
Part V

Appendices



Portrait of the Countess D'Haussonville, 1845 (oil on canvas)
by Jean-Auguste-Dominique Ingres (1780-1867)
The Frick Collection, New York

Appendix 1

Tensor Notation

Since tensor notation is not commonly used in computer vision, it seems appropriate to give a brief introduction to its use. For more details, the reader is referred to [Triggs-95]. For simplicity, these concepts will be explained here in the context of low-dimensional projective spaces, rather than in their full generality. However, the ideas apply in arbitrary dimensional vector spaces.

Consider a set of basis vectors $\mathbf{e}_i$, $i = 1, \dots, 3$ for a 2-dimensional projective space $\mathbb{P}^2$. For reasons to become clear, we will write the indices as subscripts. With respect to this basis, a point in $\mathbb{P}^2$ is represented by a set of coordinates x^i , which represents the point $\sum_{i=1}^3 x^i \mathbf{e}_i$. We write the coordinates with an upper index, as shown. Let $\mathbf{x}$ represent the triple of coordinates, $\mathbf{x} = (x^1, x^2, x^3)^T$.

Now, consider a change of coordinate axes in which the basis vectors $\mathbf{e}_i$ are replaced by a new basis set $\tilde{\mathbf{e}}_j$, where $\tilde{\mathbf{e}}_j = \sum_i H_j^i \mathbf{e}_i$, and H is the basis transformation matrix with entries H_j^i . If $\tilde{\mathbf{x}} = (\tilde{x}^1, \tilde{x}^2, \tilde{x}^3)^T$ are the coordinates of the vector with respect to the new basis, then we may verify that $\tilde{\mathbf{x}} = H^{-1} \mathbf{x}$. Thus, if the basis vectors transform according to H the coordinates of points transform according to the inverse transformation H^{-1} .

Next, consider a line in $\mathbb{P}^2$ represented by coordinates $\mathbf{l}$ with respect to the original basis. With respect to the new basis, it may be verified that the line is represented by a new set of coordinates $\tilde{\mathbf{l}} = H^T \mathbf{l}$. Thus coordinates of the line transform according to H^T .

As a further example, let P be a matrix representing a mapping between projective (or vector) spaces. If G and H represent basis transformations in the domain and range spaces, then with respect to the new bases, the mapping is represented by a new matrix $\tilde{P} = H^{-1} P G$. Note in these examples that sometimes the matrix H or H^T is used in the transformation, and sometimes H^{-1} .

These three examples of coordinate transformations may be written explicitly as follows.

$$\tilde{x}^i = (H^{-1})_j^i x^j \quad \tilde{l}_i = H_i^j l_j \quad \tilde{P}_j^i = (H^{-1})_k^i G_j^l P_l^k$$

where we use the tensor summation convention that an index repeated in upper and lower positions in a product represents summation over the range of the index. Note that those indices that are written as superscripts transform according to H^{-1} , whereas

those that are written as subscripts transform as H (or G). Note that there is no distinction in tensor notation between indices that are transformed by H , and those that are transformed by H^T . In general, tensor indices will transform by either H or H^{-1} – in fact this is the characteristic of a tensor. Those indices that transform according to H are known as *covariant* indices and are written as subscripts. Those indices that transform according to H^{-1} are known as *contravariant* indices, and are written as superscripts. The number of indices is the *valency* of the tensor. The sum over an index, e.g. $H_i^j l_j$, is referred to as a *contraction*, in this case the tensor H_i^j is contracted with the line l_j .

A1.1 The tensor ϵ_{rst}

The tensor ϵ_{rst} is defined for $r, s, t = 1, \dots, 3$ as follows:

$$\epsilon_{rst} = \begin{cases} 0 & \text{unless } r, s \text{ and } t \text{ are distinct} \\ +1 & \text{if } rst \text{ is an even permutation of } 123 \\ -1 & \text{if } rst \text{ is an odd permutation of } 123 \end{cases}$$

The tensor ϵ_{ijk} (or its contravariant counterpart, ϵ^{ijk}) is connected with the cross product of two vectors. If $\mathbf{a}$ and $\mathbf{b}$ are two vectors, and $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ is their cross product, then the following formula may easily be verified.

$$c_i = (\mathbf{a} \times \mathbf{b})_i = \epsilon_{ijk} a^j b^k.$$

Related to this is the expression (A4.5–p581) for the skew-symmetric matrix $[\mathbf{a}]_\times$. Using tensor notation one writes this as

$$([\mathbf{a}]_\times)_{ik} = \epsilon_{ijk} a^j.$$

Thus, one sees that if $\mathbf{a}$ is a contravariant vector, then $[\mathbf{a}]_\times$ is a matrix with two covariant indices. A similar formula holds for $[\mathbf{v}]_\times$ where $\mathbf{v}$ is covariant, namely $([\mathbf{v}]_\times)^{ik} = \epsilon^{ijk} v_j$.

Finally, the tensor ϵ_{ijk} is related to determinants: for three contravariant tensors a^i , b^j and c^k , one verifies that $a^i b^j c^k \epsilon_{ijk}$ is the determinant of the 3×3 matrix with rows a^i , b^j and c^k .

A1.2 The trifocal tensor

The trifocal tensor $\mathcal{T}_i^{jk}$ has one covariant and two contravariant indices. For vectors and matrices, such as x^i , l_i and P_j^i , it is possible to write the transformation rules using standard linear algebra notation, e.g. $\mathbf{x}' = H\mathbf{x}$. However, for tensors with three or more indices, this cannot conveniently be done. There is really no choice but to use tensor notation when dealing with the trifocal tensor.

Transformation rule. The arrangement of indices for the trifocal tensor implies a transformation rule

$$\tilde{\mathcal{T}}_i^{jk} = F_i^r (G^{-1})_s^j (H^{-1})_t^k \mathcal{T}_r^{st} \quad (\text{A1.1})$$

with respect to changes of basis in the three images. It is worthwhile pointing out one possible source of confusion here. The transformation rule (A1.1) shows how the

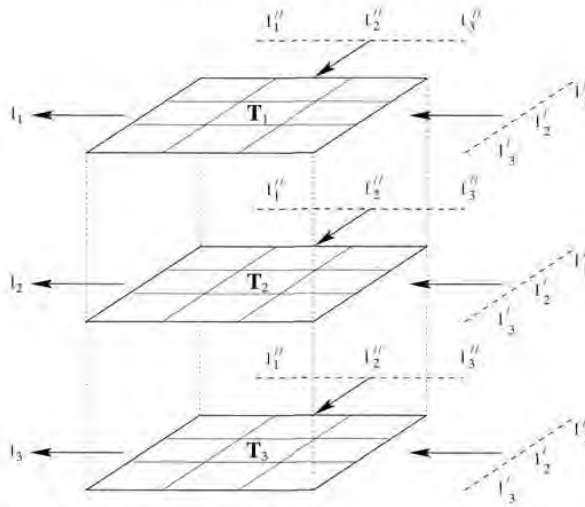


Fig. A1.1. A 3-dimensional representation of the trifocal tensor – figure after Faugeras and Papadopoulos [Faugeras-97]. The picture represents $l_i = l'_j l''_k T_i^{jk}$, which is the contraction of the tensor with the lines l' and l'' to produce a line l . In pseudo-matrix notation this can be written as $l_i = l'^T T_i l''^T$, where $(T_i)_{jk} = T_i^{jk}$.

tensor is transformed in terms of *basis* transformations in the three images. Often, we are concerned instead with point coordinate transformations. Thus, if F' , G' and H' represent *coordinate* transformations in the images, in the sense that $\hat{x}^j = F_i'^{jk} x^k$, and G' and H' are similarly defined for the other images, then the transformation rule may be written as

$$\hat{T}_i^{j'k'} = (F'^{-1})_i^r G_s'^{rj} H_t'^{rk} T_r^{st}.$$

Picture of tensors. A vector x may be thought of as a set of numbers arranged in a column or row, and a matrix H as a 2D array of numbers. Similarly, a tensor with three indices may be thought of as a 3D array of numbers. In particular the trifocal tensor is a $3 \times 3 \times 3$ cube of cells as illustrated in figure A1.1.

Appendix 2

Gaussian (Normal) and χ^2 Distributions

A2.1 Gaussian probability distribution

Given a vector $\mathbf{X}$ of random variables x_i for $i = 1, \dots, N$, with mean $\bar{\mathbf{X}} = E[\mathbf{X}]$, where $E[\cdot]$ represents the expected value, and $\Delta\mathbf{X} = \mathbf{X} - \bar{\mathbf{X}}$, the covariance matrix Σ is an $N \times N$ matrix given by

$$\Sigma = E[\Delta\mathbf{X} \Delta\mathbf{X}^T]$$

so that $\Sigma_{ij} = E[\Delta x_i \Delta x_j]$. The diagonal entries of the matrix Σ are the variances of the individual variables x_i , whereas the off-diagonal entries are the cross-covariance values.

The variables x_i are said to conform to a joint Gaussian distribution, if the probability distribution of $\mathbf{X}$ is of the form

$$P(\bar{\mathbf{X}} + \Delta\mathbf{X}) = (2\pi)^{-N/2} \det(\Sigma^{-1})^{1/2} \exp\left(-(\Delta\mathbf{X})^T \Sigma^{-1} (\Delta\mathbf{X})/2\right) \quad (\text{A2.1})$$

for some positive-semidefinite matrix Σ^{-1} . It may be verified that $\bar{\mathbf{X}}$ and Σ are the mean and covariance of the distribution. A Gaussian distribution is uniquely determined by its mean and covariance. The factor $(2\pi)^{-N/2} \det(\Sigma^{-1})^{1/2}$ is just the normalizing factor necessary to make the total integral of the distribution equal to 1.

In the special case where Σ is a scalar matrix $\Sigma = \sigma^2 \mathbf{I}$ the Gaussian PDF takes a simple form

$$P(\mathbf{X}) = (\sqrt{2\pi}\sigma)^{-N} \exp\left(-\sum_{i=1}^N (x_i - \bar{x}_i)^2 / 2\sigma^2\right)$$

where $\mathbf{X} = (x_1, x_2, \dots, x_N)^T$. This distribution is called an *isotropic Gaussian distribution*.

Mahalanobis distance. Note that in this case the value of the PDF at a point $\mathbf{X}$ is simply a function of the Euclidean distance $\left(\sum_{i=1}^N (x_i - \bar{x}_i)^2\right)^{1/2}$ of the point $\mathbf{X}$ from the mean $\bar{\mathbf{X}} = (\bar{x}_1, \dots, \bar{x}_N)^T$. By analogy with this one may define the *Mahalanobis distance* between two vectors $\mathbf{X}$ and $\mathbf{Y}$ to be

$$\|\mathbf{X} - \mathbf{Y}\|_{\Sigma} = \left((\mathbf{X} - \mathbf{Y})^T \Sigma^{-1} (\mathbf{X} - \mathbf{Y})\right)^{1/2}$$

One verifies that for a positive-definite matrix Σ , this defines a metric on $\mathbb{R}^N$. Using this notation, the general form of the Gaussian PDF may be written as

$$P(\mathbf{X}) \approx \exp \left(-\|\mathbf{X} - \bar{\mathbf{X}}\|_{\Sigma}^2 / 2 \right)$$

where the normalizing factor has been omitted. Thus, the value of the Gaussian PDF is a function of the Mahalanobis distance of the point $\mathbf{X}$ from the mean.

Change of coordinates. Since Σ is symmetric and positive-definite, it may be written as $\Sigma = \mathbf{U}^T \mathbf{D} \mathbf{U}$, where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{D} = (\sigma_1^2, \sigma_2^2, \dots, \sigma_N^2)$ is diagonal. Writing $\mathbf{X}' = \mathbf{U}\mathbf{X}$ and $\bar{\mathbf{X}}' = \mathbf{U}\bar{\mathbf{X}}$, and substituting in (A2.1), leads to

$$\begin{aligned} \exp \left(-(\mathbf{X} - \bar{\mathbf{X}})^T \Sigma^{-1} (\mathbf{X} - \bar{\mathbf{X}}) / 2 \right) &= \exp \left(-(\mathbf{X}' - \bar{\mathbf{X}}')^T \mathbf{U} \Sigma^{-1} \mathbf{U}^T (\mathbf{X}' - \bar{\mathbf{X}}') / 2 \right) \\ &= \exp \left(-(\mathbf{X}' - \bar{\mathbf{X}}')^T \mathbf{D}^{-1} (\mathbf{X}' - \bar{\mathbf{X}}') / 2 \right) \end{aligned}$$

Thus, the orthogonal change of coordinates from $\mathbf{X}$ to $\mathbf{X}' = \mathbf{U}\mathbf{X}$ transforms a general Gaussian PDF into one with diagonal covariance matrix. A further scaling by σ_i in each coordinate direction may be applied to transform it to an isotropic Gaussian distribution. Equivalently stated, a change of coordinates may be applied to transform Mahalanobis distance to ordinary Euclidean distance.

A2.2 χ^2 distribution

The χ_n^2 distribution is the distribution of the sum of squares of n independent Gaussian random variables. As applied to a Gaussian random vector $\mathbf{v}$ with non-singular covariance matrix Σ , the value of $(\mathbf{v} - \bar{\mathbf{v}})^T \Sigma^{-1} (\mathbf{v} - \bar{\mathbf{v}})$ satisfies a χ_n^2 distribution, where n is the dimension of $\mathbf{v}$. If the covariance matrix Σ is singular, then we must replace Σ^{-1} with the pseudo-inverse Σ^+ . In this case

- If $\mathbf{v}$ is a Gaussian random vector with mean $\bar{\mathbf{v}}$ and covariance matrix Σ , then the value of $(\mathbf{v} - \bar{\mathbf{v}})^T \Sigma^+ (\mathbf{v} - \bar{\mathbf{v}})$ satisfies a χ_r^2 distribution, where $r = \text{rank} \Sigma$.

The cumulative chi-squared distribution is defined as $F_n(k^2) = \int_0^{k^2} \chi_n^2(\xi) d\xi$. This represents the probability that the value of the χ_n^2 random variable is less than k^2 . Graphs of the χ_n^2 distribution and inverse cumulative χ_n^2 distributions for $n = 1, \dots, 4$ are shown in figure A2.1. A program for computing the cumulative chi-squared distribution $F_n(k^2)$ is given in [Press-88]. Since it is a monotonically increasing function, one may compute the inverse function by any simple technique such as subdivision, and values are tabulated in table A2.1 (compare with figure A2.1).

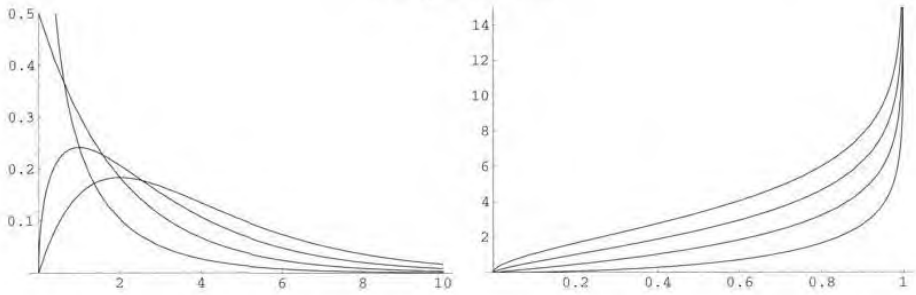


Fig. A2.1. The χ_n^2 distribution (left) and inverse cumulative χ_n^2 distribution F_n^{-1} (right) for $n = 1, \dots, 4$. In both cases, graphs are for $n = 1, \dots, 4$ bottom to top (at middle point of horizontal axis).

n	$\alpha = 0.95$	$\alpha = 0.99$
1	3.84	6.63
2	5.99	9.21
3	7.81	11.34
4	9.49	13.28

Table A2.1. Values of k^2 for which $F_n(k^2)$, the cumulative χ^2 distribution with n degrees of freedom, equals α , i.e. $k^2 = F_n^{-1}(\alpha)$, where α is the probability.

Appendix 3

Parameter Estimation

There is much theory about parameter estimation, dealing with properties such as the bias and variance of the estimate. This theory is based on analysis of the probability density functions of the measurements and the parameter space. In this appendix, we discuss such topics as bias of an estimator, the variance, the Cramér-Rao lower bound on the variance, and the posterior distribution. The treatment will be largely informal, based on examples, and exploring these concepts in the context of reconstruction.

The general lesson to be learnt from this discussion is that many of these concepts depend strongly on the particular parametrization of the model. In problems such as 3D projective reconstruction, where there is no preferred parametrization, these concepts are not well defined, or depend very strongly on assumed noise models.

A simple geometric estimation problem. The problem we shall consider is related to the triangulation problem of determining a point in space from its projection into two images. To simplify this problem, however, we consider its 2-dimensional analog. In addition, we fix one of the rays reducing the problem to one of estimating the position of a point along a known line from observing it in a single image.

Thus, consider a line camera (that is, one forming a 1D image as in section 6.4.2-(p175)) observing points on a single line. Let the camera be located at the origin $(0, 0)$ and point in the positive Y direction. Further, assume that it has unit focal length. Thus, the camera matrix for this camera is simply $[I_{2 \times 2} | 0]$. Now, suppose that the camera is observing points on the line $Y = X + 1$ (the “world line”). A point $(X, X + 1)$ on this line will be mapped to the image point $x = X/(X + 1)$. However, we assume that a point is measured with a certain inaccuracy, which may be modelled with a probability density function (PDF). The usual practice is to model the noise using a Gaussian distribution. Let us assume at least that the mode (maximum) of the distribution is at zero. The imaging setup is illustrated in figure A3.1.

The estimation problem we consider is the following: given the image coordinates of a point, estimate the position of the “world point” on the line $y = x + 1$. To consider a specific scenario, we may think of the line as a scintillator being flooded with gamma-rays. A camera is used to measure the location of each scintillation and determine its position along the line. The problem seems to be ridiculously simple, but it will turn out that there are some surprises.

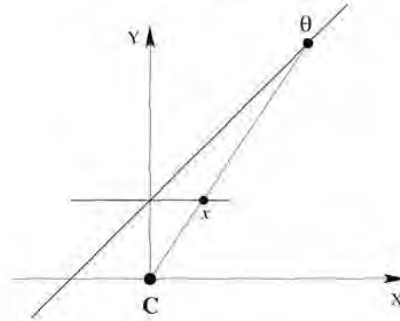


Fig. A3.1. The imaging setup for a simple estimation problem. A point on the line $Y = X + 1$ is imaged by a line camera. The projection mapping is given by $f : \theta \mapsto x = \theta / (1 + \theta)$, where θ parametrizes the points on the line $Y = X + 1$. Measurement is subject to noise with a zero-mode distribution.

Probability density function. We start by parametrizing the world line $Y = X + 1$ by a parameter θ , where the most convenient parametrization is $\theta = X$ so that the 2D point parametrized by θ is $(\theta, \theta + 1)$. This point projects to $\theta / (\theta + 1)$. We denote this projection function from the world line to the image line by f , so that $f(\theta) = \theta / (\theta + 1)$. The measurement of this point is corrupted by noise, resulting in a random variable x with probability distribution given by $p(x|\theta) = g(x - f(\theta))$. For instance, if g is a zero-mean Gaussian distribution with variance σ^2 , then

$$p(x|\theta) = (2\pi\sigma^2)^{-1/2} \exp\left(-(x - f(\theta))^2 / 2\sigma^2\right).$$

Maximum Likelihood estimate. An estimate of the parameter vector θ given a measured value x is a function denoted $\hat{\theta}(x)$, assigning a parameter vector θ to a measurement x . The maximum likelihood (ML) estimate is given by

$$\hat{\theta}_{ML} = \arg \max_{\theta} p(x|\theta).$$

In the current estimation problem it is easily seen that the ML estimate is obtained simply by back-projecting the measured point x and selecting its intersection with the world line, according to the formula

$$\hat{\theta}(x) = f^{-1}(x) = x / (1 - x).$$

This is the ML estimate, because the resulting point, with parameter $\hat{\theta}(x)$, projects forward to x , hence $p(x|\hat{\theta}) = g(x - x) = g(0)$ which by assumption gives the maximum (mode) of the probability density function g . Any other choice of parameter θ will give a smaller value of $p(x|\theta)$.

A3.1 Bias

A desirable property of an estimator is that it can be expected to give the right answer on the average. Given a parameter θ , or equivalently in our case a point on the world line, we consider all possible measurements x and from them reestimate the parameter θ , namely $\hat{\theta}(x)$. The estimator is known as *unbiased* if on the average we obtain

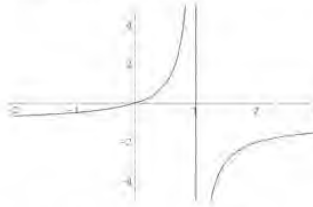


Fig. A3.2. The ML estimate of the world-point position $\hat{\theta}(x) = f^{-1}(x) = x/(1-x)$ for different measurements of the image point x . Note that for values of x greater than 1, the ML estimate switches to “behind” the camera.

the original parameter θ (the true value). In forming this average, we weight the measurements x according to their probability. More formally, the bias of the estimator is defined as

$$E_{\theta}[\hat{\theta}(x)] - \theta = \int_x p(x|\theta) \hat{\theta}(x) dx - \theta$$

and the estimator is unbiased if $E_{\theta}[\hat{\theta}(x)] = \theta$ for all θ . Here E_{θ} stands for the *expected value* given θ , defined as shown.

Another way of thinking of bias is in terms of repeating an experiment many times with identical model parameters and a different instance of the noise at each trial. The bias is the difference between the average value of the estimated parameters and the true parameter value. It is worth noting that for the bias to be defined, it is not necessary that the parameter space have an *a priori* distribution defined on it, not even that it be a measure space. It is necessary, however that it have some affine structure so that the average (or integral) can be formed.

Now, we determine whether the ML estimate of θ is unbiased in the case where $f(\theta) = \theta/(\theta + 1)$. The integral becomes

$$\int_x p(x|\theta) \hat{\theta}(x) dx = \int_x \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \theta/(\theta + 1))^2}{2\sigma^2}\right) \frac{x}{1-x} dx$$

It turns out that this integral diverges, and hence the bias is undefined. The difficulty is that with an assumed Gaussian distribution of noise, for any value of θ , there is always a finite (though perhaps very small) probability $p(x|\theta)$ that $x > 1$. For values of $x > 1$, the corresponding ray does not meet the world line in front of the camera (since the ray is parallel to the world line at $x = 1$). The estimate $\hat{\theta}(x)$ as a function of x is shown in figure A3.2, showing how it results in estimated world-points behind the camera. Even if values of $\hat{\theta}(x)$ behind the camera are ignored, the ML estimator has infinite bias, as is explained in figure A3.3.

Limiting the range of parameters. Since the range of the parameter θ is from -1 to ∞ , it makes sense to limit its range. In fact, we may have knowledge that all “events” on the world-line lie in a more restricted range. As an example, suppose that we assume that θ lies in the range $-1 \leq \theta \leq 1$, and hence noise-free projected points are in the range $-\infty < x < 1/2$. In this case, the ML estimate for any image point $x > 1/2$ will

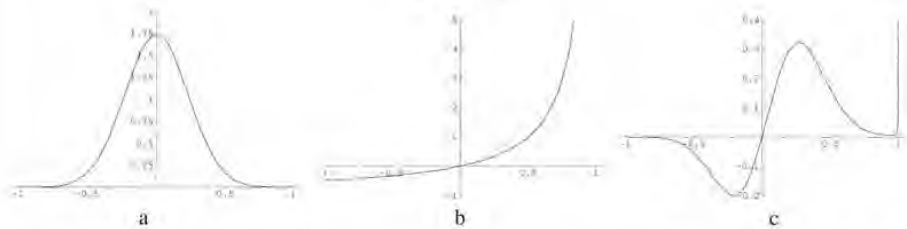


Fig. A3.3. The reason that the ML estimate with Gaussian noise model has infinite bias. (a) The distribution of possible values of the image measurement given a world-point at $x = 0, y = 1$ assuming Gaussian noise distribution with $\sigma = 0.4$ – in symbols $p(x|\theta = 0)$. (b) The ML estimate of the world point for different values of the image point, $\hat{\theta}(x) = x/(1-x)$. Note that as the image point approaches 1, the estimated point on the world-line recedes to infinity. (c) Product $\hat{\theta}(x)p(x|\theta = 0)$. The integral of this function from $x = -\infty$ to $x = 1$ gives the bias of the ML estimator. Note that as x approaches 1, the graph increases abruptly to infinity. The integral is unbounded, meaning that the estimator has infinite bias.

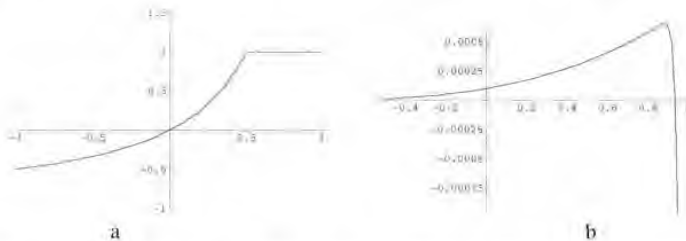


Fig. A3.4. (a) If the range of possible values of the world-point parameter θ is limited to the range $\theta \leq 1$, then the ML estimate of any point $x > 1/2$ will be $\theta = 1$. This will prevent infinite bias in the estimate, but there will still be bias. (b) The bias $E_{\theta}[\hat{\theta}(x)] - \theta = \int_{-\infty}^{\infty} \hat{\theta}(x)p(x|\theta)dx - \theta$ as a function of θ . Measurement noise is Gaussian with $\sigma = 0.01$.

be at $\hat{\theta}(x) = 1$. With this restriction on the parameter θ , the ML estimate is still biased, as shown in figure A3.4.

If the noise-distribution has finite support, then the bias will also be finite for most values of θ , even if the range of the parameter θ is unrestricted. This is shown in figure A3.5. One learns from this that the bias of the estimator can be very dependent on the noise-model – a factor that is usually not within our control.

Dependency of bias on parametrization. The reason for the infinite bias for the ML estimator with Gaussian noise model in this example is the projective mapping between the world line and the image line. It is possible to parametrize the world line differently in a way that will change the bias.

Let the world line $Y = X + 1$ be parametrized in such a way that parameter θ represents the point $(\theta/(1-\theta), 1/(1-\theta))$ on the line. The part of the line in front of the camera (having positive y coordinate) is parametrized by θ in the range $-\infty < \theta < 1$. Under the projection $(x, y) \mapsto x/y$, the projection map f is given by $f(\theta) = x/y = \theta$, thus the point with parameter θ maps to θ . In other words, points on the world line are

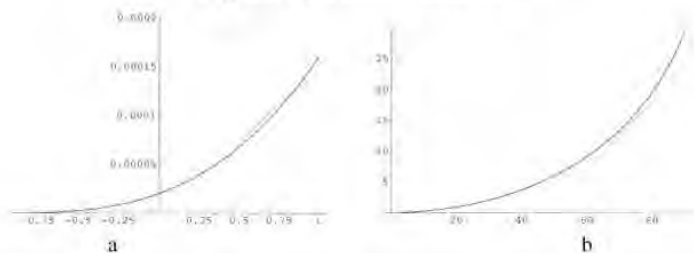


Fig. A3.5. If the noise-model has finite support then the bias will be finite. In this example the noise model is $3(1 - (x/\sigma)^2)/4\sigma^2$, for $-\sigma \leq x \leq \sigma$, where $\sigma = 0.01$. (a) Bias as a function of θ for small values of θ . (b) Percentage bias $(E_\theta[\hat{\theta}(x)] - \theta)/\theta$. The bias is relatively small for small values of θ , but larger (up to 20%) for large values. The position of the point along the world-line is always over-estimated.

parametrized by the coordinate of the point that they project to under the camera mapping.

Now, in this case, the ML estimate is given by $\hat{\theta}(x) = f^{-1}(x) = x$, and the bias is

$$\begin{aligned} \int_x \tilde{\theta}(x)p(x|\theta)dx - \theta &= \int_x xg(x - f(\theta))dx - \theta \\ &= \int_x (x + f(\theta))g(x)dx - \theta \\ &= \int_x xg(x)dx + f(\theta) \int_x g(x)dx - \theta \\ &= 0 + f(\theta) - \theta = 0 \end{aligned}$$

Assuming that the distribution $g(x)$ is zero-mean. Note, this shows that the estimate of the *original* measurement x is unbiased.

Lessons about bias. By a change of parametrization of the world line, the bias has been changed from infinite to zero. In the present example, there is a natural affine parametrization of the world line, for which we have seen that the bias is infinite. However, if we are working in a projective context, then the parameter space has no natural affine parametrization. In this case, it is somewhat meaningless to speak of any absolute measurement of bias. A second lesson from the above example is that the ML estimate of the corrected measurement (as opposed to the world point) is unbiased.

It was also seen that the value of bias is strongly dependent on the noise distribution. Even the very small tails of the Gaussian distribution have a very large effect on the computed bias. Of course, a Gaussian distribution of the noise is only a convenient model for modelling image measurement errors. The exact distribution of image measurement noise is generally unknown, and the conclusion is inescapable that one can not theoretically compute an exact value for the bias of a given estimator, except for synthetic data with known noise model.

A3.2 Variance of an estimator

The other important attribute of an estimator is its variance. Consider an experiment being repeated many times with the same model parameters, but a different instantiation of the noise at each trial. Applying our estimator to the measured data, we obtain an estimate for each of these trials. The variance of the estimator is the variance (or covariance matrix) of the estimated values. More precisely, we can define the variance for an estimation problem involving a single parameter as

$$\text{Var}_\theta(\hat{\theta}) = E_\theta[(\hat{\theta}(x) - \bar{\theta})^2] = \int_x (\hat{\theta}(x) - \bar{\theta})^2 p(x|\theta) dx$$

where

$$\bar{\theta} = E_\theta[\hat{\theta}(x)] = \int_x \hat{\theta}(x) p(x|\theta) dx = \theta + \text{bias}(\hat{\theta})$$

In the case where the parameters θ form a vector, $\text{Var}_\theta(\hat{\theta})$ is the covariance matrix

$$\text{Var}_\theta(\hat{\theta}) = E_\theta[(\hat{\theta}(x) - \bar{\theta})(\hat{\theta}(x) - \bar{\theta})^\top] \quad (\text{A3.1})$$

In many cases we might be more interested in the variability of the estimate with respect to the original parameter θ , which is the mean-squared error of the estimator. This is easily computed from

$$E_\theta[(\hat{\theta}(x) - \theta)(\hat{\theta}(x) - \theta)^\top] = \text{Var}_\theta(\hat{\theta}) + \text{bias}(\hat{\theta}) \text{bias}(\hat{\theta})^\top.$$

It should be noted that, as with the bias, the variance of an estimator makes good sense only when there is a natural affine structure on the parameter set, at least locally.

Most estimation algorithms will give the right answer if there is no noise. If an algorithm performs badly when noise is added, this means that either the bias or variance of the algorithm is high. This is the case, for instance with the DLT algorithm 4.1(p91), or the unnormalized 8-point algorithm discussed in section 11.1(p279). The variance of the algorithm grows quickly with added noise.

The Cramér-Rao lower bound. It is evident that by adding noise to a set of measurements information is lost. Consequently, it is not to be expected that any estimator can have zero bias and variance in the presence of noise on the measurements. For unbiased estimators, this notion is formalized in the Cramér-Rao lower bound, which is a bound on the variance of an unbiased estimator. To explain the Cramér-Rao bound, we need a few definitions. Given a probability distribution $p(x|\theta)$, the Fisher score is defined as $V_\theta(x) = \partial_\theta \log p(x|\theta)$. The Fisher Information Matrix is defined to be

$$\begin{aligned} F(\theta) &= E_\theta[V_\theta(x) V_\theta(x)^\top] \\ &= \int_x V_\theta(x) V_\theta(x)^\top p(x|\theta) dx. \end{aligned}$$

The relevance of the Fisher Information Matrix is expressed in the following result.

Result A3.1. Cramér-Rao lower bound. For an unbiased estimator $\hat{\theta}(x)$,

$$\det(E[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^\top]) \geq 1/\det F(\theta).$$

A Cramér-Rao lower bound may also be given in the case of a biased estimator.

A3.3 The posterior distribution

An alternative to the ML estimate is to consider the probability distribution for the parameters, given the measurements, namely $p(\theta|x)$. This is known as the posterior distribution, namely the distribution for the parameters *after* the measurements have been taken. To compute it, we need a *prior* distribution $p(\theta)$ for the parameters *before* any measurement has been taken. The posterior distribution can then be computed from Bayes Law

$$p(\theta|x) = \frac{p(x|\theta) p(\theta)}{p(x)}.$$

Since the measurement x is fixed, so is its probability $p(x)$, so we may ignore it, leading to $p(\theta|x) \approx p(x|\theta) p(\theta)$. The maximum of the posterior distribution is known as the Maximum A Posteriori (MAP) estimate.

Note. Though the MAP estimate may seem like a good idea, it is important to realize that it depends on the parametrization of the parameter space. The posterior probability distribution is proportional to $p(x|\theta)p(\theta)$. However, $p(\theta)$ is dependent on the parametrization of θ . For instance if $p(\theta)$ is a uniform distribution in one parametrization, it will not be a uniform distribution in a different parametrization that differs by a non-affine transformation. On the other hand, $p(x|\theta)$ does not depend on the parametrization of θ . Therefore, the result of a change of parametrization is to alter the posterior distribution in such a way that its maximum will change. If the parameter space does not have a natural affine coordinate system (for instance if the parameter space is projective) then the MAP estimate does not really make a lot of sense.

Other estimates based on the posterior distribution are also possible. Given a measurement x , and the posterior distribution $p(\theta|x)$, we may wish to make a different estimate of the parameter θ . One sensible choice is the estimate that minimizes the expected squared error in the estimate, namely

$$\hat{\theta}(x) = \operatorname{argmin}_{\theta} E[\|Y - \theta\|^2] = \operatorname{argmin}_{\theta} \int \|Y - \theta\|^2 p(\theta|x) d\theta,$$

which is the mean of the posterior distribution.

A further alternative is to minimize the expected absolute error

$$\hat{\theta}(x) = \operatorname{argmin}_{\theta} E[\|Y - \theta\|] = \operatorname{argmin}_{\theta} \int \|Y - \theta\| p(\theta|x) d\theta,$$

which is the median of the posterior distribution. Examples of these estimates are shown in figure A3.6 and figure A3.7.

More properties of these estimates are listed in the notes at the end of this appendix.

A3.4 Estimation of corrected measurements

We have seen that in geometric estimation problems, particularly those that involve projective models, concepts such as bias and variance of the estimator are dependent on the particular parametrization of the model, for instance a particular projective coordinate frame chosen. Even in cases where a natural affine parametrization of the model

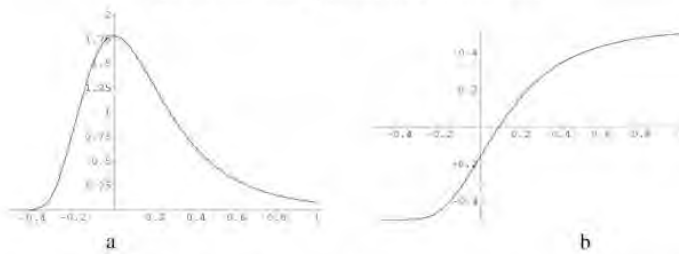


Fig. A3.6. **Different estimators for θ .** (a) For the imaging setup of figure A3.1: the a posteriori distribution $p(\theta|x=0)$ assuming a Gaussian noise distribution with $\sigma = 0.2$, and a prior distribution for θ , uniform on the interval $[-1/2, 1]$. The mode (maximum) of this distribution is the Maximal Apriori (MAP) estimate of θ , which is identical with the ML estimate, because of the assumed uniform distribution for θ . The mean of this distribution ($\theta = 0.1386$) is the estimate that minimizes the expected squared error $E[(\hat{\theta}(x) - \bar{\theta})^2]$ with respect to the true measurement $\bar{\theta}$. (b) The cumulative a posteriori distribution (offset by -0.5). The zero point of this graph is the median of the distribution, namely the estimate that minimizes $E[|\hat{\theta}(x) - \bar{\theta}|]$. The median lies at $\theta = 0.09137$.

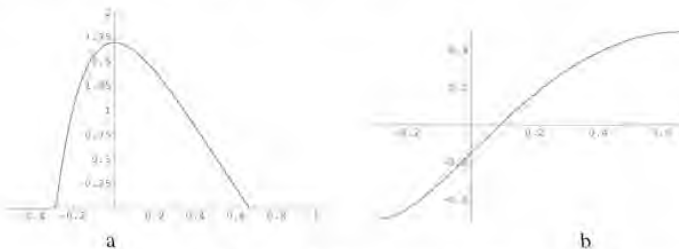


Fig. A3.7. **Different estimators with the parabolic noise model.** The two graphs show the a posteriori distribution of θ and its cumulative distribution. In this example, the noise model is $3(1 - (x/\sigma)^2)/4\sigma^2$, for $-\sigma \leq x \leq \sigma$, where $\sigma = 0.4$. The mode of the distribution ($\theta = 0$) is the MAP estimate, identical with the ML estimate, the mean ($\theta = 0.1109$) minimizes the expected squared error in θ and the median ($\theta = 0.0911$) minimizes the expected absolute error.

exists, it may be difficult to find an unbiased estimator. For instance, the ML estimator for problems such as triangulation is biased.

We saw however in the example of 1D back-projection discussed in section A3.1, that if instead of attempting to compute the model (namely the back-projected point) we estimate the corrected measurement instead, then the ML estimator is unbiased. We explore this notion further in this section, and show that in a general setting, the ML estimator of the corrected measurements is not only unbiased but attains the Cramér-Rao lower bound, when the noise-model is Gaussian.

Consider an estimation problem that involves fitting a parametrized model to a set of image measurements. As seen in section 5.1.3(p134) this problem may be viewed as an estimation problem in a high-dimensional Euclidean space $\mathbb{R}^N$, which is the space of all image measurements. This is illustrated in figure 5.2(p135). The estimation problem is, given a measurement vector $\mathbf{X}$, to find the closest point lying on a surface representing the set of all allowable exact measurements. The vector $\hat{\mathbf{X}}$ represents the

set of "corrected" image measurements that conform to the given model. The model itself may depend on a set of parameters θ , such as the fundamental matrix, hypothesized 3D points or other parameters appropriate to the problem.

The estimation of the parameters θ is subject to bias in the same way as we have seen with the simple problem discussed in section A3.1, and the exact degree of bias is dependent on the precise parametrization. Generally (for instance in projective-reconstruction problems) there is no natural affine coordinate system for the model parameters, though there is a natural affine coordinate system for the image plane.

If we think of the problem differently, as the problem of directly finding the corrected measurement vector $\hat{\mathbf{X}}$, then we find a more favourable situation. The measurements are carried out in the images, which have a natural affine coordinate system, and so questions of bias in estimating the corrected measurements make more sense. We will show that, provided the measurement surface is well approximated by its tangent plane, the ML estimate of the corrected measurement vector is unbiased, provided the noise is zero-mean. In addition, if the noise is Gaussian and isotropic, then the ML estimate meets the Cramér-Rao lower-bound.

The geometric situation is follows. A point $\bar{\mathbf{X}}$ lies on a measurement surface, as shown in figure 5.2(p135). Noise is added to this point to obtain a measured point $\mathbf{X}$. The estimate $\hat{\mathbf{X}}$ of the true point $\bar{\mathbf{X}}$ is obtained by selecting the closest point on the measurement surface to the measured point. We make an assumption that the measurement surface is effectively planar near $\bar{\mathbf{X}}$.

We may choose a coordinate system in which the measurement surface close to $\bar{\mathbf{X}}$ is spanned by the first d coordinates. We may write $\bar{\mathbf{X}} = (\bar{\mathbf{X}}_1^T, \mathbf{0}_{N-d}^T)^T$, where $\bar{\mathbf{X}}_1$ is a d -vector. The measured point may similarly be written as $\mathbf{X} = (\mathbf{X}_1^T, \mathbf{X}_2^T)^T$, and its projection onto the tangent plane is $\hat{\mathbf{X}} = (\mathbf{X}_1^T, \mathbf{0}^T)^T = (\hat{\mathbf{X}}_1^T, \mathbf{0}^T)^T$. We suppose that the noise-distribution is given by $p(\mathbf{X}|\bar{\mathbf{X}}) = g(\mathbf{X} - \bar{\mathbf{X}})$. Now, the bias of the estimate $\hat{\mathbf{X}}$ is

$$\begin{aligned} E[(\hat{\mathbf{X}} - \bar{\mathbf{X}})] &= \int_{\mathbf{X}} (\hat{\mathbf{X}} - \bar{\mathbf{X}}) p(\mathbf{X}|\bar{\mathbf{X}}) d\mathbf{X} = \int_{\mathbf{X}} (\hat{\mathbf{X}} - \bar{\mathbf{X}}) g(\mathbf{X} - \bar{\mathbf{X}}) d\mathbf{X} \\ &= \int_{\mathbf{X}} \mathbf{J}(\mathbf{X} - \bar{\mathbf{X}}) g(\mathbf{X} - \bar{\mathbf{X}}) d\mathbf{X} \\ &= \mathbf{J} \int_{\mathbf{X}} (\mathbf{X} - \bar{\mathbf{X}}) g(\mathbf{X} - \bar{\mathbf{X}}) d\mathbf{X} \\ &= \mathbf{0} \end{aligned}$$

where $\mathbf{J}$ is the matrix $[\mathbf{I}_{d \times d} | \mathbf{0}_{d \times (N-d)}]$. This shows that the estimate of $\mathbf{X}$ is unbiased as long as g has zero-mean. The variance of the estimate is equal to

$$\begin{aligned} E[(\hat{\mathbf{X}} - \bar{\mathbf{X}})(\hat{\mathbf{X}} - \bar{\mathbf{X}})^T] &= \int_{\mathbf{X}} (\hat{\mathbf{X}} - \bar{\mathbf{X}})(\hat{\mathbf{X}} - \bar{\mathbf{X}})^T p(\mathbf{X}|\bar{\mathbf{X}}) d\mathbf{X} = \int_{\mathbf{X}} (\hat{\mathbf{X}} - \bar{\mathbf{X}})(\hat{\mathbf{X}} - \bar{\mathbf{X}})^T g(\mathbf{X} - \bar{\mathbf{X}}) d\mathbf{X} \\ &= \int_{\mathbf{X}} \mathbf{J}(\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T \mathbf{J}^T g(\mathbf{X} - \bar{\mathbf{X}}) d\mathbf{X} \\ &= \mathbf{J} \int_{\mathbf{X}} (\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T g(\mathbf{X} - \bar{\mathbf{X}}) d\mathbf{X} \mathbf{J}^T \\ &= \mathbf{J} \Sigma_g \mathbf{J}^T \end{aligned}$$

where Σ_g is the covariance matrix of g .

We now compute the Cramér-Rao lower bound for this estimator, supposing that the distribution $g(\mathbf{X})$ is Gaussian defined by $g(\mathbf{X}) = k \exp(-\|\mathbf{X}\|^2/2\sigma^2)$. In this case, the variance of the estimator is simply $\sigma^2 \mathbf{I}_{d \times d}$.

We next compute the Fisher information matrix. The probability distribution is

$$\begin{aligned} p(\mathbf{X}|\bar{\mathbf{X}}) &= g(\mathbf{X} - \bar{\mathbf{X}}) = k \exp(-\|\mathbf{X} - \bar{\mathbf{X}}\|^2/2\sigma^2) \\ &= k \exp(-\|\mathbf{X}_1 - \bar{\mathbf{X}}_1\|^2/2\sigma^2) \exp(-\|\mathbf{X}_2\|^2/2\sigma^2). \end{aligned}$$

Taking logarithms and derivatives with respect to $\bar{\mathbf{X}}_1$ gives

$$\partial_{\bar{\mathbf{X}}_1} \log p(\mathbf{X}|\bar{\mathbf{X}}) = -(\mathbf{X}_1 - \bar{\mathbf{X}}_1)/\sigma^2.$$

The Fisher information matrix is then

$$1/\sigma^2 \int (\mathbf{X}_1 - \bar{\mathbf{X}}_1)(\mathbf{X}_1 - \bar{\mathbf{X}}_1)^T g(\mathbf{X}_1 - \bar{\mathbf{X}}_1) g(\mathbf{X}_2) d\mathbf{X}_1 d\mathbf{X}_2 = \mathbf{I}_{d \times d}/\sigma^2.$$

Thus, the Fisher information matrix is the inverse of the covariance matrix for the estimator. Thus, for the case where the noise distribution is Gaussian, the ML estimator meets the Cramér-Rao lower bound, to the extent that the measurement surface is flat.

It should be noticed that the Fisher Information Matrix does not depend on the specific shape of the measurement surface, but only on its first-order approximation, the tangent plane. The properties of the estimate does however depend on the shape of the measurement surface, both as regards its bias and variance. It may also be shown that if the Cramér-Rao bound is met, then the noise distribution must be Gaussian. In other words, if the noise distribution is not Gaussian, then we can not meet the lower bound.

A3.5 Notes and exercises

- (i) Show by a specific example that the a posteriori distribution is altered by a change of coordinates in the parameter space. Show also that the mean of the distribution may be altered by such a coordinate change. Thus the mode (MAP estimate) and mean of the posterior distribution are dependent on the choice of coordinates for the parameter space.
- (ii) Show for any PDF $p(\theta)$ defined on $\mathbb{R}^n$, that $\operatorname{argmin}_Y \int \|Y - \theta\|^2 p(\theta) d\theta$ is the mean of the distribution.
- (iii) Show for any PDF $p(\theta)$ defined on $\mathbb{R}$, that $\operatorname{argmin}_Y \int |Y - \theta| p(\theta) d\theta$ is the median of the distribution. In higher dimensions, show that $\int \|Y - \theta\| p(\theta) d\theta$ is a convex function of Y , and hence has a single minimum. The value of Y that minimizes this is a higher-dimensional generalization of the median of a 1-dimensional distribution.
- (iv) Show that the median of a PDF defined on $\mathbb{R}$ is invariant to reparametrization of $\mathbb{R}$. Show by an example that this is not true in higher dimensions, however.

Appendix 4

Matrix Properties and Decompositions

In this appendix we discuss matrices with particular forms that occur throughout the book, and also various matrix decompositions.

A4.1 Orthogonal matrices

A square matrix U is known as *orthogonal* if its transpose is its inverse – symbolically $U^T U = I$, where I is the identity matrix. This means that the column vectors of U are all of unit norm and are orthogonal. This may be written $u_i^T u_j = \delta_{ij}$. From the condition $U^T U = I$ one easily deduces that $U U^T = I$. Hence the row vectors of U are also of unit norm and are orthogonal. Consider once more the equation $U^T U = I$. Taking determinants leads to the equation $(\det U)^2 = 1$, since $\det U = \det U^T$. Thus if U is orthogonal, then $\det U = \pm 1$.

One easily verifies that the orthogonal matrices of a given fixed dimension form a group, denoted O_n , since if U and V are orthogonal, then $(UV)^T UV = V^T U^T UV = I$. Furthermore, the orthogonal matrices of dimension n with positive determinant form a group, called SO_n . An element of SO_n is called an n -dimensional rotation.

Norm-preserving properties of orthogonal matrices. Given a vector x , the notation $\|x\|$ represents its Euclidean length. This can be written as $\|x\| = (x^T x)^{1/2}$. An important property of orthogonal matrices is that multiplying a vector by an orthogonal matrix preserves its norm. This is easily seen by computing

$$(Ux)^T (Ux) = x^T U^T U x = x^T x.$$

By the QR decomposition of a matrix is usually meant the decomposition of the matrix A into a product $A = QR$, where Q is orthogonal, and R is an upper-triangular matrix. The letter R stands for *Right*, meaning upper-triangular. Similar to the QR decomposition, there are also QL, LQ and RQ decompositions, where L denotes a *Left* or lower-triangular matrix. In fact, the RQ decomposition of a matrix is the one that will be of most use in this book, and which will therefore be discussed here. The most important case is the decomposition of a 3×3 matrix and we will concentrate on this in the following section.

A4.1.1 Givens rotations and RQ decomposition

A 3-dimensional Givens rotation is a rotation about one of the three coordinate axes. The three Givens rotations are

$$Q_x = \begin{bmatrix} 1 & & \\ & c & -s \\ & s & c \end{bmatrix} \quad Q_y = \begin{bmatrix} c & & s \\ & 1 & \\ -s & & c \end{bmatrix} \quad Q_z = \begin{bmatrix} c & -s & \\ s & c & \\ & & 1 \end{bmatrix} \quad (\text{A4.1})$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$ for some angle θ and blank entries represent zeros.

Multiplying a 3×3 matrix A on the right by (for instance) Q_z has the effect of leaving the last column of A unchanged, and replacing the first two columns by linear combinations of the original two columns. The angle θ may be chosen so that any given entry in the first two columns becomes zero.

For instance, to set the entry A_{21} to zero, we need to solve the equation $ca_{21} + sa_{22} = 0$. The solution to this is $c = -a_{22}/(a_{22}^2 + a_{21}^2)^{1/2}$ and $s = a_{21}/(a_{22}^2 + a_{21}^2)^{1/2}$. It is required that $c^2 + s^2 = 1$ since $c = \cos(\theta)$ and $s = \sin(\theta)$, and the values of c and s given here satisfy that requirement.

The strategy of the RQ algorithm is to clear out the lower half of the matrix one entry at a time by multiplication by Givens rotations. Consider the decomposition of a 3×3 matrix A as $A = RQ$ where R is upper-triangular and Q is a rotation matrix. This may take place in three steps. Each step consists of multiplication on the right by a Givens rotation to set a chosen entry of the matrix A to zero. The sequence of multiplications must be chosen in such a way as not to disturb the entries that have already been set to zero. An implementation of the RQ decomposition is given in algorithm A4.1.

Objective

Carry out the RQ decomposition of a 3×3 matrix A using Givens rotations.

Algorithm

- (i) Multiply by Q_x so as to set A_{32} to zero.
- (ii) Multiply by Q_y so as to set A_{31} to zero. This multiplication does not change the second column of A , hence A_{32} remains zero.
- (iii) Multiply by Q_z so as to set A_{21} to zero. The first two columns are replaced by linear combinations of themselves. Thus, A_{31} and A_{32} remain zero.

Algorithm A4.1. *RQ decomposition of a 3×3 matrix.*

Other sequences of Givens rotations may be chosen to give the same result. As a result of these operations, we find that $AQ_xQ_yQ_z = R$ where R is upper-triangular. Consequently, $A = RQ_z^TQ_y^TQ_x^T$, and so $A = RQ$ where $Q = Q_z^TQ_y^TQ_x^T$ is a rotation. In addition, the angles θ_x , θ_y and θ_z associated with the three Givens rotations provide a parametrization of the rotation by three Euler angles, otherwise known as roll, pitch and yaw angles.

It should be clear from this description of the decomposition algorithm how similar QR, QL and LQ factorizations may be carried out. Furthermore, the algorithm is easily generalized to higher dimensions.

A4.1.2 Householder matrices and QR decomposition

For matrices of larger dimension, the QR decomposition is more efficiently carried out using Householder matrices. The symmetric matrix

$$H_v = I - 2vv^T/v^Tv \quad (\text{A4.2})$$

has the property that $H_v^T H_v = I$, and so H_v is orthogonal.

Let e_1 be the vector $(1, 0, \dots, 0)^T$, and let x be any vector. Let $v = x \pm \|x\|e_1$. One easily verifies that $H_v x = \mp \|x\|e_1$; thus H_v is an orthogonal matrix that transforms the vector x to a multiple of e_1 . Geometrically H_v is a reflection in the plane perpendicular to v , and $v = x \pm \|x\|e_1$ is a vector that bisects x and $\pm \|x\|e_1$. Thus reflection in the v direction takes x to $\mp \|x\|e_1$. For reasons of stability, the sign ambiguity in defining v should be resolved by setting

$$v = x + \text{sign}(x_1)\|x\|e_1. \quad (\text{A4.3})$$

If A is a matrix, x is the first column of A , and v is defined by (A4.3), then forming the product $H_v A$ will clear out the first column of the matrix, replacing the first column by $(\|x\|, 0, 0, \dots, 0)^T$. One continues left multiplication by orthogonal Householder matrices to clear out the below-diagonal part of the matrix A . In this way, one finds that eventually $QA = R$, where Q is a product of orthogonal matrices and R is an upper-triangular matrix. Therefore, one has $A = Q^T R$. This is the QR decomposition of the matrix A .

When multiplying by Householder matrices it is inefficient to form the Householder matrix explicitly. Multiplication by a vector a may be carried out most efficiently as

$$H_v a = (I - 2vv^T/v^Tv)a = a - 2v(v^T a)/v^Tv \quad (\text{A4.4})$$

and the same holds for multiplication by a matrix A . For more about Householder matrices and the QR decomposition, the reader is referred to [Golub-89].

Note. In the QR or RQ decomposition, R refers to an upper-triangular matrix and Q refers to an orthogonal matrix. In the notation used elsewhere in this book, R refers usually to a rotation (hence orthogonal) matrix.

A4.2 Symmetric and skew-symmetric matrices

Symmetric and skew-symmetric matrices play an important role in this book. A matrix is called *symmetric* if $A^T = A$ and *skew-symmetric* if $A^T = -A$. The eigenvalue decompositions of these matrices are summarized in the following result.

Result A4.1. Eigenvalue decomposition.

- (i) If A is a real symmetric matrix, then A can be decomposed as $A = UDU^T$, where U is an orthogonal matrix and D is a real diagonal matrix. Thus, a real symmetric matrix has real eigenvalues, and the eigenvectors are orthogonal.
- (ii) If S is real and skew-symmetric, then $S = UBU^T$ where B is a block-diagonal

matrix of the form $\text{diag}(a_1 Z, a_2 Z, \dots, a_m Z, 0, \dots, 0)$, where $Z = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. The eigenvectors of S are all purely imaginary, and a skew-symmetric matrix of odd order is singular.

A proof of this result is given in [Golub-89].

Jacobi's method. In general, eigenvalue extraction from arbitrary matrices is a difficult numerical problem. For real symmetric matrices however, a very stable method exists: Jacobi's method. An implementation of this algorithm is given in [Press-88].

Cross products

Of particular interest are 3×3 skew-symmetric matrices. If $\mathbf{a} = (a_1, a_2, a_3)^T$ is a 3-vector, then one defines a corresponding skew-symmetric matrix as follows:

$$[\mathbf{a}]_{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \quad (\text{A4.5})$$

Note that any skew-symmetric 3×3 matrix may be written in the form $[\mathbf{a}]_{\times}$ for a suitable vector $\mathbf{a}$. Matrix $[\mathbf{a}]_{\times}$ is singular, and $\mathbf{a}$ is its null-vector (right or left). Hence, a 3×3 skew-symmetric matrix is defined up to scale by its null-vector.

The cross product (or vector product) of two 3-vectors $\mathbf{a} \times \mathbf{b}$ (sometimes written $\mathbf{a} \wedge \mathbf{b}$) is the vector $(a_2 b_3 - a_3 b_2, a_3 b_1 - a_1 b_3, a_1 b_2 - a_2 b_1)^T$. The cross product is related to skew-symmetric matrices according to

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_{\times} \mathbf{b} = (\mathbf{a}^T [\mathbf{b}]_{\times})^T. \quad (\text{A4.6})$$

Cofactor and adjoint matrices. Let M be a square matrix. By M^* is meant the matrix of cofactors of M . That is, the (i,j) -th entry of the matrix M^* is equal to $(-1)^{i+j} \det \hat{M}_{ij}$, where $\hat{M}_{ij}$ is the matrix obtained from M by striking out the i -th row and j -th column. The transpose of the cofactor matrix M^* is known as the *adjoint* of M , and denoted $\text{adj}(M)$.

If M is invertible, then it is well known that

$$M^* = \det(M) M^{-T} \quad (\text{A4.7})$$

where M^{-T} is the inverse transpose of M . This formula does not hold for non-invertible matrices, but $\text{adj}(M) M = M \text{adj}(M) = \det(M) \mathbf{I}$ is always valid.

The cofactor matrix is related to the way matrices distribute with respect to the cross product.

Lemma A4.2. If M is any 3×3 matrix (invertible or not), and $\mathbf{x}$ and $\mathbf{y}$ are column vectors, then

$$(M\mathbf{x}) \times (M\mathbf{y}) = M^*(\mathbf{x} \times \mathbf{y}). \quad (\text{A4.8})$$

This equation may be written as $[Mx]_{\times} M = M^*[x]_{\times}$, dropping y which is inessential. Now, putting $t = Mx$ and assuming M is invertible, one obtains a rule for commuting a skew-symmetric matrix $[t]_{\times}$ with any non-singular matrix M . One may write (A4.8) as follows.

Result A4.3. *For any vector t and non-singular matrix M one has*

$$[t]_{\times} M = M^*[M^{-1}t]_{\times} = M^{-T}[M^{-1}t]_{\times} \text{ (up to scale).}$$

Note (see result 9.9(p254)) that $[t]_{\times} M$ is the form of the fundamental matrix for a pair of cameras $P = [I \mid 0]$ and $P' = [M \mid t]$. The formula of result A4.3 is used in deriving alternative forms (9.2–p244) for the fundamental matrix.

A curious property of 3×3 skew-symmetric matrices is that up to scale, $[a]_{\times} = [a]_{\times}[a]_{\times}[a]_{\times}$ (including scale $[a]_{\times}^3 = -\|a\|^2[a]_{\times}$). This is easily verified, since the right hand side is clearly skew-symmetric and its null-space generator is a . The next result follows immediately:

Result A4.4. *If $F = [e']_{\times} M$ is a fundamental matrix (a 3×3 singular matrix), then $[e']_{\times}[e']_{\times} F = F$ (up to scale). Hence one may decompose F as $F = [e']_{\times} M$, where $M = [e']_{\times} F$.*

A4.2.1 Positive-definite symmetric matrices

The special class of real symmetric matrices that have positive real eigenvalues are called *positive-definite* symmetric matrices. We list some of the important properties of a positive-definite symmetric real matrix.

Result A4.5. Positive-definite symmetric real matrix.

- (i) *A symmetric matrix A is positive-definite if and only if $x^T A x > 0$ for any non-zero vector x .*
- (ii) *A positive-definite symmetric matrix A may be uniquely decomposed as $A = K K^T$ where K is an upper-triangular real matrix with positive diagonal entries.*

Proof. The first part of this result follows almost immediately from the decomposition $A = U D U^T$. As for the second part, since A is symmetric and positive-definite, it may be written as $A = U D U^T$ where D is diagonal, real and positive and U is orthogonal. We may take the square root of D , writing $D = E E^T$ where E is diagonal. Then $A = V V^T$ where $V = U E$. The matrix V is not upper-triangular. However, we may apply the RQ-decomposition (section A4.1.1) to write $V = K Q$ where K is upper-triangular and Q is orthogonal. Then $A = V V^T = K Q Q^T K^T = K K^T$. This is the Cholesky factorization of A . One may ensure that the diagonal entries of K are all positive by multiplying K on the right by a diagonal matrix with diagonal entries equal to ± 1 . This will not change the product $K K^T$.

Now, we prove uniqueness of the factorization. Specifically, if K_1 and K_2 are two upper-triangular matrices satisfying $K_1 K_1^T = K_2 K_2^T$ then $K_2^{-1} K_1 = K_2^T K_1^{-T}$. Since the left side of this equation is upper-triangular, and the right side is lower-triangular, they must both in fact be diagonal. Thus $D = K_2^{-1} K_1 = K_2^T K_1^{-T}$. However, $K_2^{-1} K_1$ is the inverse transpose of $K_2^T K_1^{-T}$, and so D is equal to its own inverse transpose, and hence is a diagonal matrix with diagonal entries equal to ± 1 . If both K_1 and K_2 have positive diagonal entries then $D = I$, and $K_1 = K_2$. $\square$

The above proof gives a constructive method for computing the Cholesky factorization. There is, however, a very simple and more efficient direct method for computing the Cholesky factorization. See [Press-88] for an implementation.

A4.3 Representations of rotation matrices

A4.3.1 Rotations in n-dimensions

Given a matrix T , we define e^T to be the sum of the series

$$e^T = I + T + T^2/2! + \dots + T^k/k! + \dots$$

This series converges absolutely for all values of T . Now, we consider powers of a skew-symmetric matrix. According to result A4.1 a skew-symmetric matrix can be written as $S = UBU^T$ where B is block diagonal of the form $B = \text{diag}(a_1 Z, a_2 Z, \dots, a_m Z, 0, \dots, 0)$, matrix U is orthogonal, and $Z^2 = -I_{2 \times 2}$. We observe that the powers of Z are

$$Z^2 = -I; Z^3 = -Z; Z^4 = I$$

and so on. Thus,

$$e^Z = I + Z - I/2! - Z/3! + \dots = \cos(1)I + \sin(1)Z = R_{2 \times 2}(1)$$

Where $R_{2 \times 2}(1)$ means the 2×2 matrix representing a rotation through 1 radian. More generally,

$$e^{aZ} = \cos(a)I + \sin(a)Z = R_{2 \times 2}(a)$$

With $S = UBU^T$ as above, it now follows that

$$e^S = U e^B U^T = U \text{diag}(R(a_1), R(a_2), \dots, R(a_m), 1, \dots, 1) U^T$$

Thus e^S is a rotation matrix. On the other hand, any rotation matrix may be written in the block-diagonal form $U \text{diag}(R(a_1), R(a_2), \dots, R(a_m), 1, \dots, 1) U^T$, and it follows that the matrices e^S where S is an $n \times n$ skew-symmetric are exactly the set of n -dimensional rotation matrices.

A4.3.2 Rotations in 3-dimensions

If t is a 3-vector, then $[t]_\times$ is a skew-symmetric matrix, and any 3×3 skew-symmetric matrix is of this form. Consequently, any 3-dimensional rotation can be written as $e^{[t]_\times}$. We seek to describe the rotation $e^{[t]_\times}$.

Let $[\mathbf{t}]_{\times} = \mathbf{U} \text{diag}(aZ, 0) \mathbf{U}^T$. Then by matching the Frobenius norms of the matrices on both sides, we see that $a = \|\mathbf{t}\|$. Thus,

$$e^{[\mathbf{t}]_{\times}} = \mathbf{U} \text{diag}(R(\|\mathbf{t}\|), 1) \mathbf{U}^T.$$

Thus, $e^{[\mathbf{t}]_{\times}}$ represents a rotation through an angle $\|\mathbf{t}\|$. It is easily verified that $\mathbf{u}_3$, the 3-rd column of $\mathbf{U}$ is the eigenvector of $\mathbf{U} \text{diag}(R, 1) \mathbf{U}^T$ with unit eigenvalue, hence the axis of rotation. However, $[\mathbf{t}]_{\times} \mathbf{u}_3 = \mathbf{U} \text{diag}(aZ, 0) \mathbf{U}^T \mathbf{u}_3 = \mathbf{U} \text{diag}(aZ, 0) (0, 0, 1)^T = 0$. Since $\mathbf{u}_3$ is the generator of the null space of $[\mathbf{t}]_{\times}$, it must be that $\mathbf{u}_3$ is a unit vector in the direction of $\mathbf{t}$. We have shown

Result A4.6. *The matrix $e^{[\mathbf{t}]_{\times}}$ is a rotation matrix representing a rotation through an angle $\|\mathbf{t}\|$ about the axis represented by the vector $\mathbf{t}$.*

This representation of a rotation is called the *angle-axis* representation.

We may write a specific formula for the rotation matrix corresponding to $e^{[\mathbf{t}]_{\times}}$. We observe that $[\mathbf{t}]_{\times}^3 = -\|\mathbf{t}\|^2 [\mathbf{t}]_{\times} = -\|\mathbf{t}\|^3 [\hat{\mathbf{t}}]_{\times}$, where $\hat{\mathbf{t}}$ represents a unit vector in the direction $\mathbf{t}$. Then, with $\text{sinc}(\theta)$ representing $\sin(\theta)/\theta$, we have

$$\begin{aligned} e^{[\mathbf{t}]_{\times}} &= \mathbf{I} + [\mathbf{t}]_{\times} + [\mathbf{t}]_{\times}^2/2! + [\mathbf{t}]_{\times}^3/3! + [\mathbf{t}]_{\times}^4/4! + \dots \\ &= \mathbf{I} + \|\mathbf{t}\| [\hat{\mathbf{t}}]_{\times} + \|\mathbf{t}\|^2 [\hat{\mathbf{t}}]_{\times}^2/2! - \|\mathbf{t}\|^3 [\hat{\mathbf{t}}]_{\times}/3! - \|\mathbf{t}\|^4 [\hat{\mathbf{t}}]_{\times}^2/4! + \dots \\ &= \mathbf{I} + \sin \|\mathbf{t}\| [\hat{\mathbf{t}}]_{\times} + (1 - \cos \|\mathbf{t}\|) [\hat{\mathbf{t}}]_{\times}^2 \\ &= \mathbf{I} + \text{sinc} \|\mathbf{t}\| [\mathbf{t}]_{\times} + \frac{1 - \cos \|\mathbf{t}\|}{\|\mathbf{t}\|^2} [\mathbf{t}]_{\times}^2 \\ &= \cos \|\mathbf{t}\| \mathbf{I} + \text{sinc} \|\mathbf{t}\| [\mathbf{t}]_{\times} + \frac{1 - \cos \|\mathbf{t}\|}{\|\mathbf{t}\|^2} \mathbf{t} \mathbf{t}^T \end{aligned} \quad (\text{A4.9})$$

where the last line follows from the identity $[\mathbf{t}]_{\times}^2 = \mathbf{t} \mathbf{t}^T - \|\mathbf{t}\|^2 \mathbf{I}$.

Some properties of these representations:

- (i) Extraction of the axis and rotation angle from a rotation matrix $\mathbf{R}$ is just a little tricky. The unit rotation axis $\mathbf{v}$ can be found as the eigenvector corresponding to the unit eigenvalue – that is by solving $(\mathbf{R} - \mathbf{I})\mathbf{v} = \mathbf{0}$. Next, it is easily seen from (A4.9) that the rotation angle ϕ satisfies

$$\begin{aligned} 2 \cos(\phi) &= (\text{trace}(\mathbf{R}) - 1) \\ 2 \sin(\phi) \mathbf{v} &= (\mathbf{R}_{32} - \mathbf{R}_{23}, \mathbf{R}_{13} - \mathbf{R}_{31}, \mathbf{R}_{21} - \mathbf{R}_{12})^T. \end{aligned} \quad (\text{A4.10})$$

Writing this second equation as $2 \sin(\phi) \mathbf{v} = \hat{\mathbf{v}}$, we can then compute $2 \sin(\phi) = \mathbf{v}^T \hat{\mathbf{v}}$. Now, the angle ϕ can be computed from $\sin(\phi)$ and $\cos(\phi)$ using a two-argument arctan function (such as the C-language function `atan2(y, x)`).

It has often been written that ϕ can be computed directly from (A4.10) using `arccos` or `arcsin`. However, this method is not numerically accurate, and fails to find the axis when $\phi = \pi$.

- (ii) To apply a rotation $\mathbf{R}(\mathbf{t})$ to some vector $\mathbf{x}$, it is not necessary to construct the

matrix representation of $\mathbf{t}$. In fact

$$\begin{aligned} \mathbf{R}(\mathbf{t})\mathbf{x} &= \left(\mathbf{I} + \text{sinc}(\|\mathbf{t}\|) [\mathbf{t}]_{\times} + \frac{1 - \cos \|\mathbf{t}\|}{\|\mathbf{t}\|^2} [\mathbf{t}]_{\times}^2 \right) \mathbf{x} \\ &= \mathbf{x} + \text{sinc}(\|\mathbf{t}\|) \mathbf{t} \times \mathbf{x} + \frac{1 - \cos \|\mathbf{t}\|}{\|\mathbf{t}\|^2} \mathbf{t} \times (\mathbf{t} \times \mathbf{x}) \quad (\text{A4.11}) \end{aligned}$$

- (iii) If $\mathbf{t}$ is written as $\mathbf{t} = \theta \hat{\mathbf{t}}$, where $\hat{\mathbf{t}}$ is a unit vector in the direction of the axis, and $\theta = \|\mathbf{t}\|$ is the angle of rotation, then (A4.9) is equivalent to the Rodrigues formula for a rotation matrix:

$$\mathbf{R}(\theta, \hat{\mathbf{t}}) = \mathbf{I} + \sin \theta [\hat{\mathbf{t}}]_{\times} + (1 - \cos \theta) [\hat{\mathbf{t}}]_{\times}^2 \quad (\text{A4.12})$$

A4.3.3 Quaternions

Three-dimensional rotations may also be represented by unit quaternions. A unit quaternion is a 4-vector and may be written in the form $\mathbf{q} = (\mathbf{v} \sin(\theta/2), \cos(\theta/2))^T$, as may indeed any unit 4-vector. Such a quaternion represents a rotation about the vector $\mathbf{v}$ through the angle θ . This is a 2-to-1 representation in that both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation. To check this, note that $-\mathbf{q} = (\mathbf{v} \sin(\theta/2 + \pi), \cos(\theta/2 + \pi))^T$, which represents a rotation through $\theta + 2\pi = \theta$.

The relationship between the angle-axis representation of a rotation and the quaternion representation is easily determined. Given a vector $\mathbf{t}$, the angle-axis representation of a rotation, the corresponding quaternion is easily seen to be

$$\mathbf{t} \leftrightarrow \mathbf{q} = (\text{sinc}(\|\mathbf{t}\|/2) \mathbf{t}, \cos(\|\mathbf{t}\|/2))^T$$

A4.4 Singular value decomposition

The singular value decomposition (SVD) is one of the most useful matrix decompositions, particularly for numerical computations. Its most common application is in the solution of over-determined systems of equations.

Given a square matrix $\mathbf{A}$, the SVD is a factorization of $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{D}$ is a diagonal matrix with non-negative entries. Note that it is conventional to write $\mathbf{V}^T$ instead of $\mathbf{V}$ in this decomposition. The decomposition may be carried out in such a way that the diagonal entries of $\mathbf{D}$ are in descending order, and we will assume that this is always done. Thus a circumlocutory phrase such as “the column of $\mathbf{V}$ corresponding to the smallest singular value” is replaced by “the last column of $\mathbf{V}$.”

The SVD also exists for non-square matrices $\mathbf{A}$. Of most interest is the case where $\mathbf{A}$ has more rows than columns. Specifically, let $\mathbf{A}$ be an $m \times n$ matrix with $m \geq n$. In this case $\mathbf{A}$ may be factored as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ where $\mathbf{U}$ is an $m \times m$ matrix with orthogonal columns, $\mathbf{D}$ is an $n \times n$ diagonal matrix and $\mathbf{V}$ is an $n \times n$ orthogonal matrix. The fact that $\mathbf{U}$ has orthogonal columns means that $\mathbf{U}^T\mathbf{U} = \mathbf{I}_{n \times n}$. Furthermore $\mathbf{U}$ has the norm-preserving property that $\|\mathbf{U}\mathbf{x}\| = \|\mathbf{x}\|$ for any vector $\mathbf{x}$, as one readily verifies. On the other hand, $\mathbf{U}\mathbf{U}^T$ is in general not the identity unless $m = n$.

Not surprisingly, one can also define a singular value decomposition for matrices

with more columns than rows, but generally this will not be of interest to us. Instead on the occasional instances in this book where we need to take the singular value decomposition of a matrix A with $m < n$, it is appropriate to extend A by adding rows of zeros to obtain a square matrix, and then take the SVD of this resulting matrix. Usually this will be done without special remark.

Common implementations of the SVD, such as the one in [Press-88], assume that $m \geq n$. Since in this case the matrix U has the same dimension $m \times n$ as the input, matrix A may be overwritten by the output matrix U .

Implementation of the SVD. A description of the singular value decomposition algorithm, or a proof of its existence, is not given in this book. For a description of how the algorithm works, the reader is referred to [Golub-89]. A practical implementation of the SVD is given in [Press-88]. However, the implementation of the SVD given in the first edition of "Numerical Recipes in C" can sometimes give incorrect results. The version of the algorithm given in the second edition [Press-88] of "Numerical Recipes in C" corrects mistakes in the earlier version.

Singular values and eigenvalues. The diagonal entries of matrix D in the SVD are non-negative. These entries are known as the singular values of the matrix A . They are not the same thing as eigenvalues. To see the connection of the singular values of A with eigenvalues, we start with $A = UDV^T$. From this it follows that $A^T A = VDU^T UDV^T = VD^2 V^T$. Since V is orthogonal, $V^T = V^{-1}$, and so $A^T A = VD^2 V^{-1}$. This is the defining equation for eigenvalues, indicating that the entries of D^2 are the eigenvalues of $A^T A$ and the columns of V are the eigenvectors of $A^T A$. In short, the singular values of A are the square-roots of the eigenvalues of $A^T A$.

Note, $A^T A$ is symmetric and positive-semi-definite (see section A4.2.1 above), so its eigenvalues are real and non-negative. Consequently, singular values are real and non-negative.

Computational complexity of the SVD

The computational complexity of the SVD depends on how much information needs to be returned. For instance in algorithm A5.4 to be considered later, the solution to the problem is the last column of the matrix V in the SVD. The matrix U is not used, and does not need to be computed. On the other hand, algorithm A5.1 of section A5.1 requires the complete SVD to be computed. For systems of equations with many more rows than columns, the extra effort required to compute the matrix U is substantial.

Approximate numbers of floating-point operations (flops) required to compute the SVD of an $m \times n$ matrix are given in [Golub-89]. To find matrices U , V and D , a total of $4m^2n + 8mn^2 + 9n^3$ flops are needed. However, if only the matrices V and D are required, then only $4mn^2 + 8n^3$ flops are required. This is an important distinction, since this latter expression does not contain any term in m^2 . Specifically, the number of operations required to compute U varies as the square of m , the number of rows. On the other hand, the complexity of computing D and V is linear in m . For cases where there are many more rows than columns, therefore, it is important (supposing

computation time is an issue) to avoid computing the matrix U unless it is needed. To illustrate this point we consider the DLT algorithm for camera resection, described in chapter 7. In this algorithm a 3×4 camera matrix is computed from a set of n 3D to 2D point matches. The solution involves using algorithm A5.4, to solve a set of equations $A\mathbf{p} = \mathbf{0}$ where A is a $2n \times 12$ matrix. The solution vector $\mathbf{p}$ is the last column of the matrix V in an SVD, $A = UDV^T$. Thus the matrix U is not required. Table A4.1 gives the total number of flops for carrying out the SVD for six (the minimum number), 100 or 1000 point correspondences.

# points	# equations per point	# equations (m)	#unknowns (n)	# operations not computing U	# operations computing U
6	2	12	12	20,736	36,288
100	2	200	12	129,024	2,165,952
1000	2	2000	12	1,165,824	194,319,552

Table A4.1. Comparison of the number of flops required to compute the SVD of a matrix of size $m \times n$, for varying values of m and for $n = 12$. Note that the computational complexity increases sub-linearly in the number of equations when U is not computed. On the other hand, the extra computational burden of computing U is very large, especially for large numbers of equations.

Further reading. Two invaluable text books for this area are [Golub-89] and [Lutkepohl-96].

Appendix 5

Least-squares Minimization

In this appendix we discuss numerical algorithms for solving linear systems of equations under various constraints. As will be seen such problems are conveniently solved using the SVD.

A5.1 Solution of linear equations

Consider a system of equations of the form $Ax = b$. Let A be an $m \times n$ matrix. There are three possibilities:

- (i) If $m < n$ there are more unknowns than equations. In this case, there will not be a unique solution, but rather a vector space of solutions.
- (ii) If $m = n$ there will be a unique solution as long as A is invertible.
- (iii) If $m > n$ then there are more equations than unknowns. In general the system will not have a solution unless by chance b lies in the span of the columns of A .

Least-squares solutions: full-rank case. We consider the case $m \geq n$ and assume for the present that A is known to be of rank n . If a solution does not exist, then in many cases it still makes sense to seek a vector x that is closest to providing a solution to the system $Ax = b$. In other words, we seek x such that $\|Ax - b\|$ is minimized, where $\|\cdot\|$ represents the vector norm. Such an x is known as the *least-squares solution* to the over-determined system. The least-squares solution is conveniently found using the SVD as follows.

We seek x that minimizes $\|Ax - b\| = \|UDV^T x - b\|$. Because of the norm-preserving property of orthogonal transforms, $\|UDV^T x - b\| = \|DV^T x - U^T b\|$, and this is the quantity that we want to minimize. Writing $y = V^T x$ and $b' = U^T b$, the problem becomes one of minimizing $\|Dy - b'\|$ where D is an $m \times n$ matrix with vanishing

off-diagonal entries. This set of equations is of the form

$$\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \\ \hline & & & & 0 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \\ b'_{n+1} \\ \vdots \\ b'_m \end{pmatrix}.$$

Clearly, the nearest Dy can approach to b' is the vector $(b'_1, b'_2, \dots, b'_n, 0, \dots, 0)^T$, and this is achieved by setting $y_i = b'_i/d_i$ for $i = 1, \dots, n$. Note that the assumption $\text{rank} A = n$ ensures that $d_i \neq 0$. Finally, one retrieves x from $x = Vy$. The complete algorithm is

Objective

Find the least-squares solution to the $m \times n$ set of equations $Ax = b$, where $m > n$ and $\text{rank} A = n$.

Algorithm

- (i) Find the SVD $A = UDV^T$.
- (ii) Set $b' = U^T b$.
- (iii) Find the vector y defined by $y_i = b'_i/d_i$, where d_i is the i -th diagonal entry of D .
- (iv) The solution is $x = Vy$.

Algorithm A5.1. *Linear least-squares solution to an over-determined full-rank set of linear equations.*

Deficient-rank systems. Sometimes one is called upon to solve a system of equations that is expected not to be of full column rank. Thus, let $r = \text{rank} A < n$, where n is the number of columns of A . It is possible that because of noise corruption, the matrix A actually has rank greater than r , but we wish to enforce the rank r constraint because of theoretical considerations, derived from the particular problem being considered. In this case, there will be an $(n - r)$ -parameter family of solutions to the set of equations, where $r = \text{rank} A < n$. This family of solutions is appropriately solved using the SVD, as follows:

This algorithm gives an $(n - r)$ -parameter family (parametrized by the indeterminate values λ_i) of least-squares solutions to the deficient-rank system. The justification of this algorithm is similar to that of algorithm A5.1 for the least-squares solution of full-rank systems.

Systems of unknown rank. In most cases encountered in this book, the rank of a system of linear equations will be known theoretically in advance of solution. If the rank of the system of equations is not known, then one must guess at its rank. In this case, it is appropriate to set singular values that are small compared with the largest

Objective

Find the general solution to a set of equations $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is an $m \times n$ matrix of rank $r < n$.

Algorithm

- (i) Find the SVD $\mathbf{A} = \mathbf{UDV}^T$, where the diagonal entries d_i of $\mathbf{D}$ are in descending numerical order.
- (ii) Set $\mathbf{b}' = \mathbf{U}^T \mathbf{b}$.
- (iii) Find the vector $\mathbf{y}$ defined by $y_i = b'_i/d_i$ for $i = 1, \dots, r$, and $y_i = 0$ otherwise.
- (iv) The solution $\mathbf{x}$ of minimum norm $\|\mathbf{x}\|$ is $\mathbf{Vy}$.
- (v) The general solution is $\mathbf{x} = \mathbf{Vy} + \lambda_{r+1} \mathbf{v}_{r+1} + \dots + \lambda_n \mathbf{v}_n$, where $\mathbf{v}_{r+1}, \dots, \mathbf{v}_n$ are the last $n - r$ columns of $\mathbf{V}$.

Algorithm A5.2. General solution to deficient-rank system

singular value to zero. Thus, if $d_i/d_0 < \delta$ where δ is a small constant of the order of the machine precision¹ then one sets $y_i = 0$. A least-squares solution is then given by $\mathbf{x} = \mathbf{Vy}$ as before.

A5.2 The pseudo-inverse

Given a square diagonal matrix $\mathbf{D}$, we define its *pseudo-inverse* to be the diagonal matrix $\mathbf{D}^+$ such that

$$D_{ii}^+ = \begin{cases} 0 & \text{if } D_{ii} = 0 \\ D_{ii}^{-1} & \text{otherwise,} \end{cases}$$

Now, consider an $m \times n$ matrix $\mathbf{A}$ with $m \geq n$. Let the SVD of $\mathbf{A}$ be $\mathbf{A} = \mathbf{UDV}^T$. We define the *pseudo-inverse* of $\mathbf{A}$ to be the matrix

$$\mathbf{A}^+ = \mathbf{VD}^+ \mathbf{U}^T. \quad (\text{A5.1})$$

One very simply verifies that the vector $\mathbf{y}$ in algorithm A5.1 or algorithm A5.2 is nothing more than $\mathbf{D}^+ \mathbf{b}'$ where $\mathbf{b}' = \mathbf{U}^T \mathbf{b}$. Thus,

Result A5.1. *The least-squares solution to an $m \times n$ system of equations $\mathbf{Ax} = \mathbf{b}$ of rank n is given by $\mathbf{x} = \mathbf{A}^+ \mathbf{b}$. In the case of a deficient-rank system, $\mathbf{x} = \mathbf{A}^+ \mathbf{b}$ is the solution that minimizes $\|\mathbf{x}\|$.*

As remarked when discussing the SVD, if $\mathbf{A}$ has fewer rows than columns, then this result may be applied after extending $\mathbf{A}$ to a square matrix by adding rows of zeros.

Symmetric matrices. For symmetric matrices, one may generalize the pseudo-inverse as follows. This generalization was used in chapter 5, section 5.2.3(p142) for discussing singular covariance matrices. If $\mathbf{A}$ is a non-invertible symmetric matrix, and $\mathbf{X}^T \mathbf{A} \mathbf{X}$ is invertible, then we write $\mathbf{A}^{+\mathbf{X}} \stackrel{\text{def}}{=} \mathbf{X}(\mathbf{X}^T \mathbf{A} \mathbf{X})^{-1} \mathbf{X}^T$. One can see that this depends only on the span of the columns of $\mathbf{X}$. In other words, if $\mathbf{X}$ is replaced by $\mathbf{XB}$ for any invertible matrix $\mathbf{B}$, then $\mathbf{A}^{+\mathbf{X}} = \mathbf{A}^{+\mathbf{XB}}$. Otherwise stated, $\mathbf{A}^{+\mathbf{X}}$ depends only on the (left)

¹ Machine precision is the largest floating point value ϵ such that $1.0 + \epsilon = 1.0$.

null-space of X , namely the space of vectors perpendicular to the columns of X . Define the null-space $N_L X = \{x^T \mid x^T X = 0\}$. One finds that under a simple condition, A^{+X} is the pseudo-inverse of A :

Result A5.2. *Let A be a symmetric matrix, then $A^{+X} \stackrel{\text{def}}{=} X(X^T A X)^{-1} X^T = A^+$ if and only if $N_L(X) = N_L(A)$.*

Only a sketch proof is given. The necessity is obvious, since $N_L(X)$ and $N_L(A)$ are the null-spaces of the left and right sides of the equation. To prove the converse, one may assume that the columns of X are orthonormal, since as shown above, only the null-space of X is of importance. Thus, X may be extended by adding further columns X' to form an orthogonal matrix $U = [X|X']$. Now, the rows of X'^T span the null-space of X , and hence, by assumption, of A . Now, the proof is completed in a few lines by comparing the definition $A^{+X} \stackrel{\text{def}}{=} X(X^T A X)^{-1} X^T$ with the definition (A5.1) of the pseudo-inverse.

A5.2.1 Linear least-squares using normal equations

The linear least-squares problem may also be solved by a method involving the so-called *normal equations*. Once more, we consider the set of linear equations $Ax = b$ where A is an $m \times n$ matrix with $m > n$. In general, no solution x will exist for this set of equations. Consequently, the task is to find the vector x that minimizes the norm $\|Ax - b\|$. As the vector x varies over all values, the product Ax varies over the complete column space of A , that is, the subspace of $\mathbb{R}^m$ spanned by the columns of A . The task therefore is to find the closest vector to b that lies in the column space of A , where closeness is defined in terms of vector norm. Let x be the solution to this problem; thus Ax is the closest point to b . In this case, the difference $Ax - b$ must be a vector orthogonal to the column space of A . This means, explicitly, that $Ax - b$ is perpendicular to each of the columns of A , and hence $A^T(Ax - b) = 0$. Multiplying out and separating terms gives an equation

$$(A^T A)x = A^T b. \quad (\text{A5.2})$$

This is a square $n \times n$ set of linear equations, called the *normal equations*. This set of equations may be solved to find the least-squares solution to the problem $Ax = b$. Even if A is not of full rank (rank n), this set of equations should have a solution, since $A^T b$ lies in the column space of $A^T A$. In the case where A has rank n , the matrix $A^T A$ is

Objective

Find x that minimizes $\|Ax - b\|$.

Algorithm

- (i) Solve the normal equations $A^T A x = A^T b$.
- (ii) If $A^T A$ is invertible, then the solution is $x = (A^T A)^{-1} A^T b$.

Algorithm A5.3. *Linear least-squares using the normal equations.*

invertible, and so $\mathbf{x}$ may be found by $\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$. Since $\mathbf{x} = \mathbf{A}^+ \mathbf{b}$, this implies the following result, which is also easily verified directly:

Result A5.3. *If $\mathbf{A}$ is an $m \times n$ matrix of rank n , then $\mathbf{A}^+ = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$.*

This result is useful in theoretical analysis, as well as being a computationally simpler method than using the SVD to compute a pseudo-inverse if n is small compared with m (so that computing the inverse of $(\mathbf{A}^T \mathbf{A})$ is inexpensive compared to computing the SVD of $\mathbf{A}$).

Vector space norms. One sometimes wishes to minimize $\mathbf{Ax} - \mathbf{b}$ with respect to a different norm on the vector space $\mathbb{R}^n$. The usual norm in a vector space $\mathbb{R}^n$ is given in terms of the usual inner product. Thus, for two vectors $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{R}^n$ one may define the inner product $\mathbf{a} \cdot \mathbf{b}$ to be $\mathbf{a}^T \mathbf{b}$. The norm of a vector $\mathbf{a}$ is then $\|\mathbf{a}\| = (\mathbf{a} \cdot \mathbf{a})^{1/2} = (\mathbf{a}^T \mathbf{a})^{1/2}$. One notes the properties:

- (i) The inner product is a symmetric bilinear form on $\mathbb{R}^n$.
- (ii) $\|\mathbf{a}\| > 0$ for all non-zero vectors $\mathbf{a} \in \mathbb{R}^n$.

We say that the inner product is a positive-definite symmetric bilinear form. It is possible to define other inner products on a vector space $\mathbb{R}^n$. Let $\mathbf{C}$ be a real symmetric positive-definite matrix, and define a new inner product $\mathbf{C}(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{C} \mathbf{b}$. The symmetry of the inner product follows from the symmetry of $\mathbf{C}$. A norm may be defined by $\|\mathbf{a}\|_{\mathbf{C}} = (\mathbf{a}^T \mathbf{C} \mathbf{a})^{1/2}$, and this is defined and positive-definite, because $\mathbf{C}$ is assumed to be a positive-definite matrix.

Weighted linear least-squares problems. Sometimes one desires to solve a weighted least-squares problem of the form $\mathbf{Ax} - \mathbf{b} = 0$ by minimizing the $\mathbf{C}$ -norm $\|\mathbf{Ax} - \mathbf{b}\|_{\mathbf{C}}$ of the error. Here $\mathbf{C}$ is a positive-definite symmetric matrix defining an inner product and a norm $\|\cdot\|_{\mathbf{C}}$ on $\mathbb{R}^n$. As before, one can argue that the minimum error vector $\mathbf{Ax} - \mathbf{b}$ must be orthogonal in the inner product defined by $\mathbf{C}$ to the column space of $\mathbf{A}$. This leads to a requirement $\mathbf{A}^T \mathbf{C}(\mathbf{Ax} - \mathbf{b}) = 0$. Rearranging this one obtains the weighted normal equations:

$$(\mathbf{A}^T \mathbf{C} \mathbf{A}) \mathbf{x} = \mathbf{A}^T \mathbf{C} \mathbf{b}. \quad (\text{A5.3})$$

The most common weighting will be where $\mathbf{C}$ is a diagonal matrix, corresponding to independent weights in each of the axial directions in $\mathbb{R}^n$. However, general weighting matrices $\mathbf{C}$ may be used also.

A5.3 Least-squares solution of homogeneous equations

Similar to the previous problem is that of solving a set of equations of the form $\mathbf{Ax} = 0$. This problem comes up frequently in reconstruction problems. We consider the case where there are more equations than unknowns – an over-determined set of equations. The obvious solution $\mathbf{x} = 0$ is not of interest – we seek a non-zero solution to the set of equations. Observe that if $\mathbf{x}$ is a solution to this set of equations, then so is $k\mathbf{x}$ for any scalar k . A reasonable constraint would be to seek a solution for which $\|\mathbf{x}\| = 1$.

In general, such a set of equations will not have an exact solution. Suppose A has dimension $m \times n$ then there is an exact solution if and only if $\text{rank}(A) < n$ – the matrix A does not have full column rank. In the absence of an exact solution we will normally seek a least-squares solution. The problem may be stated as

- Find the x that minimizes $\|Ax\|$ subject to $\|x\| = 1$.

This problem is solvable as follows. Let $A = UDV^T$. The problem then requires us to minimize $\|UDV^T x\|$. However, $\|UDV^T x\| = \|DV^T x\|$ and $\|x\| = \|V^T x\|$. Thus, we need to minimize $\|DV^T x\|$ subject to the condition $\|V^T x\| = 1$. We write $y = V^T x$, and the problem is: minimize $\|Dy\|$ subject to $\|y\| = 1$. Now, D is a diagonal matrix with its diagonal entries in descending order. It follows that the solution to this problem is $y = (0, 0, \dots, 0, 1)^T$ having one non-zero entry, 1 in the last position. Finally $x = Vy$ is simply the last column of V . The method is summarized in algorithm A5.4.

Objective

Given a matrix A with at least as many rows as columns, find x that minimizes $\|Ax\|$ subject to $\|x\| = 1$.

Solution

x is the last column of V , where $A = UDV^T$ is the SVD of A .

Algorithm A5.4. *Least-squares solution of a homogeneous system of linear equations.*

As mentioned in section A4.4 the last column of V may alternatively be described as the eigenvector of $A^T A$ corresponding to the smallest eigenvalue.

A5.4 Least-squares solution to constrained systems

In the previous section, we considered a method of least-squares solution of equations of the form $Ax = 0$. Such problems may arise from situations where measurements are made on a set of image features. With exact measurements and an exact imaging model the mathematical model predicts an exact solution to this system. In the case of inexact image measurements, or noise, there will not be an exact solution. In this case, it makes sense to find a least-squares solution.

On other occasions, however, some of the equations represented by rows of the matrix A are derived from precise mathematical constraints, and should be satisfied exactly. This set of constraints may be described by a matrix equation $Cx = 0$, which should be satisfied exactly. Others of the equations are derived from image measurements and are subject to noise. This leads to a problem of the following sort:

- Find the x that minimizes $\|Ax\|$ subject to $\|x\| = 1$ and $Cx = 0$.

This problem can be solved in the following manner. The condition that x satisfies $Cx = 0$ means that x lies perpendicular to each of the rows of C . The set of all such x is a vector space called the orthogonal complement of the row space of C . We wish to find this orthogonal complement.

First, if C has fewer rows than columns, then extend it to a square matrix by adding rows of zero elements. This has no effect on the set of constraints $Cx = 0$. Now, let $C = UDV^T$ be the Singular Value Decomposition of C , where D is a diagonal matrix with r non-zero diagonal entries. In this case, C has rank r and the row-space of C is generated by the first r rows of V^T . The orthogonal complement of the row-space of C consists of the remaining rows of V^T . Define $C^\perp$ to be the matrix V with the first r columns deleted. Then $CC^\perp = 0$, and so the set of vectors x satisfying $Cx = 0$ is spanned by the columns of $C^\perp$ and we may write any such x as $x = C^\perp x'$ for suitable x' . Since $C^\perp$ has orthogonal columns, one observes that $\|x\| = \|C^\perp x'\| = \|x'\|$. The minimization problem now becomes

- Find the x' that minimizes $\|AC^\perp x'\|$ subject to $\|x'\| = 1$.

This is simply an instance of the problem discussed in section A5.3, solved by algorithm A5.4. The complete algorithm for solution of the constrained minimization problem is given as algorithm A5.5.

Objective

Given an $m \times n$ matrix A with $m \geq n$, find the vector x that minimizes $\|Ax\|$ subject to $\|x\| = 1$ and $Cx = 0$.

Algorithm

- If C has fewer rows than columns, then add zero-filled rows to C to make it square. Compute the SVD $C = UDV^T$ where diagonal entries of D are sorted with non-zero ones first. Let $C^\perp$ be the matrix obtained from V by deleting the first r columns of V , where r is the number of non-zero entries in D (the rank of C).
- Find the solution to the minimization problem $AC^\perp x' = 0$ using algorithm A5.4. The solution is given by $x = C^\perp x'$.

Algorithm A5.5. Algorithm for constrained minimization.

A5.4.1 More constrained minimization

A further constrained minimization problem arises in the algebraic estimation method used frequently throughout this book – for instance, for computation of the fundamental matrix (section 11.3(p282)) or the trifocal tensor (section 16.3(p395)).

The problem is:

- Minimize $\|Ax\|$ subject to $\|x\| = 1$ and $x = G\tilde{x}$ for a given matrix G and some unknown vector $\tilde{x}$.

Note that this is very similar to the previous minimization problem of section A5.4, which was reduced to the form of the present problem in which the matrix G had orthonormal columns. The condition that $x = G\tilde{x}$ for some $\tilde{x}$ means nothing more than that x lies in the span of the columns of G . Thus, to solve the present problem using algorithm A5.5, we need only to replace G by a matrix with the same column space (i.e. the space spanned by the columns), but with orthonormal columns. If $G = UDV^T$ where D has r non-zero entries (that is G has rank r), then let U' be the matrix consisting of the

first r columns of U . Then G and U' have the same column space. As in section A5.4 the solution is found by setting x' to be the unit vector that minimizes $\|AU'x'\|$, then setting $x = U'x'$.

If $\hat{x}$ is also required, then it may be obtained by solving $G\hat{x} = x = U'x'$. The solution is expressed in terms of the pseudo-inverse (section A5.2) as $\hat{x} = G^+x = G^+U'x'$; it may not be unique if G does not have full column rank. Since $G^+ = VD^+U^T$, we may write $\hat{x} = VD^+U^T U'x'$, which simplifies to $\hat{x} = V'D'^{-1}x'$ where V' consists of the first r columns of V and D' is the upper $r \times r$ block of D .

The complete method is summarized in algorithm A5.6.

Objective

Find the vector x that minimizes $\|Ax\|$ subject to the conditions $\|x\| = 1$ and $x = G\hat{x}$, where G has rank r .

Algorithm

- (i) Compute the SVD $G = UDV^T$, where the non-zero values of D appear first down the diagonal.
- (ii) Let U' be the matrix comprising the first r columns of U .
- (iii) Find the unit vector x' that minimizes $\|AU'x'\|$, using algorithm A5.4.
- (iv) The required solution is $x = U'x'$.
- (v) If desired, one may compute $\hat{x}$ as $V'D'^{-1}x'$, where V' consists of the first r columns of V and D' is the upper $r \times r$ block of D .

Algorithm A5.6. *Algorithm for constrained minimization, subject to a span-space constraint.*

A5.4.2 Yet another minimization problem

A very similar problem is

- Minimize $\|Ax\|$ subject to a condition $\|Cx\| = 1$.

This problem comes up for instance in the solution to the DLT camera calibration problem (section 7.3(p184)). In general it will be the case that $\text{rank } C < n$ where n is the dimension of the vector x . Geometrically the problem may be thought of as finding the "lowest" point on a quadratic surface (specified by $x^T A^T A x$), with the constraint that the point must lie on the (inhomogeneous) "conic" $x^T C^T C x = 1$.

We start by taking the SVD of the matrix C , obtaining $C = UDV^T$. The condition $\|UDV^T x\| = 1$ is equivalent to $\|DV^T x\| = 1$, and it is not necessary to compute U explicitly. Then writing $x' = V^T x$, the problem becomes: minimize $\|AVx'\|$ subject to the condition $\|Dx'\| = 1$. Writing $A' = AV$, this becomes: minimize $\|A'x'\|$ subject to $\|Dx'\| = 1$. Thus we have reduced to the case where the constraint matrix is a diagonal matrix, D .

We suppose that D has r non-zero diagonal entries, and s zero entries, where $r + s = n$, the non-zero entries appearing first on the diagonal of D . Then the entries x'_i of x' for $i > r$ do not affect the value of $\|Dx'\|$, since the corresponding diagonal entries of D are zero. Then, for a specific choice of the x'_i for $i = 1, \dots, r$, the other entries x'_i ,

$i = r + 1, \dots, n$ should be chosen so as to minimize the value of $\|A'x'_i\|$. We write $A' = [A'_1 \mid A'_2]$ where A'_1 consists of the first r columns of A' and A'_2 consists of the remaining s columns. Similarly, let x'_1 be an r -vector consisting of the first r elements of x' , and let x'_2 consist of the remaining s elements of x' . Further, let D_1 be the $r \times r$ diagonal matrix consisting of the first r diagonal entries of D . Then $A'x' = A'_1x'_1 + A'_2x'_2$, and the minimization problem is to minimize

$$\|A'_1x'_1 + A'_2x'_2\| \quad (\text{A5.4})$$

subject to the condition $\|D_1x'_1\| = 1$. Now, temporarily fixing x'_1 , (A5.4) takes the form of a least-squares minimization problem of the type discussed in section A5.1. According to result A5.1, the value of x'_2 that minimizes (A5.4) is $x'_2 = -A_2'^+ A_1' x'_1$. Substituting this in (A5.4) gives $\|(A_2' A_2'^+ - I) A_1' x'_1\|$, which we are required to minimize, subject to the condition $\|D_1 x'_1\| = 1$. Finally, writing $x'' = D_1 x'_1$, the problem reduces at last to a problem of the form of the familiar minimization problem of algorithm A5.4.

- Minimize $\|(A_2' A_2'^+ - I) A_1' D_1^{-1} x''\|$, subject to $\|x''\| = 1$.

We now summarize the algorithm.

Objective

Minimize $\|Ax\|$ subject to $\|Cx\| = 1$.

Algorithm

- Compute the SVD $C = UDV^T$, and write $A' = AV$.
- Suppose $\text{rank} D = r$ and let $A' = [A'_1 \mid A'_2]$ where A'_1 consists of the first r columns of A' , and A'_2 is formed from the remaining columns.
- Let D_1 be the upper $r \times r$ minor of D .
- Compute $A'' = (A_2' A_2'^+ - I) A_1' D_1^{-1}$. This is an $n \times r$ matrix.
- Minimize $\|A''x'\|$ subject to $\|x''\| = 1$ using algorithm A5.4.
- Set $x'_1 = D_1^{-1} x''$, and $x'_2 = -A_2'^+ A_1' x'_1$. Let $x' = \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix}$.
- The solution is given by $x = Vx'$.

Algorithm A5.7. Least-squares solution of homogeneous equations subject to the constraint $\|Cx\| = 1$.

Appendix 6

Iterative Estimation Methods

In this appendix we describe the various components involved in building an efficient and robust iterative estimation algorithm.

We start with two of the most common iterative parameter minimization methods, namely Newton iteration (and the closely related Gauss-Newton method) and Levenberg–Marquardt iteration. The general idea of Newton iteration is familiar to most students of numerical methods as a way of finding the zeros of a function of a single variable. Its generalization to several variables and application to finding least-squares solutions rather than exact solutions to sets of equations is relatively straightforward. The Levenberg–Marquardt method is a simple variation on Newton iteration designed to provide faster convergence and regularization in the case of over-parametrized problems. It may be seen as a hybrid between Newton iteration and a gradient descent method.

For the type of problem considered in this book, important reductions of computational complexity are obtained by dividing the set of parameters into two parts. The two parts generally consist of a set of parameters representing camera matrices or homographies, and a set of parameters representing points. This leads to a sparse structure to the problem that is described starting at section A6.3.

We discuss two further implementation issues – the choice of cost function, with respect to their robustness to outliers and convexity (section A6.8); and the parametrization of rotations, and homogeneous and constrained vectors (section A6.9). Finally, those readers who want to learn more about iterative techniques and bundle-adjustment are referred to [Triggs-00a] for more details.

A6.1 Newton iteration

Suppose we are given a hypothesized functional relation $\mathbf{X} = \mathbf{f}(\mathbf{P})$ where $\mathbf{X}$ is a *measurement vector* and $\mathbf{P}$ is a *parameter vector* in Euclidean spaces $\mathbb{R}^N$ and $\mathbb{R}^M$ respectively. A measured value of $\mathbf{X}$ approximating the true value $\bar{\mathbf{X}}$ is provided, and we wish to find the vector $\hat{\mathbf{P}}$ that most nearly satisfies this functional relation. More precisely, we seek the vector $\hat{\mathbf{P}}$ satisfying $\mathbf{X} = \mathbf{f}(\hat{\mathbf{P}}) - \epsilon$ for which $\|\epsilon\|$ is minimized. Note that the linear least-squares problem considered in section A5.1 is exactly of this type, the function $\mathbf{f}$ being defined as a linear function $\mathbf{f}(\mathbf{P}) = \mathbf{A}\mathbf{P}$.

To solve the case where f is not a linear function, we may start with an initial estimated value P_0 , and proceed to refine the estimate under the assumption that the function f is locally linear. Let ϵ_0 be defined by $\epsilon_0 = f(P_0) - X$. We assume that the function f is approximated at P_0 by $f(P_0 + \Delta) = f(P_0) + J\Delta$, where J is the linear mapping represented by the Jacobian matrix $J = \partial f / \partial P$. We seek a point $f(P_1)$, with $P_1 = P_0 + \Delta$, which minimizes $f(P_1) - X = f(P_0) + J\Delta - X = \epsilon_0 + J\Delta$. Thus, it is required to minimize $\|\epsilon_0 + J\Delta\|$ over Δ , which is a linear minimization problem. The vector Δ is obtained by solving the *normal* equations (see (A5.2))

$$J^T J \Delta = -J^T \epsilon_0 \quad (\text{A6.1})$$

or by using the pseudo-inverse $\Delta = -J^+ \epsilon_0$. Thus, the solution vector $\hat{P}$ is obtained by starting with an estimate P_0 and computing successive approximations according to the formula

$$P_{i+1} = P_i + \Delta_i$$

where Δ_i is the solution to the linear least-squares problem

$$J \Delta_i = -\epsilon_i.$$

Matrix J is the Jacobian $\partial f / \partial P$ evaluated at P_i and $\epsilon_i = f(P_i) - X$. One hopes that this algorithm will converge to the required least-squares solution $\hat{P}$. Unfortunately, it is possible that this iteration procedure converges to a local minimum value, or does not converge at all. The behaviour of the iteration algorithm depends very strongly on the initial estimate P_0 .

Weighted iteration. As an alternative to all the dependent variables being equally weighted it is possible to provide a weight matrix specifying the weights of the dependent variables X . To be more precise, one may assume that the measurement X satisfies a Gaussian distribution with covariance matrix Σ_X , and one wishes to minimize the Mahalanobis distance $\|f(\hat{P}) - X\|_\Sigma$. This covariance matrix may be diagonal, specifying that the individual coordinates of X are independent, or more generally it may be an arbitrary symmetric and positive definite matrix. In this case, the normal equations become $J^T \Sigma^{-1} J \Delta_i = -J^T \Sigma^{-1} \epsilon_i$. The rest of the algorithm remains unchanged.

Newton's method and the Hessian. We pass now to consideration of finding minima of functions of many variables. For the present, we consider an arbitrary scalar-valued function $g(P)$ where P is a vector. The optimization problem is simply to minimize $g(P)$ over all values of P . We make two assumptions: that $g(P)$ has a well-defined minimum value, and that we know a point P_0 reasonably close to this minimum.

We may expand $g(P)$ about P_0 in a Taylor series to get

$$g(P_0 + \Delta) = g + g_P \Delta + \Delta^T g_{PP} \Delta / 2 + \dots$$

where subscript P denotes differentiation, and the right hand side is evaluated at P_0 .

We wish to minimize this quantity with respect to Δ . To this end, we differentiate with respect to Δ and set the derivative to zero, arriving at the equation $g_P + g_{PP}\Delta = 0$ or

$$g_{PP}\Delta = -g_P. \quad (\text{A6.2})$$

In this equation, g_{PP} is the matrix of second derivatives, the *Hessian* of g , the (i, j) -th entry of which is $\partial^2 g / \partial p_i \partial p_j$, and p_i and p_j are the i -th and j -th parameters. Vector g_P is the gradient of g . The method of *Newton iteration* consists in starting at an initial value of the parameters, P_0 , and iteratively computing parameter increments Δ using (A6.2) until convergence occurs.

Now we turn to the sort of cost function that arises in the least-squares minimization problem considered above. Specifically, $g(P)$ is the squared norm of an error function

$$g(P) = \frac{1}{2} \|\epsilon(P)\|^2 = \epsilon(P)^T \epsilon(P) / 2$$

where $\epsilon(P)$ is a vector-valued function of the parameter vector P . In particular, $\epsilon(P) = f(P) - X$. The factor $1/2$ is present for simplifying the succeeding computations.

The gradient vector g_P is easily computed to be $\epsilon_P^T \epsilon$. However, we may write $\epsilon_P = f_P = J$ using the notation introduced previously. In short $g_P = J^T \epsilon$. Differentiating $g_P = \epsilon_P^T \epsilon$ a second time, we compute the following formula for the Hessian.¹

$$g_{PP} = \epsilon_P^T \epsilon_P + \epsilon_{PP}^T \epsilon. \quad (\text{A6.3})$$

Now, under the assumption that $f(P)$ is linear, the second term on the right vanishes, leaving $g_{PP} = \epsilon_P^T \epsilon_P = J^T J$. Now, substituting for the gradient and Hessian in (A6.2) yields $J^T J \Delta = -J^T \epsilon$ which is nothing more than the normal equations (A6.1). Thus we have arrived at the same iterative procedure as previously solving the parameter estimation problem, under the assumption that $J^T J = \epsilon_P^T \epsilon_P$ is a reasonable approximation for the Hessian of the function $g(P)$. This procedure in which $J^T J$ is used as an approximation for the Hessian is known as the *Gauss-Newton* method.

Gradient descent. The negative (or down-hill) gradient vector $-g_P = -\epsilon_P^T \epsilon$ defines the direction of most rapid decrease of the cost function. A strategy for minimization of g is to move iteratively in the gradient direction. This is known as *gradient descent*. The length of the step may be computed by carrying out a line search for the function minimum in the negative gradient direction. In this case, the parameter increment Δ is computed from the equation $\lambda \Delta = -g_P$, where λ controls the length of the step.

We may consider this as related to Newton iteration as expressed by the update equation (A6.2) in which the Hessian is approximated (somewhat arbitrarily) by the scalar matrix λI . Gradient descent by itself is not a very good minimization strategy, typically characterized by slow convergence due to zig-zagging. (See [Press-88] for a closer analysis.) It will be seen in the next section, however, that it can be useful in conjunction with Gauss-Newton iteration as a way of getting out of tight corners. The

¹ The last term in this formula needs some clarification. Since ϵ is a vector, ϵ_{PP} is a 3-dimensional array (a tensor). The sum implied by the product $\epsilon_{PP}^T \epsilon$ is over the components of ϵ . It may be written more precisely as $\sum_i (\epsilon_i)_{PP} \epsilon_i$ where ϵ_i is the i -th components of the vector ϵ , and $(\epsilon_i)_{PP}$ is its Hessian.

Levenberg–Marquardt method is essentially a Gauss-Newton method that transitions smoothly to gradient descent when the Gauss-Newton updates fail.

To summarize, we have so far considered three methods of minimization of a cost function $g(\mathbf{P}) = \|\epsilon(\mathbf{P})\|^2/2$:

- (i) **Newton.** Update equation:

$$g_{PP}\Delta = -g_P$$

where $g_{PP} = \epsilon_P^T \epsilon_P + \epsilon_{PP}^T \epsilon$ and $g_P = \epsilon_P^T \epsilon$. Newton iteration is based on the assumption of an approximately quadratic cost function near the minimum, and will show rapid convergence if this condition is met. The disadvantage of this approach is that the computation of the Hessian may be difficult. In addition, far from the minimum the assumption of quadratic behaviour is probably invalid, so a lot of extra work is done with little benefit.

- (ii) **Gauss-Newton.** Update equation:

$$\epsilon_P^T \epsilon_P \Delta = -\epsilon_P^T \epsilon$$

This is equivalent to Newton iteration in which the Hessian is approximated by $\epsilon_P^T \epsilon_P$. Generally this is a good approximation, particularly close to a minimum, or when ϵ is nearly linear in $\mathbf{P}$.

- (iii) **Gradient descent.** Update equation:

$$\lambda \Delta = -\epsilon_P^T \epsilon = -g_P.$$

The Hessian in Newton iteration is replaced by a multiple of the identity matrix. Each update is in the direction of most rapid local decrease of the function value. The value of λ may be chosen adaptively, or by a line search in the downward gradient direction. Generally, gradient descent by itself is not recommended, but in conjunction with Gauss-Newton it yields the commonly used Levenberg–Marquardt method.

A6.2 Levenberg–Marquardt iteration

The Levenberg–Marquardt (abbreviated LM) iteration method is a slight variation on the Gauss-Newton iteration method. The normal equations $J^T J \Delta = -J^T \epsilon$ are replaced by the *augmented normal equations* $(J^T J + \lambda I) \Delta = -J^T \epsilon$, for some value of λ that varies from iteration to iteration. Here I is the identity matrix. A typical initial value of λ is 10^{-3} times the average of the diagonal elements of $N = J^T J$.

If the value of Δ obtained by solving the augmented normal equations leads to a reduction in the error, then the increment is accepted and λ is divided by a factor (typically 10) before the next iteration. On the other hand if the value Δ leads to an increased error, then λ is multiplied by the same factor and the augmented normal equations are solved again, this process continuing until a value of Δ is found that gives rise to a decreased error. This process of repeatedly solving the augmented normal equations for different values of λ until an acceptable Δ is found constitutes one iteration of the LM algorithm. An implementation of the LM algorithm is given in [Press-88].

Justification of LM. To understand the reasoning behind this method, consider what happens for different values of λ . When λ is very small, the method is essentially the same as Gauss-Newton iteration. If the error function $\|\epsilon\|^2 = \|\mathbf{f}(\mathbf{P}) - \mathbf{x}\|^2$ is close to being quadratic in $\mathbf{P}$, then this method will converge fast to the minimum value. On the other hand when λ is large then the normal equation matrix is approximated by $\lambda\mathbf{I}$, and the normal equations become $\lambda\Delta = -\mathbf{J}^T\epsilon$. Recalling that $\mathbf{J}^T\epsilon$ is simply the gradient vector of $\|\epsilon\|^2$, we see that the direction of the parameter increment Δ approaches that given by gradient descent. Thus, the LM algorithm moves seamlessly between Gauss-Newton iteration, which will cause rapid convergence in the neighbourhood of the solution, and a gradient descent approach, which will guarantee a decrease in the cost function when the going is difficult. Indeed, as λ becomes larger and larger, the length of the increment step Δ decreases and eventually it will lead to a decrease of the cost function $\|\epsilon\|^2$.

To demonstrate that the parameter increment Δ obtained by solving the augmented normal equations for all values of λ is in the direction of decreasing cost, we will show that the inner product of Δ and the negative gradient direction for the function $g(\mathbf{P}) = \|\epsilon(\mathbf{P})\|^2$ is positive. This results from the following computation.

$$\begin{aligned} -g_{\mathbf{P}} \cdot \Delta &= -g_{\mathbf{P}}^T \Delta \\ &= (\mathbf{J}^T \epsilon)^T (\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I})^{-1} \mathbf{J}^T \epsilon \end{aligned}$$

However, $\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}$ is positive-definite for any value of λ , and so therefore is its inverse. By definition, this means that $(\mathbf{J}^T \epsilon)^T (\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I})^{-1} \mathbf{J}^T \epsilon$ is positive, unless $\mathbf{J}^T \epsilon$ is zero. Thus, the increment Δ is in a direction of locally decreasing cost, unless of course the gradient $\mathbf{J}^T \epsilon$ is zero.

A different augmentation. In some implementations of Levenberg-Marquardt, notably that given in [Press-88], a different method of augmenting the normal equations is used. The augmented normal equation matrix $\mathbf{N}'$ is defined in terms of the matrix $\mathbf{N} = \mathbf{J}^T \mathbf{J}$ by $N'_{ii} = (1 + \lambda)N_{ii}$ and $N'_{ij} = N_{ij}$ for $i \neq j$. Thus the diagonal of $\mathbf{N}$ is augmented by a multiplicative factor $(1 + \lambda)$ instead of an additive factor. As before, a small value of λ , results essentially in a Gauss-Newton update. For large λ , the off-diagonal entries of the normal equation matrix become insignificant with respect to the diagonal ones.

The i -th diagonal entry of $\mathbf{N}'$ is simply $(1 + \lambda)\mathbf{J}_i^T \mathbf{J}_i$ where $\mathbf{J}_i = \partial \mathbf{f} / \partial p_i$, and p_i is the i -th parameter. The update equation is then $(1 + \lambda)\mathbf{J}_i^T \mathbf{J}_i \delta_i = \mathbf{J}_i^T \epsilon$ where δ_i is the increment in the i -th parameter. Apart from the factor $(1 + \lambda)$, this is the normal equation that would result from minimizing the cost by varying only the i -th parameter δ_i . Thus, in the limit as λ becomes large, the increment to the parameters is the direction that would result from minimizing each one separately.

With this sort of augmentation, the parameter increments for large λ are not the same as for gradient descent. Nevertheless, the same analysis as before shows that the resulting increment is still in the down-hill direction for any value of λ .

One small problem may arise: if some parameter p_i has no effect on the value of the function $\mathbf{f}$, then $\mathbf{J}_i = \partial \mathbf{f} / \partial p_i$ is zero, and the i -th diagonal entry of $\mathbf{N}$ and hence $\mathbf{N}'$

is zero. The augmented normal equation matrix N' is then singular, which can cause trouble. In practice, this is a rare occurrence, but it can occur.

Implementation of LM. To carry out Levenberg–Marquardt minimization in its simplest form it is necessary only to provide a routine to compute the function being minimized, a goal vector $\hat{X}$ of observed or desired values of the function, and an initial estimate P_0 . Computation of the Jacobian matrix J may be carried out either numerically, or by providing a custom routine.

Numerical differentiation may be carried out as follows. Each independent variable x_i is incremented in turn to $x_i + \delta$, the resulting function value is computed using the routine provided for computing f and the derivative is computed as a ratio. Good results have been found by setting the value δ to the maximum of $|10^{-4} \times x_i|$ and 10^{-6} . This choice seemingly gives a good approximation to the derivative. In practice, there seems to be little disadvantage in using numerical differentiation, though for simple functions f one may prefer to provide a routine to compute J , partly for aesthetic reasons, partly because of a possible slightly improved convergence and partly for speed.

A6.3 A sparse Levenberg–Marquardt algorithm

The LM algorithm as described above in section A6.2 is quite suitable for minimization with respect to a small number of parameters. Thus, in the case of 2D homography estimation (see chapter 4), for the simple cost functions (4.6–p94) and (4.7–p95) which are minimized with respect only to the entries of the homography matrix H the LM algorithm works well. However, for minimizing cost functions with respect to large numbers of parameters, the bare LM algorithm is not very suitable. This is because the central step of the LM algorithm, the solution of the normal equations (A5.2), has complexity N^3 in the number of parameters, and this step is repeated many times. However, in solving many estimation problems of the type considered in this book, the normal equation matrix has a certain sparse block structure that one may take advantage of to realize very great time savings.

An example of the situation in which this method is useful is the problem of estimating a 2D homography between two views assuming errors in both images, by minimizing the cost function (4.8–p95). This problem may be parametrized in terms of a set of parameters characterizing the 2D homography (perhaps the 9 entries of the homography matrix), plus parameters for each of the n points in the first view, a total of $2n + 9$ parameters.

Another instance where these methods are useful is the reconstruction problem in which one has image correspondences across two or more (let us say m) views and one wishes to estimate the camera parameters of all the cameras and also the 3D positions of all the points. One can assume arbitrary projective cameras or fully or partially calibrated cameras. Furthermore, to remove some of the incidental degrees of freedom one can fix one of the cameras. For instance in the projective reconstruction problem, the problem may be parametrized by the entries of all the camera matrices (a total of $12m$ or $11m$ parameters depending on how the cameras are parametrized), plus a set of $3n$ parameters for the coordinates of the 3D points.

The sparse LM algorithm is often perceived as complex and difficult to implement. To help overcome this, the algorithms are given cook-book style. Given a suitable library of standard matrix manipulation routines, a reader should be able to implement the algorithm without difficulty.

- **Notation:** If $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ are vectors, then the vector obtained by placing them one after the other in a single column vector is denoted by $(\mathbf{a}_1^T, \mathbf{a}_2^T, \dots, \mathbf{a}_n^T)^T$. A similar notation is used for matrices.

A6.3.1 Partitioning the parameters in the LM method

The sparse LM method will be described primarily in terms of the reconstruction problem, since this is the archetypal problem to which this method relates. The estimation problem will be treated in general terms first of all, since this will illuminate the general approach without excessive detail. At this level of abstraction, the benefits of this approach may not be apparent, but they will become clearer in section A6.3.3. We start with the simple observation that the set of parameters in this problem may be divided up into two sets: a set of parameters describing the cameras, and a set of parameters describing the points. More formally, the “parameter vector” $\mathbf{P} \in \mathbb{R}^M$ may be partitioned into parameter vectors $\mathbf{a}$ and $\mathbf{b}$ so that $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}^T)^T$. We are given a “measurement vector”, denoted by $\mathbf{X}$ in a space $\mathbb{R}^N$. In the reconstruction problem, this consists of the vector of all image point coordinates. In addition let $\Sigma_{\mathbf{X}}$ be the covariance matrix for the measurement vector.¹ We consider a general function $f: \mathbb{R}^M \rightarrow \mathbb{R}^N$ taking the parameter vector $\mathbf{P}$ to the estimated measurement vector $\hat{\mathbf{X}} = f(\mathbf{P})$. Denoting by ϵ the difference $\mathbf{X} - \hat{\mathbf{X}}$ between the measured and estimated quantities, one seeks the set of parameters that minimize the squared Mahalanobis distance $\|\epsilon\|_{\Sigma_{\mathbf{X}}}^2 = \epsilon^T \Sigma_{\mathbf{X}}^{-1} \epsilon$.

Corresponding to the division of parameters $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}^T)^T$, the Jacobian matrix $\mathbf{J} = [\partial \hat{\mathbf{X}} / \partial \mathbf{P}]$ has a block structure of the form $\mathbf{J} = [\mathbf{A} | \mathbf{B}]$, where Jacobian submatrices are defined by

$$\mathbf{A} = [\partial \hat{\mathbf{X}} / \partial \mathbf{a}]$$

and

$$\mathbf{B} = [\partial \hat{\mathbf{X}} / \partial \mathbf{b}].$$

The set of equations $\mathbf{J}\delta = \epsilon$ solved as the central step in the LM algorithm (see section A6.2) now has the form

$$\mathbf{J}\delta = [\mathbf{A} | \mathbf{B}] \begin{pmatrix} \delta_{\mathbf{a}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \epsilon. \quad (\text{A6.4})$$

Then, the normal equations $\mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{J} \delta = \mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \epsilon$ to be solved at each step of the LM algorithm are of the form

$$\begin{bmatrix} \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{A} & \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{B} \\ \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{A} & \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{B} \end{bmatrix} \begin{pmatrix} \delta_{\mathbf{a}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \epsilon \\ \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \epsilon \end{pmatrix}. \quad (\text{A6.5})$$

¹ In the absence of other knowledge, one usually assumes that $\Sigma_{\mathbf{X}}$ is the identity matrix.

At this point in the LM algorithm the diagonal blocks of this matrix are augmented by multiplying their diagonal entries by a factor $1 + \lambda$, for the varying parameter λ . This augmentation alters the matrices $A^T \Sigma_X^{-1} A$ and $B^T \Sigma_X^{-1} B$. The resulting matrices will be denoted by $(A^T \Sigma_X^{-1} A)^*$ and $(B^T \Sigma_X^{-1} B)^*$. The reader may wish to look ahead to figure A6.1 and figure A6.2 which shows graphically the form of the Jacobian and normal equations in an estimation problem involving several cameras and points.

The set of equations (A6.5) may now be written in block form as

$$\begin{bmatrix} U^* & W \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \epsilon_A \\ \epsilon_B \end{pmatrix}. \quad (\text{A6.6})$$

As a first step to solving these equations, both sides are now multiplied on the left by $\begin{bmatrix} I & -WV^{*-1} \\ 0 & I \end{bmatrix}$ resulting in

$$\begin{bmatrix} U^* - WV^{*-1}W^T & 0 \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \epsilon_A - WV^{*-1}\epsilon_B \\ \epsilon_B \end{pmatrix}. \quad (\text{A6.7})$$

This results in the elimination of the top right hand block. The top half of this set of equations is

$$(U^* - WV^{*-1}W^T)\delta_a = \epsilon_A - WV^{*-1}\epsilon_B. \quad (\text{A6.8})$$

These equations may be solved to find δ_a . Subsequently, the value of δ_b may be found by back-substitution, giving

$$V^*\delta_b = \epsilon_B - W^T\delta_a. \quad (\text{A6.9})$$

As in section A6.2, if the newly computed value of the parameter vector $P = ((a + \delta_a)^T, (b + \delta_b)^T)^T$ results in a diminished value of the error function, then one accepts the new parameter vector P , diminishes the value of λ by a factor of 10, and proceeds to the next iteration. On the other hand if the error value is increased, then one rejects the new P and tries again with a new value of λ , increased by a factor of 10.

The complete partitioned Levenberg-Marquardt algorithm is given in algorithm A6.1.

Although in this method we solve first for δ_a and subsequently for δ_b based on the new value of a , it should not be thought that the method amounts to no more than independent iterations over a and b . If one were to solve for a holding b constant, then the normal equations used to solve for δ_a would take the simpler form $U\delta_a = \epsilon_A$, compared with (A6.8). The method of alternating between solving for δ_a and δ_b is not recommended, however, because of potential slow convergence.

A6.3.2 Covariance

It was seen in result 5.12(p144) that the covariance matrix of the estimated parameters is given by

$$\Sigma_P = (J^T \Sigma_X^{-1} J)^+. \quad (\text{A6.10})$$

Given A vector of measurements $\mathbf{X}$ with covariance matrix $\Sigma_{\mathbf{X}}$, an initial estimate of a set of parameters $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}^T)^T$ and a function $f: \mathbf{P} \mapsto \hat{\mathbf{X}}$ taking the parameter vector to an estimate of the measurement vector.

Objective Find the set of parameters $\mathbf{P}$ that minimizes $\epsilon^T \Sigma_{\mathbf{X}}^{-1} \epsilon$ where $\epsilon = \mathbf{X} - \hat{\mathbf{X}}$.

Algorithm

- (i) Initialize a constant $\lambda = 0.001$ (typical value).
- (ii) Compute the derivative matrices $\mathbf{A} = [\partial \hat{\mathbf{X}} / \partial \mathbf{a}]$ and $\mathbf{B} = [\partial \hat{\mathbf{X}} / \partial \mathbf{b}]$ and the error vector ϵ .
- (iii) Compute intermediate expressions

$$\mathbf{U} = \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{A} \quad \mathbf{V} = \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{B} \quad \mathbf{W} = \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{B}$$

$$\epsilon_{\mathbf{A}} = \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \epsilon \quad \epsilon_{\mathbf{B}} = \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \epsilon$$

- (iv) Augment $\mathbf{U}$ and $\mathbf{V}$ by multiplying their diagonal elements by $1 + \lambda$.
- (v) Compute the inverse $\mathbf{V}^{*-1}$, and define $\mathbf{Y} = \mathbf{W} \mathbf{V}^{*-1}$. The inverse may overwrite the value of $\mathbf{V}^*$ which will not be needed again.
- (vi) Find $\delta_{\mathbf{a}}$ by solving $(\mathbf{U}^* - \mathbf{Y} \mathbf{W}^T) \delta_{\mathbf{a}} = \epsilon_{\mathbf{A}} - \mathbf{Y} \epsilon_{\mathbf{B}}$.
- (vii) Find $\delta_{\mathbf{b}}$ by back-substitution: $\delta_{\mathbf{b}} = \mathbf{V}^{*-1} (\epsilon_{\mathbf{B}} - \mathbf{W}^T \delta_{\mathbf{a}})$.
- (viii) Update the parameter vector by adding the incremental vector $(\delta_{\mathbf{a}}^T, \delta_{\mathbf{b}}^T)^T$ and compute the new error vector.
- (ix) If the new error is less than the old error, then accept the new values of the parameters, diminish the value of λ by a factor of 10, and start again at step (ii), or else terminate.
- (x) If the new error is greater than the old error, then revert to the old parameter values, increase the value of λ by a factor of 10, and try again from step (iv).

Algorithm A6.1. A partitioned Levenberg–Marquardt algorithm.

In the over-parametrized case, the covariance matrix $\Sigma_{\mathbf{P}}$ given by (A6.10) is singular, and in particular, no variation in the parameters is allowed in directions perpendicular to the constraint surface – the variance is zero in these directions.

In the case where the parameter set is partitioned as $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}^T)^T$, matrix $(\mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{J})$ has the block form given in (A6.5) and (A6.6) (but without augmentation, represented by stars). Thus we may write

$$\mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{J} = \begin{bmatrix} \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{A} & \mathbf{A}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{B} \\ \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{A} & \mathbf{B}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix}. \quad (\text{A6.11})$$

The covariance matrix $\Sigma_{\mathbf{P}}$ is the pseudo-inverse of this matrix. Under the assumption that $\mathbf{V}$ is invertible, redefine $\mathbf{Y} = \mathbf{W} \mathbf{V}^{-1}$. Then the matrix may be diagonalized according to

$$\mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{J} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{Y} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U} - \mathbf{W} \mathbf{V}^{-1} \mathbf{W}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{Y}^T & \mathbf{I} \end{bmatrix} \quad (\text{A6.12})$$

For matrices $\mathbf{G}$ and $\mathbf{H}$ with $\mathbf{G}$ invertible, we assume an identity

$$(\mathbf{G} \mathbf{H} \mathbf{G}^T)^+ = \mathbf{G}^{-T} \mathbf{H}^+ \mathbf{G}^{-1}$$

This identity is valid under conditions explored in the exercise at the end of this ap-

Objective Compute the covariance of the parameters $\mathbf{a}$ and $\mathbf{b}$ estimated using algorithm A6.1.
Algorithm

- (i) Compute matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ as in algorithm A6.1, and also $\mathbf{Y} = \mathbf{WV}^{-1}$.
- (ii) $\Sigma_{\mathbf{a}} = (\mathbf{U} - \mathbf{WV}^{-1}\mathbf{W}^T)^+$.
- (iii) $\Sigma_{\mathbf{b}} = \mathbf{Y}^T \Sigma_{\mathbf{a}} \mathbf{Y} + \mathbf{V}^{-1}$.
- (iv) The cross-covariance $\Sigma_{\mathbf{ab}} = -\Sigma_{\mathbf{a}} \mathbf{Y}$.

Algorithm A6.2. *Computation of the covariance matrix of the LM parameters.*

pendix. Applying this to (A6.12) and multiplying out provides a formula for the pseudo inverse

$$\Sigma_{\mathbf{P}} = (\mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{J})^+ = \begin{bmatrix} \mathbf{X} & -\mathbf{XY} \\ -\mathbf{Y}^T \mathbf{X} & \mathbf{Y}^T \mathbf{XY} + \mathbf{V}^{-1} \end{bmatrix} \quad (\text{A6.13})$$

where $\mathbf{X} = (\mathbf{U} - \mathbf{WV}^{-1}\mathbf{W}^T)^+$.

The condition for this to be true is that $\text{span}(\mathbf{A}) \cap \text{span}(\mathbf{B}) = \emptyset$, where $\text{span}(\cdot)$ represents the span of the columns of the matrix. Here $\mathbf{A}$ and $\mathbf{B}$ are as in (A6.11). Proof of this fact and interpretation of the condition $\text{span}(\mathbf{A}) \cap \text{span}(\mathbf{B}) = \emptyset$ is outlined in exercise (i) on page 626.

The division of matrix $\mathbf{J}^T \Sigma_{\mathbf{X}}^{-1} \mathbf{J}$ into blocks as in (A6.11) corresponds in (A6.5) to the division of $\mathbf{P}$ into parameters $\mathbf{a}$ and $\mathbf{b}$. Truncation of the covariance matrix for parameters $\mathbf{P}$ gives covariance matrices for parameters $\mathbf{a}$ and $\mathbf{b}$ separately. The result is summarized in algorithm A6.2.

A6.3.3 General sparse LM method

In the previous pages, a method has been described for carrying out LM iteration and computing covariances of the solution in the case where the parameter vector may be partitioned into two sub-vectors $\mathbf{a}$ and $\mathbf{b}$. It is not clear from the foregoing discussion that the methods described there actually give any computational advantage in the general case. However, such methods become important when the Jacobian matrix obeys a certain sparseness condition, as will be described now.

We suppose that the “measurement vector” $\mathbf{X} \in \mathbb{R}^N$ may be broken up into pieces as $\mathbf{X} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_n^T)^T$. Similarly, suppose that the “parameter vector” $\mathbf{P} \in \mathbb{R}^M$ may be divided up as $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}_1^T, \mathbf{b}_2^T, \dots, \mathbf{b}_n^T)^T$. The estimated values of $\mathbf{x}_i$ corresponding to a given assignment of parameters will be denoted by $\hat{\mathbf{x}}_i$. We make the *sparseness assumption* that each $\hat{\mathbf{x}}_i$ is dependent on $\mathbf{a}$ and $\mathbf{b}_i$ only, but not on the other parameters $\mathbf{b}_j$. In this case, $\partial \hat{\mathbf{x}}_i / \partial \mathbf{b}_j = 0$ for $i \neq j$. No assumption is made about $\partial \hat{\mathbf{x}}_i / \partial \mathbf{a}$. This situation arises in the reconstruction problem described at the start of this discussion, in which $\mathbf{b}_i$ is the vector of parameters of the i -th point, and $\hat{\mathbf{x}}_i$ is the vector of measurements of the image of this point in all the views. In this case, since the image of a point does not depend on any other point, one sees that $\partial \hat{\mathbf{x}}_i / \partial \mathbf{b}_j = 0$, as required, unless $i = j$.

Corresponding to this division, the Jacobian matrix $\mathbf{J} = [\partial \hat{\mathbf{X}} / \partial \mathbf{P}]$ has a sparse block

structure. We define Jacobian matrices by

$$A_i = [\partial \hat{\mathbf{x}}_i / \partial \mathbf{a}] \quad B_i = [\partial \hat{\mathbf{x}}_i / \partial \mathbf{b}_i].$$

Given an error vector of the form $\epsilon = (\epsilon_1^\top, \dots, \epsilon_n^\top)^\top = \mathbf{X} - \hat{\mathbf{X}}$ the set of equations $\mathbf{J}\delta = \epsilon$ now has the form

$$\mathbf{J}\delta = \begin{bmatrix} \begin{matrix} A_1 & B_1 \\ A_2 & \\ \vdots & \\ A_n & \end{matrix} & \begin{matrix} B_2 \\ \\ \ddots \\ B_n \end{matrix} \end{bmatrix} \begin{pmatrix} \delta_a \\ \delta_{b_1} \\ \vdots \\ \delta_{b_n} \end{pmatrix} = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}. \quad (\text{A6.14})$$

We suppose further that all of the measurements $\mathbf{x}_i$ are independent with covariance matrices $\Sigma_{\mathbf{x}_i}$. Thus the covariance matrix for the complete measurement vector $\Sigma_{\mathbf{X}}$ has the block-diagonal form $\Sigma_{\mathbf{X}} = \text{diag}(\Sigma_{\mathbf{x}_1}, \dots, \Sigma_{\mathbf{x}_n})$.

In the notation of algorithm A6.1, we have

$$\begin{aligned} \mathbf{A} &= [A_1^\top, A_2^\top, \dots, A_n^\top]^\top \\ \mathbf{B} &= \text{diag}(B_1, B_2, \dots, B_n) \\ \Sigma_{\mathbf{X}} &= \text{diag}(\Sigma_{\mathbf{x}_1}, \dots, \Sigma_{\mathbf{x}_n}) \\ \delta_{\mathbf{b}} &= (\delta_{b_1}^\top, \delta_{b_2}^\top, \dots, \delta_{b_n}^\top)^\top \\ \epsilon &= (\epsilon_1^\top, \epsilon_2^\top, \dots, \epsilon_n^\top)^\top \end{aligned}$$

Now, it is a straightforward task to substitute these formulae into algorithm A6.1. The result of this substitution is given in algorithm A6.3, representing the computation of one step of the LM algorithm. The important observation is that in this form each step of the algorithm requires computation time linear in n . Without the advantage resulting from the sparse structure, the algorithm (for instance a blind adherence to algorithm A6.1) would have complexity of order n^3 .

A6.4 Application of sparse LM to 2D homography estimation

We apply the foregoing discussion to the estimation of a 2D homography $\mathbf{H}$ given a set of corresponding image points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ in two images. The points in each image are subject to noise, and one seeks to minimize the cost function (4.8–p95). We define a measurement vector $\mathbf{X}_i = (\mathbf{x}_i^\top, \mathbf{x}'_i^\top)^\top$. In this case, the parameter vector $\mathbf{P}$ may be divided up as $\mathbf{P} = (\mathbf{h}^\top, \hat{\mathbf{x}}_1^\top, \hat{\mathbf{x}}_2^\top, \dots, \hat{\mathbf{x}}_n^\top)^\top$, where the values $\hat{\mathbf{x}}_i$ are the estimated values of the image points in the first image, and $\mathbf{h}$ is a vector of entries of the homography $\mathbf{H}$. Thus, one must simultaneously estimate the homography $\mathbf{H}$ and the parameters of each point in the first image. The function f maps $\mathbf{P}$ to $(\hat{\mathbf{x}}_1^\top, \hat{\mathbf{x}}_2^\top, \dots, \hat{\mathbf{x}}_n^\top)^\top$, where each $\hat{\mathbf{x}}_i = (\hat{\mathbf{x}}_i^\top, \mathbf{H}\hat{\mathbf{x}}_i^\top)^\top = (\hat{\mathbf{x}}_i^\top, \hat{\mathbf{x}}'_i{}^\top)^\top$. Then algorithm A6.3 applies directly.

The Jacobian matrices have a special form in this case, since one observes that

$$A_i = \partial \hat{\mathbf{x}}_i / \partial \mathbf{h} = \begin{bmatrix} 0 \\ \partial \hat{\mathbf{x}}'_i / \partial \mathbf{h} \end{bmatrix}$$

Objective Formulate LM algorithm in the case where the parameter vector is partitioned as $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}_1^T, \dots, \mathbf{b}_n^T)^T$, and the measurement vector as $\mathbf{X} = (\mathbf{X}_1^T, \dots, \mathbf{X}_n^T)^T$, such that $\partial \hat{\mathbf{X}}_i / \partial \mathbf{b}_j = 0$ for $i \neq j$.

Algorithm Steps (ii) to (vii) of algorithm A6.1 become:

- (i) Compute the derivative matrices $\mathbf{A}_i = [\partial \hat{\mathbf{X}}_i / \partial \mathbf{a}]$ and $\mathbf{B}_i = [\partial \hat{\mathbf{X}}_i / \partial \mathbf{b}_i]$ and the error vectors $\epsilon_i = \mathbf{X}_i - \hat{\mathbf{X}}_i$.
 (ii) Compute the intermediate values

$$\begin{aligned} \mathbf{U} &= \sum_i \mathbf{A}_i^T \Sigma_{\mathbf{X}_i}^{-1} \mathbf{A}_i \\ \mathbf{V} &= \text{diag}(V_1, \dots, V_n) \text{ where } V_i = \mathbf{B}_i^T \Sigma_{\mathbf{X}_i}^{-1} \mathbf{B}_i \\ \mathbf{W} &= [\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_n] \text{ where } \mathbf{W}_i = \mathbf{A}_i^T \Sigma_{\mathbf{X}_i}^{-1} \mathbf{B}_i \\ \epsilon_{\mathbf{A}} &= \sum_i \mathbf{A}_i^T \Sigma_{\mathbf{X}_i}^{-1} \epsilon_i \\ \epsilon_{\mathbf{B}} &= (\epsilon_{\mathbf{B}_1}^T, \epsilon_{\mathbf{B}_2}^T, \dots, \epsilon_{\mathbf{B}_n}^T)^T \text{ where } \epsilon_{\mathbf{B}_i} = \mathbf{B}_i^T \Sigma_{\mathbf{X}_i}^{-1} \epsilon_i \\ \mathbf{Y}_i &= \mathbf{W}_i \mathbf{V}_i^{*-1}. \end{aligned}$$

- (iii) Compute $\delta_{\mathbf{a}}$ from the equation

$$(\mathbf{U}^* - \sum_i \mathbf{Y}_i \mathbf{W}_i^T) \delta_{\mathbf{a}} = \epsilon_{\mathbf{A}} - \sum_i \mathbf{Y}_i \epsilon_{\mathbf{B}_i}.$$

- (iv) Compute each $\delta_{\mathbf{b}_i}$ in turn from the equation

$$\delta_{\mathbf{b}_i} = \mathbf{V}_i^{*-1} (\epsilon_{\mathbf{B}_i} - \mathbf{W}_i^T \delta_{\mathbf{a}}).$$

Covariance

- (i) Redefine $\mathbf{Y}_i = \mathbf{W}_i \mathbf{V}_i^{-1}$
 (ii) $\Sigma_{\mathbf{a}} = (\mathbf{U} - \sum_i \mathbf{Y}_i \mathbf{W}_i^T)^+$
 (iii) $\Sigma_{\mathbf{b}_i \mathbf{b}_j} = \mathbf{Y}_i^T \Sigma_{\mathbf{a}} \mathbf{Y}_j + \delta_{ij} \mathbf{V}_i^{-1}$
 (iv) $\Sigma_{\mathbf{a} \mathbf{b}_i} = -\Sigma_{\mathbf{a}} \mathbf{Y}_i$

Algorithm A6.3. A sparse Levenberg–Marquardt algorithm.

since $\hat{\mathbf{x}}_i$ is independent of $\mathbf{h}$. Also,

$$\mathbf{B}_i = \partial \hat{\mathbf{X}}_i / \partial \hat{\mathbf{x}}_i = \begin{bmatrix} \mathbf{I} \\ \partial \hat{\mathbf{x}}_i' / \partial \hat{\mathbf{x}}_i \end{bmatrix}$$

because of the form of $\hat{\mathbf{X}}_i = (\hat{\mathbf{x}}_i^T, \hat{\mathbf{x}}_i'^T)^T$.

A6.4.1 Computation of the covariance

As an example of covariance computation, we consider the same problem as in section 5.2.4(p145), in which a homography is estimated from point correspondences. As in that example, we consider a case in which the estimated homography is actually the identity mapping, $\mathbf{H} = \mathbf{I}$. For the purposes of this example, the number of points or their distribution is not important. It will be assumed, however, that the errors of all

point measurements are independent. We recall from (5.12–p146) that in the case of errors in the second image only, $\Sigma_h = \left(\sum_i J_i^T \Sigma_i^{-1} J_i \right)^+$, where $J_i = [\partial \tilde{x}'_i / \partial \mathbf{h}]$.

We now proceed to compute the covariance of the camera parameter vector $\mathbf{h}$ in the case where the points in the first image are subject to noise as well. We assume further that $\Sigma_{x_i}^{-1} = \Sigma_{x'_i}^{-1}$, and denote them by S_i . In this case, the inverse covariance matrix Σ_x^{-1} is block-diagonal, $\Sigma_x^{-1} = \text{diag}(\Sigma_{x_i}^{-1}, \Sigma_{x'_i}^{-1})$. Then, applying the steps of algorithm A6.3 to compute the covariance matrix for $\mathbf{h}$,

$$\begin{aligned} A_i &= [0^T, J_i^T]^T \\ B_i &= [I^T, I^T]^T \text{ since } H = I \\ U &= \sum_i A_i^T \text{diag}(S_i, S_i) A_i = \sum_i J_i^T S_i J_i \\ V_i &= B_i^T \text{diag}(S_i, S_i) B_i = 2S_i \\ W_i &= A_i^T \text{diag}(S_i, S_i) B_i = J_i^T S_i \\ U - \sum_i W_i V_i^{-1} W_i^T &= \sum_i J_i^T (S_i - S_i/2) J_i = \sum_i J_i^T S_i J_i / 2 \\ \Sigma_h &= 2 \left(\sum_i J_i^T S_i J_i \right)^+ \end{aligned}$$

Hence, the covariance matrix for $\mathbf{h}$ is just twice the value of the covariance matrix for the case of error in one image. This is not generally true, and results from the fact that H is the identity mapping. As exercises, one may verify the following.

- If H represents a scaling by a factor s , then $\Sigma_h = (s^2 + 1) \left(\sum_i J_i^T S_i J_i \right)^+$.
- If H is an affine transformation and D is the upper 2×2 part of H (the non-translational part), and if $S_i = I$ for all i (isotropic and independent noise), then $\Sigma_h = \left(\sum_i J_i^T (I - D(I + D^T D)^{-1} D^T) J_i \right)^+$.

A6.5 Application of sparse LM to fundamental matrix estimation

In estimating the fundamental matrix and a set of 3D points, the algorithm is effectively that described in section A6.3.3, a sparse LM algorithm for the estimation of 2D homographies, but slightly modified to the present case. The analogy with the 2D homography estimation problem is as follows: in estimating 2D homographies, one has a mapping H that takes points $\mathbf{x}_i$ to corresponding points $\mathbf{x}'_i$; in the present problem, one has a mapping represented by a pair of camera matrices P and P' that map a 3D point to a pair of corresponding points $(\mathbf{x}_i, \mathbf{x}'_i)$.

For convenience, the notation of section A6.3.3 will be used here. In particular, note that in section A6.3.3, and here the notation $\mathbf{X}$ is used to denote the total measurement vector (in this case $(\mathbf{x}_1, \mathbf{x}'_1, \dots, \mathbf{x}_n, \mathbf{x}'_n)$) and not a 3D point. Also, be careful to distinguish between $\mathbf{P}$ the parameter vector and P the camera matrix.

The parameter vector $\mathbf{P}$ is partitioned as $\mathbf{P} = (\mathbf{a}^T, \mathbf{b}_1^T, \dots, \mathbf{b}_n^T)^T$, where

- (i) $\mathbf{a} = \mathbf{p}'$ is made up of the entries of camera matrix P' , and

(ii) $\mathbf{b}_i = (x_i, y_i, t_i)^T$ is a 3-vector, parametrizing the i -th 3D point $(x_i, y_i, 1, t_i)$.

Thus, there are a total of $3n+12$ parameters, where n is the number of points. Parameter vector $\mathbf{a}$ provides a parametrization for the camera P' and the other camera P is taken as $[I \mid 0]$. Note also that it is convenient and permissible to set the third coordinate of the 3D space point to 1 as here, since a point $(x_i, y_i, 0, t_i)^T$ maps to $(x_i, y_i, 0)^T$ which is an infinite point, not anywhere close to the measured point $(x_i, y_i, 1)^T$.

The measurement vector $\mathbf{X}$ is partitioned as $\mathbf{X} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_n^T)^T$, where each $\mathbf{x}_i = (\mathbf{x}_i^T, \mathbf{x}_i'^T)^T$, the measured images of the i -th point.

Now, the Jacobian matrices $A_i = \partial \hat{\mathbf{x}}_i / \partial \mathbf{a}$ and $B_i = \partial \hat{\mathbf{x}}_i / \partial \mathbf{b}_i$ may be computed and algorithm A6.3 applied to estimate the parameters, and hence P' , from which F may be computed.

Partial derivative matrices

Since $\hat{\mathbf{x}}_i = (\hat{\mathbf{x}}_i^T, \hat{\mathbf{x}}_i'^T)^T$ one may compute that A_i and B_i have a form similar to the Jacobian matrices in section A6.4:

$$A_i = \begin{bmatrix} 0 \\ \partial \hat{\mathbf{x}}_i' / \partial \mathbf{a} \end{bmatrix} \quad B_i = \begin{bmatrix} I_{2 \times 2} \mid \mathbf{0} \\ \partial \hat{\mathbf{x}}_i' / \partial \mathbf{b}_i \end{bmatrix}.$$

The covariance matrix $\Sigma_{\mathbf{x}_i}$ breaks up into diagonal blocks $\text{diag}(S_i, S_i')$, where $S_i = \Sigma_{\mathbf{x}_i}^{-1}$ and $S_i' = \Sigma_{\mathbf{x}_i'}^{-1}$. Now, calculating the intermediate expressions in step 2 of algorithm A6.3, we find

$$V_i = B_i^T \text{diag}(S_i, S_i') B_i = [I_{2 \times 2} \mid \mathbf{0}]^T S_i [I_{2 \times 2} \mid \mathbf{0}] + (\partial \hat{\mathbf{x}}_i' / \partial \mathbf{b})^T S_i' (\partial \hat{\mathbf{x}}_i' / \partial \mathbf{b}) \quad (\text{A6.15})$$

The abstract form of $A_i^T \Sigma_{\mathbf{x}_i}^{-1} A_i$ is the same as in the 2D homography case, and the other expressions $W_i = A_i^T \Sigma_{\mathbf{x}_i}^{-1} B_i$, $\epsilon_{B_i} = B_i^T \Sigma_{\mathbf{x}_i}^{-1} \epsilon_i$, and $\epsilon_{A_i} = A_i^T \Sigma_{\mathbf{x}_i}^{-1} \epsilon_i$ may easily be computed. The estimation procedure otherwise proceeds precisely as in algorithm A6.3.

Covariance of F

According to the discussion of section A6.3.3 and in particular algorithm A6.3, the covariance matrix of the camera parameters, namely the entries of P' , is given by

$$\Sigma_{P'} = (U - \sum_k W_k V_k^{-1} W_k^T)^+ \quad (\text{A6.16})$$

with notation as in algorithm A6.3.

In computing this pseudo-inverse, it is useful to know the expected rank of $\Sigma_{P'}$. In this case, this rank is 7, since the total number of degrees of freedom in the solution involving two cameras and n point matches is $3n + 7$. Looked at another way, P' is not determined uniquely, since if $P' = [M \mid \mathbf{m}]$, then any other matrix $[M + \mathbf{t}\mathbf{m}^T \mid \alpha\mathbf{m}]$ also determines the same fundamental matrix. Thus, in computing the pseudo-inverse appearing in the right hand side of (A6.16), one should set five of the singular values to zero.

The foregoing discussion shows how to compute the covariance matrix of the entries of P' . We desire to compute the covariance matrix of the entries of F . However, there is a simple formula for the entries of F in terms of the entries of $P' = [M \mid \mathbf{m}]$, namely

$F = [\mathbf{m}]_x \mathbf{M}$. If one desires to compute the covariance matrix of F normalized so that $\|F\| = 1$, then one writes $F = [\mathbf{m}]_x \mathbf{M} / (\|[\mathbf{m}]_x \mathbf{M}\|)$. Therefore, one may express the entries of F as a simple function in terms of the entries of P' . Let J be the Jacobian matrix of this function. The covariance of F is then computed by propagating the covariance of P' using result 5.6(p139) to get

$$\Sigma_F = J \Sigma_{P'} J^T = J \left(U - \sum_i w_i V_i^{-1} w_i^T \right)^+ J^T \quad (\text{A6.17})$$

where $\Sigma_{P'}$ is given by (A6.16). This is the covariance of the fundamental matrix *estimated according to an ML algorithm* from the given point correspondences.

A6.6 Application of sparse LM to multiple image bundle adjustment

In the previous section, the application of the sparse Levenberg–Marquardt algorithm to the computation of the fundamental matrix was considered. This is essentially a reconstruction problem from two views. It should be clear how this may easily be extended to the computation of the trifocal tensor and the quadrifocal tensor. More generally, one may apply it to the simultaneous estimation of multiple camera and points to compute projective structure, or perhaps affine or metric structure given appropriate constraints. This technique is called *bundle adjustment*.

In the case of multiple cameras, one may also take advantage of the lack of interaction between parameters of the different cameras, as will be shown now. In the following discussion, for simplicity of notation it will be assumed that each point is visible in all the views. This is not at all necessary – points may in general be visible in some arbitrary subset of the available views.

We use the same notation as in section A6.3.3. The measurement data may be expressed as a vector $\mathbf{X}$, which may be divided up into parts $\mathbf{X}_i$, representing the measured image coordinates of some 3D point in all views. One may further subdivide $\mathbf{X}_i$ writing $\mathbf{X}_i = (\mathbf{x}_{i1}^T, \mathbf{x}_{i2}^T, \dots, \mathbf{x}_{im}^T)^T$ where $\mathbf{x}_{ij}$ is the image of the i -th point in the j -th image. The parameter vector $\mathbf{a}$ (camera parameters) may correspondingly be partitioned as $\mathbf{a} = (\mathbf{a}_1^T, \mathbf{a}_2^T, \dots, \mathbf{a}_m^T)^T$, where $\mathbf{a}_j$ are the parameters of the j -th camera. Since the image point $\mathbf{x}_{ij}$ does not depend on the parameters of any but the j -th camera, one observes that $\partial \mathbf{x}_{ij} / \partial \mathbf{a}_k = 0$ unless $j = k$. In a similar way for derivatives with respect to the parameters $\mathbf{b}_k$ of the k -th 3D point, one has $\partial \mathbf{x}_{ij} / \partial \mathbf{b}_k = 0$ unless $i = k$.

The form of the Jacobian matrix J for this problem and the resulting normal equations $J^T J \delta = J^T \epsilon$ are shown schematically in figure A6.1 and figure A6.2. Referring to the Jacobian matrices defined in algorithm A6.3, one sees that $A_i = [\partial \mathbf{X}_i / \partial \mathbf{a}]$ is a block diagonal matrix $A_i = \text{diag}(A_{i1}, \dots, A_{im})$, where $A_{ij} = \partial \mathbf{x}_{ij} / \partial \mathbf{a}_j$. Similarly, matrix $B_i = [\partial \mathbf{X}_i / \partial \mathbf{b}_i]$ decomposes as $B_i = [B_{i1}^T, \dots, B_{im}^T]^T$, where $B_{ij} = \partial \mathbf{x}_{ij} / \partial \mathbf{b}_i$. It may normally be assumed also that $\Sigma_{\mathbf{X}_i}$ has a diagonal structure $\Sigma_{\mathbf{X}_i} = \text{diag}(\Sigma_{\mathbf{x}_{i1}}, \dots, \Sigma_{\mathbf{x}_{im}})$, meaning that the measurements of the projected points in separate images are independent (or more precisely, uncorrelated). With these assumptions, one is easily able to adapt algorithm A6.3, as shown in algorithm A6.4, as the reader is left to verify.

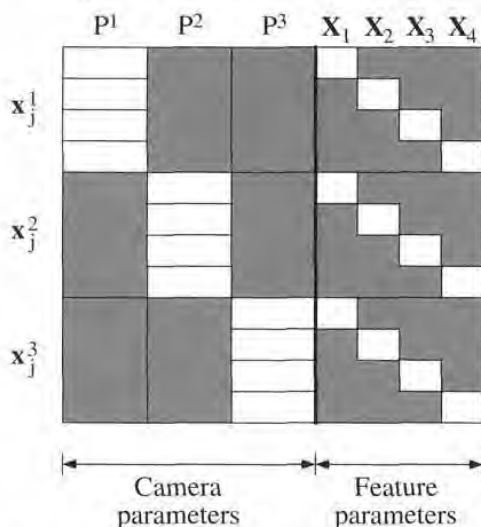


Fig. A6.1. Form of the Jacobian matrix for a bundle-adjustment problem consisting of 3 cameras and 4 points.

The diagram shows the normal equations for the bundle-adjustment problem with 3 cameras and 4 points. The left side is a block matrix representing the normal equations. It is partitioned into two main sections: 'Camera parameters' (columns U_1, U_2, U_3) and 'Feature parameters' (columns V_1, V_2, V_3, V_4). The 'Camera parameters' section is a block matrix where each camera row has non-zero entries only in its corresponding camera parameter columns. The 'Feature parameters' section is a block matrix where each camera row has non-zero entries only in the feature parameter columns of the points it observes. For example, camera 1 observes points 1, 2, and 3, so its row has non-zero entries in columns V_1, V_2, V_3 . Camera 2 observes points 2, 3, and 4, so its row has non-zero entries in columns V_2, V_3, V_4 . Camera 3 observes points 3 and 4, so its row has non-zero entries in columns V_3, V_4 . The right side of the equation is a vector of residuals, partitioned into camera residuals $\Delta(P^1), \Delta(P^2), \Delta(P^3)$ and feature residuals $\Delta(X_1), \Delta(X_2), \Delta(X_3), \Delta(X_4)$. The equation is written as:

$$\begin{bmatrix} U_1 & & & & & \\ & U_2 & & & & \\ & & U_3 & & & \\ & & & V_1 & & \\ & & & & V_2 & \\ & & & & & V_3 \\ & & & & & & V_4 \end{bmatrix} \times \begin{bmatrix} \Delta(P^1) \\ \Delta(P^2) \\ \Delta(P^3) \\ \Delta(X_1) \\ \Delta(X_2) \\ \Delta(X_3) \\ \Delta(X_4) \end{bmatrix} = \begin{bmatrix} \varepsilon(P^1) \\ \varepsilon(P^2) \\ \varepsilon(P^3) \\ \varepsilon(X_1) \\ \varepsilon(X_2) \\ \varepsilon(X_3) \\ \varepsilon(X_4) \end{bmatrix}$$

Fig. A6.2. Form of the normal equations for the bundle-adjustment problem consisting of 3 cameras and 4 points.

Missing data. Typically, in a bundle-adjustment problem some points are not visible in every image. Thus, some measurement x_{ij} may not exist, meaning that i -th point is not visible in the j -th image. Algorithm A6.4 is easily adapted to this situation, by ignoring terms subscripted by ij where the measurement x_{ij} does not exist. Such missing terms are simply omitted from the relevant summations. This includes all of A_{ij} , B_{ij} , $\Sigma_{x_{ij}}^{-1}$, W_{ij} and Y_{ij} . It may be seen that this is equivalent to setting A_{ij} and B_{ij} to zero, thus effectively giving zero weight to the missing measurements.

This is convenient when programming this algorithm, since the above quantities subscripted by ij may be associated only with existing measurements in a common data structure.

- (i) Compute the derivative matrices $A_{ij} = [\partial \tilde{x}_{ij} / \partial \mathbf{a}_j]$ and $B_{ij} = [\partial \tilde{x}_{ij} / \partial \mathbf{b}_i]$ and the error vectors $\epsilon_{ij} = \mathbf{x}_{ij} - \tilde{\mathbf{x}}_{ij}$.
- (ii) Compute the intermediate values

$$U_j = \sum_i A_{ij}^T \Sigma_{ij}^{-1} A_{ij} \quad V_i = \sum_j B_{ij}^T \Sigma_{ij}^{-1} B_{ij} \quad W_{ij} = A_{ij}^T \Sigma_{ij}^{-1} B_{ij}$$

$$\epsilon_{a_j} = \sum_i A_{ij}^T \Sigma_{ij}^{-1} \epsilon_{ij} \quad \epsilon_{b_i} = \sum_j B_{ij}^T \Sigma_{ij}^{-1} \epsilon_{ij} \quad Y_{ij} = W_{ij} V_i^{*-1}$$

where in each case $i = 1, \dots, n$ and $j = 1, \dots, m$.

- (iii) Compute $\delta_a = (\delta_{a_1}^T, \dots, \delta_{a_m}^T)^T$ from the equation

$$S \delta_a = (\mathbf{e}_1^T, \dots, \mathbf{e}_m^T)^T$$

where S is an $m \times m$ block matrix with blocks S_{jk} defined by

$$S_{jj} = -\sum_i Y_{ij} W_{ij}^T + U_j^*$$

$$S_{jk} = -\sum_i Y_{ij} W_{ik}^T \text{ if } j \neq k$$

and

$$\mathbf{e}_j = \epsilon_{a_j} - \sum_i Y_{ij} \epsilon_{b_i}$$

- (iv) Compute each δ_{b_i} in turn from the equation

$$\delta_{b_i} = V_i^{*-1} (\epsilon_{b_i} - \sum_j W_{ij}^T \delta_{a_j})$$

Covariance

- (i) Redefine $Y_{ij} = W_{ij} V_i^{-1}$.
- (ii) $\Sigma_a = S^+$, where S is defined as above, without the augmentation represented by the $*$.
- (iii) $\Sigma_{b_i b_k} = Y_i^T \Sigma_a Y_k + \delta_{ik} V_i^{-1}$.
- (iv) $\Sigma_{ab_i} = -\Sigma_a Y_i$.

Algorithm A6.4. General sparse Levenberg–Marquardt algorithm.

A6.7 Sparse methods for equation solving

In a long sequence of images, it is rare for a point to be tracked through the whole sequence, and usually point tracks disappear and new ones start, causing the set of point tracks to have a banded structure, as seen in figure 19.7(p492)(c). This banded structure of the point track set leads to a banded structure for the set of equations that are solved to compute structure and motion – we refer here to the matrix S in algorithm A6.4. Thus, for bundle-adjustment problems with banded track structure, sparseness can appear at two levels, first at the level of independence of the individual point measurements, as exploited in algorithm A6.4, and secondly arising from the banded track structure as will be explained in this section.

Another similar context in which this will occur is solution for structure and motion

in section 18.5.1(p448) or simply motion, as in section 18.5.2(p450). In both these methods, large sparse sets of equations may arise. In order for large problems of this kind to be tractable, it is necessary to take advantage of this sparse structure in order to minimize the amount of storage and computation involved. In this section, we will consider sparse matrix techniques that are valuable in this context.

A6.7.1 Banded structure in bundle-adjustment

The time-consuming step in finding the parameter increments in the iterative step of bundle-adjustment, as given in algorithm A6.4 is the solution of the equations $S\delta_u = e$ in step (iii) of the algorithm. As shown there, the matrix S is a symmetric matrix with off-diagonal blocks of the form $S_{jk} = -\sum_i W_{ij} V_i^{-1} W_{ik}^T$. We see that the block S_{jk} is non-zero only when for some i , both W_{ij} and W_{ik} are non-zero. Since $W_{ij} = [\partial \hat{x}_{ij} / \partial a_j]^T \Sigma_{x_{ij}}^{-1} [\partial \hat{x}_{ij} / \partial b_i]$ it follows that W_{ij} is non-zero only when the corresponding measurement $\hat{x}_{ij}$ depends on parameters a_j and b_i . To be more concrete, if a_j represents the parameters of the j -th camera, and b_i represents the parameters of the i -th point, then W_{ij} is non-zero only when the i -th point is visible in the j -th image, and $\hat{x}_{ij}$ is its measured image position.

The condition that for some i both W_{ij} and W_{ik} are non-zero means that there exists some point with index i that is visible in both the j -th and k -th images. To summarize,

- The block S_{jk} is non-zero only if there exists a point that is visible in both the j -th and k -th images.

Thus, if point tracks extend only over consecutive views, then the matrix S will be banded. In particular, if no point track extends over more than B views (B representing bandwidth), then the block S_{jk} is zero unless $|j - k| < B$.

Consider tracking points over a long sequence, for instance along a path that may loop back and cross itself. In this case, it may be possible to recognize a point that had been seen previously in the sequence, and pick up its track again. The set of views in which a 3D point is seen will not be a consecutive set of views. This will destroy the banded nature of the matrix S , by introducing non-zero blocks possibly far away from the central band. Nevertheless, if there is not too much filling-in of off-diagonal blocks, sparse solution techniques may still be utilized as we shall see later.

A6.7.2 Solution of symmetric linear equations

In solving a set of linear equations $Ax = b$ in which the matrix A is symmetric, it is best not to use a general purpose equation solver, such as Gaussian-elimination, but rather to take advantage of the symmetry of the matrix A . One way of doing this is the use the LDL^T decomposition of the matrix A . This relies on the following observation:

Result A6.1. Any positive-definite symmetric matrix A can be factored as $A = LDL^T$, in which L is a lower-triangular matrix with unit diagonal entries, and D is diagonal.

The reader is advised to consult [Golub-89] for details of efficient implementation and numeric properties of LDL^T decomposition. The normal equations derived from structure and motion problems are always at least positive semi-definite, and with stabiliza-

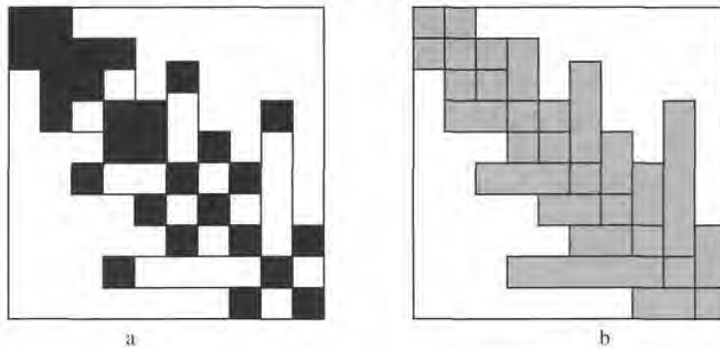


Fig. A6.3. (a) A sparse matrix and (b) its corresponding "skyline" structure. Non-zero entries in the original matrix are shown in black. In the skyline format, all non-zero entries lie in the shaded area. For each row i , there is an integer m_i representing the first non-zero entry in that row. Note that non-zero off-diagonal elements cause "fill-in" of the skyline format for that row (or symmetric above-diagonal column). If such entries are relatively rare, the skyline format will remain sparse, and techniques for skyline matrices may be usefully applied. The LDL^T decomposition of a matrix in skyline format has the same skyline structure. Thus, the LDL^T decomposition of the original sparse matrix shown in (a), will have non-zero entries only within the shaded area of (b).

tion through enhancement, they are positive-definite, and symmetric factorization is the recommended method of solution.

Given the LDL^T factorization, the set of linear equations $Ax = LDL^Tx = b$ may be solved for x in three steps as (i) $Lx' = b$, (ii) $x'' = D^{-1}x'$, (iii) $L^Tx = x''$.

Solving the equations $Lx' = b$ is carried out by the process of "forward-substitution." In particular (bearing in mind that L has unit diagonal entries), the components of x may be computed in order as

$$\text{forward-substitution: } x'_i = b_i - \sum_{j=1}^{i-1} L_{ij}x'_j.$$

Since L^T is upper-triangular, the second set of equations is solved in a similar fashion, except that the values x_i are computed in inverse order in a process known as "back-substitution."

$$\text{back-substitution: } x_i = x''_i - \sum_{j=i+1}^n L_{ji}x_j.$$

The number of operations involved in this computation is given in [Golub-89], and is equal to $n^3/3$, where n is the dimension of the matrix.

A6.7.3 Solution of sparse symmetric linear systems

We consider a special type of sparse structure for a symmetric matrix, known as "skyline" format. This is illustrated in figure A6.3. An $n \times n$ symmetric matrix A in skyline format is characterized by the existence of an array of integers m_i for $i = 1, \dots, n$ such that $A_{ij} = 0$ for $j < m_i$. A diagonally banded matrix is a special case of a matrix with skyline structure.

Although a diagonally banded or skyline matrix may be sparse, its inverse is not, and in fact will be completely filled out with non-zero elements. Thus, it is a very bad idea actually to compute the inverse of A to find the solution $\mathbf{x} = A^{-1}\mathbf{b}$ to the set of equations. However, the importance of matrices in skyline (or banded) form is that skyline structure of the matrix *is preserved* by the LDL^T decomposition, as expressed in the following result.

Result A6.2. *Let A be a symmetric matrix such that $A_{ij} = 0$ for $j < m_i$. Let $A = LDL^T$. Then $L_{ij} = 0$ for $j < m_i$.*

In other words, the skyline structure of L is the same as that of A .

Proof. Suppose that j is the smallest index such that $L_{ij} = 0$. Then in multiplying out $A = LDL^T$, it may be verified that only one product contributes to the (i, j) -th element of A_{ij} . Specifically, $A_{ij} = L_{ij}D_{jj}L_{jj} \neq 0$. $\square$

Thus, in computing the LDL^T decomposition of A having a sparse skyline structure, we know in advance that many of the entries of L will be zero. The algorithm for computing the LDL^T decomposition for such a matrix is much the same as the that for a full symmetric matrix, except that we do not need to consider the zero elements.

Forward and back substitution involving a matrix L with skyline structure easily take advantage of the sparse structure. In fact the forward substitution formula becomes

$$\mathbf{x}'_i = \mathbf{b}_i - \sum_{j=m_i}^{i-1} L_{ij}\mathbf{x}'_j.$$

Back-substitution is left for the reader to work out. More details of implementation are given in [Bathe-76].

A6.8 Robust cost functions

In estimation problems of the Newton or Levenberg-Marquardt type, an important decision to make is the precise form of the cost function. As we have seen, an assumption of Gaussian noise without outliers implies that the Maximum Likelihood estimate is given by a least-squares cost function involving the predicted errors in the measurements, where the noise is introduced.

The same analysis may be carried out for other assumed probability models for the measurements. Thus, if all measurements are assumed to be independent, and $f(\delta)$ is the probability distribution of an error δ in the measurement, then the probability of a set of measurement with errors δ_i is given by $p(\delta_1, \dots, \delta_n) = \prod_{i=1}^n f(\delta_i)$. Taking the negative logarithm gives $-\log(p(\delta_1, \dots, \delta_n)) = -\sum_{i=1}^n \log(f(\delta_i))$ and the right-hand side of this expression is a suitable cost function for a set of measurements. It is usually appropriate to set the cost of an exact measurement to be zero, by subtracting $\log(f(0))$, though this is not strictly necessary if our purpose is cost minimization. Graphs of various specific cost functions to be discussed next are shown in figure A6.4.

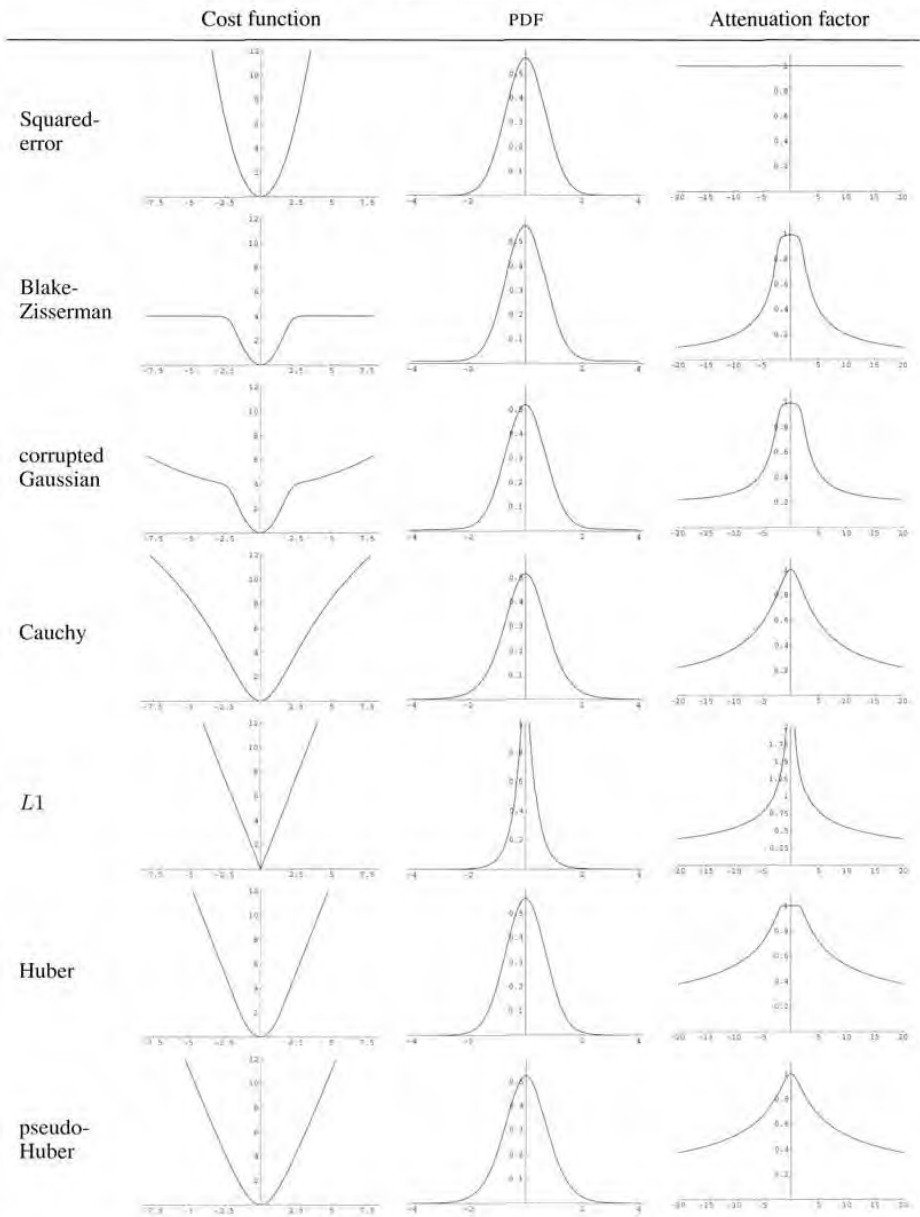


Fig. A6.4. A comparison of different cost functions, $C(\delta)$, for robust estimation. Their corresponding PDFs, $\exp(-C(\delta))$, and attenuation factors ($w = C(\delta)^{1/2}/\delta$ see text) are also shown.

Statistically based cost functions. Determination of a suitable cost function may be approached by estimating or guessing the distribution of errors for the particular measurement process involved, such as point extraction in an image. In the following list, for simplicity, we ignore the normalization constant for Gaussian distributions $(2\pi\sigma^2)^{-1/2}$, and assume that $2\sigma^2 = 1$.

- (i) **Squared error.** Assuming the data is Gaussian distributed, the Probability Distribution Function (PDF) is $p(\delta) = \exp(-\delta^2)$ which leads to a cost function

$$C(\delta) = \delta^2.$$

- (ii) **Blake-Zisserman.** The data is assumed to have a Gaussian distribution for inliers with a uniform distribution of outliers. The PDF is taken to be of the form $p(\delta) = \exp(-\delta^2) + \epsilon$. This is not actually a PDF, since it integrates to infinity. Nevertheless, it leads to a cost function of the form

$$C(\delta) = -\log(\exp(-\delta^2) + \epsilon).$$

For inliers (small δ), this approximates δ^2 , whereas for outliers (large δ) the asymptotic cost is $-\log \epsilon$. Thus, the crossover point from inliers to outliers is given approximately by $\delta^2 = -\log \epsilon$. The actual cost function used by Blake and Zisserman in [Blake-87] was $\min(\delta^2, \alpha^2)$ and $\epsilon = \exp(-\alpha^2)$.

- (iii) **Corrupted Gaussian.** The previous example has the theoretical disadvantage that it is not actually a PDF. An alternative is to model the outliers by a Gaussian with larger standard deviation, leading to a mixture model probability distribution of the form $p(\delta) = \alpha \exp(-\delta^2) + (1 - \alpha) \exp(-\delta^2/w^2)/w$ where w is the ratio of standard deviations of the outliers to the inliers, and α is the expected fraction of inliers. Then

$$C(\delta) = -\log(\alpha \exp(-\delta^2) + (1 - \alpha) \exp(-\delta^2/w^2)/w).$$

Heuristic cost functions. Next we consider cost functions justified more by heuristics and required noise-immuneness properties than by adherence to a specific noise-distribution model. For this reason they will be introduced directly as a cost function, rather than a PDF.

- (i) **Cauchy cost function.** The cost function is given by

$$C(\delta) = b^2 \log(1 + \delta^2/b^2)$$

for some constant b . For small values of δ , this curve approximates δ^2 , and the value of b determines for what range of δ this approximation is close. The cost function is derived from the Cauchy distribution $p(\delta) = 1/(\pi(1 + \delta^2))$, which is a bell-curve similar to the Gaussian, but with heavier tails.

- (ii) **The L_1 cost function.** Instead of using the sum of squares, we use the sum of absolute errors. Thus,

$$C(\delta) = 2b|\delta|$$

where $2b$ is some positive constant (which normally could just be 1). This cost function is known as the *total variation*.

- (iii) **Huber cost function.** This cost function is a hybrid between the $L1$ and least-squares cost function. Thus, we define

$$\begin{aligned} C(\delta) &= \delta^2 \text{ for } |\delta| < b \\ &= 2b|\delta| - b^2 \text{ otherwise} \end{aligned}$$

This cost function is continuous with continuous first derivative. The value of the threshold b is chosen approximately to equal the outlier threshold.

- (iv) **Pseudo-Huber cost function.** The cost function

$$C(\delta) = 2b^2(\sqrt{1 + (\delta/b)^2} - 1)$$

is very similar to the Huber cost function, but has continuous derivatives of all orders. Note that it approximates δ^2 for small δ and is linear with slope $2b$ for large δ .

A6.8.1 Properties of the different cost functions

Squared error. The most basic cost function is the squared error $C(\delta) = \delta^2$. Its main drawback is that it is not robust to outliers in the measurements, as we shall see. Because of the rapid growth of the quadratic curve, distant outliers exert an excessive influence, and can draw the cost minimum well away from the desired value.

Non-convex cost functions. The Blake-Zisserman, corrupted Gaussian and Cauchy cost functions seek to mitigate the deleterious effect of outliers by giving them diminished weight. As is seen in the plot of the first two of these, once the error exceeds a certain threshold, it is classified as an outlier, and the cost remains substantially constant. The Cauchy cost function also seeks to deemphasize the cost of outliers, but this is done more gradually. These three cost functions are non-convex, which has important effects as we will see.

Asymptotically linear cost functions. The $L1$ cost function measures the absolute value of the error. The main effect of this is to give outliers less weight compared with the squared error. The key to understanding the performance of this cost function is to observe that it acts to find the *median* of a set of data. Consider a set of real valued data $\{a_i\}$ and a cost function defined by $C(x) = \sum_i |x - a_i|$. The minimum of this function is at the median of the set $\{a_i\}$. To see this, note that the derivative of $|x - a_i|$ with respect to x is $+1$ when $x > a_i$ and -1 when $x < a_i$. Thus, the derivative is zero when there are as many values of a_i less than x as there are greater than x . Thus, the cost is minimized at the median of the values a_i . Note that the median is immune to changes in the values of data a_i that lie far from the median. The value of the cost function changes, but not the position of its minimum.

For higher dimensional data $\mathbf{a}_i \in \mathbb{R}^n$, the minimum of the cost function $C(\mathbf{x}) = \sum_i \|\mathbf{x} - \mathbf{a}_i\|$ has similar stability properties. Note that $\|\mathbf{x} - \mathbf{a}_i\|$ is a *convex* function

of x , and therefore so is a sum of such terms, $\sum_i \|x - a_i\|$. Consequently, this cost function has a single minimum (as do all convex functions).

The Huber cost function takes the form of a quadratic for small values of the error, δ , and becomes linear for values of δ beyond a given threshold. As such, it retains the outlier stability of the $L1$ cost function, while for inliers it reflects the property that the squared-error cost function gives the Maximum Likelihood estimate.

The Pseudo-Huber cost function is also near-quadratic for small δ , and linear for large δ . Thus, it may be used as a smooth approximation to the Huber cost function, and gives similar results. It is important to note that each of these three cost functions has the very desirable property of being convex.

A6.8.2 Performance of the different cost functions

To illustrate the properties of the different cost functions we will evaluate the cost $\sum_i C(x - a_i)$ for two synthetic example data sets $\{a_i\}$. Of the group of asymptotically linear cost functions, only the Huber cost function will be shown, since the other two ($L1$ and pseudo-Huber) give very similar results.

The data $\{a_i\}$ may be thought of as the outcome of an experiment to measure some quantity, with repeated measurements. The measurements are subject to random Gaussian noise, with outliers. The purpose of the estimation process is to estimate the value of the quantity by minimizing a cost function. The experiments and the results for the two data sets are described in the captions of figure A6.5 and figure A6.6.

Summary of findings. The squared-error cost function is generally very susceptible to outliers, and may be regarded as unusable as long as outliers are present. If outliers have been thoroughly eradicated, using for instance RANSAC, then it may be used.

The non-convex cost functions, though generally having a stable minimum, not much effected by outliers have the significant disadvantage of having local minima, which can make convergence to a global minimum chancy. The estimate is not strongly attracted to the minimum from outside of its immediate neighbourhood. Thus, they are not useful, unless (or until) the estimate is close to the final correct value.

The Huber cost function has the pleasant property of being convex, which makes convergence to a global minimum more reliable. The minimum is quite immune to the baleful influence of outliers since it represents a compromise between the Maximum Likelihood estimate of the inliers and the median of the outliers. The pseudo-Huber cost function is a good alternative to Huber, but use of $L1$ should be approached with care, because of its non-differentiability at the origin.

These findings were illustrated on one-dimensional data, but they carry over to higher dimensional data also.

Parameter minimization. We have seen that the Huber and related cost functions are convex, and hence have a single minimum. We refer here to the cost $C(\delta)$ as a function of the error δ . In general in problems such as structure from motion, the error δ itself is a non-linear function of the parameters (such as camera and point positions). For this reason, the total cost expressed as a function of the motion and structure parameters

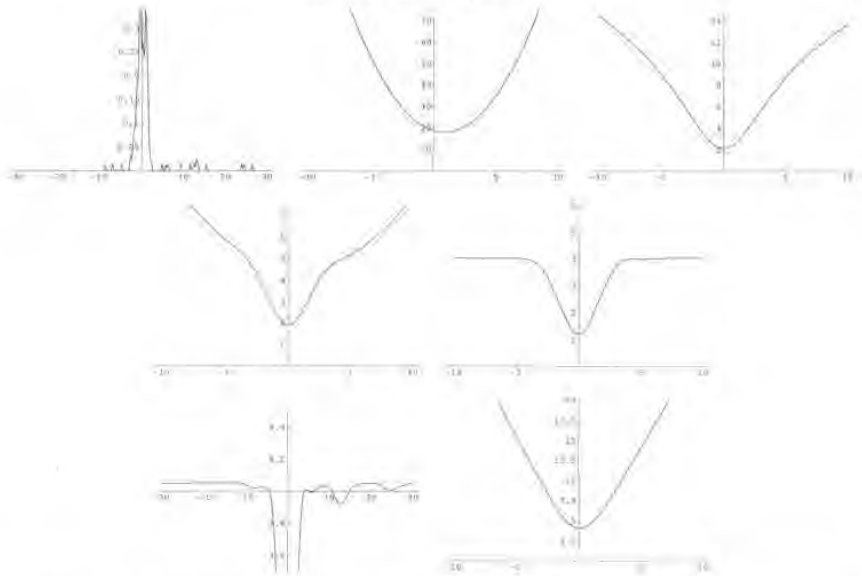


Fig. A6.5. The data $\{a_i\}$ (illustrated in the top left graph) consists of a set of measurements centred around 0 with unit Gaussian noise, plus 10% of outliers biased towards the right of the true value. The graphs of $\sum_i C(|x - a_i|)$ correspond (left-to-right and top-to-bottom) to the cost functions Squared error, Cauchy, corrupted-Gaussian, Blake-Zisserman, a zoom of the Blake-Zisserman, and Huber cost functions. Note that the minimum of the squared-error cost function is pulled significantly to the right by the outliers, whereas the other cost-functions are relatively outlier-independent.

The Blake-Zisserman cost function, which is based most nearly on the distribution of the data, has a very clear minimum. However, close examination (the zoomed plot) shows an undesirable characteristic, which is the presence of local minima near each of the outliers. An iterative method to find the cost minimum will fail if it is initiated outside the narrow basin of attraction surrounding the minimum.

By contrast, the Huber cost function is convex, which means the estimate will be drawn towards the single minimum from any initialization point.

can not be expected to be convex, and local minima are inevitable. Nevertheless, an important principle is:

- choose a parameterization in which the error is as close as possible to being a linear function of the parameters, at least locally.

Observing this principle will lead to simpler cost surfaces with fewer local minima, and generally quicker convergence.

Cost functions and least-squares. Usually cost functions of the type we have discussed are used in the context of a parameter minimization procedure, such as Levenberg-Marquardt or Newton iteration. Commonly, these procedures seek to minimize the norm of some vector Δ depending on a set of parameters $\mathbf{p}$. Thus, they minimize $\|\Delta(\mathbf{p})\|^2$ over all choices of the parameter vector $\mathbf{p}$. For instance in a structure and motion problem we may seek to minimize $\sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 = \sum_i \|\delta_i\|^2 = \|\Delta\|^2$ where the values $\mathbf{x}_i$ are measured image coordinates, the $\hat{\mathbf{x}}_i$ are the predicted values de-

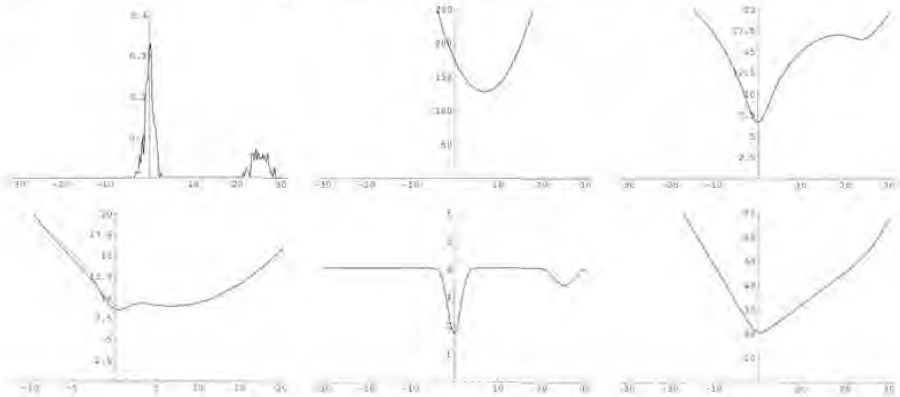


Fig. A6.6. In this experiment, as in figure A6.5, the main part of the data (70%) is centred at the origin, with 30% of “outliers” concentrated in a block away from the origin (see top left graph). This type of measurement distribution is quite realistic in many imaging scenarios, for instance where point or edge measurement is confused by ghost edges. The cost functions in order from the top are: Square error, Cauchy, corrupted Gaussian, Blake-Zisserman and Huber. The Squared error cost function finds the mean of the measurement distribution, which is significantly pulled to the right by the block of outliers. The effect of the outlier block on the non-convex cost functions is also shown clearly here. Because of the non-convexity, the total cost function does not have a single minimum, but rather two minima around the separate blocks of measurements. Because of its convexity, the Huber cost function has a single minimum, which is located close to the median of the data, and is hardly influenced by the presence of the 30% of outliers.

rived from the current parameter values, and Δ is the vector formed by concatenating the individual error vectors δ_i .

Since minimization of a squared vector norm $\|\Delta\|^2$ is built into most implementations of Levenberg-Marquardt, we need to see how to apply the robust cost function in this case. The answer is to replace each vector δ_i by a weighted vector $\delta'_i = w_i \delta_i$ such that

$$\|\delta'_i\|^2 = w_i^2 \|\delta_i\|^2 = C(\|\delta_i\|)$$

for then $\sum_i C(\|\delta_i\|) = \sum_i \|\delta'_i\|^2$ as desired. From this equality, we find

$$w_i = C(\|\delta_i\|)^{1/2} / \|\delta_i\|. \quad (\text{A6.18})$$

Thus, the minimization problem is to minimize $\|\Delta'\|^2$ where Δ' is the vector obtained by concatenating the vectors $\delta'_i = w_i \delta_i$, and each w_i is computed from (A6.18). Note that w_i is a function of $\|\delta_i\|$, which normally seeks to attenuate the cost of the outliers. This attenuation function is shown in the final column of figure A6.4 for the different cost functions. For the Squared error cost function, the attenuation factor is 1, meaning no attenuation occurs. For the other cost functions, there is little attenuation within an inlier region, and points outside this region are attenuated to different degrees.

A6.9 Parametrization

In the sort of iterative estimate problems we are considering here, an important consideration is how the solution space is to be parametrized. Most of the geometric entities that we consider do not have a simple Euclidean parametrization. For example, in solving a geometric optimization problem, we often wish to iterate over all camera orientations, represented by rotations. Rotations are most conveniently represented for computational purposes by rotation matrices, but this is a major overparametrization. The set of 3D rotations is naturally 3-dimensional (in fact, it forms a 3-dimensional Lie group, $SO(3)$), and we may wish to parametrize it with only 3 parameters.

Homogeneous vectors or matrices are another common type of representation. It is tempting to use a parametrization of a homogeneous n -vector just by using the components of the vector itself. However, there are really only $n - 1$ degrees of freedom in a homogeneous n -vector, and it is sometimes advantageous to parametrize it with only $n - 1$ parameters.

Gauge freedom. There has been much attention paid to *gauge freedom* and *gauge independence* in recent papers (for instance [McLauchlan-00]). In this context, the word gauge means a coordinate system for a parameter set, and gauge-freedom essentially refers to a change in the representation of the parameter set that does not essentially change the underlying geometry, and hence has no effect on the cost function. The most important gauge freedoms commonly encountered are projective or other ambiguities, such as those arising in reconstruction problems. However, the scale ambiguity of homogeneous vectors can be counted as gauge freedom also. Gauge freedoms in the parametrization of an optimization problem cause the normal equations to be singular, and hence allow multiple solutions. This problem is avoided by the regularization (or enhancement) step in Levenberg-Marquardt, but there is evidence that excessive gauge freedoms, causing slop in the parametrization, can lead to slower convergence. In addition, when gauge freedoms are present the covariance matrix of the estimated parameters is troublesome, in that there will be infinite variance in unconstrained parameter directions. For instance, it makes no sense to talk of the covariance matrix of an estimated homogeneous vector, unless the scale of the vector is constrained. Therefore, in the following pages, we will give some methods for obtaining minimal parametrizations for certain common geometric parameter sets.

What makes a good parametrization? The foremost requirement of a good parametrization is that it be singularity-free, at least in the areas that are visited during the course of an iterative optimization. This means that the parametrization should be locally continuous, differentiable and one-to-one – in short a diffeomorphism. A simple example of what is meant by this is given by the latitude-longitude parametrization of a sphere, that is, spherical-polar coordinates. This has a singularity at the poles, where the mapping from the lat-long coordinates to a neighbourhood of the pole is not one-to-one. The difficulty is that the point with coordinates (latitude = 89° , longitude = 0°) is very close to the point (latitude = 89° , longitude = 90°) – they are both points very close to the pole. However, they are a long way apart in parameter space. To see

the effect of this, suppose that an optimization is taking place on the sphere, tracking down a minimum of a cost function, which exists at the point $(89^\circ, 90^\circ)$. If the current estimate is proceeding in the general direction of this minimum, up the line of zero longitude, when it gets near the pole it will find that although close to the minimum, it can not get there without a long detour in lat-long parameter space. The difficulty is that arbitrarily close points on the sphere, in the neighbourhood of the singularity (the pole) can have large differences in parameter values. The same sort of thing happens with representations of rotations using Euler angles.

Now, we move on to consider some specific parametrizations.

A6.9.1 Parametrization of 3D rotations

Using the angle-axis formula of (A4.9-p584) we may parametrize a 3×3 rotation by a 3-vector $\mathbf{t}$. This represents a rotation through an angle $\|\mathbf{t}\|$ about the axis determined by the vector $\mathbf{t}$. We denote the corresponding rotation matrix by $R(\mathbf{t})$.

Regarding this representation, we may make certain simple observations:

- (i) The identity map (no rotation) is represented by the zero vector $\mathbf{t} = \mathbf{0}$.
- (ii) If some rotation R is represented by a vector $\mathbf{t}$, then the inverse rotation is represented by $-\mathbf{t}$. In symbols: $R(\mathbf{t})^{-1} = R(-\mathbf{t})$.
- (iii) If $\mathbf{t}$ is small, then the rotation matrix is approximated by $I + [\mathbf{t}]_\times$.
- (iv) For small rotations represented by $\mathbf{t}_1$ and $\mathbf{t}_2$, the composite rotation is represented to first-order by $\mathbf{t}_1 + \mathbf{t}_2$. In other words $R(\mathbf{t}_1)R(\mathbf{t}_2) \approx R(\mathbf{t}_1 + \mathbf{t}_2)$. Thus the mapping $\mathbf{t} \mapsto R(\mathbf{t})$ is to first order a group isomorphism for small $\mathbf{t}$. In fact, for small $\mathbf{t}$, the map is an isometry (distance preserving map) in terms of the distance between two rotations R_1 and R_2 defined to equal to the angle of the rotation $R_1 R_2^{-1}$.
- (v) Any rotation may be represented as $R(\mathbf{t})$ for some $\mathbf{t}$ such that $\|\mathbf{t}\| \leq \pi$. That is, any rotation is a rotation through an angle of at most π radians about some axis. The mapping $\mathbf{t} \mapsto R(\mathbf{t})$ is one-to-one for $\|\mathbf{t}\| < \pi$ and two-to-one for $\|\mathbf{t}\| < 2\pi$. If $\|\mathbf{t}\| = 2\pi$, then $R(\mathbf{t})$ is the identity map, regardless of $\mathbf{t}$. Thus, the parametrization has a singularity at $\|\mathbf{t}\| = 2\pi$.
- (vi) **Normalization:** In parametrizing rotations by a vector $\mathbf{t}$ it is best to maintain the condition that $\|\mathbf{t}\| \leq \pi$, in order to keep away from the singularity when $\|\mathbf{t}\| = 2\pi$. If $\|\mathbf{t}\| > \pi$, then it may be replaced by the vector $(\|\mathbf{t}\| - 2\pi)\mathbf{t}/\|\mathbf{t}\| = \mathbf{t}(1 - 2\pi/\|\mathbf{t}\|)$, which represents the same rotation.

A6.9.2 Parametrization of homogeneous vectors

The quaternion representation of a rotation (section A4.3.3(p585)) is a redundant representation in that it contains 4 parameters where 3 will suffice. The angle-axis representation on the other hand is a minimum parametrization. Many entities in projective geometry are represented by homogeneous quantities, either vectors or matrices, for instance points in projective space or the fundamental matrix to name a few. For computational purposes, it is possible to represent such quantities as vectors with a min-

imum number of parameters in a similar way to which the angle-axis representation gives an alternative to the quaternion representation of a rotation.

Let $\mathbf{v}$ be a vector of any dimension, and represent by $\hat{\mathbf{v}}$ the unit vector $(\text{sinc}(\|\mathbf{v}\|/2)\mathbf{v}^T, \cos(\|\mathbf{v}\|/2))^T$. This mapping $\mathbf{v} \mapsto \hat{\mathbf{v}}$ maps the disk of radius π (that is, the set of vectors of length at most π) smoothly and one-to-one onto the set of unit vectors $\hat{\mathbf{v}}$ with non-negative final coordinate. Thus, it provides a mapping onto the set of homogeneous vectors. The only difficult point with this mapping is that it takes any vector of length 2π to the same vector $(0, -1)^T$. However, this singular point may be avoided by renormalizing any vector $\mathbf{v}$ of length $\|\mathbf{v}\| > \pi$, replacing it with $(\|\mathbf{v}\| - 2\pi)\mathbf{v}/\|\mathbf{v}\|$ which represents the same homogeneous vector $\hat{\mathbf{v}}$.

A6.9.3 Parametrization of the n -sphere

Commonly it occurs in geometric optimization problems that some vector of parameters is required to lie on a unit sphere. As an example, consider a complete Euclidean bundle-adjustment with two views. The two cameras may be taken as $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}' = [\mathbf{R} \mid \mathbf{t}]$, where $\mathbf{R}$ is a rotation and $\mathbf{t}$ is a translation. In addition, 3D points $\mathbf{X}_i$ are defined, which are to map via the two camera matrices to image points. This defines an optimization problem in which the parameters are $\mathbf{R}$, $\mathbf{t}$ and the points $\mathbf{X}_i$, and the cost to be minimized is a geometric residual of the projections with respect to the image measurements. In this problem, there is an overall scale ambiguity, and this is conveniently resolved by requiring that $\|\mathbf{t}\| = 1$.

A minimum parametrization for the rotation matrix $\mathbf{R}$ is given by the rotation parametrization of section A6.9.1. Similarly, the points $\mathbf{X}_i$ are conveniently parametrized as homogeneous 4-vectors, using the parametrization of section A6.9.2. We consider in this section how one may parametrize the unit vector $\mathbf{t}$. Note that we can not simply parametrize $\mathbf{t}$ as a homogeneous vector, since change of sign of $\mathbf{t}$ changes the projection $\mathbf{P}'\mathbf{X} = [\mathbf{R} \mid \mathbf{t}]\mathbf{X}$.

The same problem arises in multiple-view Euclidean bundle-adjustment. In this case we have many camera matrices $\mathbf{P}^i = [\mathbf{R}^i \mid \mathbf{t}^i]$ and we may fix $\mathbf{P}^0 = [\mathbf{I} \mid \mathbf{0}]$. The set of translations $\mathbf{t}^i$ for all $i > 0$ are subject to scale ambiguity. We can minimally parametrize the translations by requiring that $\|\mathbf{T}\| = 1$, where $\mathbf{T}$ is the vector formed by concatenating all the $\mathbf{t}^i$ for $i > 0$.

There may be several ways of parametrizing a unit vector. We consider one particular parametrization here based on a local parametrization of the tangent plane to the unit sphere. We consider a sphere of dimension n , which consists of the set of $(n+1)$ -vectors of unit length. Let $\mathbf{x}$ be a such a vector. Let $\mathbf{H}_{\mathbf{v}(\mathbf{x})}$ be a Householder matrix (see section A4.1.2(p580)) such that $\mathbf{H}_{\mathbf{v}(\mathbf{x})}\mathbf{x} = (0, \dots, 0, 1)^T$. Thus, we have transformed the vector $\mathbf{x}$ to lie along the coordinate axis. Now, we consider a parametrization of the unit sphere in the vicinity of $(0, \dots, 0, 1)^T$. Such a parametrization is a map $\mathbb{R}^n \rightarrow S^n$ that is well behaved in the vicinity of the origin. There are many choices, of which two possibilities are

- (i) $f(\mathbf{y}) = \hat{\mathbf{y}}/\|\hat{\mathbf{y}}\|$ where $\hat{\mathbf{y}} = (\mathbf{y}^T, 1)^T$
- (ii) $f(\mathbf{y}) = (\text{sinc}(\|\mathbf{y}\|/2)\mathbf{y}^T, \cos(\|\mathbf{y}\|/2))^T$.

Both these functions map the origin $(0, \dots, 0)^T$ to $(0, \dots, 0, 1)^T$, and their Jacobian is $\partial f / \partial \mathbf{y} = [\mathbf{I} | \mathbf{0}]^T$. Note that although we are interested in these functions just as local parametrizations, the first provides a parametrization for half the sphere, whereas for $\|\mathbf{y}\| \leq \pi$ the second map parametrizes the whole of the sphere with no singularity except at $\|\mathbf{y}\| = 2\pi$.

The composite map $\mathbf{y} \mapsto \mathbf{H}_{\mathbf{v}(\mathbf{x})}f(\mathbf{y})$ provides a local parametrization for a neighbourhood of the point $\mathbf{x}$ on the sphere (note here that we should write $\mathbf{H}_{\mathbf{v}(\mathbf{x})}^{-1}$, but $\mathbf{H}_{\mathbf{v}(\mathbf{x})} = \mathbf{H}_{\mathbf{v}(\mathbf{x})}^{-1}$). The Jacobian of this map is simply $\mathbf{H}_{\mathbf{v}(\mathbf{x})}[\mathbf{I} | \mathbf{0}]^T$, which consists of the first n columns of the Householder matrix, and hence is easy to compute.

In minimization problems, we usually need to compute a Jacobian matrix $\partial C / \partial \mathbf{y}$, of a vector valued cost function C with respect to a set of parameters $\mathbf{y}$. In the case where the parameters are constrained to lie on a sphere S^n in $\mathbb{R}^{n+1}$ the cost function is nonetheless usually defined for all values of the parameter $\mathbf{x}$ in $\mathbb{R}^{n+1}$. As an example, in the Euclidean bundle-adjustment problem considered at the start of this section, the cost function (for instance residual reprojection error) can be defined for all pairs of cameras $\mathbf{P} = [\mathbf{I} | \mathbf{0}]$ and $\mathbf{P}' = [\mathbf{R} | \mathbf{t}]$ with $\mathbf{t}$ taking any value. Nevertheless, we may wish to minimize the cost function constraining $\mathbf{t}$ to lie on a sphere.

Thus, consider the case where a cost function $C(\mathbf{x})$ is defined for $\mathbf{x} \in \mathbb{R}^{n+1}$, but we parametrize $\mathbf{x}$ to lie on a sphere by setting $\mathbf{x} = \mathbf{H}_{\mathbf{v}(\mathbf{x})}f(\mathbf{y})$ with $\mathbf{y} \in \mathbb{R}^n$. In this case, we see that

$$\mathbf{J} = \frac{\partial C}{\partial \mathbf{y}} = \frac{\partial C}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \frac{\partial C}{\partial \mathbf{x}} \mathbf{H}_{\mathbf{v}(\mathbf{x})}[\mathbf{I} | \mathbf{0}]^T.$$

In summary, by using local parametrizations, a parameter vector may be constrained to lie on an n -sphere with a modest added computational cost compared with the over-parametrization of allowing the vector to vary over the whole of $\mathbb{R}^{n+1}$. The key points of the method are as follows.

- (i) Store the parameter vector $\mathbf{x} \in \mathbb{R}^{n+1}$, satisfying $\|\mathbf{x}\| = 1$.
- (ii) In forming the linear update equations, compute the Jacobian matrix $\partial C / \partial \mathbf{x}$, and multiply it by $\mathbf{H}_{\mathbf{v}(\mathbf{x})}[\mathbf{I} | \mathbf{0}]^T$ to obtain the Jacobian with respect to a minimal parameter set $\mathbf{y}$. Multiplication of $\partial C / \partial \mathbf{x}$ by $\mathbf{H}_{\mathbf{v}(\mathbf{x})}[\mathbf{I} | \mathbf{0}]^T$ is efficiently carried out by the method of (A4.4–p580).
- (iii) The iteration step provides an increment parameter vector $\delta \mathbf{y}$. Compute the new value of $\mathbf{x} = \mathbf{H}_{\mathbf{v}(\mathbf{x})}f(\delta \mathbf{y})$.

Essentially the same method of using local-parametrizations may be used more generally in other situations where a minimal parametrization is required. For instance, in section 11.4.2(p285) it was seen how the fundamental matrix may be parametrized locally with a minimum number of parameters, but no minimal parametrization can cover the whole set of fundamental matrices.

A6.10 Notes and exercises

- (i) We prove in various steps the form of the pseudo-inverse of a block matrix given in (A6.13–p606).

- (a) Recall that $H^+ = G(G^T H G)^{-1} G^T$ if and only if $N_L(G) = N_L(H)$ (see section A5.2(p590)).
 - (b) Let G be invertible. Then $(GHG^T)^+ = G^{-T} H^+ G^{-1}$ if and only if $N_L(H)G^T = N_L(H)G^{-1}$.
 - (c) Applying this condition to (A6.12-p605) the necessary and sufficient condition for (A6.13-p606) to be valid is that $N_L(U - WV^{-1}W^T) \subseteq N_L(Y) = N_L(W)$.
 - (d) With U , V and W defined in terms of A and B as in (A6.11-p605) this is equivalent to the condition $\text{span}(A) \cap \text{span}(B) = \emptyset$.
- (ii) Investigate conditions under which the condition $\text{span}(A) \cap \text{span}(B) = \emptyset$ is true. It may be interpreted as meaning that the effects of varying the parameters a (for instance camera parameters) and the effects of varying b (point parameters) can not be complementary. Clearly this is not the case with (for instance) unconstrained projective reconstruction where both cameras and points may vary without affecting the measurements. In such a case, the variance of parameters a and b is infinite in directions δ_a and δ_b such that $A\delta_a = B\delta_b$.

Appendix 7

Some Special Plane Projective Transformations

Projective transformations (homographies) can be classified according to the algebraic and geometric multiplicity of their eigenvalues. The algebraic multiplicity of an eigenvalue is the number of times the root is repeated in the characteristic equation. The geometric multiplicity may be determined from the rank of the matrix $(H - \lambda I)$, where H is the homography and λ the eigenvalue. A complete classification is given in projective geometry textbooks such as [Springer-64]. Here we mention several special cases which are important in practical situations, and occur at several points throughout this book. The description will be for plane transformations where H is a 3×3 matrix, but the generalization to 3-space transformations is straightforward.

The special forms are significant because H satisfies a number of relationships (remember the only restriction a general projective transformation is that it has full rank). Since H satisfies constraints it has fewer degrees of freedom, and consequently can be computed from fewer correspondences than a general projective transformation. The special transformations also have richer geometry and invariants than in the general case.

Note that unlike the special forms (affine etc.) discussed in chapter 2, which form subgroups, the following special projectivities do not form subgroups in general since they are not closed under multiplication. They do form a subgroup if all the elements have coincident fixed points and lines (i.e. differing only in their eigenvalues).

A7.1 Conjugate rotations

A rotation matrix R has eigenvalues $\{1, e^{i\theta}, e^{-i\theta}\}$, with corresponding eigenvectors $\{\mathbf{a}, \mathbf{I}, \mathbf{J}\}$, where $\mathbf{a}$ is the rotation axis, i.e. $R\mathbf{a} = \mathbf{a}$, θ is the angle of rotation about the axis, and $\mathbf{I}$ and $\mathbf{J}$ (which are complex conjugate) are the circular points for the plane orthogonal to $\mathbf{a}$. Suppose a projective transformation between two planes has the form

$$H = T R T^{-1}$$

with T a general projective transformation; then H is a *conjugate rotation*. Eigenvalues are preserved under a conjugate relationship¹ so the eigenvalues of the projective transformation H are also $\{1, e^{i\theta}, e^{-i\theta}\}$ up to a common scale.

¹ Conjugacy is also referred to as a "similarity transformation". This meaning of "similarity" is unrelated to its use in this book as an isometry plus scaling transformation.

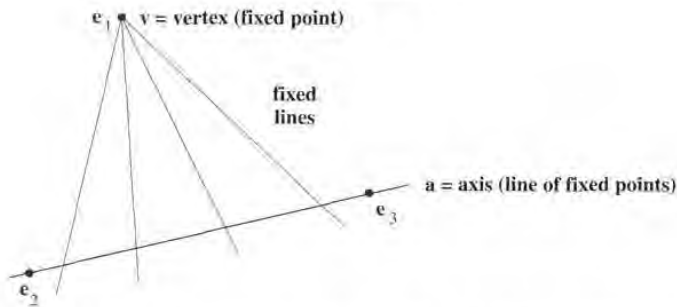


Fig. A7.1. **A planar homology.** A planar homology is a plane projective transformation which has a line **a** of fixed points, called the *axis*, and a distinct fixed point **v**, not on the line, called the *centre* or *vertex* of the homology. There is a pencil of fixed lines through the vertex. Algebraically, two of the eigenvalues of the transformation matrix are equal, and the fixed line corresponds to the 2D invariant space of the matrix (here the repeated eigenvalues are λ_2 and λ_3).

Consider two images obtained by a camera rotating about its centre (as in figure 2.5-(p36)b); then as shown in section 8.4.2(p204), the images are related by a conjugate rotation. In this case the complex eigenvalues determine the angle (θ) through which the camera rotates, and the eigenvector corresponding to the real eigenvalue is the vanishing point of the rotation axis. Note that θ – a metric invariant – can be measured directly from the projective transformation.

A7.2 Planar homologies

A plane projective transformation H is a planar homology if it has a line of fixed points (called the *axis*), together with a fixed point (called the *vertex*) not on the line, see figure A7.1. Algebraically, the matrix has two equal and one distinct eigenvalues and the eigenspace corresponding to the equal eigenvalues is two-dimensional. The axis is the line through the two eigenvectors (i.e. points) spanning this eigenspace. The vertex corresponds to the other eigenvector. The ratio of the distinct eigenvalue to the repeated one is a characteristic invariant μ of the homology (i.e. the eigenvalues are, up to a common scale factor, $\{\mu, 1, 1\}$).

Properties of a planar homology include:

- Lines joining corresponding points intersect at the vertex, and corresponding lines (i.e. lines through two pairs of corresponding points) intersect on the axis. This is an example of Desargues' Theorem. See figure A7.2a.
- The cross ratios defined by the vertex, a pair of corresponding points, and the intersection of the line joining these points with the line of fixed points, are the same for all points related by the homology. See figure A7.2b.
- For curves related by a planar homology, corresponding tangents (the limit of neighbouring points defining corresponding lines) intersect on the axis.
- The vertex (2 dof), axis (2 dof) and invariant (1 dof) are sufficient to define the homology completely. A planar homology thus has 5 degrees of freedom.

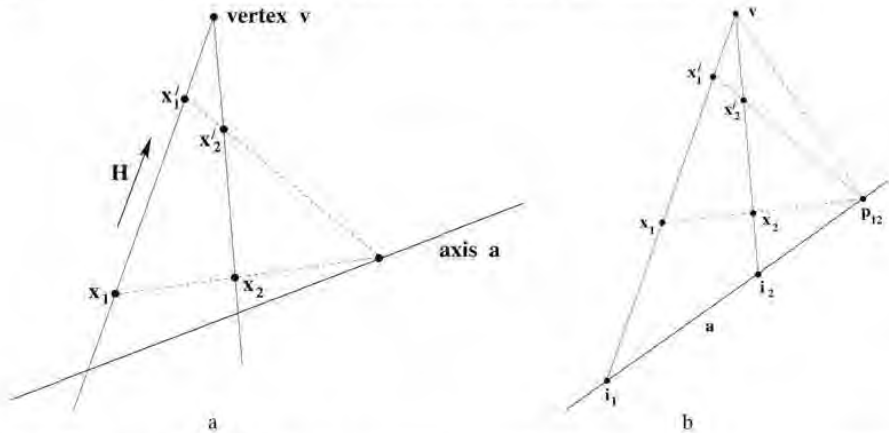


Fig. A7.2. **Homology transformation.** (a) Under the transformation points on the axis are mapped to themselves; each point off the axis lies on a fixed line through v intersecting a and is mapped to another point on the line. Consequently, corresponding point pairs $x \leftrightarrow x'$ and the vertex of the homology are collinear. Corresponding lines – i.e. lines through pairs of corresponding points – intersect on the axis: for example, the lines $\langle x_1, x_2 \rangle$ and $\langle x'_1, x'_2 \rangle$. (b) The cross ratio defined by the vertex v , the corresponding points x, x' and the intersection of their join with the axis i , is the characteristic invariant of the homology, and is the same for all corresponding points. For example, the cross ratios of the four points $\{v, x'_1, x_1, i_1\}$ and $\{v, x'_2, x_2, i_2\}$ are equal since they are perspectively related by lines concurrent at p_{12} . It follows that the cross ratio is the same for all points related by the homology.

- 3 matched points are sufficient to compute a planar homology. The 6 degrees of freedom of the point matches over-constrain the 5 degrees of freedom of the homology.

A planar homology arises naturally in an image of two planes related by a perspectivity of 3-space (i.e. lines joining corresponding points on the two planes are concurrent). An example is the transformation between the image of a planar object and the image of its shadow on a plane. In this case the axis is the imaged intersection of the two planes, and the vertex is the image of the light source, see figure 2.5(p36)c.

Parametrization. The projective transformation representing the homology can be parametrized directly in terms of the 3-vectors representing the axis a and vertex v , and the characteristic ratio μ , as

$$H = I + (\mu - 1) \frac{va^T}{v^T a}$$

where I is the identity. It is straightforward to verify that the inverse transformation is given by

$$H^{-1} = I + \left(\frac{1}{\mu} - 1 \right) \frac{va^T}{v^T a}.$$

The eigenvectors are

$$\{e_1 = v, e_2 = a_1^\perp, e_3 = a_2^\perp\}$$

with corresponding eigenvalues

$$\{\lambda_1 = \mu, \lambda_2 = 1, \lambda_3 = 1\}$$

where $\mathbf{a}_i^\perp$ are two vectors that span the space orthogonal to the 3-vector $\mathbf{a}$, i.e. $\mathbf{a}^\top \mathbf{a}_i^\perp = 0$ and $\mathbf{a} = \mathbf{a}_1^\perp \times \mathbf{a}_2^\perp$.

If the axis or the vertex is at infinity then the homology is an affine transformation. Algebraically, if $\mathbf{a} = (0, 0, 1)^\top$, then the axis is at infinity; or if $\mathbf{v} = (v_1, v_2, 0)^\top$, then the vertex is at infinity; and in both cases the transformation matrix $\mathbf{H}$ has last row $(0, 0, 1)$.

Planar harmonic homology. A specialization of a planar homology is the case that the cross ratio is harmonic ($\mu = -1$). This planar homology is called a planar harmonic homology and has 4 degrees of freedom since the invariant is known. The transformation matrix $\mathbf{H}$ obeys $\mathbf{H}^2 = \mathbf{I}$, i.e. the transformation is a square root of the identity, which is called an involution (also a collineation of period 2). The eigenvalues are, up to a common scale factor, $\{-1, 1, 1\}$. Two pairs of point correspondences determine $\mathbf{H}$.

In a perspective image of a plane object with a bilateral symmetry, corresponding points in the image are related by a planar harmonic homology. The axis of the homology is the image of the symmetry axis. Algebraically, $\mathbf{H}$ is a conjugate reflection where the conjugating element is a plane projective transformation. In an affine image (generated by an affine camera) the resulting transformation is a *skewed symmetry*, and the conjugating element is a plane affine transformation. For a skewed symmetry the vertex is at infinity, and the lines joining corresponding points are parallel.

The harmonic homology can be parametrized as

$$\mathbf{H} = \mathbf{H}^{-1} = \mathbf{I} - 2 \frac{\mathbf{v}\mathbf{a}^\top}{\mathbf{v}^\top \mathbf{a}}.$$

Again, if the axis or vertex is at infinity then the transformation is affine.

A7.3 Elations

An elation has a line of fixed points (the axis), and a pencil of fixed lines intersecting in a point (the vertex) on the axis. It may be thought of as the limit of a homology where the vertex is on the line of fixed points. Algebraically, the matrix has three equal eigenvalues, but the eigenspace is 2-dimensional. It may be parametrized as

$$\mathbf{H} = \mathbf{I} + \mu \mathbf{v}\mathbf{a}^\top \quad \text{with} \quad \mathbf{a}^\top \mathbf{v} = 0 \quad (\text{A7.1})$$

where $\mathbf{a}$ is the axis, and $\mathbf{v}$ the vertex. The eigenvalues are all unity. The invariant space of $\mathbf{H}$ is spanned by $\mathbf{a}_1^\perp, \mathbf{a}_2^\perp$. This is a line (pencil) of fixed points (which includes $\mathbf{v}$ since $\mathbf{a}^\top \mathbf{v} = 0$). The invariant space of $\mathbf{H}^\top$ is spanned by vectors $\mathbf{v}_1^\perp, \mathbf{v}_2^\perp$ orthogonal to $\mathbf{v}$. This is a pencil of fixed lines, $\mathbf{l} = \alpha \mathbf{v}_1^\perp + \beta \mathbf{v}_2^\perp$, for which $\mathbf{l}^\top \mathbf{v} = 0$, i.e. all lines of the pencil are concurrent at the point $\mathbf{v}$.

An elation has 4 degrees of freedom; one less than a homology due to the constraint $\mathbf{a}^\top \mathbf{v} = 0$. It is defined by the axis $\mathbf{a}$ (2 dof), the vertex $\mathbf{v}$ on $\mathbf{a}$ (1 dof) and the parameter μ (1 dof). It can be determined from 2 point correspondences.

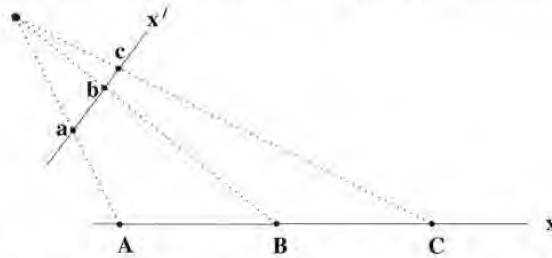


Fig. A7.3. **A line perspectivity.** The lines joining corresponding points (*a, A* etc.) are concurrent. Compare with figure A7.4.

Elations often arise in practice as conjugate translations. Consider a pattern on a plane that repeats by a translation $\mathbf{t} = (t_x, t_y)^T$, for example identical windows on the wall of a building. This action is represented on the plane of the wall as

$$H_E = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

which is an elation where $\mathbf{v} = (t_x, t_y, 0)^T$ is the translation direction of the repetition, and $\mathbf{a} = (0, 0, 1)^T$ is the line at infinity. In an image of the wall the windows are related by a conjugate translation $H = TH_E T^{-1}$, where T is the projectivity which maps the plane of the wall to the image. The image transformation H is also an elation. The vertex of this elation is the vanishing point of the translation direction, and the axis is the vanishing line of the wall plane.

A7.4 Perspectivities

One other special case of a projectivity is a *perspectivity*, which is shown in figure A7.3 for a 1D projectivity on the plane. The distinctive property of a perspectivity is that lines joining corresponding points are concurrent. The difference between a perspectivity and projectivity is made clear by considering the composition of two perspectivities. As shown in figure A7.4 the composition of two perspectivities is *not* in general a perspectivity. However, the composition *is* a projectivity because a perspectivity is a projectivity, and projectivities form a group (closed), so that the composition of two projectivities is a projectivity. To summarize:

- *The composition of two (or more) perspectivities is a projectivity, but not, in general, a perspectivity.*

A central projection image of a world plane, as in figure 2.3(p34), is an example of a 2D perspectivity between different planes. Notice that identifying the projectivity as a perspectivity requires the embedding of the planes in 3-space.

Finally, imagine that the planes and camera centre of figure 2.3(p34) are mapped (by another perspectivity) onto one of the planes. Then this imaged perspectivity is now a map between points on the same plane, and is seen to be a planar homology (section A7.2).

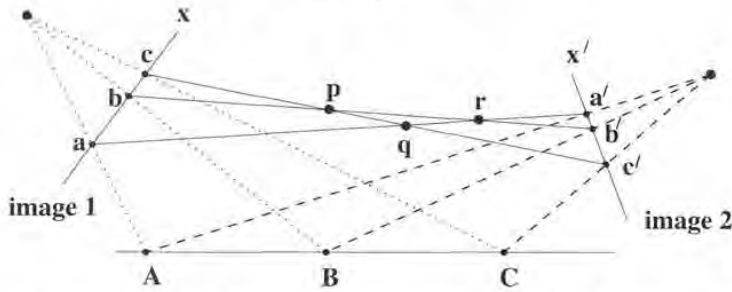


Fig. A7.4. **A line projectivity.** Points $\{a, b, c\}$ are related to points $\{A, B, C\}$ by a line-to-line perspectivity. Points $\{a', b', c'\}$ are also related to points $\{A, B, C\}$ by a perspectivity. However, points $\{a, b, c\}$ are related to points $\{a', b', c'\}$ by a projectivity; they are not related by a perspectivity because lines joining corresponding points are not concurrent. In fact the pairwise intersections result in three distinct points $\{p, q, r\}$.

Further reading. [Springer-64] classifies projectivities and covers special cases, e.g. planar homologies. Planar homologies appear in many guises: modelling imaged shadow relations in [VanGool-98]; modelling imaged extruded surfaces in [Zisserman-95a]; and modelling relations for planar pose recovery in [Basri-99]. The parametrization of planar homologies is given in Viéville and Lingrand [Viéville-95]. Elations appear in the grouping of imaged repeated patterns on a plane [Schaffalitzky-99, Schaffalitzky-00b] and in 3-space they appear in the generalized bas-relief ambiguity [Kriegman-98].

Bibliography

- [Agrawal-03] M. Agrawal and L. Davis. Camera calibration using spheres: A dual-space approach. Research Report CAR-TR-984, Center for Automation Research, University of Maryland, 2003.
- [Aloimonos-90] J. Y. Aloimonos. Perspective approximations. *Image and Vision Computing*, 8(3):177–192, August 1990.
- [Anandan-02] P. Anandan and M. Irani. Factorization with uncertainty. *International Journal of Computer Vision*, 49(2/3):101–116, 2002.
- [Armstrong-94] M. Armstrong, A. Zisserman, and P. Beardsley. Euclidean reconstruction from uncalibrated images. In *Proc. British Machine Vision Conference*, pages 509–518, 1994.
- [Armstrong-96a] M. Armstrong. *Self-Calibration from Image Sequences*. PhD thesis, University of Oxford, England, 1996.
- [Armstrong-96b] M. Armstrong, A. Zisserman, and R. Hartley. Self-calibration from image triplets. In *Proc. European Conference on Computer Vision*, LNCS 1064/5, pages 3–16. Springer-Verlag, 1996.
- [Astrom-98] K. Åström and A. Heyden. Continuous time matching constraints for image streams. *International Journal of Computer Vision*, 28(1):85–96, 1998.
- [Avidan-98] S. Avidan and A. Shashua. Threading fundamental matrices. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, pages 124–140, 1998.
- [Baillard-99] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 559–565, June 1999.
- [Barrett-92] E. B. Barrett, M. H. Brill, N. N. Haag, and P. M. Payton. Invariant linear methods in photogrammetry and model-matching. In J. L. Mundy and A. Zisserman, editors, *Geometric invariance in computer vision*. MIT Press, Cambridge, 1992.
- [Bascle-98] B. Bascle and A. Blake. Separability of pose and expression in facial tracing and animation. In *Proc. International Conference on Computer Vision*, pages 323–328, 1998.
- [Basri-99] R. Basri and D. Jacobs. Projective alignment with regions. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1158–1164, 1999.
- [Bathe-76] K.-J. Bathe and E. Wilson. *Numerical methods in finite element analysis*. Prentice Hall, 1976.
- [Beardsley-92] P. A. Beardsley, D. Sinclair, and A. Zisserman. Ego-motion from six points. Insight meeting, Catholic University Leuven, February 1992.
- [Beardsley-94] P. A. Beardsley, A. Zisserman, and D. W. Murray. Navigation using affine structure and motion. In *Proc. European Conference on Computer Vision*, LNCS 800/801, pages 85–96. Springer-Verlag, 1994.
- [Beardsley-95a] P. A. Beardsley and A. Zisserman. Affine calibration of mobile vehicles. In *Europe-China workshop on Geometrical Modelling and Invariants for Computer Vision*, pages 214–221. Xidan University Press, Xi'an, China, 1995.
- [Beardsley-95b] P. A. Beardsley, I. D. Reid, A. Zisserman, and D. W. Murray. Active visual navigation using non-metric structure. In *Proc. International Conference on Computer Vision*, pages 58–64, 1995.

- [Beardsley-96] P. A. Beardsley, P. H. S. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Proc. 4th European Conference on Computer Vision, LNCS 1065, Cambridge*, pages 683–695, 1996.
- [Blake-87] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, USA, August 1987.
- [Boehm-94] W. Boehm and H. Prautzsch. *Geometric Concepts for Geometric Design*. A. K. Peters, 1994.
- [Bookstein-79] F. Bookstein. Fitting conic sections to scattered data. *Computer Graphics and Image Processing*, 9:56–71, 1979.
- [Bougnoux-98] S. Bougnoux. From Projective to Euclidean space under any practical situation, a criticism of self-calibration. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 790–796, January 1998.
- [Boult-91] I. E. Boult and L. Gottesfeld Brown. Factorisation-based segmentation of motions. In *Proc. IEEE Workshop on Visual Motion*, 1991.
- [Brand-01] M. Brand. Morphable 3d models from video. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages II: 456–463, 2001.
- [Brown-71] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.
- [Buchanan-88] T. Buchanan. The twisted cubic and camera calibration. *Computer Vision, Graphics and Image Processing*, 42:130–132, 1988.
- [Buchanan-92] T. Buchanan. Critical sets for 3D reconstruction using lines. In *Proc. European Conference on Computer Vision*, LNCS 588, pages 730–738. Springer-Verlag, 1992.
- [Canny-86] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [Capel-98] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, pages 885–891, June 1998.
- [Caprile-90] B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4:127–140, 1990.
- [Carlsson-93] S. Carlsson. Multiple image invariance using the double algebra. In *Applications of Invariance in Computer Vision*, volume SLN Comp. Science vol 825, pages 335–350, 1993.
- [Carlsson-94] S. Carlsson. Multiple image invariance using the double algebra. In J. Mundy, A. Zisserman, and D. Forsyth, editors, *Applications of Invariance in Computer Vision LNCS 825*. Springer-Verlag, 1994.
- [Carlsson-95] S. Carlsson. Duality of reconstruction and positioning from projective views. In *IEEE Workshop on Representation of Visual Scenes, Boston*, 1995.
- [Carlsson-98] S. Carlsson and D. Weinshall. Dual computation of projective shape and camera positions from multiple images. *International Journal of Computer Vision*, 27(3):227–241, 1998.
- [Christy-96] S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iteration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104, November 1996.
- [Chum-03] O. Chum, T. Werner, and T. Pajdla. On joint orientation of epipoles. Research Report CTU-CMP-2003-10, Center for Machine Perception, K333 FEE Czech Technical University, Prague, Czech Republic, April 2003.
- [Cipolla-99] R. Cipolla, T. Drummond, and D. Robertson. Camera calibration from vanishing points in images of architectural scenes. In *Proc. British Machine Vision Conference*, September 1999.
- [Collins-93] R. T. Collins and J. R. Beveridge. Matching perspective views of coplanar structures using projective unwarping and similarity matching. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1993.
- [Costeira-98] J.P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.
- [Criminisi-98] A. Criminisi, I. Reid, and A. Zisserman. Duality, rigidity and planar parallax. In *Proc.*

- European Conference on Computer Vision*, pages 846–861. Springer-Verlag, June 1998.
- [Criminisi-99a] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 434–442, September 1999.
- [Criminisi-99b] A. Criminisi, I. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.
- [Criminisi-00] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, November 2000.
- [Criminisi-01] A. Criminisi. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. Distinguished Dissertation Series. Springer-Verlag London Ltd., July 2001. ISBN: 1852334681.
- [Cross-98] G. Cross and A. Zisserman. Quadric surface reconstruction from dual-space geometry. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 25–31, January 1998.
- [Cross-99] G. Cross, A. W. Fitzgibbon, and A. Zisserman. Parallax geometry of smooth surfaces in multiple views. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 323–329, September 1999.
- [Csurka-97] G. Csurka, C. Zeller, Z. Zhang, and O. D. Faugeras. Characterizing the uncertainty of the fundamental matrix. *Computer Vision and Image Understanding*, 68(1):18–36, October 1997.
- [Csurka-98] G. Csurka, D. Demirdjian, A. Ruf, and R. Horaud. Closed-form solutions for the euclidean calibration of a stereo rig. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, pages 426–442, June 1998.
- [DeAgapito-98] L. de Agapito, E. Hayman, and I. Reid. Self-calibration of a rotating camera with varying intrinsic parameters. In *Proc. 9th British Machine Vision Conference, Southampton*, 1998.
- [DeAgapito-99] L. de Agapito, R. I. Hartley, and E. Hayman. Linear self-calibration of a rotating and zooming camera. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 15–21, 1999.
- [Dementhon-95] D. Dementhon and L. Davis. Model based pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, 1995.
- [Devernay-95] F. Devernay and O. D. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. In *SPIE*, volume 2567, San Diego, CA, July 1995.
- [Devernay-96] F. Devernay and O. D. Faugeras. From projective to euclidean reconstruction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–269, 1996.
- [Faugeras-90] O. D. Faugeras and S. J. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4:225–246, 1990.
- [Faugeras-92a] O. D. Faugeras, Q. Luong, and S. Maybank. Camera self-calibration: Theory and experiments. In *Proc. European Conference on Computer Vision*, LNCS 588, pages 321–334. Springer-Verlag, 1992.
- [Faugeras-92b] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proc. European Conference on Computer Vision*, LNCS 588, pages 563–578. Springer-Verlag, 1992.
- [Faugeras-93] O. D. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.
- [Faugeras-94] O. D. Faugeras and L. Robert. What can two images tell us about a third one. In J. O. Eckland, editor, *Proc. 3rd European Conference on Computer Vision, Stockholm*, pages 485–492. Springer-Verlag, 1994.
- [Faugeras-95a] O. D. Faugeras and B. Mourrain. On the geometry and algebra of point and line correspondences between N images. In *Proc. International Conference on Computer Vision*, pages 951–962, 1995.
- [Faugeras-95b] O. D. Faugeras. Stratification of three-dimensional vision: projective, affine, and metric representation. *Journal of the Optical Society of America*, A12:465–484, 1995.
- [Faugeras-95c] O. D. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3-D reconstruction of urban scenes from sequences of images. Technical report, INRIA, 1995.
- [Faugeras-97] O. D. Faugeras and T. Papadopoulos. Grassmann-Cayley algebra for modeling systems of cameras and the algebraic equations of the manifold of trifocal tensors. Technical Report 3225, INRIA, Sophia-Antipolis, France, 1997.

- [Faugeras-98] O. D. Faugeras, L. Quan, and P. Sturm. Self-calibration of a 1D projective camera and its application to the self-calibration of a 2D projective camera. In *Proc. European Conference on Computer Vision*, pages 36–52, 1998.
- [Fischler-81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6):381–395, 1981.
- [Fitzgibbon-98a] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc. European Conference on Computer Vision*, pages 311–326. Springer-Verlag, June 1998.
- [Fitzgibbon-98b] A. W. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3D model construction for turn-table sequences. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments, LNCS 1506*, pages 155–170. Springer-Verlag, June 1998.
- [Fitzgibbon-99] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least-squares fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999.
- [Gear-98] C. W. Gear. Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150, 1998.
- [Giblin-87] P. Giblin and R. Weiss. Reconstruction of surfaces from profiles. In *Proc. 1st International Conference on Computer Vision, London*, pages 136–144, London, 1987.
- [Gill-78] P. E. Gill and W. Murray. Algorithms for the solution of the nonlinear least-squares problem. *SIAM J Num Anal*, 15(5):977–992, 1978.
- [Golub-89] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, second edition, 1989.
- [Gracie-68] G. Gracie. Analytical photogrammetry applied to single terrestrial photograph mensuration. In *XIIth International Conference of Photogrammetry, Lausanne, Switzerland*, July 1968.
- [Gupta-97] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1997.
- [Haralick-91] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 592–598, 1991.
- [Harris-88] C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.
- [Hartley-92a] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proc. European Conference on Computer Vision, LNCS 588*, pages 579–587. Springer-Verlag, 1992.
- [Hartley-92b] R. I. Hartley. Invariants of points seen in multiple images. GE internal report, GE CRD, Schenectady, NY 12301, USA, May 1992.
- [Hartley-92c] R. I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1992.
- [Hartley-94a] R. I. Hartley. Self-calibration from multiple views with a rotating camera. In *Proc. European Conference on Computer Vision, LNCS 800/801*, pages 471–478. Springer-Verlag, 1994.
- [Hartley-94b] R. I. Hartley. Euclidean reconstruction from uncalibrated views. In J. Mundy, A. Zisserman, and D. Forsyth, editors, *Applications of Invariance in Computer Vision, LNCS 825*, pages 237–256. Springer-Verlag, 1994.
- [Hartley-94c] R. I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:1036–1041, October 1994.
- [Hartley-94d] R. I. Hartley. Projective reconstruction from line correspondence. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994.
- [Hartley-95a] R. I. Hartley. Multilinear relationships between coordinates of corresponding image points and lines. In *Proceedings of the Sophus Lie Symposium, Nordfjordeid, Norway* (not published yet), 1995.
- [Hartley-95b] R. I. Hartley. A linear method for reconstruction from lines and points. In *Proc. International Conference on Computer Vision*, pages 882–887, 1995.
- [Hartley-97a] R. I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal*

- of *Computer Vision*, 22(2):125–140, 1997.
- [Hartley-97b] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, November 1997.
- [Hartley-97c] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580 – 593, October 1997.
- [Hartley-97d] R. I. Hartley. Kruppa’s equations derived from the fundamental matrix. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):133–135, 1997.
- [Hartley-97e] R. I. Hartley and T. Saxena. The cubic rational polynomial camera model. In *Proc. DARPA Image Understanding Workshop*, pages 649 – 653, 1997.
- [Hartley-98a] R. I. Hartley. Chirality. *International Journal of Computer Vision*, 26(1):41–61, 1998.
- [Hartley-98b] R. I. Hartley. Dualizing scene reconstruction algorithms. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments, LNCS 1506*, pages 14–31, Springer-Verlag, June 1998.
- [Hartley-98c] R. I. Hartley. Computation of the quadrifocal tensor. In *Proc. European Conference on Computer Vision*, LNCS 1406, pages 20–35. Springer-Verlag, 1998.
- [Hartley-98d] R. I. Hartley. Minimizing algebraic error in geometric estimation problems. In *Proc. International Conference on Computer Vision*, pages 469–476, 1998.
- [Hartley-99] R. Hartley, L. de Agapito, E. Hayman, and I. Reid. Camera calibration and the search for infinity. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 510–517, September 1999.
- [Hartley-00a] R. I. Hartley and N. Y. Dano. Reconstruction from six-point sequences. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages II-480 – II-486, 2000.
- [Hartley-00b] R. I. Hartley. Ambiguous configurations for 3-view projective reconstruction. In *Proc. 6th European Conference on Computer Vision, Part I, LNCS 1842, Dublin, Ireland*, pages 922–935, 2000.
- [Hartley-02a] R. Hartley and F. Kahl. Critical curves and surfaces for euclidean reconstruction. In *Proc. 7th European Conference on Computer Vision, Part II, LNCS 2351, Copenhagen, Denmark*, pages 447–462, 2002.
- [Hartley-02b] R. Hartley and R. Kaucic. Sensitivity of calibration to principal point position. In *Proc. 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 2, pages 433–446. Springer-Verlag, 2002.
- [Hartley-03] R. Hartley and F. Kahl. A critical configuration for reconstruction from rectilinear motion. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
- [Hayman-03] E. Hayman, T. Thorhallsson, and D.W. Murray. Tracking while zooming using affine transfer and multifocal tensors. *International Journal of Computer Vision*, 51(1):37–62, January 2003.
- [Heyden-95a] A. Heyden. Reconstruction from image sequences by means of relative depths. In E. Grimson, editor, *Proc. 5th International Conference on Computer Vision, Boston, Cambridge, MA*, June 1995.
- [Heyden-95b] A. Heyden. *Geometry and Algebra of Multiple Projective Transformations*. PhD thesis, Department of Mathematics, Lund University, Sweden, December 1995.
- [Heyden-97a] A. Heyden. Projective structure and motion from image sequences using subspace methods. In *Scandinavian Conference on Image Analysis, Lappeenranta*, pages 963–968, 1997.
- [Heyden-97b] A. Heyden and K. Åström. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1997.
- [Heyden-97c] A. Heyden. Reconstruction from multiple images by means of using relative depths. *International Journal of Computer Vision*, 24(2):155–161, 1997.
- [Heyden-98] A. Heyden. Algebraic varieties in multiple view geometry. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, pages 3–19, 1998.
- [Hilbert-56] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea, NY, 1956.
- [Horaud-98] R. Horaud and G. Csurka. Self-calibration and Euclidean reconstruction using motions of a

- stereo rig. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 96–103, January 1998.
- [Horn-90] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4:59–78, 1990.
- [Horn-91] B. K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America*, 8(10):1630–1638, 1991.
- [Horry-97] Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 225–232, 1997.
- [Huang-89] T. S. Huang and O. D. Faugeras. Some properties of the E-matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1310–1312, 1989.
- [Huber-81] P. J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.
- [Huynh-03] D.Q. Huynh, R. Hartley, and A. Heyden. Outlier correction of image sequences for the affine camera. In *Proc. 9th International Conference on Computer Vision, Vancouver, France*, 2003.
- [Irani-98] M. Irani, P. Anandan, and D. Weinshall. From reference frames to reference planes: Multi-view parallax geometry and applications. In *Proc. European Conference on Computer Vision*, 1998.
- [Irani-99] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *Proc. International Conference on Computer Vision*, 1999.
- [Irani-00] Michal Irani and P. Anandan. Factorization with uncertainty. In *Proc. 6th European Conference on Computer Vision, Part I, LNCS 1842, Dublin, Ireland*, pages 539–553, 2000.
- [Jiang-02] G. Jiang, H. Tsui, L. Quan, and A. Zisserman. Single axis geometry by fitting conics. In *Proc. 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 1, pages 537–550. Springer-Verlag, 2002.
- [Kahl-98a] F. Kahl and A. Heyden. Structure and motion from points, lines and conics with affine cameras. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, pages 327–341, 1998.
- [Kahl-98b] F. Kahl and A. Heyden. Using conic correspondences in two images to estimate epipolar geometry. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 761–766, 1998.
- [Kahl-99] F. Kahl. Critical motions and ambiguous euclidean reconstructions in auto-calibration. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 469–475, 1999.
- [Kahl-01a] F. Kahl, R. Hartley, and K. Åström. Critical configurations for n -view projective reconstruction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages II-158 – II-163, 2001.
- [Kahl-01b] F. Kahl. *Geometry and Critical Configurations of Multiple Views*. PhD thesis, Lund Institute of Technology, 2001.
- [Kanatani-92] K. Kanatani. *Geometric computation for machine vision*. Oxford University Press, Oxford, 1992.
- [Kanatani-94] K. Kanatani. Statistical bias of conic fitting and renormalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):320–326, 1994.
- [Kanatani-96] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, Amsterdam, 1996.
- [Kaucic-01] R. Kaucic, R. I. Hartley, and N. Y. Dano. Plane-based projective reconstruction. In *Proc. 8th International Conference on Computer Vision, Vancouver, Canada*, pages I-420–427, 2001.
- [Klein-39] F. Klein. *Elementary Mathematics from an Advanced Standpoint*. Macmillan, New York, 1939.
- [Knight-03] J. Knight, A. Zisserman, and I. Reid. Linear auto-calibration for ground plane motion. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.
- [Koenderink-84] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.
- [Koenderink-90] J. Koenderink. *Solid Shape*. MIT Press, 1990.
- [Koenderink-91] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, 8(2):377–385, 1991.

- [Krames-42] J. Krames. Über die bei der Hauptaufgabe der Luftphotogrammetrie auftretenden "gefährlichen" Flächen. *Bildmessung und Luftbildwesen (Beilage zur Allg. Vermessungs-Nachr.)*, 17, Heft 1/2:1–18, 1942.
- [Kriegman-98] D. J. Kriegman and P. Belhumeur. What shadows reveal about object structure. In *Proc. European Conference on Computer Vision*, pages 399–414, 1998.
- [Laveau-96a] S. Laveau. *Géométrie d'un système de N caméras. Théorie, estimation et applications*. PhD thesis, INRIA, 1996.
- [Laveau-96b] S. Laveau and O. D. Faugeras. Oriented projective geometry in computer vision. In *Proc. 4th European Conference on Computer Vision, LNCS 1065, Cambridge*, pages 147–156, Springer-Verlag, 1996. Buxton B. and Cipolla R.
- [Liebowitz-98] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 482–488, June 1998.
- [Liebowitz-99a] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. EuroGraphics*, volume 18, pages 39–50, September 1999.
- [Liebowitz-99b] D. Liebowitz and A. Zisserman. Combining scene and auto-calibration constraints. In *Proc. 7th International Conference on Computer Vision, Kerkira, Greece*, September 1999.
- [Liebowitz-01] D. Liebowitz. *Camera Calibration and Reconstruction of Geometry from Images*. PhD thesis, University of Oxford, Dept. Engineering Science, June 2001. D.Phil. thesis.
- [LonguetHiggins-81] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.
- [Luong-92] Q. Luong. *Matrice Fondamentale et Autocalibration en Vision par Ordinateur*. PhD thesis, Université de Paris-Sud, France, 1992.
- [Luong-94] Q. T. Luong and T. Viéville. Canonic representations for the geometries of multiple projective views. In *Proc. 3rd European Conference on Computer Vision, Stockholm*, pages 589–599, May 1994.
- [Luong-96] Q. T. Luong and T. Viéville. Canonical representations for the geometries of multiple projective views. *Computer Vision and Image Understanding*, 64(2):193–229, September 1996.
- [Lutkepohl-96] H. Lutkepohl. *Handbook of Matrices*. Wiley, ISBN 0471970158, 1996.
- [Ma-99] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. Euclidean reconstruction and reprojection up to subgroups. In *Proc. 7th International Conference on Computer Vision, Kerkira, Greece*, pages 773–780, 1999.
- [Mathematica-92] S. Wolfram. *Mathematica A System for Doing Mathematics by Computer second edition*. Addison-Wesley, 1992.
- [Maybank-90] S. J. Maybank. The projective geometry of ambiguous surfaces. *Philosophical Transactions of the Royal Society of London, SERIES A*, A 332:1–47, 1990.
- [Maybank-93] S. J. Maybank. *Theory of reconstruction from image motion*. Springer-Verlag, Berlin, 1993.
- [Maybank-98] S. J. Maybank and A. Shashua. Ambiguity in reconstruction from images of six points. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 703–708, 1998.
- [McLauchlan-00] P. F. McLauchlan. Gauge independence in optimization algorithms for 3D vision. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of LNCS, pages 183–199. Springer, 2000.
- [Mohr-92] R. Mohr. Projective geometry and computer vision. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*. World Scientific, 1992.
- [Mohr-93] R. Mohr, F. Veillon, and L. Quan. Relative 3D reconstruction using multiple uncalibrated images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 543–548, 1993.
- [Moons-94] T. Moons, L. Van Gool, M. Van Diest, and E. Pauwels. Affine reconstruction from perspective image pairs. In J. Mundy, A. Zisserman, and D. Forsyth, editors, *Applications of Invariance in Computer Vision, LNCS 825*. Springer-Verlag, 1994.
- [Muehlich-98] M. Mühlich and R. Mester. The role of total least squares in motion analysis. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, pages 305–321. Springer-Verlag,

- 1998.
- [Mundy-92] J. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.
- [Newsam-96] G. Newsam, D. Q. Huynh, M. Brooks, and H. P. Pan. Recovering unknown focal lengths in self-calibration: An essentially linear algorithm and degenerate configurations. In *Int. Arch. Photogrammetry & Remote Sensing*, volume XXXI-B3, pages 575–80, Vienna, 1996.
- [Niem-94] W. Niem and R. Buschmann. Automatic modelling of 3D natural objects from multiple views. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Hamburg, Germany*, 1994.
- [Nister-00] D. Nister. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *Proc. European Conference on Computer Vision*, 2000.
- [Oskarsson-02] M. Oskarsson, A. Zisserman, and K. Åström. Minimal projective reconstruction for combinations of points and lines in three views. In *Proc. British Machine Vision Conference*, pages 62–72, 2002.
- [Poelman-94] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *Proc. 3rd European Conference on Computer Vision, Stockholm*, volume 2, pages 97–108, 1994.
- [Pollefeys-96] M. Pollefeys, L. Van Gool, and A. Oosterlinck. The modulus constraint: a new constraint for self-calibration. In *Proc. International Conference on Pattern Recognition*, pages 31–42, 1996.
- [Pollefeys-98] M. Pollefeys, R. Koch, and L. Van Gool. Self calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 90–96, 1998.
- [Pollefeys-99a] M. Pollefeys, R. Koch, and L. Van Gool. A simple and efficient rectification method for general motion. In *Proc. International Conference on Computer Vision*, pages 496–501, 1999.
- [Pollefeys-99b] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, ESAT-PSI, K.U.Leuven, 1999.
- [Pollefeys-02] M. Pollefeys, F. Verbiest, and L. J. Van Gool. Surviving dominant planes in uncalibrated structure and motion recovery. In *ECCV (2)*, pages 837–851, 2002.
- [Ponce-94] J. Ponce, D. H. Marimont, and T. A. Cass. Analytical methods for uncalibrated stereo and motion measurement. In *Proc. 3rd European Conference on Computer Vision, Stockholm*, volume 1, pages 463–470, 1994.
- [Porrill-91] J. Porrill and S. B. Pollard. Curve matching and stereo calibration. *Image and Vision Computing*, 9(1):45–50, 1991.
- [Pratt-87] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21(4):145–151, 1987.
- [Press-88] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.
- [Pritchett-98] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 754–760, January 1998.
- [Proesmans-98] M. Proesmans, T. Tuytelaars, and L. J. Van Gool. Monocular image measurements. Technical Report Improofs-M12T21/1/P, K.U.Leuven, 1998.
- [Quan-94] L. Quan. Invariants of 6 points from 3 uncalibrated images. In J. O. Eckland, editor, *Proc. 3rd European Conference on Computer Vision, Stockholm*, pages 459–469. Springer-Verlag, 1994.
- [Quan-97a] L. Quan and T. Kanade. Affine structure from line correspondences with uncalibrated affine cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):834–845, August 1997.
- [Quan-97b] L. Quan. Uncalibrated 1D projective camera and 3D affine reconstruction of lines. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 60–65, 1997.
- [Quan-98] L. Quan and Z. Lan. Linear $n \geq 4$ -point pose determination. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 778–783, 1998.
- [Reid-96] I. D. Reid and D. W. Murray. Active tracking of foveated feature clusters using affine structure. *International Journal of Computer Vision*, 18(1):41–60, 1996.
- [Rinner-72] K. Rinner and R. Burkhardt. Photogrammetrie. In *Handbuch der Vermessungskunde*, volume

- Band III a/3. Jordan, Eggert, Kneissel, Stuttgart: J.B. Metzlersche Verlagsbuchhandlung, 1972.
- [Robert-93] L. Robert and O. D. Faugeras. Relative 3D positioning and 3D convex hull computation from a weakly calibrated stereo pair. In *Proc. 4th International Conference on Computer Vision, Berlin*, pages 540–544, 1993.
- [Rother-01] C. Rother and S. Carlsson. Linear multi view reconstruction and camera recovery. In *Proc. 8th International Conference on Computer Vision, Vancouver, Canada*, pages 1–42–49, 2001.
- [Rother-03] C. Rother. *Multi-View Reconstruction and Camera Recovery using a Real or Virtual Reference Plane*. PhD thesis, Computational Vision and Active Perception Laboratory, Kungl Tekniska Högskolan, 2003.
- [Rousseeuw-87] P. J. Rousseeuw. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.
- [Sampson-82] P. D. Sampson. Fitting conic sections to ‘very scattered’ data: An iterative refinement of the Bookstein algorithm. *Computer Vision, Graphics, and Image Processing*, 18:97–108, 1982.
- [Sawhney-98] H. S. Sawhney, S. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *Proc. European Conference on Computer Vision*, pages 103–119. Springer-Verlag, 1998.
- [Schaffalitzky-99] F. Schaffalitzky and A. Zisserman. Geometric grouping of repeated elements within images. In D.A. Forsyth, J.L. Mundy, V. Di Gesu, and R. Cipolla, editors, *Shape, Contour and Grouping in Computer Vision*, LNCS 1681, pages 165–181. Springer-Verlag, 1999.
- [Schaffalitzky-00a] F. Schaffalitzky. Direct solution of modulus constraints. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, Bangalore*, pages 314–321, 2000.
- [Schaffalitzky-00b] F. Schaffalitzky and A. Zisserman. Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing*, 18:647–658, 2000.
- [Schaffalitzky-00c] F. Schaffalitzky, A. Zisserman, R. I. Hartley, and P. H. S. Torr. A six point solution for structure and motion. In *Proc. European Conference on Computer Vision*, pages 632–648. Springer-Verlag, June 2000.
- [Schmid-97] C. Schmid and A. Zisserman. Automatic line matching across views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 666–671, 1997.
- [Schmid-98] C. Schmid and A. Zisserman. The geometry and matching of curves in multiple views. In *Proc. European Conference on Computer Vision*, pages 394–409. Springer-Verlag, June 1998.
- [Se-00] S. Se. Zebra-crossing detection for the partially sighted. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 211–217, 2000.
- [Semple-79] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, 1979.
- [Shapiro-95] L. S. Shapiro, A. Zisserman, and M. Brady. 3D motion recovery via affine epipolar geometry. *International Journal of Computer Vision*, 16(2):147–182, 1995.
- [Shashua-94] A. Shashua. Trilinearity in visual recognition by alignment. In *Proc. 3rd European Conference on Computer Vision, Stockholm*, volume 1, pages 479–484, May 1994.
- [Shashua-95a] A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, August 1995.
- [Shashua-95b] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. In *Proc. 5th International Conference on Computer Vision, Boston*, 1995.
- [Shashua-96] A. Shashua and S. J. Maybank. Degenerate N-point configurations of three views: Do critical surfaces exist? Technical Report TR 96-19, Hebrew University, Computer Science, November 1996.
- [Shashua-97] A. Shashua and S. Toelg. The quadric reference surface: Theory and applications. *International Journal of Computer Vision*, 23(2):185–198, 1997.
- [Shimshoni-99] I. Shimshoni, R. Basri, and E. Rivlin. A geometric interpretation of weak-perspective motion. Technical report, Technion, 1999.
- [Sinclair-92] D. A. Sinclair. *Experiments in Motion and Correspondence*. PhD thesis, University of Oxford, 1992.
- [Slama-80] C. Slama. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, USA, 4th edition, 1980.

- [Spetsakis-91] M. E. Spetsakis and J. Aloimonos. A multi-frame approach to visual motion perception. *International Journal of Computer Vision*, 16(3):245–255, 1991.
- [Springer-64] C. E. Springer. *Geometry and Analysis of Projective Spaces*. Freeman, 1964.
- [Stein-99] G. Stein and A. Shashua. On degeneracy of linear reconstruction from three views: Linear line complex and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):244–251, 1999.
- [Stolfi-91] J. Stolfi. *Oriented Projective Geometry*. Academic Press, 1991.
- [Strecha-02] C. Strecha and L. Van Gool. PDE-based multi-view depth estimation. *1st Int. Symp. of 3D Data Processing Visualization and Transmission*, pages 416–425, 2002.
- [Sturm-96] P. Sturm and W. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. 4th European Conference on Computer Vision, Cambridge*, pages 709–720, 1996.
- [Sturm-97a] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico*, pages 1100–1105, June 1997.
- [Sturm-97b] P. Sturm. *Vision 3D non calibrée: Contributions à la reconstruction projective et étude des mouvements critiques pour l'auto calibrage*. PhD thesis, INRIA Rhône-Alpes, 1997.
- [Sturm-99a] P. Sturm and S. J. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. In *Proc. 10th British Machine Vision Conference, Nottingham*, 1999.
- [Sturm-99b] P. Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. In *Proc. 10th British Machine Vision Conference, Nottingham*, pages 63–72, 1999.
- [Sturm-99c] P. Sturm and S. Maybank. On plane based camera calibration: A general algorithm, singularities, applications. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 432–437, June 1999.
- [Sturm-01] P. Sturm. On focal length calibration from two views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 145–150, 2001.
- [Sutherland-63] I. E. Sutherland. Sketchpad: A man-machine graphical communications system. Technical Report 296, MIT Lincoln Laboratories, 1963. Also published by Garland Publishing, New York, 1980.
- [Szeliski-96] R. Szeliski and S. B. Kang. Shape ambiguities in structure from motion. In B. Buxton and Cipolla R., editors, *Proc. 4th European Conference on Computer Vision, LNCS 1064, Cambridge*, pages 709–721. Springer-Verlag, 1996.
- [Szeliski-97] R. Szeliski and S. Heung-Yeung. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 1997.
- [Taubin-91] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *PAMI*, 13(11):1115–1138, 1991.
- [Thorhallsson-99] T. Thorhallsson and D.W. Murray. The tensors of three affine views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1999.
- [Tomas-92] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 9(2):137–154, November 1992.
- [Tordoff-01] B. Tordoff and D.W. Murray. Reactive zoom control while tracking using an affine camera. In *Proc. British Machine Vision Conference*, volume 1, pages 53–62, 2001.
- [Torr-93] P. H. S. Torr and D. W. Murray. Outlier detection and motion segmentation. In *Proc SPIE Sensor Fusion VI*, pages 432–443, Boston, September 1993.
- [Torr-95a] P. H. S. Torr, A. Zisserman, and D. W. Murray. Motion clustering using the trilinear constraint over three views. In R. Mohr and C. Wu, editors, *Europe-China Workshop on Geometrical Modelling and Invariants for Computer Vision*, pages 118–125. Xidan University Press, 1995.
- [Torr-95b] P. H. S. Torr. *Motion segmentation and outlier detection*. PhD thesis, Dept. of Engineering Science, University of Oxford, 1995.
- [Torr-97] P. H. S. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor.

- Image and Vision Computing*, 15:591–605, 1997.
- [Torr-98] P. H. S. Torr and A. Zisserman. Robust computation and parameterization of multiple view relations. In *Proc. 6th International Conference on Computer Vision, Bombay, India*, pages 727–732, January 1998.
- [Torr-99] P. H. S. Torr, A. W. Fitzgibbon, and A. Zisserman. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision*, 32(1):27–44, August 1999.
- [Torresani-01] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modelling non-rigid objects with rank constraints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages I: 493–500, 2001.
- [Triggs-95] W. Triggs. The geometry of projective reconstruction i: Matching constraints and the joint image. In *Proc. International Conference on Computer Vision*, pages 338–343, 1995.
- [Triggs-96] W. Triggs. Factorization methods for projective structure and motion. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–851, 1996.
- [Triggs-97] W. Triggs. Auto-calibration and the absolute quadric. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997.
- [Triggs-98] W. Triggs. Autocalibration from planar scenes. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, 1998.
- [Triggs-99a] W. Triggs. Camera pose and calibration from 4 or 5 known 3D points. In *Proc. International Conference on Computer Vision*, pages 278–284, 1999.
- [Triggs-99b] W. Triggs. Differential matching constraints. In *Proc. International Conference on Computer Vision*, pages 370–376, 1999.
- [Triggs-00a] W. Triggs, P. F. McLauchlan, R. I. Hartley, and A. Fitzgibbon. Bundle adjustment for structure from motion. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 2000.
- [Triggs-00b] W. Triggs. Plane + parallax, tensors and factorization. In *Proc. European Conference on Computer Vision*, pages 522–538, 2000.
- [Tsai-84] R. Y. Tsai and T. S. Huang. The perspective view of three points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:13–27, 1984.
- [VanGool-98] L. Van Gool, M. Proesmans, and A. Zisserman. Planar homologies as a basis for grouping and recognition. *Image and Vision Computing*, 16:21–26, January 1998.
- [Vieville-93] T. Viéville and Q. Luong. Motion of points and lines in the uncalibrated case. Technical Report 2054, I.N.R.I.A., 1993.
- [Vieville-95] T. Viéville and D. Lingrand. Using singular displacements for uncalibrated monocular vision systems. Technical Report 2678, I.N.R.I.A., 1995.
- [VonSanden-08] H. von Sanden. *Die Bestimmung der Kernpunkte in der Photogrammetrie*. PhD thesis, Univ. Göttingen, December 1908.
- [Weinshall-95] D. Weinshall, M. Werman, and A. Shashua. Shape descriptors: Bilinear, trilinear and quadrilinear relations for multi-point geometry and linear projective reconstruction algorithms. In *IEEE Workshop on Representation of Visual Scenes, Boston*, pages 58–65, 1995.
- [Weng-88] J. Weng, N. Ahuja, and T. S. Huang. Closed-form solution and maximum likelihood : A robust approach to motion and structure estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1988.
- [Weng-89] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: algorithms, error analysis and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–476, 1989.
- [Werner-01] T. Werner and T. Pajdla. Oriented matching constraints. In T. Cootes and C. Taylor, editors, *Proc. British Machine Vision Conference*, pages 441–450, London, UK, September 2001. British Machine Vision Association.
- [Werner-03] T. Werner. A constraint on five points in two images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.
- [Wolfe-91] W. J. Wolfe, D. Mathis, C. Weber Sklair, and M. Magee. The perspective view of three points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):66–73, January 1991.

- [Xu-96] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition*. Kluwer Academic Publishers, 1996.
- [Zeller-96] C. Zeller. *Projective, Affine and Euclidean Calibration in Computer Vision and the Application of Three Dimensional Perception*. PhD thesis, RobotVis Group, INRIA Sophia-Antipolis, 1996.
- [Zhang-95] Z. Zhang, R. Deriche, O. D. Faugeras, and Q. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78:87–119, 1995.
- [Zhang-98] Z. Zhang. Determining the epipolar geometry and its uncertainty – a review. *International Journal of Computer Vision*, 27(2):161–195, March 1998.
- [Zhang-00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.
- [Zisserman-92] A. Zisserman. Notes on geometric invariance in vision. Tutorial, British Machine Vision Conference, 1992.
- [Zisserman-94] A. Zisserman and S. Maybank. A case against epipolar geometry. In J. Mundy, A. Zisserman, and D. Forsyth, editors, *Applications of Invariance in Computer Vision LNCS 825*. Springer-Verlag, 1994.
- [Zisserman-95a] A. Zisserman, J. Mundy, D. Forsyth, J. Liu, N. Pillow, C. Rothwell, and S. Uteke. Class-based grouping in perspective images. In *Proc. International Conference on Computer Vision*, 1995.
- [Zisserman-95b] A. Zisserman, P. Beardsley, and I. Reid. Metric calibration of a stereo rig. In *IEEE Workshop on Representation of Visual Scenes, Boston*, pages 93–100, 1995.
- [Zisserman-96] A. Zisserman. A users guide to the trifocal tensor. Dept. of Engineering Science, University of Oxford, 1996.
- [Zisserman-98] A. Zisserman, D. Liebowitz, and M. Armstrong. Resolving ambiguities in auto-calibration. *Philosophical Transactions of the Royal Society of London, SERIES A*, 356(1740):1193–1211, 1998.

Index

χ^2 566

A

- absolute conic 5–6, 9, 16–18, 78, 80–84, 195, 209–210, 224, 272–277, 340, 458, 461, 464, 471, 475, 485, 487, 500
 - relation to orthogonality 83
- absolute dual quadric 83–85, 458, 462, 466–467, 469, 472–473, 475, 477
- absolute dual quadric 83–84, 463
- absolute dual quadric
 - extracting the rectifying transform 463
 - properties 462
 - simple properties 462
- adjoint matrix 31, 581
- affine camera 14, 166, 169, 172–174, 176, 184–185, 271, 344, 347, 355, 413, 425, 432, 436–437, 440–441, 520
- affine camera
 - counting arguments 425
 - decomposition 169
 - definition 166
 - definition 166
 - error in employing 168
 - estimation of 184
 - hierarchy of 170
 - more properties 172
- affine fundamental matrix 298, 308, 345, 347–349, 351–353, 360–361, 390, 413, 439
- affine fundamental matrix
 - algebraic derivation 347
 - derivation 345
 - geometric derivation 345
 - geometric interpretation 349
 - Gold Standard algorithm 349
 - linear algorithm 348
 - minimal configuration 352
 - properties 347
 - singularity constraint 349
- affine geometry. 3
- affine imaging 169
- affine imaging conditions 346
- affine imaging model 169
- affine matrix 40, 98, 347
- affine matrix 62
- affine multiple view tensors
 - computation 439
- affine properties 43, 49–50, 55, 62, 79–81, 220, 223
- affine properties of reconstruction 81
- affine properties
 - recovery from image 49
- affine reconstruction 10, 261, 266, 268–272, 274, 277, 339, 351, 355–356, 360, 434, 436–437, 439–440, 456, 459, 473, 475–476, 479, 487, 489, 496, 498, 515, 521, 525, 528
- affine reconstruction by alternation 440
- affine transformation 39
- algebraic distance 93–94, 96–97, 186, 279, 308
- algebraic distance
 - comparison with geometric distance 95
- algebraic error
 - minimizing 186
- algebraic minimization 107, 289, 400
- algebraic solution
 - limitations 468
- ambiguities
 - auto-calibration 478
- ambiguity 480
- ambiguity
 - affine 438–439, 442, 444
 - auto-calibration 473
 - bas-relief 356–359, 633
 - examples 543
 - for 2D cameras 535
 - for 3D cameras 536
 - for three views 554
 - Necker reversal 356
 - of auto-calibration 473, 479–480
 - of auto-calibration
 - for a sequence 480
 - projective 265, 387
 - single view 549
 - two views 552
- angle
 - between two lines in image 215
 - between two planes 83, 85, 218, 236
 - between two rays 9, 195, 209–210, 213, 215, 223, 265
 - between two scene lines 215
 - of rotation 204–205, 628
- angle-axis parametrization 584–585, 624–625
- augmentation 601
- auto-calibration – see also reconstruction metric, 18, 58, 86, 205, 210, 253, 340, 439, 453, 456, 458–459, 461–465, 469, 473–474, 476–479, 481, 483, 485–486, 490, 496–500
- auto-calibration 274
- auto-calibration
 - ambiguities 478
 - critical motion sequences 497
 - finding plane at infinity 473
 - from planar motion 496
 - implementation 485
 - iterative methods 467
 - limitations of absolute dual quadric method 468

- metric from affine 475
 - numbers of views needed 467
 - of stereo rig 494
 - problems if Q_{∞}^* not positive-definite 469
 - removing ambiguity 479
 - removing the ambiguity 479
 - solving for Q_{∞}^*
 - linear 464
 - non-linear 466
 - solving for Q_{∞}^*
 - non-linear 466
 - typical ambiguities 480
 - using Q_{∞}^* 463
 - varying parameters 477
 - auto-epipolar 249, 260
 - axis planes of camera 160, 199
- B**
- back-projection
 - of conics 199
 - of lines 197
 - of points to rays 161
 - banded data 614
 - bas-relief ambiguity 356
 - bias 109, 150, 184, 302, 321, 568, 570–577
 - dependency on parametrization 571
 - lessons 572
 - bundle adjustment 14, 397, 434–435, 444, 452–453, 461, 467, 498, 611
 - bundle adjustment 434
 - bundle adjustment
 - banded structure 614
 - incremental 453
 - initial solution 435
 - sparse methods 435
 - with missing data 611
- C**
- calibrating conic 195
 - orthogonality 231
 - calibration constraints 477
 - calibration constraints
 - known internal parameters 273
 - same camera in all images 273
 - calibration matrix 157, 163–165, 167–168, 170, 190, 204, 209, 212, 215, 224, 226, 231, 257, 264, 273, 275, 459–460, 463–464, 469, 485, 489
 - calibration
 - internal 176, 185, 210, 220, 233, 239, 355, 486, 489–490
 - what does it give? 208
 - camera anatomy 158
 - camera calibration 6, 17, 21, 155, 185, 191, 195, 208, 212, 216, 219, 222, 224, 228–230, 233, 235, 262, 268, 278, 293, 340, 355, 444, 458–459, 473, 482, 511, 595
 - camera centre 163
 - camera centre
 - finding it 163
 - moving 207
 - camera matrices
 - compatible with two fundamental matrices 386
 - computation from F 263
 - computed from three fundamental matrices 386
 - defined by three fundamental matrices
 - uniqueness 385
 - camera matrix
 - canonical form 254
 - computation 263
 - decomposition 163
 - in projective space 165
 - reduced – see reduced camera matrix
 - camera motion
 - recovering it 444
 - camera orientation 156, 205, 210, 623
 - finding it 163
 - camera parameters 147, 156, 164, 170, 182, 185–188, 193, 232, 399, 435, 444, 458, 467, 602, 610–611, 627
 - camera resectioning
 - covariance 188
 - degenerate configurations 179
 - from five points 549
 - minimal solution 179
 - over-determined solution 179
 - camera rotation 155
 - camera translation 204, 249–250
 - camera translation 155
 - camera
 - affine – see affine camera
 - affine 172
 - at infinity 173
 - basic pinhole model 153
 - calibrated 193, 209, 231, 234
 - CCD 156
 - centre 158
 - finite 170
 - finite projective 157
 - general projective 157
 - known orientation 448
 - line 175
 - pinhole model 153
 - projective 7, 153, 157–159, 161, 165–166, 173–176, 184–185, 193, 208, 213, 253, 361, 459, 509, 518
 - projective
 - action on points 161
 - pushbroom 174
 - resection ambiguity 550
 - rotation 195, 204, 207, 215, 250, 489
 - rotation and translation 155
 - cameras
 - as points 533
 - Carlsson–Weinshall duality 505, 513, 546
 - Carlsson–Weinshall map 546–547, 549
 - Carlsson–Weinshall map
 - definition 546
 - caution 227
 - central projection 6–7, 34, 37, 153–154, 165, 175, 200, 632
 - central projection 154
 - change of coordinates 2, 106, 137, 299, 472, 566, 577
 - change of coordinates 566
 - Chasles' theorem 535
 - cheiral inequalities
 - solving 526
 - Cholesky factorization 211, 225–226, 272–273, 275, 277, 469, 582
 - choosing the depths 445
 - circular points 5–6, 18, 25, 44, 48, 52–57, 61–63, 79, 82, 85, 211, 226, 229, 235, 341, 485–491, 495, 500–501, 628
 - circular points 52
 - cofactor matrix 581
 - collineation – see also projective transformation, 32–33, 65, 631
 - column vector 26, 143, 154, 158, 448, 578, 581, 603
 - column vector 159
 - compatibility conditions 385
 - computing a plane
 - from point and line correspondence 332
 - from three point correspondences 330
 - computing vanishing lines 218
 - concurrent lines 45–46
 - concurrent lines 45
 - condition number 108, 439

- conic 30
 - conic fitting 127, 229, 297
 - conic
 - affine classification 60
 - analog for fundamental matrix 97
 - classification 59
 - defined by 5 points 30
 - degenerate 32
 - dual – see also dual conic
 - dual 31
 - dual to circular points 53
 - projective normal form 59
 - projective normal form of 59
 - tangent lines 31
 - transformations of 36
 - conjugacy 59, 83, 212–213, 228–229, 494, 628
 - conjugacy relation 494
 - conjugate configurations – definition 540
 - consistency conditions 331
 - constraints
 - from internal parameters 273
 - from same camera 273
 - from scene orthogonality 273
 - contour 200–201, 234, 300, 322
 - contour generator 200–202, 233–234, 295
 - contour generator – definition 200
 - contour
 - apparent 200–202, 233–234, 236
 - occluding 200
 - convex hull 515
 - coordinate orientation 164
 - coordinates. 2
 - coplanarity constraint 276, 422
 - correlation 59, 124, 148, 242, 246, 388, 447, 455, 486
 - correlation – definition 59
 - correspondence
 - of curves and surfaces 295
 - of lines 294
 - correspondences
 - determining 124
 - corridor scene 289
 - corridor scene 302
 - cosine formula 209
 - cost function 88, 90, 93–95, 98–99, 101–104, 110–114, 121–123, 125, 129, 134, 187, 191, 284, 287–289, 291, 308, 310, 312, 314–315, 318–319, 349–351, 386, 397–399, 404, 435, 462, 467, 485, 496, 498, 513, 597, 599–602, 607, 616–624, 626
 - cost function 88, 102, 110, 112, 279, 285, 602, 619, 621
 - cost function
 - heuristic 618
 - Huber 617, 619–622
 - least squares 621
 - non-convex 619
 - performance 620
 - properties 619
 - robust 122
 - squared error 619
 - statistically based 618
 - cost functions
 - asymptotically linear 619
 - linear 100
 - robust 122
 - summary 620
 - counting argument 18, 43, 63, 255, 383, 387, 411, 425, 427–428, 451, 461
 - counting argument 427
 - covariance estimation 608
 - covariance estimation
 - camera resectioning 188
 - covariance
 - backward propagation 141
 - backward propagation
 - over-parametrized case 142
 - estimation 188
 - forward propagation 139
 - matrix 103, 111, 129, 137, 139–146, 148, 150, 188–189, 282, 298–301, 565–566, 573, 576–577, 598, 603–607, 609–611, 623
 - of epipolar lines 301
 - use in point transfer 148
 - Cramér-Rao lower bound 573
 - critical configurations 11, 179, 533–534, 538, 544, 554, 559
 - examples 544
 - critical motion sequences 497
 - critical set 537
 - for 2d camera
 - definition 536
 - for 2D cameras 535
 - critical surface
 - for three views 553
 - cross product 581
 - cross ratio 42, 44–46, 49, 51, 63, 534–535, 629–631
 - cross ratio 44
- D**
- deficient-rank systems. 589
 - degeneracy 92, 226, 228, 293, 295–296, 323, 342, 381–383, 499, 546, 559
 - degeneracy 225, 234, 295–296, 329–330, 352, 456, 498
 - degeneracy
 - of triangulation 323
 - transfer 383
 - degenerate configuration
 - for resectioning 179
 - for transfer 382
 - degenerate configurations – see also camera resectioning, transfer, homography 2D estimation
 - degenerate configurations 127, 179, 341, 352, 443, 468, 545, 559
 - degenerate configurations
 - 2D homography 91
 - degenerate homographies 334
 - degrees of freedom (dof) 27
 - degrees of freedom (dof)
 - of trifocal tensor 368
 - degrees of freedom
 - trifocal 368
 - depth of points 162
 - derivative matrix
 - computation 145
 - determining the correct convergence of an algorithm 138
 - determining the intersection 27
 - DIAC – see dual image of the absolute conic
 - direct metric reconstruction using ω 275
 - direct motion estimation 450
 - direct solution for structure and translation 448
 - distance ratio 270
 - distance ratios
 - on a line 270
 - distortion correction 191
 - distortion function 191
 - choice 191
 - computing 191
 - distortion
 - correction 191
 - DLT – see also homography computation, transformation invariance, triangulation
 - DLT 90–91, 93, 96, 104–110, 116, 127, 150, 179–184, 187, 193–194, 276, 312, 424, 510–511, 573, 587, 595

- DLT algorithm
 - non-invariance 105
 - using line correspondences 180
 - dual absolute quadric
 - computing
 - assuming zero skew 466
 - limitations 468
 - with constrained parameters 466
 - equivalence to calibration 463
 - linear solution 465
 - specifying linear constraints 464
 - dual circular points 52
 - dual conic 31–32, 37, 53–55, 64, 475
 - dual conic 30
 - dual image of the absolute conic 464–466, 477
 - dual image of the absolute conic
 - definition 210
 - dual quadric 73
 - dual reconstruction algorithm 504–506
 - dual reconstruction algorithm 503
 - dual reconstruction algorithm
 - justification 507
 - dual
 - of absolute conic – see absolute dual quadric
 - of circular points 53
 - duality 29–31, 66, 207, 456, 503, 507, 514, 546, 548–550, 552
 - duality 29
 - duality principle 548
- E**
- eigenvalue extraction
 - Jacobi's method 581
 - elation 43, 62, 341, 631–633
 - elliptic quartic 556–558, 560
 - epipolar distance 288
 - epipolar envelope 301–303
 - epipolar line – see also epipolar lines, 46, 208, 240–249, 251–253, 263, 269, 288, 291, 295, 298–299, 301–306, 308, 311, 315–318, 324, 328, 332, 334–335, 340, 344–348, 352–353, 358–360, 372–374, 380–385, 400, 406, 443, 470
 - epipolar line 347
 - epipolar line correspondence 251
 - epipolar line homography 246
 - epipolar line
 - affine 344
 - epipolar lines 240–243, 245–247, 249, 280, 298, 300–303, 305, 307, 315–316, 328, 342, 345–347, 358–360, 371, 373–374, 380, 384–385, 392, 470
 - affine cameras 344
 - epipolar transfer
 - degeneracies 380
 - epipoles 240–242, 244, 251–252, 260, 264–266, 280, 286–287, 289, 307, 317–318, 320, 322, 328, 347, 361, 366, 371, 373, 375, 380–382, 384–385, 395–396, 404, 413, 471, 487–490
 - epipoles
 - as parameters 286
 - as tensors 413
 - for affine cameras 344
 - retrieval from trifocal tensor 395
 - retrieving 395
 - error
 - algebraic 88, 93–95, 98, 100, 106, 134, 179, 183–187, 229, 283–284, 288, 391, 395–396, 399–400, 467
 - algebraic
 - camera resectioning 186
 - geometric interpretation 183
 - geometric 98, 100–101, 103, 105, 107, 114, 130, 134, 181–182, 184, 186, 287, 318, 391, 398–399, 436, 438
 - in both images 101, 103, 134, 137, 147
 - in one image 94, 102, 112–113, 133, 136, 145
 - minimizing algebraic 129, 186, 288, 290
 - minimizing geometric 106–107, 130, 181, 290, 400
 - reproduction 112
 - reprojection 95–97, 103, 112–113, 130, 291, 312, 314, 343, 355, 401, 406, 434–435, 437
 - residual 115–116, 133–139, 150, 188, 288, 290, 400, 513
 - transfer 94–96, 98, 342
 - essential matrix 257–262, 275, 294, 356
 - computation of 294
 - definition 257
 - extraction of cameras from 258
 - four solutions 259
 - properties 257
 - estimation error 135–139
 - estimation problem 568
 - estimator 109, 117, 122, 135–136, 145, 303, 310, 568–575, 577
 - Euclidean and affine interpretations 165
 - euclidean geometry, 4
 - Euclidean transformation
 - fixed points 495
 - extending the baseline 452
 - exterior orientation 187
 - exterior orientation of camera 187
 - extracting the fundamental matrices 374
- F**
- factorization 14, 211, 258–259, 351, 355, 406, 434–437, 439–446, 449, 456, 463, 579, 583, 585, 615
 - non-rigid 442–443, 456
 - feature detection 192
 - Fisher information matrix 573, 577
 - fixed points 487, 491
 - fixed points of a Euclidean transformation 495
 - fixed points
 - of Euclidean transformation 495
 - focal length 16, 21, 36, 157, 166–167, 169, 183–184, 189–190, 192–194, 203, 227–228, 231, 234, 265, 466–469, 472, 484, 486, 497, 499–500, 568
 - forward projection 197
 - forward projection 161
 - forward projection
 - of lines 196
 - of quadrics 201
 - Frobenius norm 108, 259, 280–281, 294, 467
 - fundamental matrix
 - affine – see affine fundamental matrix
 - algebraic derivation 243
 - algorithm results and recommendations 400
 - both epipoles as parameters 286
 - computation 263
 - computation from seven points
 - connection to quadrics 545
 - computation
 - from seven points 281
 - from two planes 337
 - Gold Standard algorithm 284
 - Maximum likelihood 284
 - condition for point correspondence 245
 - consistency conditions 331
 - covariance 610
 - definition 245
 - degeneracies
 - no translation 297
 - points on a plane 553
 - points on a ruled quadric 296
 - derivation from four planes 422
 - epipolar parametrization 286

- extraction of canonical cameras 255
 - for planar motion
 - computation of 14, 78, 86, 247, 250, 252–253, 260–261, 293, 341, 389, 457–458, 486–490, 495, 497, 499
 - form of 250, 252
 - for translation
 - computation of 247
 - form of 247
 - geometric derivation 242–243
 - geometric derivation 242
 - iterative estimation 283
 - over-parametrization 286
 - parametrization of 285
 - parametrization of
 - over-parametrization 286
 - projective ambiguity given F 254
 - projective invariance of 253
 - properties 245
 - reduced 433, 508–512
 - reduced
 - computation of 509
 - Sampson method for estimation 287, 398
 - singularity constraint 280
 - symmetric part 251
- G**
- gauge freedom 623
- Gaussian**
- distribution 102–103, 135, 137, 143–144, 299–301, 565, 569–570, 572, 598, 618
 - distribution
 - isotropic 134–136
 - error 100, 102, 111, 116, 122, 150, 314
 - noise 109, 132–133, 136, 145, 289, 301, 303, 315, 320–321, 399–400, 436, 571, 575, 616
 - general motion 204, 239, 249–250, 252–253, 293, 406, 467–469, 473, 491, 494, 496, 498–499
 - general position 31–32, 35, 44, 63, 66–67, 73, 76, 92–93, 131, 296, 326, 336, 341, 349, 352, 384, 393, 406, 411, 425, 533, 539, 545
 - general position 352
 - generator
 - of quadric 75
 - geometric distance 94
 - geometric error – see also error geometric
 - camera resectioning 186
 - invariance to coordinate transforms 106
 - geometric interpretation 63, 72, 98, 234, 461, 464, 476
 - geometric interpretation 334, 349
 - geometric interpretation
 - of estimation 101
 - geometric minimization 114, 187–188
 - geometric representation 80, 82–84, 213, 246, 250, 252–253, 463
 - Givens rotation 579
 - Gold Standard algorithm 88, 114–116, 130, 181–182, 185, 284–285, 287–289, 293, 303, 310, 344, 349, 351, 397, 399–401
 - Gold Standard algorithm
 - for affine fundamental matrix 349
 - for affine homography 130
 - for fundamental matrix 284
 - for homography 114
 - for trifocal tensor 396
 - gradient descent 599
 - ground truth 223, 228, 320, 360
 - groups of projectivities 39
 - guided matching 125, 401
- H**
- Hessian 599–600
 - hierarchical merging 453
 - homogeneity. 2
 - homogeneous representation 27, 30, 33, 65–66, 159, 163, 175, 446, 516, 518
 - of lines 26
 - of points 27
 - homogeneous scaling 115, 224
 - homogeneous vector 2, 27–28, 51, 65, 88, 90, 144, 154, 249, 317, 447, 623, 625
 - homographies
 - what do they say about the camera matrices? 448
 - homography estimation
 - approximate solution 88
 - conic analogue 97
 - degenerate configurations 91
 - DLT 90
 - error in both images 134, 147
 - error in one image 94, 102, 112–113, 133, 145
 - errors in world points 182
 - experimental evaluation 187
 - from noisy points 332
 - function specification 111
 - Gold Standard algorithm 88
 - inhomogeneous solution 90
 - initialization 14, 110, 113–114, 400, 435, 621
 - iterative methods 114
 - linear cost function 100
 - over-determined solution 90
 - parametrization 110
 - symmetric transfer error 94, 112
 - using RANSAC 124
 - homography
 - 2D 91, 93, 99–101, 105, 109, 114, 123, 129, 133, 142, 145, 148, 180–181, 183, 188, 243, 270, 297, 602, 607, 609–610
 - 3D 423, 508
 - compatible with epipolar geometry 327
 - degenerate 334
 - induced by a plane 368
 - homology
 - parametrization of 630
 - planar harmonic 631
 - horopter 77, 251–253, 343, 487–488, 550–551
 - horopter 550
 - Householder matrix 128, 147, 268, 431, 580, 625–626
 - Huber cost function 617, 619–622
 - hyperboloid – see quadric, 74–75, 86, 296, 543–545, 548, 552–553, 559
- I**
- IAC – see image of the absolute conic
 - IAC
 - advantage over DIAC 477
 - ideal point 28
 - image of the absolute conic 9, 17–18, 210, 215, 231, 233, 236, 272–273, 275, 461, 469, 473, 475–479, 487
 - image of the absolute conic
 - connection to calibration 209
 - image plane
 - moving it 203
 - image-equivalent configurations
 - definition 535
 - incidence relations 66, 367, 370–372, 376–377, 379, 422
 - incidence relations
 - for lines 365
 - independently moving objects. 442
 - infinite homography 205, 250, 270–271, 274, 276, 325, 338–339, 341, 473, 475, 479–483, 496, 498, 500
 - infinite homography 270, 475

- infinite homography constraint
 - relationship to Kruppa equations 481
- infinite homography
 - definition 338
- inhomogeneous method 313
- initialization
 - for camera resectioning 187
- interest points 123–125, 290–291, 400–401, 403, 452–453, 468
- internal parameters 163, 167, 178, 187, 195, 204, 210, 228, 239, 247–248, 250, 264, 269, 273–274, 276, 339, 365, 439, 444, 458–459, 461–462, 464, 466, 468–469, 471–473, 475–479, 481–486, 489–490, 492–493, 495–500
- of camera
 - finding them 163
- intersection
 - of lines 27, 253, 269, 373
 - of parallel lines 28
- invariance to image coordinate transformations 104
- invariants 38–39, 41, 43–44, 63, 77, 176, 266, 489–490, 532, 628
- invariants 38–40, 42
- invariants
 - of a transformation 38–40, 42
- isometries 38
- iteration
 - weighted 598
- iterative algorithms 533
- iterative methods 114, 461, 485
- iterative minimization 110–113, 121–122, 128, 181, 186–187, 191, 193, 406, 462, 469
- J**
- Jacobi's method 581
- Jacobian 132, 140, 142, 144–145, 147–148, 188, 299, 301, 305, 315, 523, 598, 602–604, 606–607, 610–612, 626
- K**
- Klein
 - Erlangen Program 32
- known principal point 467, 472, 477
- Kruppa equations 458, 469–474, 481, 498, 500
- Kruppa equations
 - relationship to infinite homography 481
- L**
- least squares 110
- least squares solution 589, 597
- least-squares problems 592
- least-squares solution 589, 597
- least-squares solution
 - of constrained systems 594–595
 - using normal equations 591
 - weighted 592
- least-squares solutions
 - full rank case 588
- length ratios 6, 41, 43, 51, 54–56, 221–222, 270
- Levenberg-Marquardt 111, 186, 285, 398, 600–604, 606–609, 616, 621
- Levenberg-Marquardt
 - covariance 604
 - implementation 602
 - justification 601
 - sparse 603, 606
- line at infinity 2–6, 10, 25, 28–29, 43–44, 48–50, 62, 79, 81, 91, 160, 218, 518, 632
- line at infinity 28, 48
- line correspondences 92, 131, 180, 391–392, 394, 396, 398, 406, 411, 424, 428, 432, 447, 450, 532
- line estimation 117, 233
- line projection matrix
 - definition 198
- line reconstruction 323
- line transfer 377
- line transfer 382
- linear equations
 - systems of unknown rank 589
- linear solution 465
- lines
 - concurrent 45
 - parallel 269
 - representation 68, 393
 - transformations of 36
- M**
- Mahalanobis
 - distance 103, 112, 137, 141, 183, 282, 398, 565–566, 598, 603
 - norm 100, 129, 143
- matching
 - using trifocal tensor 123, 126–127, 291–292, 401–403, 455
- matrix
 - Euclidean 62
 - orthogonal
 - norm-preserving properties 578
 - skew-symmetric part 251
 - symmetric 590
 - symmetric part 251
- Maximum likelihood estimate 114, 127, 134, 144, 181, 184–185, 216, 279, 284–285, 291, 312, 323, 332, 349–351, 353, 397, 403, 437, 439, 456, 569, 616, 620
- Maximum likelihood estimate
 - robust 121, 125
- Maximum likelihood estimate
 - robust 125
- measurement matrix
 - reduced 186
- metric calibration 495
- metric properties 25, 48, 53, 55, 79, 82–84, 266, 458
- metric properties of reconstruction 82
- metric properties
 - recovery from images 55
- metric reconstruction
 - via the infinite homography 475
- metric rectification 55–57, 236, 479
- metric rectification 55
- metric structure 39
- metric structure : structure-
 - metric 39
- metric-calibration
 - of a stereo rig 495
- minimal solution 91–92, 113, 181, 193–194, 279, 348, 352, 406
- minimization
 - iterative 435
- MLE – see Maximum likelihood estimate
- modulus constraint 473–474, 499
- modulus constraint 473
- multiview tensor
 - affine 439
- N**
- Necker reversal 356
- Necker reversal ambiguity 356
- Newton iteration 111, 114, 597, 599–601, 616, 621
- newton's method and the hessian, 598
- non-collinearity conditions 386
- non-isotropic scaling – see scaling – non-isotropic
- non-rigid factorization 442–443, 456

- non-zero skew 164
- norm
 - in vector space 592
- normal equations 440, 591, 598–604, 611–612, 614, 623
- normalization 91, 94, 97, 107–110, 127, 129, 146, 179–181, 185, 259, 281–282, 284, 313, 391–392, 394, 396, 446, 513, 618, 624
 - for camera resectioning 180
 - for trifocal tensor 394
 - why is it essential? 107
- normalized coordinates 257
- normalized coordinates
 - for calibrated camera 257
- normalizing transformation 105, 107, 282, 394
- note. 574
- null-space 67–70, 72, 76, 90, 144, 158, 175, 179, 198–199, 234, 258, 280–281, 283, 296, 299, 309, 348, 367, 374, 405, 451, 471, 480, 538, 582, 591
- numbers of equations generated. 451
- O**
- orientation – see also camera orientation
- orientation 11, 30, 37–39, 41–42, 47–48, 105, 156, 163–165, 167, 174, 186–187, 193, 210, 215–216, 218, 230–231, 262, 264, 273, 301–302, 358, 427, 448, 486, 490, 493, 496, 523, 531
- orthogonal matrices
 - norm-preserving properties 578
- orthogonal regression 351, 489
- orthogonal regression 406
- orthogonality and ω 212
- orthogonality in the image 213
- orthogonality relationships 219
- orthographic projection 170–172, 174, 177, 456
- over-parametrization 142
- over-parametrization 70, 141–143, 177, 286, 435, 597, 605
- P**
- parallax 204, 207–208, 325, 334–337, 339–340, 352–353, 388, 404, 407
 - induced by a plane 335
- parallel lines 1–2, 4, 9, 25, 28–29, 34, 39, 41, 44, 49–50, 65, 86, 174, 176, 215, 217–219, 226, 247–248, 269–270, 293–294, 496
- parallel projection 153, 170, 172–173, 200, 344, 346, 356–357, 361
- parameter minimization 620
- parameter space 113, 134, 141–143, 145, 475, 498, 568, 570, 572, 574, 577, 623–624
- parameters
 - constant internal 466
 - limit range 570
- parametrization 76, 100, 110–111, 113–114, 127, 134, 141, 164, 181, 186, 284–286, 293–294, 316, 392, 398, 406, 467, 491, 499, 512, 568–569, 572, 574, 576, 579, 597, 610, 623–626, 633
- epipolar 286
 - of 3D rotations 624
 - of homogeneous vectors 624
 - of homology 630
 - of the n -sphere 625
 - what makes a good one? 623
- Plücker
 - coordinates 72–73, 86, 197
 - line coordinates 72, 198–199, 233
 - line representation 197
 - matrix 70, 198
- planar harmonic homology 64, 631
- planar harmonic homology 631
- planar homology
 - parametrization 630
- planar motion
 - fixed image points 487
- planar projective transformation
 - definition 33
- plane at infinity 2–4, 6, 9, 16–18, 78, 80, 82–85, 160–162, 166, 172–173, 175, 195, 209, 214, 268–272, 274–277, 313, 338, 347, 353, 425–426, 448–449, 458, 460–461, 463, 471, 473–475, 481, 487, 489, 494–496, 500, 515, 517, 519, 521–522, 527–528, 530
 - computation 495
 - methods of finding 474
- plane induced parallax 335–336
- plane induced parallax 335
- plane plus parallax 404
- plane plus parallax reconstruction
 - counting arguments 425
- plane
 - defined by three points 66
 - homogeneous vector representation 66
- point equations
 - recommended method 431
- point transfer
 - using fundamental matrix 380
 - using trifocal tensor 381
- point
 - defined by three planes 67
- points on a plane 296
- polarity 83
- pole–polar relationship 58
- positive definite matrix 598
- positive-definite matrix
 - symmetric 582
- preserve the convex hull – definition 515
- principal axis vector 158
- principal axis vector 161
- principal plane 154, 158, 160–163, 165, 173, 183, 199, 215, 271, 323, 414, 418, 425, 456–457, 518–519, 530
- principal plane 160
- principal plane
 - of camera 160
- principal point 16, 21, 154–155, 157–158, 160–161, 163, 168, 170, 185, 191, 203–204, 226–228, 231–232, 235–236, 260, 464–467, 469, 475, 477–478, 480, 484, 486, 492, 498–499
- principal point 160
- principal point offset 155
- probability density function 569
- profile
 - of a surface 200
- programming hint 129, 399
- projection
 - orthographic 171
 - parallel 170
 - scaled orthographic 171
 - weak perspective 171
- projective ambiguity 265
- projective camera
 - action on conics 199
 - action on lines 196
 - action on planes 196
 - definition 157
- projective depth 407, 444–447
- projective depth 336
- projective factorization
 - normalizing image coordinates 446
 - normalizing the weights 446
 - what is being minimized 446
- projective plane model 29, 47
- projective plane

- model 29
- topology 47
- projective realization – see realization – projective
- projective reconstruction
 - dual algorithm 504
 - from reduced cameras matrices 509
 - from seven points in n views 512
 - from six points 510
 - from six points in n views 511
- projective transformation 68
- projective transformation
 - decomposition of 42
- projectivity
 - definition 32
- pseudo-inverse 144–145, 147, 161, 185, 244, 246, 347–348, 440, 566, 590–592, 595, 598, 605, 610, 626
- Q**
- QR decomposition 186, 578–580
- quadratic 6, 72–76, 83–85, 97, 176, 195, 199–202, 234, 267, 295–298, 309, 340, 462–464, 473, 499, 541–549, 552–560
 - dual – see dual quadric
 - ruled 543, 549, 557
 - ruled
 - definition 75
- quadrics 73
- quadrics
 - classification of 74
- quadrifocal tensor 13, 411, 421, 423, 426, 432–433, 450
- quadrifocal tensor
 - derivation from four planes 421
- quartic 97, 474, 559
- quasi-affine mapping
 - definition 517
 - two-dimensional 518
- quasi-affine reconstruction
 - algorithm 527
 - bounds on π_∞ 527
- quaternions 585
- quaternions 585, 624–625
- R**
- radial distortion 188–193, 223
 - estimating 191
- RANSAC 88, 114, 117–118, 121–124, 126–128, 193, 290–292, 352, 391, 400–403, 620
- RANSAC
 - adaptive 120
 - application domain 125
 - distance threshold 118
 - for 2D homography 124
 - for homography 124
 - number of samples 119
 - run details 125
 - size of consensus set 120
- realizable points
 - definition 503
- realization 503
- realization
 - definition 503
 - strong 524
- realizations
 - equivalent
 - definition 540
 - non-equivalent 541
 - oppositely oriented 526
- recommendations 289
- reconstruction
 - affine – see affine reconstruction
- affine 339
- affine properties 81
- affine
 - from affine camera 271
 - from scene constraints 269
 - from translation 268
- Euclidean 19, 266–267, 526
- metric – see also auto-calibration
- metric 268, 272–277, 339, 439, 454, 458–461, 463–464, 467–468, 473, 476, 479, 481, 488–490, 492, 497–499, 520, 532
- metric properties 82
- projective 13, 16–18, 150, 239, 266–268, 270–271, 275–277, 310–312, 320, 330, 339, 387, 398, 434, 445–446, 448, 453, 456, 458–464, 467, 475, 479, 485, 489, 491, 495–496, 502–503, 506, 508, 511, 514–515, 519–521, 525–531, 535, 539–540, 602, 627
- quasi-affine 475, 515, 517–521, 523, 525–528, 530, 532
- rectifying homography 463
- reduced camera matrix 207
- reduced camera matrix
 - definition 502
- reduced fundamental matrix – see fundamental matrix – reduced
- reduced measurement matrix 186
- reduced measurement matrix 186
- reduced trifocal tensor – see trifocal tensor – reduced
- relation
 - bilinear 419
 - quadrilinear 420
 - trilinear 391, 397, 414–417
- reprojection error 401
- reprojection error
 - both images 95
 - geometric interpretation 96
- resection 276, 424, 435, 453, 533–534, 536–539, 549–550, 553, 558, 587
- resectioning – see camera resectioning, 77, 178, 453, 536–537, 559
 - covariance estimation 188
 - degenerate configuration 179
 - initialization 187
 - normalization 180
- residuals 187, 192, 194, 216, 298, 472
- retrieving the camera matrices 374
- robust algorithms 122
- robust methods 88, 113, 117–118, 121–123, 125, 128, 281, 291, 352, 391, 393, 401, 455, 514, 597, 617, 619, 622
- Rodrigues formula 585
- rotations
 - in n dimensions 583
 - in 3 dimensions 583
- row vector 26, 521, 578
- row vector 159
- S**
- Sampson approximation 99, 113–114, 287, 291, 308, 314, 349, 391, 398, 401
- Sampson distance 287
- Sampson error 98
- scaled orthographic projection 171
- scaling
 - isotropic 62, 107, 445
 - non-isotropic 109
 - with points near infinity 110
- scene constraints
 - distance ratios
 - on a line 270
 - parallel lines 269
 - scene orthogonality 273

- screw axis 78–79, 86, 253, 487–488, 490, 500
- screw decomposition of a rigid motion 77–79
- similarity matrix 62
- similarity transformation 39, 43, 53, 107, 184
- Singular Value Decomposition 40, 55, 70, 91, 108, 163, 186, 225, 258, 280–282, 294, 308, 344, 430, 437–439, 442, 444–445, 469, 471, 585–590, 592–596
- Singular Value Decomposition
 - computational complexity 586
 - implementation 439, 586
- singular values
 - relation to eigenvalues 586
- singularity constraint 281–282, 288, 349
- singularity constraint 349
- skew-symmetric part 250–252
- skyline structure 615–616
- solution
 - over-determined 179
- space curve 295
- span representation 70, 322
- sparse linear systems
 - solution 615
- special points 5
- stability 469
- stereo correspondence 240
- stereo correspondence 340
- stereo rectification 47, 49–50, 228–230, 249, 279, 302–303, 308–309, 339, 479
- stereo rectification
 - affine 308
 - algorithm outline 307
 - mapping the epipole to infinity 303
 - matching transforms 305
- stereo-rig
 - metric calibration 495
- stratification 276–277
- stratification 57
- stratified reconstruction
 - affine 268
 - metric 272
- structure
 - affine 9, 356, 481, 570, 573
 - metric 233, 276, 461, 491, 611
 - projective 19, 386, 423, 426, 448, 494, 535
- subspaces and tensors 442
- SVD – see Singular Value Decomposition
- symmetric epipolar distance 287
- symmetric linear equations 614
- symmetric linear equations
 - sparse 615
- symmetric matrices 590
- symmetric part 250–252, 261, 293, 489, 541
- symmetric transfer error 94, 112
- synthetic data 132, 138, 399, 572
- T**
- tangent line
 - to conic 31
- tensor notation 376
- tensor
 - pictorial representation 564
- terminology 172, 240, 445
- topology of $\mathbb{P}^1$ 47
- total least squares 110
- transfer 14, 16, 94–98, 109, 113, 123–124, 131, 143, 147–150, 221–222, 242–243, 270, 274, 299, 318, 325, 336, 338, 340, 346, 365, 368–369, 374, 377, 379–383, 385, 388, 392, 398, 462, 473, 476, 485
- transfer
 - degenerate configurations 382–383
- transformation invariance 129
- transformation invariance 184
- transformation invariance
 - of DLT 184
- transformation rule 36–37, 53–54, 63, 231, 394, 406, 475, 563–564
- transformation
 - of conic 36
 - of conics 36
 - of lines 36
 - projective 1, 3–4, 7–9, 11–13, 16, 25, 30, 33–37, 41–49, 51–55, 57, 59, 61–65, 74, 76–77, 80–84, 86–88, 91–92, 94, 101, 103–105, 115, 132–133, 137, 165, 177–178, 195–196, 201–202, 204, 210, 222, 239, 249–250, 253–255, 265–268, 271–272, 296, 303, 305–308, 310, 313, 320, 366, 374–375, 387, 389, 404, 420, 423, 434, 445, 463, 475, 482, 491, 494, 502, 508, 511, 513, 516–517, 519–522, 524, 527, 529, 531, 533–534, 539–540, 557, 559, 628–631
 - similarity 39, 43, 53, 107, 184
- translational motion 293
- triangulation 12, 263, 267, 276–277, 285, 291, 293, 310–315, 318, 320, 323–324, 332, 344, 380, 385, 397, 406, 435, 439–440, 442, 453–454, 456, 491, 568, 575
- triangulation 263
- triangulation
 - DLT method 312
 - evaluation on real images 320
 - for affine cameras 353
 - for affine cameras 439–440, 442
 - inhomogenous method 313
 - local minima 319
 - minimizing the cost function 316
 - problem formulation 315
- trifocal tensor 12–13, 16, 87, 108, 119, 123, 164, 365, 367–378, 381–383, 386–407, 411, 414–419, 422–423, 426, 431, 439, 443, 446, 452, 456, 488–489, 508, 510, 512, 514, 533, 563–564, 594, 611
- trifocal tensor
 - affine 390, 406, 418
 - algebraic properties 373
 - computation from camera matrices 377
 - computation
 - from 6 points 393
 - Gold Standard algorithm 396
 - iterative 396
 - reprojection error 401
 - results and recommendations 400
 - Sampson distance 287, 398
 - specifying lines 404
 - using algebraic minimization 395
 - constraints 392
 - definition 367
 - definition 377
 - derivation from four planes 421
 - geometrically valid 392
 - line relations 416
 - parametrization 398
 - picture 564
 - point and line relations 369
 - point relations 414
 - reduced 433, 512
 - relation to two views 418
 - transformation rule 563
- trilinearities – see also relations – trilinear
- trilinearities 378
- twisted cubic 75–77, 179–180, 251, 253, 343, 487, 536–537, 539–540, 547, 549–553, 558–559
- properties 76
- twisted pair 259, 275

V

vanishing lines 176, 213, 216, 218–219, 230, 236, 270
vanishing lines 339
vanishing points 9, 42, 51, 110, 158–159, 161, 176, 180, 195,
208, 212–213, 215, 217–219, 222–224, 226–230,
232–233, 236, 269–271, 273, 278, 294, 339, 341, 448,
458, 497
vanishing points 339
vanishing points
 computing 215
variance 102, 109, 122, 132–137, 139–141, 143–145,
147–148, 150, 188, 300–301, 321–322, 565, 568–569,
573–574, 576–577, 605, 623, 627
vector geometry 67
vector space norms 592

W

weak perspective 170–172, 357

Z

zero skew 164, 193, 224–226, 228–229, 231, 235, 273,
465–466, 472, 477–479, 484, 497–499
zero skew 466

Multiple View Geometry

A basic problem in computer vision is to reconstruct a real world scene given several images of it. Techniques for solving this problem are taken from projective geometry and photogrammetry. Here, the authors cover the geometric principles and their algebraic representation in terms of camera projection matrices, the fundamental matrix and the trifocal tensor. The theory and methods of computation of these entities are discussed with real examples, as is their use and the reconstruction of scenes from multiple images. The new edition features an extended introduction covering the key ideas in the book (which itself has been updated with additional examples and appendices) and significant new results that have appeared since the first edition.

Comprehensive background material is provided, so a reader familiar with linear algebra and basic numerical methods can understand the projective geometry and estimation algorithms presented, and implement the algorithms directly from the book.

'I am very positive about this book. The authors have succeeded very well in describing the main techniques in mainstream multiple view geometry, both classical and modern, in a clear and consistent way ... I heartily recommend this book to anyone interested in the theoretical basis of conventional and modern techniques in multiple view geometry.' *Computing Reviews*

'Overall, this book provides a clear explanation of some difficult concepts and an introduction to the mathematical equations required to use them. It presents both the theory and resulting algorithms in a style which makes them easy to understand and implement. It is well laid out, easy to navigate for reference, provides good examples (although a few more may be useful) and helpful notes and exercises at the end of each chapter.

[the book] stands out with its clear approach and explanations as being ideal for both teaching and reference for any researcher in the field.' *BMVA News*

Cover Image: Irises of human eyes © Tony Stone Images Ltd

PENN STATE UNIVERSITY LIBRARIES



A000053319586

www.cambridge.org

ISBN 0-521-54051-8



9 780521 540513