaching $180°$. In practice, one thus usually generates a cylindrical image (more details are given during the lecture).

3.3 Self-calibration

Remember how the homography between two images depends on the intrinsic parameters and the orientations of these images:

$$\mathsf{H} \sim \mathsf{K}\mathsf{R}_2\mathsf{R}_1^\mathsf{T}\mathsf{K}^{-1} \ .$$

Until now, we applied the homography as such, without considering these intrinsic and orientation parameters separately. It would be interesting to study if one can, for instance, compute the "relative rotation" $\mathsf{R}_2\mathsf{R}_1^\mathsf{T}$ between two images, given the homography. This would

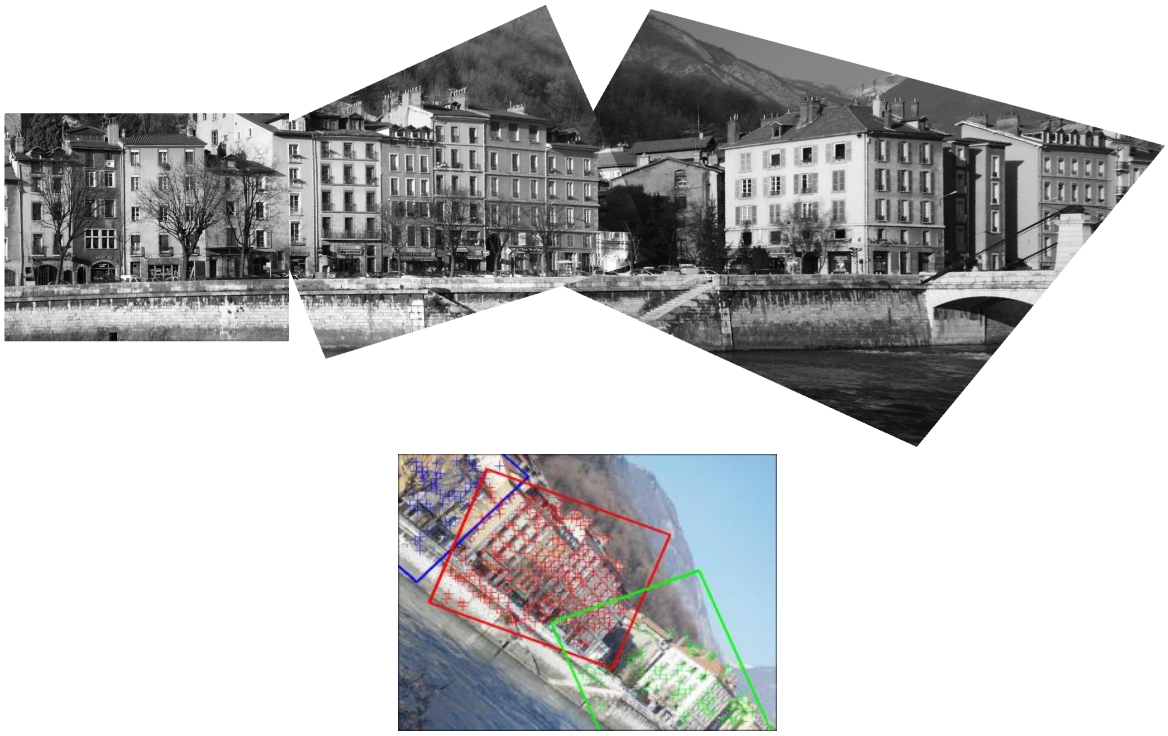


Figure 5: Mosaic generated from the first three images and an illustration of their relative situation shown on top of the fourth (wide angle) image.

for example allow to determine when a complete $360°$ rotation has been achieved, which in turn might help to refine (make more accurate) a generated mosaic.

If the camera is already calibrated (i.e. *off-line* calibrated), we can compute the relative rotation from the above equation and the knowledge of K: $\mathsf{R} \sim \mathsf{R}_2\mathsf{R}_1^\mathsf{T} \sim \mathsf{K}^{-1}\mathsf{H}\mathsf{K}$. In the opposite case, we may try to proceed like for camera calibration (section 2), where we decomposed a projection matrix into individual parameters, amongst which rotation parameters.

Let us transform the above equation:

$$\mathsf{K}^{-1}\mathsf{H}\mathsf{K} \sim \mathsf{R}_2\mathsf{R}_1^\mathsf{T} \ .$$

Upon multiplying each side of the equation with its own transpose, we get:

$$\mathsf{K}^{-1}\mathsf{H}\mathsf{K}\ \mathsf{K}^\mathsf{T}\mathsf{H}^\mathsf{T}\mathsf{K}^{-\mathsf{T}} \sim \mathsf{R}_2\mathsf{R}_1^\mathsf{T}\ \mathsf{R}_1\mathsf{R}_2^\mathsf{T} \ .$$

The right hand side can be simplified since $\mathsf{R}_1^\mathsf{T}\mathsf{R}_1 = \mathtt{I}$ and analogously for R_2. Hence:

$$\mathsf{K}^{-1}\mathsf{H}\mathsf{K}\ \mathsf{K}^\mathsf{T}\mathsf{H}^\mathsf{T}\mathsf{K}^{-\mathsf{T}} \sim \mathtt{I} \ .$$

It follows that:

$$\mathsf{H}\mathsf{K}\mathsf{K}^\mathsf{T}\mathsf{H}^\mathsf{T} \sim \mathsf{K}\mathsf{K}^\mathsf{T} \ .$$

Let us denote:

$$\mathsf{A} = \mathsf{K}\mathsf{K}^{\mathsf{T}} \ .$$

Then:

$$\mathsf{H}\mathsf{A}\mathsf{H}^{\mathsf{T}} \sim \mathsf{A} \ .$$

This equation is only defined up to a scalar factor. We can make it an actual element-wise equality by computing the appropriate scale factor for matrix H. Concretely, we may determine a scalar λ such that the determinants of both sides of the equations, are equal:

$$\det\left((\lambda\mathsf{H})\,\mathsf{A}\left(\lambda\mathsf{H}^{\mathsf{T}}\right)\right) = \det\mathsf{A} \ ,$$

or:

$$\lambda^6\left(\det\mathsf{H}\right)^2\det\mathsf{A} = \det\mathsf{A} \ .$$

The solution to this is unique and is given by:

$$\lambda = \sqrt[3]{1/\det\mathsf{H}} \ .$$

If we denote $\lambda\mathsf{H}$ by $\bar{\mathsf{H}}$, then we obtain the following *exact* equation on the unknown matrix A:

$$\bar{\mathsf{H}}\mathsf{A}\bar{\mathsf{H}}^{\mathsf{T}} = \mathsf{A} \ .$$

This equation is linear in the elements of A. It can be shown that a single equation of this form, i.e. a single homography, is not sufficient to completely compute A. However, with two or more homographies a unique solution is in general possible (in this case, the solution should be computed using for instance a linear least squares method).

Once A is computed, we can extract from it, the calibration matrix K, due to its upper triangular form, *via* a Cholesky decomposition, much like in §2.2.1. Now, the camera is calibrated and we can for example compute the relative rotations between images, as shown above.

Despite some common aspects between the calibration process (§2) and the method for computing K described in this section, we can pinpoint an essential difference: for calibration, an image of a *perfectly known* 3D object was used, whereas in this section, we did not use any information at all about the objet(s) visible in the images. Indeed, the camera is calibrated here without the usage of an actual calibration object. This is the simplest instance of the concept of *self-calibration* (or, auto-calibration). The information that allows to compute the camera's intrinsic parameters, is exactly the knowledge that the camera does not move in space but that it only rotates about its optical center. More general self-calibration problems have been considered in the literature, in particular the self-calibration of a camera that moves freely in space. The associated algorithms are usually more complex than the one outlined in this section.

An **off-line** calibration consists in taking one or more images of a calibration object, after which an actual application can be launched, e.g. 3D modeling. In this scheme, the image(s) of the calibration object are usually only used for calibration, whereas the subsequent images only serve the actual application. In this section however, we directly considered images

used for the target application (here, mosaic generation); still, these images proved useful for calibration too. One thus also talks about **on-line** calibration instead of self-calibration.

Remark. Here, we treated K as if it were totally unknown (besides its particular upper triangular form). In practice, it is often possible to use simplifying hypotheses, for example on the position of the principal point (u_0, v_0). Hence, fewer parameters have to be estimated and it is possible to use specialized algorithms that require fewer images. In general (if all intrinsic parameters are unknown), one needs at least 3 images (at least 2 homographies); in addition, the relative rotations between images must correspond to rotations about at least 2 different axes (otherwise, the obtained equations are redundant).

3.4 Bibliography

- R. Szeliski, *Image Alignment and Stitching: A Tutorial*, Foundations and Trends in Computer Graphics and Vision, Volume 2, Issue 1, 2006.

4 3D reconstruction from two completely calibrated images

We now consider two images of scene, taken from **different** viewpoints. Mosaic generation is then no longer possible in general; however we can now perform a 3D reconstruction of the scene.

The simplest 3D reconstruction scenario concerns the case where everything is known about the two images: the intrinsic parameters of both cameras as well as their position and orientation. In other words, we supposed here that we know the two cameras' projection matrices, P_1 and P_2.

The basic problem in this case is thus to compute the coordinates of a 3D point $\mathbf{Q}$, from its two images $\mathbf{q}_1$ and $\mathbf{q}_2$. We need to determine $\mathbf{Q}$ (a 4-vector of homogeneous coordinates) such that:

$$\begin{aligned} \mathsf{P}_1\mathbf{Q} &\sim \mathbf{q}_1 \\ \mathsf{P}_2\mathbf{Q} &\sim \mathbf{q}_2 \ . \end{aligned}$$

As explained in section 2.1, the data ($\mathbf{q}_1$ and $\mathbf{q}_2$) are usually noisy. Hence, the above equations do not in general have an exact solution and has to find an estimate for $\mathbf{Q}$ that is optimal according to some criterion. Many methods have been proposed in the literature; we briefly describe a few:

- the line of sight of each image point can be computed from the projection matrix of the associated image; in our case, we thus get access to two lines of sight. A possible solution for $\mathbf{Q}$ is the "mid-point" of these two lines, that is the point that is equidistant from them and minimizes the distance to these lines. This point is unique in general; it lies on the common perpendicular of the two lines.

- a similar though simpler solution consists of supposing that $\mathbf{Q}$ lies exactly on one of the two lines of sight. Among all points on that line of sight, one then determines the one that is closest to the second line of sight (the solution is again unique in general).

- from the above equations, one can deduce four linear equations in the coordinates of $\mathbf{Q}$:

$$\begin{aligned} u_1 \left(\mathsf{P}_1\mathbf{Q}\right)_3 &= \left(\mathsf{P}_1\mathbf{Q}\right)_1 \\ v_1 \left(\mathsf{P}_1\mathbf{Q}\right)_3 &= \left(\mathsf{P}_1\mathbf{Q}\right)_2 \\ u_2 \left(\mathsf{P}_2\mathbf{Q}\right)_3 &= \left(\mathsf{P}_2\mathbf{Q}\right)_1 \\ v_2 \left(\mathsf{P}_2\mathbf{Q}\right)_3 &= \left(\mathsf{P}_2\mathbf{Q}\right)_2 \ . \end{aligned}$$

  This system can be solved to least squares, in the same manner as for the computation of homographies and projection matrices in previous sections.

The third method can be directly extended to the case where more than two images of $\mathbf{Q}$ are available.

Remark. All above methods are sub-optimal (more details in the lecture) but usually give sufficiently good solutions. A more complex optimal method is given in: R. Hartley and P. Sturm, *Triangulation*, Computer Vision and Image Understanding, Vol. 68, No. 2, pp. 146-157, 1997.

5 Computing the pose of an object

Section 2 dealt with the calibration of a camera: given an image of an object whose structure is perfectly known, it is in general possible to determine the intrinsic parameters of the camera as well as its position and orientation relative to the object. It was also mentioned that in order to do so, at least 6 object point must be identified in the image.

An important sub-problem of this is **pose computation**: the same scenario as for calibration is considered, with the difference that the camera is already calibrated (its intrinsic parameters are known). The problem then reduces to the determination of the position and orientation – the **pose** – of the camera relative to the object (or vice-versa). A direct consequence of the reduction of the number of unknowns is that we no longer need 6 points; we shall see that with only 3 points, one already obtains a finite number of solutions and with 4 or more points, there is in general a unique solution for the pose. A second consequence is that pose estimation from a single image is possible when the object is **planar**, whereas calibration using a planar calibration object requires two or more images. In the following, we indeed focus on this special case. It is for example very relevant for augmented reality applications (examples will be given during the lecture).

We first examine the pose computation problem for the minimal case where the images of 3 object points are used and we then give a method that can take advantage of multiple object points.

5.1 First method, using 3 points

We consider that the camera is calibrated, i.e. we know its calibration matrix K. The images of 3 object point have been determined; they are given by:

$$\mathbf{q}_1 = \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \quad \mathbf{q}_2 = \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \quad \mathbf{q}_3 = \begin{pmatrix} u_3 \\ v_3 \\ 1 \end{pmatrix} \quad .$$

We also know the structure of the object. In this section, we do not use the representation of the structure *via* the object points' coordinates (like in §2 for calibration), but we simply use the knowledge of the pairwise distances of the 3 points: d_{12}, d_{13} and d_{23}.

The goal of this section is to compute the position and orientation of the camera relative to the 3 3D points. Equivalently, this is determined if the position of the 3D points relative to the camera is computed; from that information, one may, if needed, recover the translation and rotation that constitute the change of coordinates between the object's coordinate system and that of the camera.

One may formulate the problem as follows: we search for 3 3D points $\mathbf{Q}_1, \mathbf{Q}_2$ and $\mathbf{Q}_3$ (with coordinates relative to the camera system) such that the pairwise distances between them are equal to the known values d_{ij} and such that these 3D points get projected to the given image

points $\mathbf{q}_i$:

$$dist\left(\mathbf{Q}_i, \mathbf{Q}_j\right) = d_{ij} \qquad \forall i, j \in \{1,2,3\} \tag{16}$$

$$\mathsf{P}\mathbf{Q}_i \sim \mathbf{q}_i \qquad \forall i \in \{1,2,3\} \ . \tag{17}$$

Since we search these 3D points in the camera coordinate system, we have $\mathsf{R} = \mathtt{I}$ and $\mathbf{t} = \mathbf{0}$ in equation (8)), meaning that the projection matrix in our case is:

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{K} & \mathbf{0} \end{pmatrix} \ .$$

Now, we need to find a parameterization of the unknowns. We may suppose without any problem that the 3D points are finite (since we consider the image of a real object). Hence, their homogeneous coordinates can be written as:

$$\mathbf{Q}_i = \begin{pmatrix} \bar{\mathbf{Q}}_i \\ 1 \end{pmatrix} \quad \forall i \in \{1,2,3\}$$

with 3-vectors $\bar{\mathbf{Q}}_i$.

The projection of these points into the camera is thus given by:

$$\mathbf{q}_i \sim \mathsf{P}\mathbf{Q}_i \sim \mathsf{K}\bar{\mathbf{Q}}_i \quad \forall i \in \{1,2,3\} \ .$$

Since the calibration matrix K and the image points $\mathbf{q}_i$ are known, we have at our disposal some partial information on the 3D points:

$$\bar{\mathbf{Q}}_i \sim \mathsf{K}^{-1}\mathbf{q}_i \quad \forall i \in \{1,2,3\} \ .$$

The only missing information is due to the fact that these equations are defined up to scale only. We introduce the unknown scale factors explicitly and write the equations as exact ones:

$$\bar{\mathbf{Q}}_i = \lambda_i \mathsf{K}^{-1}\mathbf{q}_i \quad \forall i \in \{1,2,3\} \ .$$

The homogeneous coordinates of the 3D points are thus given by:

$$\mathbf{Q}_i = \begin{pmatrix} \lambda_i \mathsf{K}^{-1}\mathbf{q}_i \\ 1 \end{pmatrix} \quad \forall i \in \{1,2,3\} \ .$$

The only unknowns in our problem are the 3 scalars λ_1, λ_2 and λ_3. They encode the position of the 3D points along the lines of sight given by the corresponding image points.

So far we have reduced the problem given by equations (16) and (17): we have constructed $\mathbf{Q}_i$ that satisfy equation (17). The remaining problem is to determine λ_1, λ_2 and λ_3 such that:

$$dist\left(\mathbf{Q}_i, \mathbf{Q}_j\right) = d_{ij} \quad \forall i, j \in \{1,2,3\} \ .$$

To simplify notations, let us write:

$$\mathsf{K}^{-1}\mathbf{q}_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} \quad \forall i \in \{1,2,3\} \ ,$$

Then, the distance between two points $\mathbf{Q}_i$ and $\mathbf{Q}_j$ is written as:

$$dist\left(\mathbf{Q}_i, \mathbf{Q}_j\right) = \sqrt{\left(\lambda_i X_i - \lambda_j X_j\right)^2 + \left(\lambda_i Y_i - \lambda_j Y_j\right)^2 + \left(\lambda_i Z_i - \lambda_j Z_j\right)^2} \quad \forall i, j \in \{1, 2, 3\} \ .$$

We "get rid" of the square root by considering squared distances. Hence, we obtain equations of the form:

$$\left(\lambda_i X_i - \lambda_j X_j\right)^2 + \left(\lambda_i Y_i - \lambda_j Y_j\right)^2 + \left(\lambda_i Z_i - \lambda_j Z_j\right)^2 = d_{ij}^2 \quad \forall i, j \in \{1, 2, 3\} \ . \tag{18}$$

We now have 3 equations in the 3 unknowns λ_1, λ_2 and λ_3, which implies that in general there will be only a finite number of solutions. Concretely, the 3 equations are quadratic; thus, there are in general up to 2^3 solutions for the unknowns (some of the may correspond to complex conjugate numbers, which is not of interest for us). There exist many numerical methods for solving systems of quadratic equations (one of them will be briefly sketched in the lecture).

5.1.1 How to determine a unique solution

We thus obtain a maximum of 8 real solutions for the triplet of unknowns $(\lambda_1, \lambda_2, \lambda_3)$, thus 8 solutions for the pose. In practice, some of these solutions sometimes correspond to a case where one or several of the 3D points lie "behind" the camera (negative Z coordinate); these can be directly discarded.

In addition, note that for every solution, there exists a "mirror solution" which consists in changing the sign of each λ_i: one may observe that this does not change equations (18). For each such pair of solutions, one can thus reject at least one that corresponds to one 3D point lying behind the camera. Consequently, from the 8 "mathematical" solutions, at most 4 are physically plausible.

In order to find a unique solution, one needs in general more information. Especially, if we know the image $\mathbf{q}_4$ of a fourth object point, this is sufficient: for each one among the potential solutions for pose computed from the first 3 points, we can determine which would be the position of the 4th 3D point. If the considered solution for pose is correct, so will be the position of that 4th point and hence, the projection of it *via* P will coincide with $\mathbf{q}_4$. On the contrary, if the considered solution for pose is wrong, then the position of the 4th point will be wrong too and its projection will be different from $\mathbf{q}_4$. This reasoning (it is possible to obtain actual proofs) allows in general to eliminate all wrong solutions and to identify the single correct solution for the pose problem.

5.2 Second method, using multiple points

In the previous section we have seen that with more than 3 points, one can determine a unique solution for the pose problem. However, the described method has one drawback: the quality of the solution depends exactly on the considered 3 points (others are only used for choosing one among the potential multiple solutions). Hence, if image points are extracted with only

bad precision, the computed pose may be of bad quality. In case we have more than 3 points available, we thus would like to compute a pose that in some sence corresponds to the "average" across all points. One method to do so, is described in the following.

Compared to the first method, we now adopt as unknowns, the camera's rotation matrix R and position vector $\mathbf{t}$. Also, we do not anymore use the distances between points as representation of the object's structure, but directly the coordinates of the points, expressed in some coordinate system attached to the object. We still consider a planar object. Thus, we may choose a coordinate system such that the points' Z coordinates are all equal to 0. Hence, the 3D coordinates of our n points are given as:

$$\mathbf{Q}_i = \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix} \quad \forall i \in \{1 \dots n\} \ .$$

The camera's projection matrix, expressed relative to the object's coordinate system, can be written (cf. (8)):

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{KR} & -\mathsf{KRt} \end{pmatrix} = \mathsf{KR} \begin{pmatrix} \mathtt{I} & -\mathbf{t} \end{pmatrix} \ .$$

The projection equations are given as:

$$\mathbf{q}_i \sim \mathsf{KR} \begin{pmatrix} \mathtt{I} & -\mathbf{t} \end{pmatrix} \mathbf{Q}_i \quad \forall i \in \{1 \dots n\} \ .$$

Remind that all entities in these equations are known, with the exception of R and $\mathbf{t}$.

In the following, we proceed much like during camera calibration, by first computing intermediate variables (there, the projection matrix) and then extracting from these, the actual unknowns one is interested in (there, intrinsic and extrinsic parameters). Let us develop the projection equation associated with a 3D point $\mathbf{Q}_i$:

$$\begin{aligned} \mathbf{q}_i &\sim \mathsf{KR} \begin{pmatrix} \mathtt{I} & -\mathbf{t} \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix} \\ &= \mathsf{KR} \begin{pmatrix} 1 & 0 & 0 & -t_1 \\ 0 & 1 & 0 & -t_2 \\ 0 & 0 & 1 & -t_3 \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix} \\ &= \mathsf{KR} \begin{pmatrix} 1 & 0 & -t_1 \\ 0 & 1 & -t_2 \\ 0 & 0 & -t_3 \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} \ . \end{aligned}$$

Whereas in general, the 3D-to-2D projection equation is not invertible, we now have a projective transformation between points on the object and corresponding image points. This

transformation is in general invertible; it is represented by the 3×3 matrix:

$$\mathsf{H} \sim \mathsf{K}\mathsf{R} \begin{pmatrix} 1 & 0 & -t_1 \\ 0 & 1 & -t_2 \\ 0 & 0 & -t_3 \end{pmatrix} \quad , \tag{19}$$

such that:

$$\mathbf{q}_i \sim \mathsf{H} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} \quad \forall i \in \{1 \dots n\} \ .$$

Note that this invertible transformation is due to the fact that the object points are all coplanar; if one considers a set of non-coplanar 3D points, the projection is no longer invertible in the same manner.

The method for computing the pose consists of two steps: first, we estimate the projective transformation H between the planar object and the image plane. Then, we extract, from H, the pose parameters: rotation matrix R and vector $\mathbf{t}$.

As for the first step, we refer to the method explained in paragraph 3.1. This method can be directly applied here – the only difference is the "meaning" of the considered 2D points: in §3.1, the computed transformation happens between two image planes (within a mosaic generation) whereas here, one of the planes is contained in the 3D scene.

Once H is computed, we examine how to extract first R, then $\mathbf{t}$, starting from equation (19). We first determine the first two columns of R, then its third column, then $\mathbf{t}$.

5.2.1 Computation of the first two columns of R

Since the calibration matrix K is known, we can compute the matrix $\mathsf{M} = \mathsf{K}^{-1}\mathsf{H}$, for which we have:

$$\mathsf{M} \sim \mathsf{R} \begin{pmatrix} 1 & 0 & -t_1 \\ 0 & 1 & -t_2 \\ 0 & 0 & -t_3 \end{pmatrix} \quad . \tag{20}$$

Let us denote the column vectors of matrix R by 3-vectors $\mathbf{r}_1, \mathbf{r}_2$ and $\mathbf{r}_3$ such that:

$$\mathsf{R} = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{pmatrix} \ .$$

Equation (20) is then rewritten as follows, after explicitly introducing a scale factor λ (the equality now holds element-wise and not only up to scale):

$$\lambda\mathsf{M} = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & -\mathsf{R}\mathbf{t} \end{pmatrix} \ . \tag{21}$$

First, we will determine the first two columns of R. Before this, let us remind that all column vectors of a rotation matrix are of norm 1 (cf. §1.2.3). This helps in the determination of the scalar λ. The following two relations must be satisfied (one each for the first two columns of M):

$$\begin{aligned} \lambda^2 \left(M_{11}^2 + M_{21}^2 + M_{31}^2 \right) &= 1 \\ \lambda^2 \left(M_{12}^2 + M_{22}^2 + M_{32}^2 \right) &= 1 \ . \end{aligned}$$

We observe two things:

- there are two solutions for λ, which differe by their sign: $\pm\lambda$;
- because of noise, for instance in the positions of image points used to compute H then M, there is in general no exact solution to the two above equations. We may thus compute the solutions for each equation separately, and adopt the average as the final value of λ (one positive and one negative value, as explained just before).

In the following, we denote by $\mathbf{m}_1, \mathbf{m}_2$ and $\mathbf{m}_3$ the three columns of $\lambda\mathsf{M}$ (let us note that the following operations have also to be done for $-\lambda$). Theoretically, we could directly determine the first two columns of R as: $\mathbf{r}_1 = \mathbf{m}_1$ et $\mathbf{r}_2 = \mathbf{m}_2$ (according to equation (21)). Once again however, noise in the data will occur in practice. In particular, one condition on the columns of a rotation matrix R is that they are mutually orthogonal (cf. §1.2.3), i.e. that their dot product is zero. The vectors $\mathbf{m}_1$ and $\mathbf{m}_2$ computed as above will in the presence of noise not satisfy this condition exactly.

One possible method to find "admissible" vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, is sketched in the following. Let $\mathbf{v}$ be the bi-sector of the 3-vectors $\mathbf{m}_1$ and $\mathbf{m}_2$. The three vectors $\mathbf{v}, \mathbf{m}_1$ and $\mathbf{m}_2$ lie in the same plane; let this plane be Π. In Π, there exist two directions that form an angle of $45°$ with $\mathbf{v}$. For each of these two, we can compute the associated vector that is of length (norm) equal to 1 and that is "close" to either $\mathbf{m}_1$ or $\mathbf{m}_2$. The two selected vectors will satsify all required constraints: they are mutually orthogonal and are both of norm 1. We can thus adopt them as our estimate of $\mathbf{r}_1$ and $\mathbf{r}_2$.

5.2.2 Computation of the third column of R

Let us remind again that the columns of R are mutually orthogonal and of norm 1. Given $\mathbf{r}_1$ and $\mathbf{r}_2$, there exist exactly two vectors $\mathbf{r}_3$ that satisfy these conditions. They are pointing in opposite directions (one can be obtained from the other by multiplying it with -1).

Remark. A vector $\mathbf{r}_3$ that is orthogonal to two vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ can be computed, up to scale, by the cross-product: $\mathbf{r}_3 \sim \mathbf{r}_1 \times \mathbf{r}_2$.

Up to now, we have two potential solutions for the rotation matrix:

$$\mathsf{R} = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \pm\mathbf{r}_3 \end{pmatrix} \quad .$$

It can be observed that the determinants of the two solution matrices, are of opposite signs (changing the sign of all elements in one column of a matrix, causes the sign of its determinant to switch). Since the determinant of a rotation matrix must be positive (equal to $+1$, cf. §1.2.3), we can eliminate one of the two solutions for $\mathbf{r}_3$ and thus for R.

5.2.3 Computation of $\mathbf{t}$

The computation of $\mathbf{t}$ is rather simple. Let $\mathbf{m}_3$ be the third column of $\lambda\mathsf{M}$. According to equation (21), we have:

$$\mathbf{m}_3 = -\mathsf{R}\mathbf{t} \quad .$$

Hence:

$$\mathbf{t} = -\mathsf{R}^\mathsf{T}\mathbf{m}_3 \ .$$

This finalizes the computation of the pose – up to one last detail, see next.

5.2.4 Remarks

Remember that during the computation of the first two columns of R, two solutions for λ were possible (with opposite sign). With the above steps, we thus actually obtain two solutions for rotation and position.

It can be shown that these two solutions correspond two camera positions on opposite sides of the planar object: one is actually the reflection of the other, in the object's support plane. Without additional information, it is not possible to tell which solution is the correct one.

In order to obtain a unique solution (thus, the correct one), one may use the constraint that often, only a single side of the object is visible. In this case, we may know in advance in which half-space, relative to the object, the camera must be situated. Since the object points have, with our choice of coordinate system, Z coordinates equal to 0, we may for instance define the system such that the admissible half-space for the camera corresponds to positive Z coordinates. Among the two solutions for the pose, the one with positive third coordinate of $\mathbf{t}$ must then be the correct one.

5.3 Bibliography

- R.J. Holt and A.N. Netravali, *Camera Calibration Problem: Some New Results,* CVGIP - Computer Vision, Graphics and Image Processing, Vol. 54, No. 3, pp. 368-383, 1991.

- R.M. Haralick, C. Lee, K. Ottenberg and M. Nölle, *Analysis and Solutions of the Three Point Perspective Pose Estimation Problem,* IEEE International Conference on Computer Vision and Pattern Recognition, pp. 592-598, 1991.

- D. Dementhon and L.S. Davis, *Model-Based Object Pose in 25 Lines of Code,* International Journal on Computer Vision, Vol. 15, No. 1/2, pp. 123-141, 1995.

- P. Sturm, *Algorithms for Plane-Based Pose Estimation,* IEEE International Conference on Computer Vision and Pattern Recognition, pp. 706-711, 2000.

6 Geometric relations between two images taken from different viewpoints – Epipolar geometry

6.1 Introduction

Remember the mosaic generation scenario (cf. §3) and an observation made there: two images taken from the *same* viewpoint, are linked by a projective transformation (or, homography). Given the image point of a scene point in one image, this transformation allows to determine where in the other image the same scene point's image must be. As soon as the images are taken from *different* viewpoints, there is in general no longer such a bijective transformation between images. Hence, given a point in one image, one can no longer determine exactly where in the other image the same scene points will be seen.

However, we will see that it is possible to compute a line in the other image, on which the corresponding image point must lie. Hence, geometric relations/links do exist between two images taken from different viewpoints, but they are weaker than in the case of images taken from the same viewpoint.

6.2 Basic case: looking for the correspondence of an image point

Consider two images of a scene, taken from different viewpoints. The key question in this section is: given a point $\mathbf{q}_1$ in the first image, what can one tell about the position of the corresponding point $\mathbf{q}_2$ in the second image?

Let us first examine what one can tell about the position of the original 3D point $\mathbf{Q}$ that is seen in $\mathbf{q}_1$. We suppose here that the camera is calibrated, hence that we can trace the line of sight (the line spanned by the optical center and the point $\mathbf{q}_1$ on the image plane). Obviously, we can ascertain that $\mathbf{Q}$ must be somewhere along this line.

Let us now project this line of sight into the second image; this gives rise to a line $\mathbf{l}_2$ in the second image plane. The point $\mathbf{q}_2$ must lie on this line (see also figure 6). There is thus a geometric relationship between the two images (the point $\mathbf{q}_2$ cannot be anywhere in the second image, but must lie on $\mathbf{l}_2$), but it is not "bijective": all points on $\mathbf{l}_2$ are potential correspondences of $\mathbf{q}_1$.

The fact that the corresponding point lies on a line is usually called "epipolar constraint" (the expression will become clearer below). This constraint is very useful for image matching: the search for corresponding points can be considerably accelerated (one does not need to search all over the image but only along a single line) and the probability of accidentally finding a wrong correspondence, is strongly reduced.

In the following, we consider the epipolar constraint from a more general standpoint.

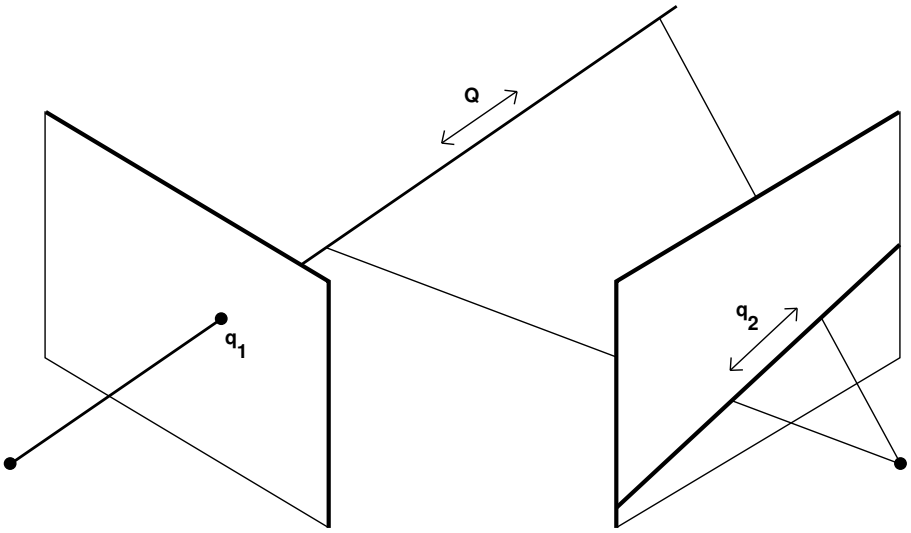


Figure 6: Epipolar geometry: the 3D point lies on the line of sight and the corresponding point $\mathbf{q}_2$ on the image of that line.

6.3 Epipolar geometry

Before proceeding further, we introduce some notations (cf. figures 6 and 7). The optical centers and the line of sight considered in the previous paragraph, define a plane, a so-called **epipolar plane**. This plane cuts the two image planes in two lines, the **epipolar lines** – $\mathbf{l}_1$ in the first image plane and $\mathbf{l}_2$ in the second one. The point in the second image that corresponds to $\mathbf{q}_1$ must lie on the the *associated epipolar line* $\mathbf{l}_2$.

The configuration is perfectly symmetrical: the point in the first image that corresponds to some point $\mathbf{q}_2$ in the second image on line $\mathbf{l}_2$, must lie on the associated epipolar line, $\mathbf{l}_1$.

Consider now a different point $\mathbf{q}'_1$ on $\mathbf{l}_1$ and the task of finding its correspondence in the second image. We may observe that the epipolar plane as well as the two epipolar lines, are the same as those associated with $\mathbf{q}_1$. We can thus conclude that all pairs of points on $\mathbf{l}_1$ and $\mathbf{l}_2$, are potential correspondences.

We now consider a point $\mathbf{q}'_1$ that does *not* lie on $\mathbf{l}_1$. It is clear that this now gives rise to a different epipolar plane. According to the construction of epipolar planes (defined by the two optical centers and a line of sight), it is also clear that this second epipolar plane contains the line spanned by the two optical centers.

Upon repeating this construction for other image points, we can "produce" the pencil of all epipolar planes, which consists of all planes containing both optical centers. In addition, to the pencil of epipolar planes, correspond naturally the two pencils of epipolar lines in the two image planes. The basis of the epipolar plane pencil is the so-called **baseline** – the line spanned by the optical centers. As for the epipolar line pencils, their bases are two points, the so-called **epipoles**.

When considering figure 8, one can give another definition of the epipoles: the epipole of the first image – $\mathbf{e}_1$ – is the image of the second camera's optical center. Vice-versa, $\mathbf{e}_2$ is the

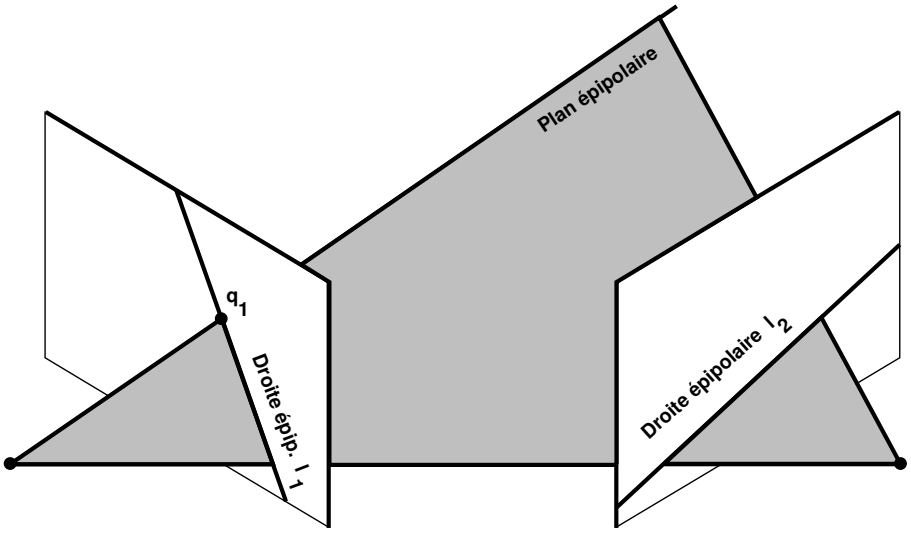


Figure 7: The optical centers and the line of sight define an epipolar plane. This plane cuts the image planes in epipolar lines.

image, in the second camera, of the first camera's optical center.

Remark. Intuitively, knowing the epipoles is useful for estimating the relative position of the two cameras. For example, the first epipole tells us in which direction, relative to the first camera, lies the optical center of the second camera.

Let us examine the two pencils of epipolar lines more closely. We may interpret them as two one-dimensional spaces whose elements are the lines passing through one epipole respectively. There exists a *bijective* relationship between these two spaces: a line $\mathbf{l}_1$ in the first pencil corresponds to exactly one epipolar plane and consequently, to exactly one epipolar line $\mathbf{l}_2$ in the second image. This relationship (actually, a one-dimensional projective transformation) is called **epipolar transformation**.

Epipolar geometry is determined or defined by the position of the two epipoles and by the epipolar transformation. Knowing these allows to utilize the epipolar constraint in order to, as explained above, restrict the search space for the point corresponding to a point $\mathbf{q}_1$: knowing $\mathbf{e}_1$ enables the computation of the epipolar line $\mathbf{l}_1$ passing through $\mathbf{q}_1$. The epipolar transformation in association with the position of $\mathbf{e}_2$ then allow to infer the corresponding epipolar line $\mathbf{l}_2$.

In the following section, we derive the algebraic description of epipolar geometry.

6.4 Algebraic representation of epipolar geometry – The fundamental matrix

Let P_1 and P_2 be the projection matrices of the two cameras:

$$\begin{aligned} \mathsf{P}_1 &\sim \mathsf{K}_1\mathsf{R}_1\begin{pmatrix}\mathtt{I} & -\mathbf{t}_1\end{pmatrix} \\ \mathsf{P}_2 &\sim \mathsf{K}_2\mathsf{R}_2\begin{pmatrix}\mathtt{I} & -\mathbf{t}_2\end{pmatrix} \end{aligned}$$

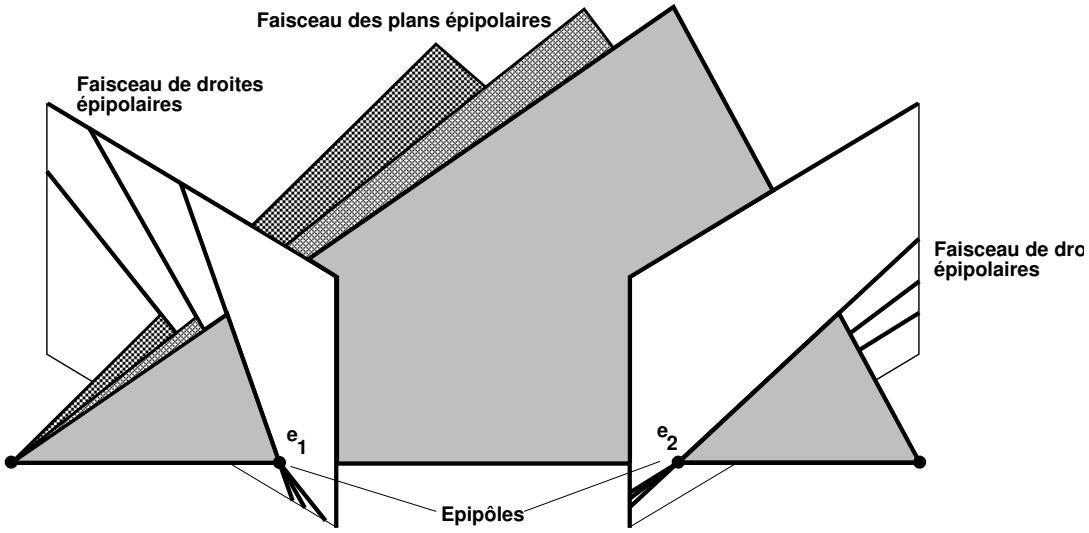


Figure 8: All epipolar planes contain the two optical centers. Hence, they form a pencil of planes. Their intersections with the image planes form two pencils of epipolar lines. The epipoles are the basis points of these pencils.

Given a point $\mathbf{q}_1$ in the first image, we wish to compute the associated epipolar line $\mathbf{l}_2$ in the second image. We proceed as follows. First, we compute the line of sight of $\mathbf{q}_1$; it may be represented by two 3D points. We then project these two points onto the second image. The line $\mathbf{l}_2$ can then be computed from the cross-product of the coordinate vectors of the two 2D points.

The natural choice for the first point on the line of sight, is the optical center of the first camera (since it lies on all lines of sight of the first image):

$$\begin{pmatrix} \mathbf{t}_1 \\ 1 \end{pmatrix} \quad .$$

As for the second point, one that gives rise to simple formulas, is the point at infinity of the line of sight:

$$\begin{pmatrix} \mathsf{R}_1^{\mathsf{T}} \mathsf{K}_1^{-1} \mathbf{q}_1 \\ 0 \end{pmatrix}$$

(it is easy to verify that this point at infinity gets projected by P_1 onto $\mathbf{q}_1$).

These two points define the line of sight. We project them onto the second image. Note that the projection of the first camera's optical center gives the epipole $\mathbf{e}_2$, as explained earlier:

$$\begin{aligned} \mathbf{a} &\sim \mathsf{P}_2 \begin{pmatrix} \mathbf{t}_1 \\ 1 \end{pmatrix} \sim \mathsf{K}_2 \mathsf{R}_2 \left(\mathbf{t}_1 - \mathbf{t}_2 \right) \\ \mathbf{b} &\sim \mathsf{P}_2 \begin{pmatrix} \mathsf{R}_1^{\mathsf{T}} \mathsf{K}_1^{-1} \mathbf{q}_1 \\ 0 \end{pmatrix} \sim \mathsf{K}_2 \mathsf{R}_2 \mathsf{R}_1^{\mathsf{T}} \mathsf{K}_1^{-1} \mathbf{q}_1 \quad . \end{aligned}$$

The epipolar line can now be computed by the cross-product $\mathbf{a} \times \mathbf{b}$. By using the rule

$(\mathsf{M}\mathbf{x}) \times (\mathsf{M}\mathbf{y}) \sim \mathsf{M}^{-\mathsf{T}} (\mathbf{x} \times \mathbf{y})$, we get:

$$\begin{aligned} \mathbf{l}_2 &\sim \mathbf{a} \times \mathbf{b} \\ &\sim \left\{\mathsf{K}_2\mathsf{R}_2 (\mathbf{t}_1 - \mathbf{t}_2)\right\} \times \left\{\mathsf{K}_2\mathsf{R}_2\mathsf{R}_1^{\mathsf{T}}\mathsf{K}_1^{-1}\mathbf{q}_1\right\} \\ &\sim \left(\mathsf{K}_2\mathsf{R}_2\right)^{-\mathsf{T}} \left\{(\mathbf{t}_1 - \mathbf{t}_2) \times \left(\mathsf{R}_1^{\mathsf{T}}\mathsf{K}_1^{-1}\mathbf{q}_1\right)\right\} \quad . \end{aligned}$$

Finally, using the notation $[\cdot]_\times$ (cf. §0.2), we can write:

$$\begin{aligned} \mathbf{l}_2 &\sim \left(\mathsf{K}_2\mathsf{R}_2\right)^{-\mathsf{T}} [\mathbf{t}_1 - \mathbf{t}_2]_\times \left(\mathsf{R}_1^{\mathsf{T}}\mathsf{K}_1^{-1}\mathbf{q}_1\right) \\ &\sim \left(\mathsf{K}_2\mathsf{R}_2\right)^{-\mathsf{T}} [\mathbf{t}_1 - \mathbf{t}_2]_\times \left(\mathsf{R}_1^{\mathsf{T}}\mathsf{K}_1^{-1}\right) \mathbf{q}_1 \quad . \end{aligned} \tag{22}$$

Equation (22) represents the transformation that maps a point $\mathbf{q}_1$ of the first image, to the associated epipolar line $\mathbf{l}_2$ in the second image. The matrix representing this transformation is the so-called **fundamental matrix** F_{12}:

$$\mathsf{F}_{12} \sim \left(\mathsf{K}_2\mathsf{R}_2\right)^{-\mathsf{T}} [\mathbf{t}_1 - \mathbf{t}_2]_\times \left(\mathsf{R}_1^{\mathsf{T}}\mathsf{K}_1^{-1}\right) \quad . \tag{23}$$

It encodes entirely the epipolar geometry: the epipolar transformation and the two epipoles can be extracted from this matrix.

The fundamental matrix is "oriented" – it gives epipolar lines in the second image from points in the first image. How about the opposite direction? To obtain the fundamental matrix for the opposite direction, it suffices to swap the indices 1 and 2 in equation (23). Thus, the fundamental matrix mapping points of the second image to epipolar lines in the first image, is:

$$\mathsf{F}_{21} \sim \left(\mathsf{K}_1\mathsf{R}_1\right)^{-\mathsf{T}} [\mathbf{t}_2 - \mathbf{t}_1]_\times \left(\mathsf{R}_2^{\mathsf{T}}\mathsf{K}_2^{-1}\right) \quad .$$

We may observe[5] that F_{21} is nothing else than the transpose of F_{12} (possibly up to a multiplicative scale factor):

$$\mathsf{F}_{21} \sim \mathsf{F}_{12}^{\mathsf{T}} \quad .$$

In the following, we will simply write F, assuming that the direction is clear from the context.

We can now express the epipolar constraint introduced earlier in §6.2. The point $\mathbf{q}_2$ corresponding to $\mathbf{q}_1$ must lie on the epipolar line $\mathbf{l}_2$, hence the scalar product of $\mathbf{q}_2$ and $\mathbf{l}_2$ must vanish, i.e. $\mathbf{q}_2^{\mathsf{T}}\mathbf{l}_2 = 0$. By replacing$\mathbf{l}_2$ with the above findings, we obtain the standard formula for the **epipolar constraint**:

$$\mathbf{q}_2^{\mathsf{T}}\mathsf{F}\mathbf{q}_1 = 0 \quad . \tag{24}$$

Remark. Knowing the fundamental matrix, i.e. the epipolar geometry, of two images, we may thus simplify image matching by using the epipolar constraint to narrow the search space for corresponding points. Reciprocally, given point correspondences, equation (24) provides constraints that allow to estimate the fundamental matrix. In section 7.2 it will be shown that this may then allow to estimate camera trajectories.

[5] Using the fact that for every $\mathbf{x}$: $[\mathbf{x}]_\times \sim \left([\mathbf{x}]_\times\right)^{\mathsf{T}}$.

6.5 Some details on the fundamental matrix

The fundamental matrix represents a transformation that maps points to lines; in projective geometry this is sometimes called a *correlation*.

The main algebraic property of the fundamental matrix is that it is singular (rank 2). Two ways of seeing this are as follows. First, the matrix $[\mathbf{t}_1 - \mathbf{t}_2]_\times$ in equation (23) is singular (any skew-symmetric matrix is singular); consequently, the fundamental matrix must be singular too (the rank of a product of square matrices is less or equal the smallest rank of each individual matrix). Second, let us consider figure 8. The fundamental matrix is a function whose domain and target are two-dimensional spaces – the space of points of the first image and the space of lines of the second image. However, only the lines in the epipolar pencil – a one-dimensional projective space – are reached by the transformation. In addition, the transformation is not bijective, meaning finally that the fundamental matrix must be singular.

A singular matrix has a non-empty null-space (or, kernel), i.e. there exist non-zero vectors for which the matrix-vector product is zero. The kernel of the fundamental matrix represents nothing else than the epipole $\mathbf{e}_1$, i.e. we have:

$$\mathsf{F}\mathbf{e}_1 = \mathbf{0} \ .$$

This can be proven analytically, but there is also an intuitive explication. The epipolar line in the first image that is associated with a point $\mathbf{q}_1$, is the line spanned by that point and the epipole $\mathbf{e}_1$. Hence, the epipolar line associated with the epipole itself, is not defined. Algebraically, this must mean that $\mathsf{F}\mathbf{e}_1 = \mathbf{0}$. A vector of zeroes is not admitted as vector of homogeneous coordinates; obtaining this as result of a projective transformation nicely expresses that the result of a transformation is not defined, as it is the case here.

As for the second epipole, the analogous finding is:

$$\mathsf{F}^\mathsf{T}\mathbf{e}_2 = \mathbf{0} \ .$$

6.6 Epipolar geometry for calibrated images – Essential matrix

The fundamental matrix depends on the pose of the two cameras as well as on their intrinsic parameters, i.e. their calibration matrices K_1 and K_2. In this section, we suppose that the cameras are calibrated, i.e. that we know K_1 and K_2. Hence, we can compute, from the fundamental matrix, the so-called **essential matrix** E (cf. equation (23)):

$$\mathsf{E} \sim \mathsf{K}_2^\mathsf{T}\mathsf{F}\mathsf{K}_1 \sim \mathsf{R}_2 \left[\mathbf{t}_1 - \mathbf{t}_2\right]_\times \mathsf{R}_1^\mathsf{T} \ . \tag{25}$$

What does the essential matrix represent? It represents the epipolar geometry of two images if the coordinate systems used for 2D image points, are the *image systems*[6], instead of the *pixel systems* as for the fundamental matrix. One often says that the essential matrix represents the **calibrated epipolar geometry**, since it is the calibration information K_1 and K_2 that allows to go from the pixel to the image coordinate systems.

[6] Cf. §1.2. To be precise, the required coordinate systems are the image systems, after applying a change of unit that makes one unit equal the focal length.

One may observe (cf. equation (25)) that the essential matrix only depends on the position and orientation of the cameras. This will be used in §7.2 for the estimation of a camera's motion from two images, respectively for the estimation of the relative pose between two different cameras.

Remark. Previously, we observed that fundamental matrices are singular. This also holds true for the essential matrix. In addition, any essential matrix has the property that its two non-zero singular values (related to the singular value decomposition and the eigenvalues of a matrix) are identical.

6.7 Estimating the epipolar geometry – Basic method

In §6.4, it was described how to compute the fundamental matrix F from the intrinsic and extrinsic parameters of the two cameras. One of the usages of F is to constraint the search of correspondences between images. Reciprocally, knowledge of sufficiently many correspondences allows to estimate the fundamental matrix, even if the cameras are not calibrated and if nothing is known about their pose, which is often the case in practical applications of computer vision.

In this section, we describe the most basic method for estimating the fundamental matrix from point correspondences. In the next section, an extension is presented that is designed to handle correspondences that are not reliable – in practice one always will have this case.

The basic method uses equation (24):

$$\mathbf{q}_2^\mathsf{T}\,\mathsf{F}\,\mathbf{q}_1 = 0 \ .$$

By making the coefficients of F explicit, this is written as:

$$\begin{aligned} & F_{11}q_{1,1}q_{2,1} + F_{12}q_{1,2}q_{2,1} + F_{13}q_{1,3}q_{2,1} \\ +& F_{21}q_{1,1}q_{2,2} + F_{22}q_{1,2}q_{2,2} + F_{23}q_{1,3}q_{2,2} \\ +& F_{31}q_{1,1}q_{2,3} + F_{32}q_{1,2}q_{2,3} + F_{33}q_{1,3}q_{2,3} \quad = \quad 0 \ . \end{aligned}$$

This equation is linear in the coefficients of F. The equations of this type, for n point correspondences, can be gathered in a single equation system:

$$\mathsf{A}\mathbf{f} = \mathbf{0} \ . \tag{26}$$

where $\mathbf{f}$ is a vector containing the coefficients of F (the unknowns):

$$\mathbf{f} = (F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33})^\mathsf{T} \ ,$$

and the matrix A, of size $n \times 9$ has the following form:

$$\mathsf{A} = \begin{pmatrix} q_{1,1}q_{2,1} & q_{1,2}q_{2,1} & q_{1,3}q_{2,1} & q_{1,1}q_{2,2} & q_{1,2}q_{2,2} & q_{1,3}q_{2,2} & q_{1,1}q_{2,3} & q_{1,2}q_{2,3} & q_{1,3}q_{2,3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}_{n\times 9} \ .$$

Due to noise in the data (the point correspondences), there is in general no exact solution to equation (26). One thus computes a linear least squares solution $\hat{\mathbf{f}}$ such that:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \|\mathsf{A}\mathbf{f}\|^2 \quad \text{subject to} \quad \|\mathbf{f}\| = 1 \ .$$

As for the solution of this problem, we refer to the remarks made in §2.1.

Remark. The notation $\|\mathbf{v}\|$ refers to the norm of vector $\mathbf{v}$, i.e. the square root of the sum of its squared coefficients. To be precise, this is the L_2 norm. There are many other norms but in this lecture we always use L_2.

6.7.1 A little problem...

The above method neglects a constraint on the structure of fundamental matrices: in section 6.5, it was explained that any "valid" fundamental matrix (i.e. any 3×3 matrix that ought to represent the epipolar geometry of two images), must be singular. This constraint however is non-linear (it is equivalent to saying that the determinant is zero) and cannot be enforced by the above method, which is based on solving a linear equation system.

There is a solution to this problem: given an estimate of F that is not singular, one may estimate the matrix $\hat{\mathsf{F}}'$ that is perfectly singular and that is "closest" to F, according to some matrix norm. If we use the so-called Frobenius norm (the square root of the sum of squared coefficients of a matrix), this can be formulated as:

$$\hat{\mathsf{F}}' = \arg\min_{\mathsf{F}'} \sum_{i,j=1}^{3} \left(F'_{ij} - F_{ij}\right)^2 \quad \text{subject to} \quad \mathsf{F}' \text{ being singular} \ .$$

Remark. This problem can be solved using the singular value decomposition (SVD) of F'. More generally, suppose here that we wish to compute, given an arbitrary matrix A, the matrix B that is of a desired rank r and that is closest to A, in the sense of the Frobenius norm. To do so, let us consider the singular value decomposition of A: $\mathsf{A} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$, where the elements of the diagonal matrix Σ are ordered: $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$. Then, B is given by: $\mathsf{B} = \mathsf{U}\Sigma'\mathsf{V}^\mathsf{T}$, with Σ' the diagonal matrix defined by:

$$\sigma'_i = \begin{cases} \sigma_i & \text{if } i \leq r \\ 0 & \text{if } i > r \end{cases}$$

In our case, we have $r = 2$ (fundamental matrices are of rank 2).

Remark. The Frobenius norm used above is defined as:

$$\|\mathsf{A}\|_F = \sqrt{\sum_{i,j} A_{ij}^2} \ .$$

It is the analogous for matrices to what the L_2 is for vectors.

6.7.2 How many correspondences are needed?

Equation (26) is a *linear* and *homogeneous* equation system. In general, $m - 1$ linearly independent equations are needed to solve a linear homogenous system in m unknowns.

When fewer equations are available, the system is under-constrained[7]. If more equations than required are available, there is in general, due to noise in the data, no exact solution and one usually computes a linear least squares solution.

In our case, we have 9 unknowns to estimate, hence a minimum of 8 point correspondences is required to solve the linear equation system considered above. Due to this, the above basic algorithm is usually called the **8 point method**.

Remark. The constraint that the fundamental matrix must be singular, can actually be used to estimate it from only 7 point correspondences. With 7 linear homogeneous equations on 9 unknowns (which are defined up to scale, since they are coefficients of a matrix that is applied to vectors of homogeneous point coordinates) there exists a linear family of solutions that can be expressed, as mentioned above, as a linear combination of two basis matrices:

$$\mathsf{F} = \mathsf{F}_1 + \lambda F_2$$

F being singular means that its determinant is zero. The determinant of matrix F in the above equation, is a cubic polynomial in λ. Hence, instead of having infinitely many solutions for F, there are at most 3 (corresponding to the 3 roots of the cubic polynomial).

6.8 Robust estimation of the epipolar geometry

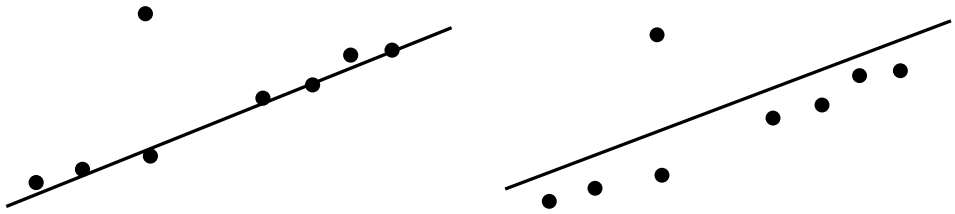
In this section, we encounter the principle of **robust estimation** which is truly essential to make many computer vision algorithms work in applications. In many applications, data are affected by noise. On the one hand this refers to an imprecise localization of interest points in images for example. On the other hand, and what is even more important, this refers to **outliers** in the data, e.g. point correspondences that are not only imprecise but plain wrong (for example, images of scenes containing repeated structures, suffer from that problem, but not only those kinds of images). Figure 9 illustrates this problem on the rather simple illustrative example of fitting a line to a set of 2D points. As it can be seen, a single outlier can have a "catastrophic" impact on the result. Intuitively, it should not be too hard to tackle this simple line fitting problem. However, in other estimation problems one may be confronted with many more data that live in many more dimensions and that contain many more outliers...

In the sequel, we explain the basic principles for one method of **robust estimation**, i.e. the estimation of parameters while identifying outliers and discarding or otherwise handling them. It is based on the use of so-called robust statistics, used as measure of the goodness-of-fit of estimated parameters. Two such measures, for the case illustrated in figure 9, i.e. for the goodness-of-fit of a line, are:

- the number of data points that are closer to the line than a given threshold distance.
- the median of the distances between points and the line.

The first measure is easier to compute but the choice of the threshold may be delicate. As for the second measure, it does not require an explicit threshold. However, if more than 50

[7]Thus, there exists a linear subspace of solutions to the system; in other words, all possible solutions can be expressed as linear combinations of some basis vectors.



(a) The result one would like to obtain: the outlying point (which may for example be caused by a measurement error) is correctly identified as such.

(b) This is the result if outliers are not identified and handled: the fitted line is deviated towards the outlier.

Figure 9: Illustration of the impact of outliers in the data. The goal here is to fit a line to a set of points.

% of the data are outliers, one should use another quantile instead of the median (the 50 % quantile). Hence, an assumption on the expected number of outliers must be made, which is somewhat equivalent to choosing a threshold as for the first measure. The first measure can work correctly (see below) with more than 50 % outliers in the data (meaning that the line that maximizes the measure, is likely to be close to the true line).

Remark. The opposite of outliers, are **inliers**.

Let us return to our application, the estimation of epipolar geometry. Here, outliers are incorrect point correspondences. If such correspondences are present, the result of the estimation method of §6.7 will in general be totally useless.

Let us consider now how to measure the quality (or, goodness-of-fit) of an estimated fundamental matrix F. The ingredient used most often is the so-called "epipolar distance", explained as follows. If the correspondence between two points $\mathbf{q}_1$ and $\mathbf{q}_2$ is correct, then $\mathbf{q}_2$ will be close to the epipolar line $\mathbf{l}_2 \sim \mathsf{F}\mathbf{q}_1$, and vice-versa. Thus, one can consider the distance of points to epipolar lines "predicted" by F, to construct our quality measure. Analogously to the case of line fitting considered above, we may conceive a measure that counts the number of correspondences where the distance to epipolar lines is less than a threshold. Or, one can take the median of all these epipolar distances.

We now have means of evaluating the quality of estimates of the fundamental matrix. The remaining question is how to obtain these estimates. The basic idea is very simple: depending on the method used, one needs 7 or 8 correspondences to estimate the fundamental matrix (cf. §6.7.2). All one needs to do is to carry out random selections of sets of 7 or 8 correspondences among the input. For each such random sample, one estimated the fundamental matrix and then computes one of the above measures *using all correspondences*. If sufficiently many random samples are processed, one will, with a high probability, have at least one sample that only contains inliers, that is correct correspondences. In general, only samples with inliers only, will lead to an estimate of the fundamental matrix that receives a good quality measure. One simply retains the fundamental matrix that has the highest quality measure.

Once a good estimate of the fundamental matrix is obtained this way, one can still improve

it by re-estimating it, using all inliers among the point correspondences (all correspondences where the epipolar distance is below a threshold).

This process is illustrated in figure 10, for the example of line fitting.

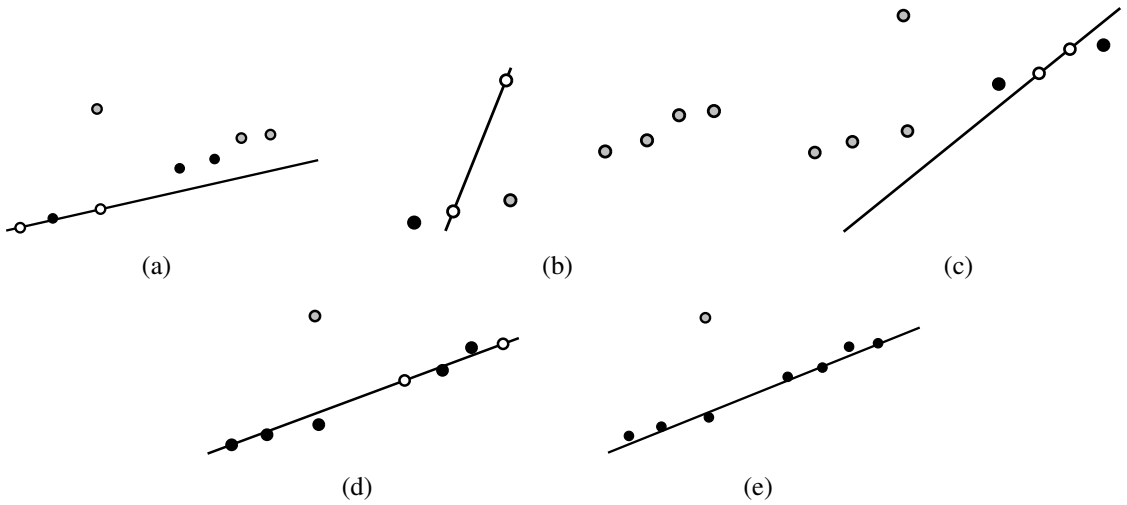


Figure 10: Illustration of the robust estimation process outlined in the text. At each iteration, two points are randomly selected (shown in white) and used to define a line. The outliers with respect to lines estimated this way, are shown in grey and the inliers, in black. (a) – (c) samples that will not be retained. (d) the best sample of this example. (e) result of re-estimating the line by fitting it to the inliers of sample (d).

One last issue remain to be discussed: how many random samples does one need to process in order to be quite sure to encounter at least one good sample, i.e. a sample with only inliers? We remark immediately that an absolute certainty is impossible, unless one samples the data exhaustively which quickly becomes intractable, even for problems with moderate amounts of data.

Given a percentage of outliers in the data, one may apply formulas of elementary statistics to compute the probability of encountering at least one good sample among x random samples. The percentage of outliers is obviously not known in advance but for many estimation problems, typical values met in practice, are often available. For example, image matching typically yields a few tens of percent of wrong matches in practice.

Let now g be the fraction of "good" data, s the size of a sample and x the number of random samples. The probability of encountering at least one good sample, is:

$$1 - (1 - g^s)^x$$

Hence, the number of samples one needs to process in order to attain a probability α of finding at least one good sample, is:

$$x \geq \frac{ln(1-\alpha)}{ln(1-g^s)}$$

Figure 11 shows some examples for the problem of fundamental matrix estimation (sample size is $s = 8$).

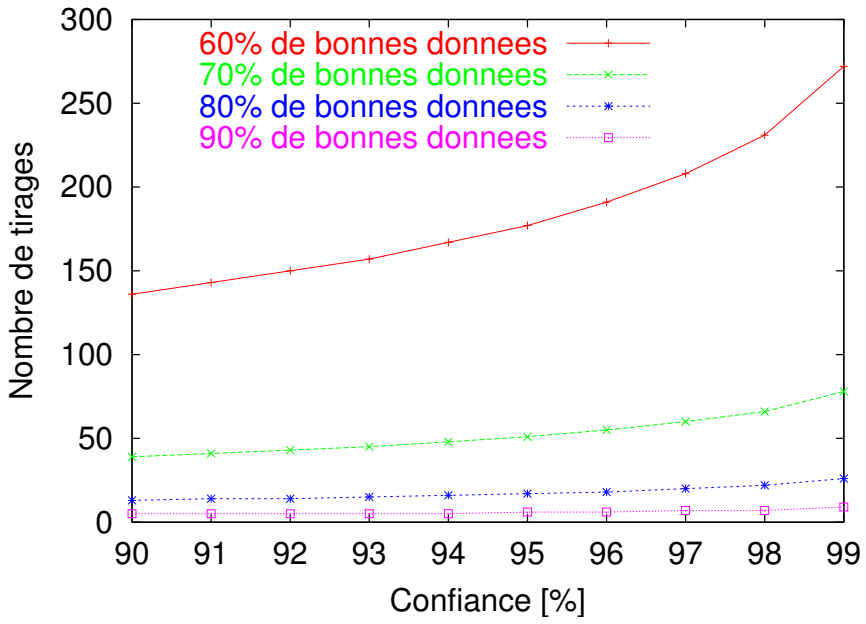


Figure 11: Number of samples that should be processed in order to reach a probability α ('confiance") of encountering at least one sample of 8 good correspondences.

Remark. In this section, we have described the basic principle of robust estimation methods. Such methods can be applied to any estimation problem that is affected by outliers in the data. Their main principle is surprisingly simple; however, the usage of these methods usually comes at the prize of an increased compuation time compared to non-robust methods, due to the repeated processing of random samples. However, non-robust methods, when applied to data containing outliers, will most often fail completely...

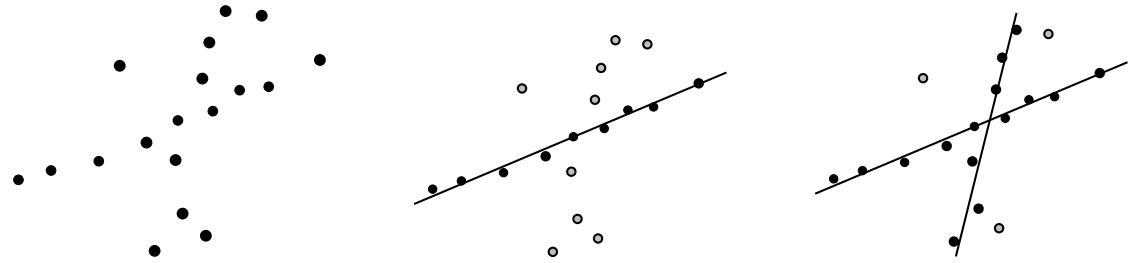
7 Estimating and segmenting motions

7.1 A basic method for motion segmentation

Consider a scene that contains objects that move independently one from another (in different directions, with different velocities, etc.). The goal of this section is to segment, in images taken by a camera (which may move too), the different objects. This will be done here by grouping the points (point matches actually) that appear as moving similarly; these will be considered here (a bit naively) as composing objects. This problem is known as *motion segmentation*.

We illustrate the principle behind the basic method that is considered in this section, with the help of the example started above, on the robust estimation of a line based on points. Consider now figure 12 (a). A robust estimation of the line probably gives a result looking similar to what is shown in figure 12 (b). This corresponds to the "dominant" line. In order to detect the second line, one may restart the robust estimation, but this time after having discarded the points considered as inliers to the first line. A possible result is shown in figure 12 (c).

One may iterate this process several times... In the example shown in figure 12 (c) however, the remaining points seem random and no further line can be found that has sufficient "support" from the remaining points, thus the process would stop here.



(a) Input data.

(b) Result of robust estimation.

(c) Result of second robust estimation, after discarding inliers of (b).

Figure 12: Iterated robust estimation, to estimate several lines from a set of points.

Let us now return to our target application, motion segmentation. The key of the method we are going to describe is the fact that objects that move differently, give rise to different epipolar geometries in a pair of images. To see this, consider figure 13: on the left, the case of a static camera is shown that takes two images of a dynamic scene, at two different time instants. The exact same images would have been acquired by a moving camera, that moves in the opposite direction than the lorry, with the lorry being static now. The second interpretation of the two images also corresponds to the case of two cameras, located in different positions and taking one image each, both at the same time (thus, the lorry does not move between the two image acquisitions). We may thus, as seen previously, define the epipolar geometry of the two cameras: matching points in the two images will satisfy this

epipolar geometry (for each point in the first image, the corresponding point in the second image will lie on the associated epipolar line).

What happens if the scene contains a second object, that moves differently from the first one? The second (virtual) camera, that appears as if moving in the opposite direction compared to the actual object, will thus come to lie at another position, compared to when considering the first object's motion. Hence, there will be a different epipolar geometry, associated with the points of the second object.

Remark. Epipolar geometry, as it was introduced in the previous chapter, characterizes the relative positioning of two cameras. It can be estimated from point matches between two images. Until now, we did not take into account temporal aspects of this, i.e. it was not specified if the two cameras take images at the same time or not.

In this section, we consider the case of a moving camera, taking images across time. If the objects in the scene move meanwhile, we can thus define several epipolar geometries, one each characterizing the relative positioning of the two camera poses *and* of a considered object.

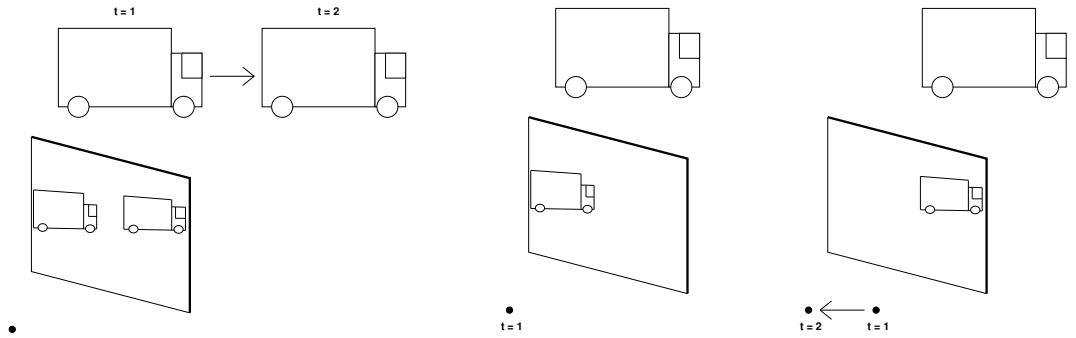


Figure 13: Left: a static camera that acquires two images of a moving object. Right: this is equivalent to two images taken by a moving camera, of a static object or, by two images taken by two different cameras, of the same static object.

The fact that each individually moving object gives in general rise to a particular epipolar geometry, gives us the means to construct a method for motion segmentation. This method follows the same scheme as outlined above for the estimation of lines from 2D points. Now however, input data are point matches across two images, as opposed to a single set of 2D points. A second difference is of course that the basic "model" to be estimated is no longer a 2D line, but the epipolar geometry (fundamental or essential matrix).

The robust method of §6.8 will usually produce the "dominant" epipolar geometry, that corresponds to the object with most available point matches. The iterative application of the method, after discarding the inliers of the previous found epipolar geometry, will successively reveal the different objects/motions.

Figure 14 gives an example. The scene contains two differently moving objects: the lorry and the rest of the scene (the camera moves, hence the background of the scene moves relative to the camera).

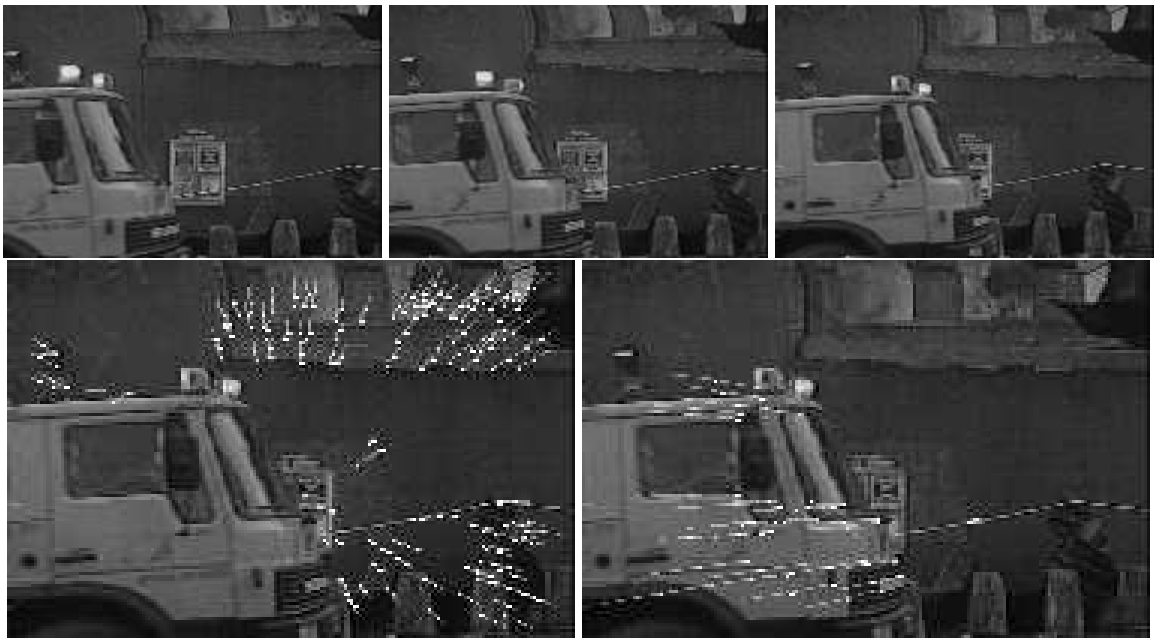


Figure 14: Three images of a sequence: the lorry move to the right, the camera moves away from the scene in a backward direction. The two images at the bottom show the two motions that were segmented: background and lorry. Images taken from: P. Torr, "Motion Segmentation and Outlier Detection", PhD thesis, Oxford University, 1995.

7.2 Estimating camera motion from the essential matrix

We consider how to extract, from the essential matrix, the extrinsic parameters of the two cameras. One also talks about **motion estimation**, since the main application of the principle concerns the estimation of the motion carried out by a single camera, between two image acquisitions. Nevertheless, we will talk of "two cameras" in the following, for ease of expression.

Consider equation (25) where the essential matrix is defined. The extrinsic parameters $\mathsf{R}_1, \mathsf{R}_2, \mathbf{t}_1$ and $\mathbf{t}_2$ are defined with respect to a world coordinate system, i.e. they express the *absolute* pose of the two cameras. However, all that interests us here is the motion *between* the image, or, their *relative* pose. One may thus assume, without loss of generality, that the first camera is in "canonic" pose, i.e.:

$$\begin{aligned} \mathsf{R}_1 &= \mathtt{I} \\ \mathbf{t}_1 &= \mathbf{0} \ . \end{aligned}$$

Equation (25) then becomes:

$$\mathsf{E} \sim \mathsf{R}\left[-\mathbf{t}\right]_\times \ ,$$

where R and $\mathbf{t}$ represent the orientation and position of the second camera, relative to the first one. Upon applying the fact that $\left[-\mathbf{v}\right]_\times = -\left[\mathbf{v}\right]_\times$ for any vector $\mathbf{v}$, the above equation

can be simplified further:

$$\mathsf{E} \sim \mathsf{R}\,[\mathbf{t}]_\times \quad . \tag{27}$$

From this equation, one may directly compute $\mathbf{t}$. In particular, $\mathbf{t}$ is a null-vector of E: since $[\mathbf{t}]_\times \mathbf{t} \sim \mathbf{t} \times \mathbf{t} = \mathbf{0}$, we have $\mathsf{E}\mathbf{t} = \mathbf{0}$. The essential matrix being singular, a null-vector exists and can be computed. However, given one null-vector, any multiple thereof is a null-vector too. This implies that $\mathbf{t}$ can only be computed up to a scale factor. Note that this is different from the ambiguity up to scale we encountered regularly for vectors of homogeneous coordinates: $\mathbf{t}$ was defined in section 1.2.3 as a vector of 3 non-homogeneous coordinates representing the camera position.

The scale ambiguity in estimating $\mathbf{t}$ is in the end not a problem: the scale of $\mathbf{t}$ represents the distance between the optical centers of the two cameras and this cannot be computed from images alone in any case. This can be seen by considering a "large" scene imaged by two distant cameras and a "reduced" but otherwise identical scene imaged by appropriately located close-by cameras. The images will be identical in the two cases, meaning that the size of the scene, as well as the distance between cameras and the scene and between the cameras themselves, cannot be inferred from images alone. As a consequence, we may only estimate the *direction* of camera motion, not its length.

Importantly however, if motion estimation is carried out for successive pairs of images, then it is possible to estimate the relative lengths of the inter-image motions. This means for example that when considering an entire image sequence acquired by a moving camera, one can estimate the entire camera trajectory up to a single scale ambiguity only, which is not that bad!

Let us now consider the computation of the rotation matrix R. A good method is based on the singular value decomposition of E. Let this be given as:

$$\mathsf{E} \sim \mathsf{U}\Sigma\mathsf{V}^\mathsf{T} \quad .$$

If E has been estimated from noisy data, it will in general not satisfy all characteristics of an essential matrix (one singular value being zero and the two non-zero ones being identical, cf. §6.6). In that case, the "perfect" essential matrix that is closest to E is given by[8]:

$$\hat{\mathsf{E}} = \mathsf{U}\ \mathrm{diag}(1,1,0)\ \mathsf{V}^\mathsf{T} \quad .$$

The vecteur $\mathbf{t}$ (null-vector of E) is given by the third column of V (it is easy to verify that $\hat{\mathsf{E}}\mathbf{t} = \mathbf{0}$). As for the rotation matrix, there exist two solutions:

$$\mathsf{R} = \mathsf{U} \begin{pmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathsf{V}^\mathsf{T} \quad .$$

If $\det \mathsf{R} = -1$ for any of the two solutions, one has to multiply R by -1.

It is easy to verify, from the properties of the matrices in a singular value decomposition, that the R computed above, are indeed rotation matrices.

[8]The notation $\mathrm{diag}(a, b, c)$ represents the diagonal matrix whose coefficients on the diagonal are a, b and c.

Remark. Both above solutions for R are mathematically correct. However, only one corresponds of course to the true camera motion. Often, the wrong solution can be discarded in practice. To do so, let us carry out a 3D reconstruction, based on the point correspondences used to estimate the essential matrix, for both solutions of R (cf. §4). Often, the wrong R will result in (some) 3D points that lie *behind* one or both cameras...

7.3 Summary: estimating the motion of a calibrated camera

In the last two sections, we have introduced techniques that allow to estimate the motion of a calibrated camera, solely from images acquires by an unknown scene. The skeleton of a complete practical approach is:

1. Matching of the images, with a general method (epipolar geometry is not yet available). The result will contain many outliers.
2. Robust estimation of the fundamental matrix.
3. Optional: redo image matching using the now available fundamental matrix. This can be used to find many more matches than in the first step. Then, re-estimate the fundamental matrix, to improve its quality.
4. Computation of the essential matrix (equation (25)).
5. Extraction of camera motion, as described in the previous section.

8 Reconstruction 3-D à partir de plusieurs images

Dans le chapitre précédent, nous avons décrit des méthodes permettant d'obtenir la position relative de deux caméras, ou bien le mouvement qu'effectue une caméra entre deux prises de vue. Ensuite, la méthode du §4 peut être appliquée afin d'obtenir une reconstruction 3-D des points à partir des deux images.

C'est déjà pas mal de pouvoir faire ceci – on peut alors regarder en avant et se poser d'autres questions :

- est-il possible d'estimer le mouvement de caméra et la structure 3-D de la scène *simultanément*, et non l'un après l'autre ?
- est-il possible d'utiliser plus de deux images à la fois ?

On peut s'attendre à ce que des solutions à ces questions permettent d'obtenir des résultats plus précis, surtout en ce qui concerne la deuxième question. La réponse aux questions est « oui, mais » ... Le premier « mais » concerne le modèle de caméra – avec le modèle de projection perspective (modèle sténopé) utilisé jusqu'ici, il est possible de travailler dans la direction indiquée par les questions. Par contre, c'est moins intuitif et plus complexe à mettre en œuvre qu'avec un modèle de caméra plus simple – la *projection affine*, que nous allons détailler dans le paragraphe suivant. En §8.2, nous développons une méthode de reconstruction « multi-images » qui s'appuie sur ce modèle de caméra et qui permet effectivement de déterminer les mouvements de caméra et la structure 3-D simultanément, et ce en utilisant plusieurs images à la fois.

8.1 Le modèle de caméra affine

La représentation algébrique du modèle de projection perspective ou modèle sténopé est une matrice de projection – une matrice de dimension 3×4 (qui est définie à un facteur scalaire près). Si nous imposons que la matrice de projection ait la forme :

$$\mathsf{P} \sim \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix} \tag{28}$$

on obtient une matrice de *projection affine*.

Pourquoi cette dénomination ? Considérons les projections de deux droites parallèles. Deux droites qui sont parallèles ont un point d'intersection qui est un point à l'infini. Soit

$$\mathbf{Q} \sim \begin{pmatrix} \bar{\mathbf{Q}} \\ 0 \end{pmatrix}$$

le vecteur des coordonnées homogènes de ce point. Les projections des deux droites sont des droites dans l'image qui contiennent la projection de $\mathbf{Q}$, qui est donnée par (pour P comme

dans l'équation (28)) :

$$\mathbf{q} \sim \mathsf{P}\mathbf{Q} \sim \begin{pmatrix} \times \\ \times \\ 0 \end{pmatrix}$$

On note que $\mathbf{q}$ est un point à l'infini dans le plan image. Par conséquent, les droites dans l'image sont parallèles, tout comme c'est le cas pour les droites originales en 3-D. On dit que la projection préserve le parallélisme de droites.

Plus généralement, une projection de la forme (28) préserve le parallélisme et des rapports de distances le long d'une droite. Puisque ces propriétés caractérisent les transformations affines, on parle alors de *projection affine*.

Il est intéressant de déterminer le centre de projection d'une projection affine. Puisque son image n'est pas définie, le centre de projection

$$\mathbf{C} \sim \begin{pmatrix} X \\ Y \\ Z \\ t \end{pmatrix}$$

doit vérifier :

$$\mathsf{P}\mathbf{C} = \mathbf{0}$$

Ceci donne :

$$\begin{pmatrix} \times \\ \times \\ P_{34}t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Le coefficient P_{34} est en général non nul (sinon, la projection serait dégénérée, contenant toute une ligne de zéros). Par conséquent, $t = 0$, ce qui veut dire rien d'autre que le centre de projection est un point à l'infini ! Les rayons de projection contiennent le centre de projection et sont donc tous parallèles entre eux, c'est pourquoi on parle aussi de *projection parallèle*.

Le modèle de projection affine (on parlera parfois aussi de *caméra affine*) est moins riche que celui de la projection perspective et il décrit donc en général moins bien ce qui se passe dans une caméra réelle. Pourtant, le modèle affine est une bonne approximation par exemple dans les situations suivantes :

- On utilise un objectif avec un très grand zoom ou bien une mise au point à l'infini.
- La profondeur de la scène est petite par rapport à la distance entre la caméra et la scène.

L'avantage du modèle affine par rapport au modèle perspectif réside dans le fait qu'on peut éviter la prise en compte parfois embarassante des facteurs scalaires imposés par l'utilisation des coordonnées homogènes. Concrètement, on peut fixer l'échelle des matrices de projection telle que les matrices sont de la forme :

$$\mathsf{P} = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Aussi bien, si l'on fait de la reconstruction 3-D, on peut imposer que les vecteurs-4 représentant les points 3-D aient un 1 comme dernière coordonnée.

Alors, on peut écrire l'équation de projection entre points 3-D et points 2-D dans l'image, avec une égalité *exacte* entre vecteurs (il n'y a plus de « $\sim$ ») :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \times \\ \times \\ 1 \end{pmatrix}$$

Dans la suite, nous utilisons les définitions suivantes :

$$\mathbf{q} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{q}} \\ 1 \end{pmatrix} \qquad \mathbf{Q} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{Q}} \\ 1 \end{pmatrix} \qquad \mathsf{P} = \begin{pmatrix} \mathsf{A} & \mathbf{b} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}$$

avec A de dimension 2×3 et $\mathbf{b}$ de longueur 2. Ainsi, l'équation de projection ci-dessus s'écrit comme :

$$\begin{pmatrix} \bar{\mathbf{q}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathsf{A} & \mathbf{b} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{Q}} \\ 1 \end{pmatrix} \tag{29}$$

8.2 Estimation du mouvement et reconstruction 3-D multi-images par factorisation

8.2.1 Formulation du problème

Considérons maintenant la situation suivante. On dispose de m images d'une scène statique et on a réussi à extraire et mettre en correspondance les projections de n points de la scène. Soit

$$\mathbf{q}_{ij} \quad i = 1, \ldots, m; j = 1, \ldots, n$$

la projection du jth point sur la ith image. Les $\mathbf{q}_{ij}$ sont nos seules données. Regardons d'où viennent ces points image : il y a m matrices de projection P_i et n points 3-D $\mathbf{Q}_j$ tels que :

$$\mathsf{P}_i \mathbf{Q}_j = \mathbf{q}_{ij} \quad \forall i, j$$

ou bien (d'après l'équation (29)) :

$$\begin{pmatrix} \mathsf{A}_i & \mathbf{b}_i \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{Q}}_j \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{q}}_{ij} \\ 1 \end{pmatrix} \quad \forall i, j \tag{30}$$

Le but est de reconstruire les P_i et les $\mathbf{Q}_j$. Nous avons la liberté du choix pour le repère de coordonnées dans lequel la reconstruction sera exprimée – liberté de choix dont on pourra

profiter pour simplifier le problème à résoudre. Nous pouvons par exemple « imposer » que l'un des points 3-D reconstruits se trouve à l'origine du repère. Sans perte de généralité :

$$\mathbf{Q}_1 = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

Considérons alors les m projections de ce point :

$$\begin{pmatrix} \bar{\mathbf{q}}_{i1} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathsf{A}_i & \mathbf{b}_i \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_i \\ 1 \end{pmatrix} \quad \forall i$$

Ces équations nous donnent directement la dernière colonne $\mathbf{b}_i$ pour chacune des matrices de projection : $\mathbf{b}_i = \bar{\mathbf{q}}_{i1}$. En remplaçant ceci dans l'équation (30), on obtient alors :

$$\begin{pmatrix} \mathsf{A}_i & \bar{\mathbf{q}}_{i1} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{Q}}_j \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{q}}_{ij} \\ 1 \end{pmatrix} \quad \forall i, j$$

ou bien :

$$\begin{pmatrix} \mathsf{A}_i \bar{\mathbf{Q}}_j + \bar{\mathbf{q}}_{i1} \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{q}}_{ij} \\ 1 \end{pmatrix} \quad \forall i, j$$

Seulement les deux premiers coefficients de ces vecteurs-3 sont intéressants, et après soustraction du vecteur $\bar{\mathbf{q}}_{i1}$ on obtient :

$$\mathsf{A}_i \bar{\mathbf{Q}}_j = \bar{\mathbf{q}}_{ij} - \bar{\mathbf{q}}_{i1} \quad \forall i, j$$

Comment peut-on interpréter le côté droit de cette équation ? Il correspond en effet à un changement de repère dans le plan image ; plus concrètement, une translation qui ramène les points $\bar{\mathbf{q}}_{i1}$ à l'origine (respectivement pour chaque i). Soient les $\mathbf{v}_{ij}$ les coordonnées des points image dans les nouveaux repères[9] :

$$\mathbf{v}_{ij} = \bar{\mathbf{q}}_{ij} - \bar{\mathbf{q}}_{i1} \quad \forall i, j$$

On a donc mn équations de la forme :

$$\mathsf{A}_i \bar{\mathbf{Q}}_j = \mathbf{v}_{ij}$$

où les inconnues sont les matrices A_i de dimension 2×3 et les vecteurs $\bar{\mathbf{Q}}_j$ de longueur 3.

8.2.2 Méthode de factorisation

L'ensemble des mn équations peut être écrit sous forme d'un système matriciel :

$$\underbrace{\begin{bmatrix} \mathsf{A}_1 \\ \mathsf{A}_2 \\ \vdots \\ \mathsf{A}_m \end{bmatrix}}_{\mathsf{A}} \underbrace{\begin{bmatrix} \bar{\mathbf{Q}}_1 & \bar{\mathbf{Q}}_2 & \cdots & \bar{\mathbf{Q}}_n \end{bmatrix}}_{\mathsf{Q}} = \underbrace{\begin{bmatrix} \mathbf{v}_{11} & \mathbf{v}_{12} & \cdots & \mathbf{v}_{1n} \\ \mathbf{v}_{21} & \mathbf{v}_{22} & \cdots & \mathbf{v}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_{m1} & \mathbf{v}_{m2} & \cdots & \mathbf{v}_{mn} \end{bmatrix}}_{\mathsf{V}}$$

[9] Naturellement, on a $\mathbf{v}_{i1} = \mathbf{0}$.

Les dimensions de ces matrices sont :

$$\mathsf{A}_{2m\times 3}\mathsf{Q}_{3\times n} = \mathsf{V}_{2m\times n} \tag{31}$$

Faisons quelques observations :

- On dispose de $2mn$ données (les coordonnées de tous les points dans toutes les images) – les coefficients de la matrice V. Pourtant, ces valeurs ne sont pas indépendantes les unes des autres, puisqu'elles peuvent être reproduites à partir de seulement $6m + 3n$ paramètres ($6m$ pour les matrices de projection et $3n$ pour les points 3-D). Implicitement, c'est cette redondance dans les données qui permettra d'obtenir la reconstruction 3-D.

- La matrice A n'a que 3 colonnes ; elle est donc de rang 3 au plus. Idem pour la matrice Q, qui n'a que 3 lignes. Le rang d'un produit de matrices est toujours inférieur ou égal au minimum des rangs des matrices individuelles. Par conséquent, la matrice V est de rang 3 au plus (c'est une manifestation de la redondance dans les données, mentionnée ci-dessus). Cette observation est la clef de l'algorithme de reconstruction énoncé dans la suite.

Si l'on effectue la décomposition en valeurs singulières de V, on obtient :

$$\mathsf{V}_{2m\times n} = \mathsf{U}_{2m\times n}\Sigma_{n\times n}\mathsf{X}_{n\times n} \tag{32}$$

Le fait que le rang de V soit 3 au plus implique qu'au plus 3 valeurs singulières sont non nulles. La matrice diagonale Σ est alors de la forme suivante :

$$\Sigma = \begin{pmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \sigma_3 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

Soit $\mathbf{u}_i$ la ith colonne de U et $\mathbf{x}_j^\mathsf{T}$ la jth ligne de X. L'équation (32) peut alors s'écrire :

$$\begin{aligned}
\mathsf{V} &= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 & \cdots & \mathbf{u}_n\end{pmatrix}
\begin{pmatrix}\sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \sigma_3 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0\end{pmatrix}
\begin{pmatrix}\mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \mathbf{x}_3^\mathsf{T} \\ \mathbf{x}_4^\mathsf{T} \\ \vdots \\ \mathbf{x}_n^\mathsf{T}\end{pmatrix} \\
&= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 & \cdots & \mathbf{u}_n\end{pmatrix}
\begin{pmatrix}\sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0\end{pmatrix}
\begin{pmatrix}\mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \mathbf{x}_3^\mathsf{T}\end{pmatrix} \\
&= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{pmatrix}
\begin{pmatrix}\sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3\end{pmatrix}
\begin{pmatrix}\mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \mathbf{x}_3^\mathsf{T}\end{pmatrix}
\end{aligned}$$

Si nous introduisons les notations

$$\begin{aligned}
\mathsf{A}' &= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{pmatrix}
\begin{pmatrix}\sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3\end{pmatrix} \qquad (33) \\
\mathsf{Q}' &= \begin{pmatrix}\mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \mathbf{x}_3^\mathsf{T}\end{pmatrix}
\end{aligned}$$

on obtient alors une équation matricielle avec les dimensions suivantes :

$$\mathsf{V}_{2m\times n} = \mathsf{A}'_{2m\times 3}\mathsf{Q}'_{3\times n} \ .$$

On note qu'on a construit des matrices A' et Q' qui « reproduisent » les données V et qui ont les mêmes dimensions que les matrices A et Q dans l'équation (31). Si l'on extrait les sous-matrices A'_i de dimension 2×3 de A' et les colonnes $\bar{\mathbf{Q}}'_j$ (des vecteurs-3) de Q', on obtient une estimation du mouvement et une reconstruction 3-D correctes : si l'on re-projette les points 3-D (représentés par les $\bar{\mathbf{Q}}'_j$) par les matrices de projection (les A'_i), on obtient les points image $\mathbf{v}_{ij}$ mesurés, nos données de départ.

8.2.3 Concernant l'unicité de la reconstruction

La méthode décrite ci-dessus permet donc d'obtenir une reconstruction 3-D, pourtant, il n'y a pas de solution unique. Afin de passer de la décomposition en valeurs singulières à deux matrices A' et Q' avec les bonnes dimensions, nous avons fait le choix arbitraire d'« absorber

» la matrice diagonale des valeurs singulières dans la matrice de gauche (voir l'équation (33)).

Nous aurions aussi bien pu choisir par exemple :

$$\begin{aligned} \mathsf{A}' &= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{pmatrix} \\ \mathsf{Q}' &= \begin{pmatrix}\sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3\end{pmatrix} \begin{pmatrix}\mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \mathbf{x}_3^\mathsf{T}\end{pmatrix} \end{aligned}$$

ou bien

$$\begin{aligned} \mathsf{A}' &= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{pmatrix} \begin{pmatrix}\sqrt{\sigma_1} & 0 & 0 \\ 0 & \sqrt{\sigma_2} & 0 \\ 0 & 0 & \sqrt{\sigma_3}\end{pmatrix} \\ \mathsf{Q}' &= \begin{pmatrix}\sqrt{\sigma_1} & 0 & 0 \\ 0 & \sqrt{\sigma_2} & 0 \\ 0 & 0 & \sqrt{\sigma_3}\end{pmatrix} \begin{pmatrix}\mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \mathbf{x}_3^\mathsf{T}\end{pmatrix} \end{aligned}$$

Plus généralement, ayant obtenu une solution pour A' et Q', les matrices définies par :

$$\mathsf{A}'' = \mathsf{A}'\mathsf{T} \qquad \mathsf{Q}'' = \mathsf{T}^{-1}\mathsf{Q}'$$

sont aussi une solution possible, pour toute matrice 3×3 inversible T, puisque :

$$\mathsf{A}''\mathsf{Q}'' = \mathsf{A}'\mathsf{Q}' = \mathsf{V}$$

Il existe donc effectivement une « famille de solutions » de 9 degrés de liberté (les 3×3 coefficients de T). Comment interpréter ce résultat ?

- Premièrement, il faut souligner qu'on a réduit le nombre d'inconnues de $6m + 3n$ à 9.
- La matrice T représente une transformation affine de l'espace 3-D. Puisque c'est exactement cette matrice qui reste indéfinie, on dit alors qu'on a obtenu une reconstruction 3-D *à une transformation affine près* (ou bien une *reconstruction affine*). Ayant choisi n'importe laquelle des solutions possibles pour A' et Q', on sait alors que cette reconstruction approche la « réalité » à seulement une transformation affine près : des points qui sont co-planaires dans la scène le seront dans la reconstruction, et aussi bien le parallélisme de droites et les rapports de longueurs le long d'une droite sont correctement retrouvés dans la reconstruction.

Il y a plusieurs moyens de déterminer, parmi la famille de solutions, la bonne solution (qui ne sera plus seulement une reconstruction affine, mais une reconstruction métrique ou Euclidienne) :

- Des connaissances sur par exemple des distances entres des points dans la scène donnent des contraintes sur la transformation affine T. Suffisamment de contraintes permettront de trouver une solution unique pour T et donc pour la reconstruction.
- Des connaissances sur les caméras (sur les A_i) pourront être utilisées de la même manière.

8.2.4 Quelques remarques

Au §8.2.1, nous avons fixé l'origine du repère 3-D de la reconstruction sur un point particulier de la scène. Cette méthode ne doit pas être appliquée directement comme telle en pratique, puisque la qualité de la reconstruction entière dépendra fortement de la qualité de la mise en correspondance concernant le seul point choisi.

En présence de bruit dans les données, la décomposition en valeurs singulières de V ne donnera pas seulement trois valeurs singulières non nulles. Pourtant, en pratique, il y aura trois valeurs singulières qui seront très grandes par rapport aux autres, en on fait comme si ces autres valeurs étaient égales à zéro.

8.3 Bibliographie

- C. Tomasi et T. Kanade, *Shape and Motion from Image Streams under Orthography: A Factorization Method,* International Journal on Computer Vision, Vol. 9, No. 2, pp. 137-154, 1992.

- P. Sturm et B. Triggs, *A factorization based algorithm for multi-image projective structure and motion,* European Conference on Computer Vision, pp. 709-720, 1996.

- B. Triggs, *Factorization Methods for Projective Structure and Motion,* IEEE International Conference on Computer Vision and Pattern Recognition, pp. 845-851, 1996.