Invariant and Calibration-Free Methods in Scene Reconstruction and Object Recognition

Richard Hartley and Joe Mundy

G.E. CRD, Schenectady, NY, 12301.

Sponsored by Defense Advanced Research Projects Agency
Software & Intelligent Systems Technology Office
Algebraic Invariants
ARPA Order No. 8228
Program Code No. N/A
Issued by DARPA/CMO under Contract #MDA972-91-C-0053

Contents

Part I

Preliminaries

1:1 Statement of Problem and Outline

This document is a report on research carried out in vision invariants under DARPA contract, number #MDA 972-91-C-0053. It contains a detailed description of the research results obtained under this project. Much of the work described here has been derived from various publications that resulted from the research under the contract.

The field of vision invariants deals with view-point and pose-independent properties of objects seen in images. Broadly interpreted, it has come to mean the study of geometric properties of objects that can be deduced from images. Generally, it is assumed that the camera, or cameras taking the images are uncalibrated. An early result in the study of vision invariants is that one can not compute any invariant of a set of unstructured points seen in a single image, other than the number of points in the set. Rather than dampening research into vision invariants, this result spurred research in two distinct areas :

1. The study of structured set of points, such as planar sets of points or features belonging to some hypothesized geometric structure. Examples of such structures are solids of revolution, extruded solids or repeated structures.

2. The study of objects seen in multiple views. This study was given special impetus by the early discovery (1991) that the projective structure of a set of points can be computed from a pair of uncalibrated images. Since projective structure is often sufficient for computer vision tasks, this result meant that camera calibration is unnecessary in many applications.

Early research in vision invariants concentrated on the computation of projective invariants, usually of planar structures. Demonstrations were given of the use of such invariants in object recognition through indexing into a database using the computed geometric invariant. However interest soon developed in the area of complete shape reconstruction from several views. This brought the area close to the more traditional field of shape from motion. What has differentiated the study from the earlier shape from motion research was the insistence on uncalibrated, or only partly calibrated cameras, and the greater reliance on tools of projective geometry. This led to the discovery and intensive study of new projectively invariant entities related to camera and point positions, most notably the fundamental matrix and the trifocal tensor.

A fundamental result of Maybank and Faugeras was that given several images of a scene, one can do better than projective reconstruction if one makes the assumption that the images are all taken with one camera. In fact, one can in principal reconstruct the scene with the only ambiguity being expressed by an unknown similarity transform. In other words, essentially one can deduce the object shape except for its overall scale. Since one can do no better with a completely calibrated camera, this result further emphasizes that prior calibration is unnecessary. Much research followed in this new area of self-calibration, both in determining new methods and in finding workable algorithms.

In the study of single view scenes, the early work on invariant recognition of planar objects gave way to research into identification and recognition of 3D objects present in a single image. New techniques evolved in areas of invariant-aided grouping and segmentation as well as indexing, which was the original goal of the study of invariants. This work emphasized the truth that the structure of the scene must be taken into account at each

step of the object recognition process, having profound influence on the low level tasks as well as the high-level task of final recognition.

This report deals exclusively with work done by researchers at GE, and their collaborators, and hence makes no claim to be a complete survey of the evolving field. Nevertheless, it does constitute a body of results at the center of the field of image invariants, and touching on a large part of the contemporary research.

The report is divided into several parts as follows :

Part I of the report lays out preliminary definitions.

Part II is concerned with scene reconstruction from two uncalibrated images of a scene. It introduces the important concept of the Fundamental Matrix, and shows how it may be used for scene reconstruction, up to an indeterminate projective transformation. There are several aspects to the problem, which are treated in the different sections of this part of the report. These aspects are the computation of the fundamental matrix, factorization of the fundamental matrix to compute the camera matrices, triangulation from the two images to determine the position of points in space, and iterative refinement of the result using the Levenberg-Marquardt estimate method.

Part III deals with geometric invariants of a scene that can be computed from two views. The invariants of 4 lines in space are given detailed examination. One way of computing invariants is to perform a scene reconstruction using the method of Part II, and then computing projective invariants directly. Other methods of computing such invariants are also discussed, based on alternative geometric or algebraic techniques, but avoiding explicit scene reconstruction.

Part IV introduces the concept of *cheirality*, which is concerned with knowledge of whether points lie in the front of a camera or behind. Knowledge of this sort was used by Longuet-Higgins to select a correct scene reconstruction among four candidates. In the projective case, it leads to constraints on the set of possible projective reconstructions, which allow an improved reconstruction, known as a *quasi-affine* reconstruction of the scene to be computed. These constraints are embodied in a set of inequalities known as the *cheiral inequalities*. The study of cheirality also leads to the definition of new integer-valued quasi-affine invariants of a set of points.

Part V of the report turns to the study of the three-view case, asking what new information can be obtained by allowing a third view of a scene. The central object in this section is the trifocal tensor which plays a role in the study of three views, similar to that played by the fundamental matrix for two views. The new thing is that both lines and points can be treated with a single algorithm. As with two views, a non-iterative algorithm is described for computing the trifocal tensor and subsequently reconstructing the scene. The technique for deriving a three-view tensor extends to four views and leads to the definition of the quadrifocal tensor, which has many interesting properties.

Part VI turns to Euclidean reconstruction and self calibration in the spirit of Maybank and Faugeras. The key object for Euclidean reconstruction is the absolute conic, and the determination of its image is equivalent to camera calibration. The constraints that are imposed on the image of the absolute conic by consideration

of several images are embodied in the Kruppa equations. Three methods for self calibration are given, each one based on different assumptions of knowledge of the camera motion or partial calibration. The are

1. unconstrained motion of the camera,
2. knowledge of the cameras' principal points and
3. constraining the camera to rotatory motion only, fixing the camera center.

Part VII concludes the report with a study of object recognition from a single view of 3D objects. It is shown how geometric and invariant information is used for edge extraction and grouping as well as efficient database indexing. A wide range of different object types may be amenable to recognition using invariants. These include planar objects, surfaces of revolution, extruded objects, symmetric objects and polyhedral objects.

There are several areas in which the research in this report has been carried forward in the last year or so, by other researchers. The trifocal and quadrifocal tensors have received extensive study by many researchers such as Shashua, Torr and Zisserman, Triggs, Heyden Anandan, Weinshall and others. A logical continuation of this thread has been the interest in applying this sort of technique to video processing, in which not 3 of 4, but perhaps hundreds of images of a scene are available. The work of Carlsson and Weinshall on duality give hope that fast linear or non-iterative algorithms may become available for the analysis of such image sets. The work reported in this document serves as no more than a starting point for wide-ranging and exciting future research.

1:2 Notation

Vectors are represented by bold lower case letters, such as $\mathbf{u}$, and all such vectors are thought of as being column vectors unless explicitly transposed (for instance $\mathbf{u}^\top$ is a row vector). Vectors are multiplied as if they were matrices. In particular, for vectors $\mathbf{u}$ and $\mathbf{v}$, the product $\mathbf{u}^\top\mathbf{v}$ represents the inner product, whereas $\mathbf{u}\mathbf{v}^\top$ is a matrix. The notation $\mathbf{x}$ usually denotes a homogeneous 4-vector representing an element in $\mathcal{P}^3$, whereas $\mathbf{u}$ represents a vector in $\mathcal{P}^2$.

Homogeneous Coordinates It is convenient to express points in space and points in the image plane in homogeneous coordinates. A point in 3-space R^3 is expressed in homogeneous coordinates by a 4-vector. Specifically, the homogeneous vector $(x, y, z, t)^\top$ with $t \neq 0$ represents the point $(x/t, y/t, z/t)^\top$ of R^3 in non-homogeneous coordinates. Similarly, a homogeneous vector $(u, v, w)^\top$ represents the point $(u/w, v/w)^\top$ in R^2. One sees immediately that two homogeneous vectors that differ by a constant non-zero factor represent the same point. Consequently, two homogeneous vectors differing by a non-zero constant factor are considered to be equivalent. We may write $\mathbf{x} \approx \mathbf{x}'$ to express this equivalence of homogeneous vectors. However, this notation quickly becomes tedious, and so usually the equivalence of two homogeneous vectors is expressed using an equality sign. Thus we write $\mathbf{x} = \mathbf{x}'$ to mean that the two vectors are equal up to a multiplicative factor.

The plane at infinity It was seen in the previous paragraph that a homogeneous vector $\mathbf{x} = (x, y, z, t)^\top$ represents a point in R^3 if and only if $t \neq 0$. The set of all non-zero homogeneous 4-vectors form the projective 3-space P^3. The set of points $\mathbf{x} = (x, y, z, 0)^\top$ form a plane consisting of points not in R^3. This is referred to as the plane at infinity. Thus, projective 3-space P^3 is made up of R^3 plus the plane at infinity. A point in R^3 may be conveniently expressed in homogeneous coordinates as $(x, y, z, 1)^\top$.

In the same way, projective 2-space P^2 is made up of R^2 plus a line at infinity, consisting of points $(u, v, w)^\top$ in homogeneous coordinates with $w = 0$.

Projective geometry is the study of the projective space $\mathcal{P}^n$. In projective geometry it is usual not to distinguish the plane (or line) at infinity from any other plane. Thus, all points, whether infinite of finite are created equal. In the area of computer vision dealing with points in space and projections of these points by pinhole cameras, we also deal with homogeneous vectors. However, in computer vision it is often appropriate to distinguish the plane at infinity and treat it differently from other planes. For instance, no person ever managed to photograph a point at infinity, nor did anyone ever manage to place a camera on the plane at infinity. Certain concepts such as front and back of the camera and affine and Euclidean reconstruction and invariants make no sense without considering the plane at infinity to be distinguished.

The norm of a vector $\mathbf{f}$ is equal to the square root of the sum of squares of its entries, that is the Euclidean length of the vector. Similarly, for matrices, we use the Frobenius norm, which is defined to be the square root of the sum of squares of the entries of the matrix.

1:3 The Camera Model

In this report we will be concerned most particularly with the projective camera model often referred to as a pinhole camera model in computer vision. We prefer to use the term pinhole model for a somewhat restricted form of camera model with just 9 parameters, as will be described. The full projective camera model is a slight generalization of the pinhole model.

The basic pinhole model We consider the central projection of points in space onto a plane. Let the centre of projection be the origin of a Euclidean coordinate system, and consider the plane $z = f$, henceforth called the focal plane. Under the pinhole camera model, a point in space with coordinates $\mathbf{x} = (x, y, z)^\top$ is mapped to the point in the focal plane where a line joining the point $\mathbf{x}$ to the centre of projection meets the focal plane. This is shown in Fig 1.1. By similar triangles, one quickly computes that the point $(x, y, z)^\top$ is mapped to the point $(fx/z, fy/z, f)^\top$ on the focal plane. Ignoring the final coordinate, we see that

$$(x, y, z)^\top \mapsto (fx/z, fy/z)^\top \tag{1.1}$$

describes the central projection mapping. This is a mapping from Euclidean space R^3 to R^2.

The centre of projection is often called the *camera centre*. The line from the camera centre perpendicular to the focal plane is called the *principal axis* of the camera, and the point where the principal axis meets the focal plane is called the *principal point*.

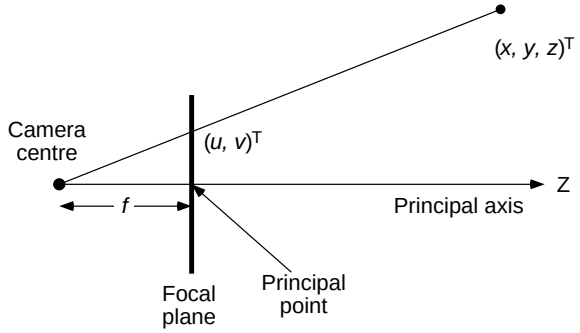


Figure 1.1: *Pinhole camera geometry*

Central projection using homogeneous coordinates If the points are written in homogeneous coordinates, then central projection is very simply expressed as a linear mapping in homogeneous coordinates. In particular, the expression (1.1) may be written in terms of matrix multiplication as

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.2}$$

The matrix in this expression may be written as $\mathrm{diag}(f, f, 1)[I \mid \mathbf{0}]$ where $\mathrm{diag}(f, f, 1)$ is a diagonal matrix and $[I \mid \mathbf{0}]^\top$ represents a matrix divided up into a 3×3 block (the identity matrix) plus a column vector, here the zero vector. If the image of a point $\mathbf{x}$ under central projection is $\mathbf{u}$ then we see that

$$\mathbf{u} = \mathrm{diag}(f, f, 1)[I \mid \mathbf{0}]\mathbf{x} \ .$$

Principal point offset The expression (1.1) assumed that the origin of coordinates in the image plane is at the principal point. In practice, the principal point may not be accurately known. In general, therefore, we will have a mapping

$$(x, y, z)^\top \mapsto (fx/z + p_u, fy/z + p_v)^\top$$

where $(p_u, p_v)^\top$ are the coordinates of the principal point, otherwise known as the *principal point offset*. This equation may be expressed conveniently in homogeneous coordinates as

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx + p_u \\ fy + p_v \\ z \end{pmatrix} = \begin{bmatrix} f & & p_u & 0 \\ & f & p_v & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.3}$$

Camera Rotation and Translation So far, it has been assumed that the camera is located at the origin of a Euclidean coordinate system with the principal axis of the camera pointing straight down the z-axis. Such a coordinate system may be called the *camera coordinate frame*. In general, however, points in space will be expressed in

terms of a different Euclidean coordinate frame, known as the *world* coordinate frame. The two coordinate frames are related via a rotation and a translation. If $\mathbf{x}$ is a non-homogeneous vector representing the coordinates of a point in the world coordinate frame, and $\mathbf{x}'$ represents the same point in the camera coordinate frame, then we may write $\mathbf{x}' = R(\mathbf{x} - \mathbf{c})$ where $\mathbf{c}$ represents the coordinates of the camera centre in the world coordinate frame, and R is a 3×3 rotation matrix representing the orientation of the camera coordinate frame. This equation may be written in homogeneous coordinates as

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\mathbf{c} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Putting this together with the equation 1.2 leads to the formula

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{bmatrix} f & & p_u \\ & f & p_v \\ & & 1 \end{bmatrix} R[I \mid -\mathbf{c}] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.4}$$

where

- f is the focal length of the camera.
- $(p_u, p_v)^\top$ are the image coordinates of the principal point.
- R is the rotation of the camera.
- $\mathbf{c}$ is the location of the camera centre.

Writing

$$K = \begin{bmatrix} f & & p_u \\ & f & p_v \\ & & 1 \end{bmatrix} \tag{1.5}$$

we see that image of a point $\mathbf{x}$ under a pinhole camera mapping is

$$\mathbf{u} = KR[I \mid -\mathbf{c}]\mathbf{x} \tag{1.6}$$

This is the general mapping given by a pinhole camera. One sees that a general pinhole camera has 9 degrees of freedom.

CCD cameras The pinhole camera model just derived assumes that both object coordinates (that is the 3D world coordinates) and image coordinates are Euclidean coordinates having equal scales in all axial directions. In the case of CCD cameras, there is the additional possibility of having unsquare pixels. If image coordinates are measured in pixels, then this has the extra effect of introducing unequal scale factors in each direction. In particular if the number of pixels per unit distance in image coordinates are m_u and m_v in the u and v directions, then the cameras transformation from world coordinates to pixel coordinates is obtained by multiplying (1.5) on the left by an extra factor

diag$(m_1, m_2, 1)$. Thus, the general form of the calibration matrix of a CCD camera is

$$K = \begin{bmatrix} k_u & & p_u \\ & k_v & p_v \\ & & 1 \end{bmatrix} \tag{1.7}$$

where $k_u = fm_u$ and $k_v = fm_v$ represent the focal length of the camera in terms of pixel dimensions in the u and v direction respectively. Similarly, (p_u, p_v) are the pixel coordinates of the principal point. A CCD camera thus has 10 degrees of freedom.

General Projective Camera For simplicity and added generality, we can consider a calibration matrix of the form

$$K = \begin{bmatrix} k_u & s & p_u \\ & k_v & p_v \\ & & 1 \end{bmatrix} . \tag{1.8}$$

The added parameter s is referred to as the *skew* parameter. The skew parameter will be zero for most normal cameras. However, in certain unusual instances it can take non-zero values. In a CCD camera if the pixel elements in the CCD array are skewed so that the u and v axes are not perpendicular, then skewing of the image can result. This is admittedly very unlikely to happen. The CCD camera model assumes that the image has been stretched by different amounts in the two axial directions. If on the other hand the image is stretched in a non-axial direction, then skewing results. To see this, consider what happens to a pair of axes if the image is stretched in a diagonal direction : the axes do not remain perpendicular. Skewing may occur if images taken by a pinhole camera (such as an ordinary film camera) is subsequently magnified. If the axis of the magnifying lens is not perpendicular to the film plane or the new image plane, then the image will be skewed. In all of these cases, the effect of skew will be small, so generally the parameter s will be very small compared with k_u.

A camera with camera matrix $P = KR[I \mid -\mathbf{c}]$ for which the calibration matrix K is of the form (1.8) will be called a *projective* camera. A projective camera has 11 degrees of freedom. This is the same number of degrees of freedom as a 3×4 matrix, defined up to an arbitrary scale. Any general 3×3 matrix M may be decomposed as a product $M = KR$ where K is upper triangular and R is a rotation matrix. Thus, the class of projective cameras with camera centre at a finite point corresponds to the class of matrices of the form $P = [M \mid -M\mathbf{c}]$. This is a general 3×4 matrix with the sole restriction that the left-hand 3×3 block is non-singular. We may further extend the class of camera matrices to include cameras at infinity. A general projective camera is one which is represented by an arbitrary 3×4 matrix of rank 3.

In summary, we may distinguish the following hierarchy of camera models.

General Projective Camera. The general projective camera is one for which the object-space to image-space transformation is represented by the mapping $\mathbf{u} = P\mathbf{x}$ in homogeneous coordinates, where P is an arbitrary 3×4 matrix of rank 3. Matrix P is defined only up to a non-zero multiplicative factor:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.9}$$

Finite Projective Camera. This terminology may be used to describe a projective camera for which the camera centre is not at infinity. The camera matrix P may be written in the form $P = [M \mid -M\mathbf{c}]$ where $\mathbf{c}$ is the camera centre and M is non-singular. Using the QR decomposition M can be decomposed as $M = KR$, where R is a rotation and K is an upper triangular matrix with positive diagonal entries, called the *calibration matrix*.

Zero-skew camera This is a finite projective camera with zero skew, such as a CCD camera. Hence the calibration matrix has the form (1.7).

Pinhole camera A finite projective camera with calibration matrix of the form (1.4) having zero skew and equal magnification in each direction.

Calibrated Camera. A camera with matrix of the form $P = [R \mid -R\mathbf{c}]$. In other words, the calibration matrix in assumed to be the identity. It is not implied, of course that the calibration matrix really is the identity matrix, but rather that the effect of the calibration matrix K, once it is known, may be removed, and that one may without loss of generality assume that $K = I$. Accordingly, the term *calibrated camera* will be used in this report to mean that $K = I$.

All these different camera models have been considered in the literature. The terminology proposed here is certainly not standardized. For instance, the term pinhole camera has been used to denote more general camera models, even the general projective camera model. For the most part, in this report we will consider either general or finite projective cameras. When we talk of camera calibration, or cameras with specific calibration, however, it is implicit that we are talking of finite projective cameras, since there is no natural way to define the calibration matrix of a camera at infinity.

Cameras with centres at infinity form a different hierarchy, which will not be discussed further in this report.

Part II

Reconstruction from Two Views

One of the major problems addressed in this report is the reconstruction problem from several views. Consider points $\mathbf{x}_i$ in space as seen by a set of cameras with camera matrices P_j. In general, neither the point locations nor the camera matrices are known. In general we assume that the cameras are projective cameras modelled by the full 11-parameter projective camera model. In the most general case, nothing is assumed about the individual cameras, and in particular, their calibration matrix K_j is assumed to be unknown.

Let the image of the i-th point $\mathbf{x}_i$ seen in the j-th camera be denoted by $\mathbf{u}_j^i$. We assume that these coordinates are known, and in particular the matching of the points across the several images is assumed to be known. The task of finding these points and carrying out cross-image matching is a significantly difficult problem in itself, but it will not be considered in this report.

The reconstruction problem is, given the coordinates $\mathbf{u}_j^i$, to find the points $\mathbf{x}_i$, and the cameras $\mathbf{P}_j$ so that

$$u_j^i = P_j \mathbf{x}_i \tag{2.1}$$

for all i and j.

Without further restriction, this problem does not have a unique solution. For instance, any translation of the points and corresponding translation of the camera positions will not change the image points. Consequently, the reconstruction of the scene is at least indeterminate with respect to translation. One may similarly see that the orientation and overall scale of the scene may not be determined. Thus, even with calibrated cameras, the geometry of the scene may not be determined more closely than up to a similarity transformation (translation, rotation and scaling).

For uncalibrated cameras, there is a further ambiguity of the scene. It may be seen that the scene may not be determined by its projections more precisely than up to a general projective transformation of space. It is shown in this report, however, that this is the full extent of reconstruction ambiguity. Provided sufficient point matches are given, a scene can be reconstructed up to an indeterminate projective transformation by its projection into two or more views with uncalibrated projective cameras. The resulting reconstruction of the scene is known as a projective reconstruction.

Under more restricted conditions, it may be possible to determine the scene more exactly than in a general projective reconstruction. A reconstruction that is known to differ from the true reconstruction by at most a 3D affine transformation is called an affine reconstruction, and one that differs by a Euclidean transformation from the true reconstruction is called a Euclidean reconstruction. The term Euclidean transformation will be used in this report to mean a similarity transform, namely the composition of a rotation, a translation and a uniform scaling. Affine or Euclidean reconstruction is possible under various circumstances, such as restricted motions of the camera, an assumption of the same calibration for each of the cameras, or imposed geometric constraints on the scene. Euclidean and affine reconstruction will be considered in part VI of this report.

In the presence of noise in the image measurements, equations (2.1) will not be satisfied exactly. An obvious approach in these circumstances is to attempt to find points $\mathbf{x}_i$ and camera matrices P_j that map the points as close as possible to the observed positions. This problem can be cast as a minimization problem in which one attempts to minimize some cost function measuring the difference between the observed positions of the points and the projections of the predicted world points $\mathbf{x}_i$. A suitable cost function is the sum

of squares of errors. Thus if

$$P_j \mathbf{x}_i = \hat{\mathbf{u}}_j^i$$

then the cost function to be minimized is

$$\sum_{i,j} d(\hat{\mathbf{u}}_j^i, \mathbf{u}_j^i)^2 \tag{2.2}$$

where $d(*,*)$ represents Euclidean distance in the image. One seeks to minimize this cost function over all choices of points $\mathbf{x}_i$ and camera matrices P_j.

Presented in this manner, the reconstruction problem may be considered simply as a non-linear least-squares minimization problem. The theme of this report is to investigate more efficient algorithms for reconstruction than the standard non-linear approaches. Often such methods (as will be discussed) provide a fast solution, sometimes at some cost in accuracy. To evaluate the proposed algorithms, their results are usually compared with the best results obtained by non-linear least-squares minimization. Therefore, we begin by describing a standard algorithm for solving such non-linear least-squares problems – the Levenberg-Marquardt algorithm.

2:1 Levenberg Marquardt Minimization

The Levenberg-Marquardt (LM) algorithm is a well known algorithm for parameter estimation ([55]). It is a variation on Newton iteration which will be described first, in general terms.

2:1.1 Newton Iteration

Given a hypothesized functional relation $\mathbf{y} = f(\mathbf{x})$ where $\mathbf{x}$ and $\mathbf{y}$ are vectors in some Euclidean spaces R^m and R^n, and a measured value $\hat{\mathbf{y}}$ for $\mathbf{y}$, we wish to find the vector $\hat{\mathbf{x}}$ that most nearly satisfies this functional relation. More precisely, we seek the vector $\hat{\mathbf{x}}$ satisfying $\hat{\mathbf{y}} = f(\hat{\mathbf{x}}) + \hat{\epsilon}$ for which $||\hat{\epsilon}||$ is minimized. The method of Newton iteration starts with an initial estimated value $\mathbf{x}_0$, and proceeds to refine the estimate under the assumption that the function f is locally linear. Let $\hat{\mathbf{y}} = f(\mathbf{x}_0) + \epsilon_0$. We assume that the function f is approximated at $\mathbf{x}_0$ by $f(\mathbf{x}_0 + \mathbf{\Delta}) = f(\mathbf{x}_0) + J\mathbf{\Delta}$, where J is the linear mapping represented by the Jacobian matrix $J = \partial\mathbf{y}/\partial\mathbf{x}$. Setting $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{\Delta}$ leads to $\hat{\mathbf{y}} - f(\mathbf{x}_1) = \hat{\mathbf{y}} - f(\mathbf{x}_0) - J\mathbf{\Delta} = \epsilon_0 - J\mathbf{\Delta}$. It is required to minimize $||\epsilon_0 - J\mathbf{\Delta}||$. Solving for $\mathbf{\Delta}$ is a linear minimization problem that can be solved by the method of normal equations. The minimum occurs when $J\mathbf{\Delta} - \epsilon_0$ is perpendicular to the row space of J, which leads to the so-called *normal equations* $J^\top(J\mathbf{\Delta} - \epsilon_0) = 0$ or $J^\top J\mathbf{\Delta} = J^\top\epsilon_0$. Thus, the solution is obtained by starting with an estimate $\mathbf{x}_0$ and computing successive approximations according to the formula

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{\Delta}_i$$

where $\mathbf{\Delta}_i$ is the solution to the normal equations

$$J^\top J\mathbf{\Delta}_i = J^\top\epsilon_i \ .$$

Matrix J is the Jacobian $\partial\mathbf{y}/\partial\mathbf{x}$ evaluated at $\mathbf{x}_i$ and $\epsilon_i = \hat{\mathbf{y}} - f(\mathbf{x}_i)$ One hopes that this algorithm will converge to the required least-squares solution $\hat{\mathbf{x}}$. 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 $\mathbf{x}_0$.

2:1.2 Levenberg-Marquardt Iteration

The Levenberg-Marquardt (abbreviated LM) iteration method is a slight variation on the Newton iteration method. The normal equations $N\boldsymbol{\Delta} = J^\top J\boldsymbol{\Delta} = J^\top \epsilon$ are replaced by the *augmented normal equations* $N'\boldsymbol{\Delta} = J^\top \epsilon$, where $N'_{ii} = (1+\lambda)N_{ii}$ and $N'_{ij} = N_{ij}$ for $i \neq j$. The value λ is initially set to some value, typically $\lambda = 10^{-3}$. If the value of $\boldsymbol{\Delta}$ obtained by solving the augmented normal equations leads to a reduction in the error, then the increment is accepted and λ is divided by 10 before the next iteration. On the other hand if the value $\boldsymbol{\Delta}$ leads to an increased error, then λ is multiplied by 10 and the augmented normal equations are solved again, this process continuing until a value of $\boldsymbol{\Delta}$ 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 $\boldsymbol{\Delta}$ is found constitutes one iteration of the LM algorithm.

2:1.3 Implementation

Based on the implementation of the LM algorithm in [55] we have coded a general minimization routine. To use this algorithm in the simplest form it is necessary only to provide a routine to compute the function being minimized, a goal vector $\hat{\mathbf{y}}$ of observed or desired values of the function and an initial estimate $\mathbf{x}_0$. If desired, it is possible to provide a function to compute the Jacobian matrix J. If a null function is specified, then the differentiation is done numerically. Numerical differentiation is 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. The value δ is set to the maximum of $|10^{-4} * x_i|$ and 10^{-6}. This choice seemingly gives a good approximation to the derivative. In practice, we have seen almost no disadvantage in using numerical differentiation, though for simple functions f we prefer to provide a routine to compute J, partly for aesthetic reasons, partly because of a possible slightly improved convergence and partly for speed.

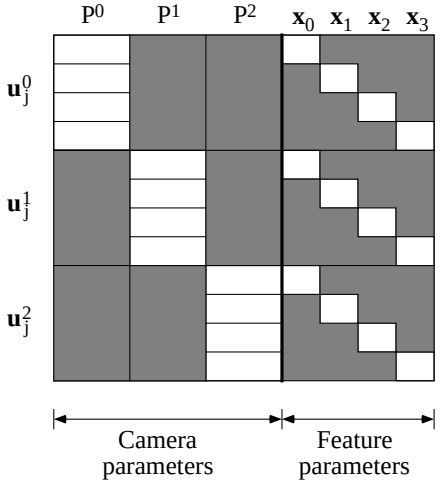
As an alternative to all the dependent variables being equally weighted, it is possible to provide a weight matrix specifying the weights of the dependent variables $\mathbf{y}$. This weight matrix may be diagonal specifying independent weights for each of the y_i, or else it may be symmetric, equal to the inverse of the covariance matrix of the variables y_i. If C is the covariance matrix of $\mathbf{y}$, then the normal equations become $J^\top C^{-1} J\boldsymbol{\Delta}_i = J^\top C^{-1}\epsilon_i$.

2:1.4 Sparse Methods in LM Scene Reconstruction

In pose estimation and scene reconstruction problems involving several cameras the LM algorithm is appropriately used to find a least-squares solution. In cases where the camera parameters and the 3D locations of the points are to be found simultaneously the Jacobian matrix, J has a special block structure. This block structure gives rise to a sparse block structure of the normal equations. It is possible to take advantage of this to

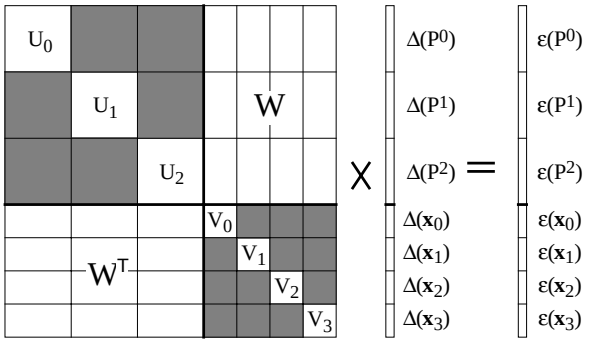
achieve an enormous simplification in the solution of the normal equations. This method is described in [68] but is presented here for the convenience of the reader.

In the case of scene reconstruction, the variable parameters fall into two classes, namely the camera parameters and the coordinates of the points $\mathbf{x}_j$. Altering the coordinates of a point $\mathbf{x}_j$ will cause a change in the coordinates of each point $\mathbf{u}_j^i$ with the same index j as the point. Similarly, altering the parameters of a camera P_i will lead to a change in the points $\mathbf{u}_j^i$ with the same index i as the camera. Consequently, the matrix J of partial derivatives of the dependent parameters with respect to the independent parameters has a particular sparse structure as shown in the following diagram.



The diagram shows the case for three camera and four points, but the general scheme is easily extended to any number of points and cameras. In the case where certain of the parameters are set to fixed values (for instance, the camera P_0 may be fixed to the value $[I \mid \mathbf{0}]$) then the corresponding columns are missing from the matrix. In addition, in the cases where some points are not visible in some views, then the corresponding rows are missing from the matrix. This all makes very little difference to the following discussion.

Because of the structure of the matrix J, the normal equations $J^\top J\mathbf{\Delta} = J^\top \epsilon$ has a special block structure as follows:



(2.3)

It is possible to give specific formulae for each of the blocks in the normal equations. Specifically, let $D(\mathbf{u}_j^i, P_i)$ represent the matrix of partial derivatives of the coordinates of the image point $\mathbf{u}_j^i$ with respect to the parameters of the matrix P_i. Similarly, let

$D(\mathbf{u}_j^i, \mathbf{x}_j)$ be the matrix of partial derivatives of the coordinates of $\mathbf{u}_j^i$ with respect to the coordinates of the point $\mathbf{x}_j$. Further, write $\epsilon(\mathbf{u}_j^i)$ to represent the current residual error in the point $\mathbf{u}_j^i$ Then, we may write

$$\begin{aligned} U_i &= \sum_j D(\mathbf{u}_j^i, P_i)^\top D(\mathbf{u}_j^i, P_i) \\ V_j &= \sum_i D(\mathbf{u}_j^i, \mathbf{x}_j)^\top D(\mathbf{u}_j^i, \mathbf{x}_j) \\ W_{ij} &= D(\mathbf{u}_j^i, P_i)^\top D(\mathbf{u}_j^i, \mathbf{x}_j) \\ \epsilon(P_i) &= \sum_j D(\mathbf{u}_j^i, P_i)^\top \epsilon(\mathbf{u}_j^i) \\ \epsilon(\mathbf{x}_j) &= \sum_i D(\mathbf{u}_j^i, \mathbf{x}_j)^\top \epsilon(\mathbf{u}_j^i) \end{aligned} \tag{2.4}$$

The normal equations (2.3) may be written in the form

$$\begin{bmatrix} U & W \\ W^\top & V \end{bmatrix} \begin{pmatrix} \Delta(P) \\ \Delta(X) \end{pmatrix} = \begin{pmatrix} \epsilon(P) \\ \epsilon(X) \end{pmatrix}$$

where each of the matrices U, V and the vectors $\Delta(P)$, $\Delta(X)$, $\epsilon(P)$ and $\epsilon(X)$ is itself made up of subblocks.

We assume that V is invertible, and multiply each side of the normal equations on the left by the matrix

$$\begin{bmatrix} I & -WV^{-1} \\ 0 & I \end{bmatrix}$$

The resulting set of equations

$$\begin{bmatrix} U - WV^{-1}W^\top & 0 \\ W^\top & V \end{bmatrix} \begin{pmatrix} \Delta(P) \\ \Delta(X) \end{pmatrix} = \begin{pmatrix} \epsilon(P) - WV^{-1}\epsilon(X) \\ \epsilon(X) \end{pmatrix} \tag{2.5}$$

may be divided into two sets of equations, to be solved separately. From the top half of (2.5), one obtains

$$(U - WV^{-1}W^\top)\Delta(P) = \epsilon(P) - WV^{-1}\epsilon(X) \tag{2.6}$$

which may be solved to get $\Delta(P)$. The resulting solution may then be substituted back into the bottom half of (2.5) providing a set of equations $V\Delta(X) = \epsilon(X) - W^\top\Delta(P)$, or

$$\Delta(X) = V^{-1}(\epsilon(X) - W^\top\Delta(P)) \ . \tag{2.7}$$

Because of the block-diagonal form of V, the equations (2.6) may be computed efficiently using the quantities computed in (2.4). Specifically, the matrix $A = U - WV^{-1}W^\top$ divides naturally into sub-blocks, where the (i, j)-th sub-block is the matrix

$$A_{ij} = \delta_{ij}U_i - \sum_k W_{ik}V_k^{-1}{W_{jk}}^\top \tag{2.8}$$

The vector $\mathbf{b} = \epsilon(P) - WV^{-1}\epsilon(X)$ also divides into blocks of the form

$$\mathbf{b}_i = \epsilon(P_i) - \sum_j W_{ij}V_j^{-1}\epsilon(\mathbf{x}_j) \tag{2.9}$$

Matrix A and the vector $\mathbf{b}$ may be computed directly, without needing to compute and store either the matrix J, or the the normal equations (2.3). The amount of computation required is linear in the number of points $\mathbf{x}_j$ involved, and also linear in the total number of observed points $\mathbf{u}_j^i$.

Similarly, the back-substitution given by (2.10) may be done block-by-block as follows :

$$\Delta(\mathbf{x}_j) = V_j^{-1}(\epsilon(\mathbf{x}_j) - \sum_i W_{ij}{}^\top \Delta(P_i)) \tag{2.10}$$

The back substitution also requires computation time linear in the number of points involved.

The above algorithm was described for the case of Newton iteration. It is easy to see how to extend this to LM iteration. One needs simply to augment the matrix $J^\top J$, which comes down to augmenting the matrices U_i and V_j in (2.3). Augmenting the matrices V_j will help to ensure that they are invertible, even in degenerate cases where V_j is singular. This effect of augmenting the normal equations is the reason that it is not essential to avoid over-parametrization of the minimization problem.

This method is easily extended in many ways :

1. To allow different weightings to errors in the different measured image points $\mathbf{u}_j^i$.
2. To allow estimated values (with confidence weightings) to be provided for individual camera or point parameters.
3. To allow for relations to be specified between the parameters of different cameras, such as specifying that two cameras have the same focal length, or that a set of cameras all lie in a straight line.

Using these sparse methods, it is possible to solve systems where there are thousands of point correspondences. In fact, we have solved in reasonable time systems in which more than 5000 point correspondences were given. If such a system were solved using the complete normal equations, then the dimension of the system of normal equations would be greater than 15000×15000, and solving it using usual methods (for instance Gaussian elimination) would be out of the question.

2:1.5 Projective Reconstruction

Mohr et. al. ([48]) have reported a direct LM approach to projective reconstruction. However, with our recoding of their algorithm we have been unable to obtain reliable convergence in all cases. Therefore, we shall describe a different (although similar) approach, also based on LM iteration.

As usual, we assume that errors in the data are manifested as errors in measurement of the pixel locations of the $\mathbf{u}_j^i$, and that these errors are independent and gaussian. As with Euclidean reconstruction, the problem is to find the camera matrices, P_i and points $\mathbf{x}_j$ such that $\hat{\mathbf{u}}_j^i = P_i \mathbf{x}_j$ and such that the squared error sum

$$\sum_{i,j} d(\hat{\mathbf{u}}_j^i, \mathbf{u}_j^i)^2$$

is minimized. Without loss of generality (and without changing the value of the error expression) it may be assumed that the first camera has matrix $P_0 = [I \mid 0]$. This least-squares minimization problem is different from the one described in Section 6:3.1. The problem is formulated in the form $\mathbf{y} = f(\mathbf{x})$ where the set of independent variables $\mathbf{x}$ comprise the 3D coordinates of all the points in space and the entries of the camera matrices P_i for $i > 0$. The dependent variables $\mathbf{y}$ are the image coordinates.

The main difference between our algorithm and that of Mohr et. al. is that whereas they fix the locations of five points in space, we fix the location and the orientation of one of the cameras. In particular, we set $P_0 = [I \mid 0]$. In the algorithm of Mohr et. al. a check is necessary to make sure that the five points chosen are not in fact coplanar. Such a check is not necessary in our algorithm. Setting $P_0 = [I \mid 0]$ still leaves three degrees of freedom. The LM method easily handles systems with redundant parameters, however, so this is not a problem. If desired, however, it is possible to constrain the solution completely by specifying three arbitrary points to lie on the plane at infinity. In doing this it is necessary to check that the points are not collinear, or coplanar with the camera centre of P_0, which may be done easily be choosing three points that do not map to collinear points in the image corresponding to P_0.

In order for this iterative refinement method to work effectively, it is necessary to start with a good estimate to be used as an initial configuration. This is done by using a linear method of projective reconstruction which will be described in the next several sections of this report.

2:2 The Fundamental Matrix

In 1982, Longuet-Higgins ([42]) gave a solution to the reconstruction problem for a pair of images taken with calibrated cameras, introducing what became known as the *essential matrix*, denoted Q. He showed that for a pair of calibrated cameras with matrices $P = [I \mid \mathbf{0}]$ and $P' = [R \mid -R\mathbf{c}]$ there exists a matrix Q with the following property. If $\mathbf{u} \leftrightarrow \mathbf{u}'$ are a pair of corresponding points in the two images, then ${\mathbf{u}'}^\top Q\mathbf{u} = 0$.

To consider this further, we need a new terminology. Let $\mathbf{t}$ be a 3-vector and M be a 3×3 matrix. We denote by $\mathbf{t} \times M$ the matrix formed by taking the cross-product of $\mathbf{t}$ with the columns of M separately. Similarly, $M \times \mathbf{t}$ is the matrix formed by taking the cross product of each rows of M with $\mathbf{t}$ separately. If $\mathbf{t} = (t_x, t_y, t_z)$, then we define a matrix

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_x & t_y \\ t_x & 0 & -t_z \\ -t_y & t_z & 0 \end{bmatrix} . \tag{2.11}$$

One quickly verifies that

$$\mathbf{t} \times M = [\mathbf{t}]_\times M$$

and

$$M \times \mathbf{t} = M[\mathbf{t}]_\times \ .$$

Longuet-Higgins showed (effectively) that the essential matrix Q corresponding to a pair of camera matrices $[I \mid \mathbf{0}]$ and $[R \mid -R\mathbf{c}] = [R \mid \mathbf{t}]$ is the matrix

$$Q = \mathbf{t} \times R \ .$$

Longuet-Higgins idea is to use sufficiently many point matches $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ to determine Q using the relation $\mathbf{u}'^{\top} Q \mathbf{u} = 0$. With at least 8 point matches, Q may be determined using linear techniques. Subsequently, the matrix Q may be factored as $Q = \mathbf{t} \times R$ to find the two camera matrices.

2:2.1 Generalization to Projective Cameras

Much of the work of Longuet-Higgins may be generalized to projective cameras. To conform with current terminology, we will define a matrix, called the *fundamental matrix*, denoted F, which is associated with a pair of camera matrices. In the case where the camera matrices correspond to calibrated cameras, F will be identical with the essential matrix Q of Longuet-Higgins.

To derive the existence and properties of the fundamental matrix, we consider a general pair of camera matrices represented by $P = (M \mid -M\mathbf{c})$ and $P' = (M' \mid -M'\mathbf{c}')$. This is the general form for camera matrices for which the camera is not located at infinity. We will consider formulas for general cameras, possibly located at infinite points later on in this report. Suppose that $\mathbf{u}$ and $\mathbf{u}'$ are a pair of matching points as seen in the two images.

We consider the ray in space consisting of all points that map to point $\mathbf{u}$ in the first image. This ray is evidently a straight line passing through the camera centre, $\begin{pmatrix} \mathbf{c} \\ 1 \end{pmatrix}$. A further point on this ray is the point at infinity $\begin{pmatrix} M^{-1}\mathbf{u} \\ 0 \end{pmatrix}$. We now consider the image of this ray as seen by the second camera. Transform P' takes these two points $\begin{pmatrix} \mathbf{c} \\ 1 \end{pmatrix}$ and $\begin{pmatrix} M^{-1}\mathbf{u} \\ 0 \end{pmatrix}$ on the ray to points $M'\mathbf{c} - M'\mathbf{c}$‘ and $M'M^{-1}\,\mathbf{u}$ in the second image. The line through these points is given by the cross product

$$M'(\mathbf{c} - \mathbf{c}') \times M'M^{-1}\mathbf{u} \ . \tag{2.12}$$

This line in the second image corresponding to the point $\mathbf{u}$ in the first image is known as the *epipolar line* corresponding to $\mathbf{u}$. It is a line passing through the point $M'(\mathbf{c} - \mathbf{c}')$ which is the image of the first camera centre as seen in the second image. This point is known as the *epipole* in the second image. As $\mathbf{u}$ varies, the set of corresponding epipolar lines form a pencil of lines all passing through the epipole.

Using notation as defined in (2.11), the expression (2.12) can be written as

$$M'(\mathbf{c} - \mathbf{c}') \times M'M^{-1}\mathbf{u} = [M'(\mathbf{c} - \mathbf{c}')]_{\times} M'M^{-1}\mathbf{u} = F\mathbf{u} \tag{2.13}$$

where $F = [M'(\mathbf{c} - \mathbf{c}')]_{\times} M'M^{-1}$. The matrix F is the fundamental matrix corresponding to the two camera matrices. Let $\mathbf{u}'$ be a point in the second image corresponding to the point $\mathbf{u}$ in the first image. The corresponding point $\mathbf{x}$ in space giving rise to these two image points must lie on the ray corresponding to image point $\mathbf{u}$. Thus the image of $\mathbf{x}$, namely point $\mathbf{u}'$, must lie on the epipolar line $F\mathbf{u}$. Consequently, $\mathbf{u}'^{\top} F\mathbf{u} = 0$.

Summary of Properties of the Fundamental Matrix. The following properties of the fundamental matrix are easily derived from the preceding discussion.

Proposition 2.1.

1. *F is a 3×3 matrix of rank 2.*
2. *If $\mathbf{u} \leftrightarrow \mathbf{u}'$ are a pair of matching points, then $\mathbf{u}'^\top F \mathbf{u} = 0$*
3. *If F is the fundamental matrix for a pair of cameras (J, J') then $F^\top$ is the fundamental matrix for the pair (J', J).*
4. *If $\mathbf{p}$ and $\mathbf{p}'$ are the two epipoles, then $\mathbf{p}'^\top F = F\mathbf{p} = 0$.*
5. *$F\mathbf{u}$ is the epipolar line in the second image corresponding to point $\mathbf{u}$ in the first image.*
6. *F factors as a product $F = \mathbf{p}'^\top \times M$, where M is non-singular.*

Formulas for the Fundamental Matrix We can give explicit formulae for the fundamental matrix.

Proposition 2.2. *If $P = [M \mid -M\mathbf{c}]$ and $P' = [M' \mid -M'\mathbf{c}']$ then*

$$\begin{aligned} F &= [M'(\mathbf{c}' - \mathbf{c})]_\times (M'M^{-1}) \\ &= M'^{-\top}[\mathbf{c}' - \mathbf{c}]_\times M^{-1} \\ &= (M'M^{-1})^{-\top}[M(\mathbf{c}' - \mathbf{c})]_\times \end{aligned}$$

The first formula for the fundamental matrix was derived above. The alternative forms may be derived from the following identity, true for any 3×3 matrix M and vector $\mathbf{t}$.

$$M^*[\mathbf{t}]_\times = [M\mathbf{t}]_\times M \ .$$

2:2.2 Determination of Camera Matrices from the Fundamental Matrix

It was seen in the previous section that the fundamental matrix is uniquely determined by a pair of camera matrices. In the present section, we will consider the converse problem. To what extent are the two camera matrices determined by the fundamental matrix. In this section the fundamental matrix will be characterized by the defining condition that $\mathbf{u}'^\top F \mathbf{u} = 0$ for any pair of matching points in two images.

Let P and P' be a pair of camera matrices and let $\mathbf{u}' \leftrightarrow \mathbf{u}$ be a pair of matched points. This means that there is a point $\mathbf{x}$ in space such that $\mathbf{u} = P\mathbf{x}$ and $\mathbf{u}' = P'\mathbf{x}$. We seek a matrix F (the fundamental matrix) such that $\mathbf{u}'^\top F \mathbf{u} = 0$. Expressing this in terms of F leads to the equation

$$\mathbf{x}^\top P'^\top F P \mathbf{x} = 0$$

This equation must hold for any point $\mathbf{x}$ in space, since any such point gives rise to a pair of matched points $\mathbf{u} = P\mathbf{x}$ and $\mathbf{u}' = P'\mathbf{x}$. However, a matrix A satisfies the equation $\mathbf{x}^\top A \mathbf{x} = 0$ for all $\mathbf{x}$ if and only if A is skew-symmetric. Consequently, we have proven the following result

Proposition 2.3. *Given a pair of camera matrices P and P', then a matrix F satisfies $\mathbf{u}'^\top F\mathbf{u} = 0$ for all possible matched points $\mathbf{u} \leftrightarrow \mathbf{u}'$ in the images taken by the two cameras, if and only if $P'^\top FP$ is skew-symmetric.*

Under this condition we say that the pair (P, P') is a *realisation* of the fundamental matrix F.

We now show that given matrix F, the matrices P and P' are not uniquely determined. Specifically, if H is a non-singular 4×4 matrix, and $P'^\top FP$ is skew-symmetric, then so is $H^\top P'^\top FPH$. This shows that (P, P') and $(P'H, PH)$ are both realizations of the matrix F. It will be shown later that this is the only ambiguity in the realization of a fundamental matrix.

Projective transform of a reconstruction. This may be seen in another way as follows. If $\mathbf{u} = P\mathbf{x}$ and H is a non-singular 4×4 matrix representing any projective transformation then replacing the camera P by PH^{-1} and the point $\mathbf{x}$ by $H\mathbf{x}$, we see that $\mathbf{u} = (PH^{-1})H\mathbf{x}$. Thus, the image point $\mathbf{u}$ is unchanged by this transformation. This shows that the camera matrices can not be determined unambiguously by a set of image correspondences, since an arbitrary projective transformation H applied to the scene and the cameras in this way does not result in any change in the images. Thus, neither the scene, nor the camera placement may be determined mor accurately than up to an unknown projective transformation. We speak of a projective transformation H being applied to a camera matrix P to mean that P is replaced by PH^{-1}.

Normal form. Given a matrix F, we have just seen that there exist a large family of camera matrix pairs, (P, P') that make a realization of F. Next, we will be interested in certain normal form realizations. The first *normal form* will be one in which one of the cameras has matrix $[I \mid \mathbf{0}]$ and the other camera has camera matrix $[M \mid -M\mathbf{c}]$, meaning that the camera centre is not at infinity. In fact, this can be done with any number of cameras, as the following proposition shows.

Proposition 2.4. *Given camera matrices P_i for $i = 0, \ldots, N$, there exists a 4×4 matrix H such that $P_0H^{-1} = [I \mid \mathbf{0}]$ and for all $i = 1, \ldots, N$ the camera centre for the matrix P_iH^{-1} does not lie at infinity.*

Proof. Since P_0 has rank 3, it may be supplemented by one extra row to form a non-singular matrix, which we will denote by H. We see immediately that $P_0H^{-1} = [I \mid \mathbf{0}]$. Multiplying each of the other matrices by H^{-1} gives matrices $P_iH^{-1} = P'_i$. If the centre of each camera P'_i lies at a finite point, then we are done. Otherwise, we must apply a further transformation. Let $\mathbf{c}'_i$ be the centre of the camera P'_i. Thus, $P'_i\mathbf{c}'_i = 0$. Now, we select a plane that contains none of the points $\mathbf{c}'_i$, and which furthermore does not pass through the point $(0, 0, 0, 1)^\top$. Such a plane may be represented as $(\mathbf{v}^\top 1)^\top$, where $\mathbf{v}$ is a 3-vector. The condition that the plane does not pass through any of the points $\mathbf{c}'_i$ means that $(\mathbf{v}^\top 1)\mathbf{c}'_i \neq 0$ for any i. Letting $H' = \begin{bmatrix} I & 0 \\ \mathbf{v}^\top & 1 \end{bmatrix}$ this implies that $H'\mathbf{c}'_i$ is not a point at infinity. However, $H'\mathbf{c}'_i$ is the centre of the camera $P'_iH'^{-1}$. One verifies further that $[I \mid \mathbf{0}]H'^{-1} = [I \mid \mathbf{0}]$. Thus, transforming each P'_i by H'^{-1} gives the required set of camera matrices. □

This is a particularly convenient normal form for many applications. It may be easily verified that a matrix of the type

$$H = \begin{bmatrix} I & 0 \\ \mathbf{v}^\top & 1 \end{bmatrix}$$

is the general form for a 4×4 matrix satisfying the condition $[I \mid \mathbf{0}]H^{-1} = [I \mid \mathbf{0}]$. The choice of different vectors $\mathbf{v}$ correspond to different choices of the plane at infinity, since a point is mapped to infinity by the transformation H if and only if it lies on the plane represented by $(\mathbf{v}^\top 1)^\top$.

Now, we may show that the two matrices P and P' uniquely determine F as long as the camera centres for P and P' are not the same. Thus, suppose that ${P'}^\top F P$ is skew-symmetric. There is a matrix H such that $PH^{-1} = [I \mid \mathbf{0}]$ and $P'H^{-1} = [M \mid -M\mathbf{c}]$, where $\mathbf{c} \neq \mathbf{0}$ and M is non-singular. Now applying $H^{-\top}$ and H^{-1} to the left and right sides of ${P'}^\top F P$, we see that

$$\begin{aligned} H^{-\top}{P'}^\top F P H^{-1} &= \begin{pmatrix} M^\top \\ -\mathbf{c}^\top M^\top \end{pmatrix} F [I \mid \mathbf{0}] \\ &= \begin{bmatrix} M^\top F & 0 \\ -\mathbf{c}^\top M^\top F & 0 \end{bmatrix} \end{aligned}$$

is skew-symmetric. This implies that $M^\top F$ is skew-symmetric and $\mathbf{c}^\top (M^\top F) = 0$. Since a 3×3 skew-symmetric matrix is determined by its kernel (in this case $\mathbf{c}$), it follows that $M^\top F = [\mathbf{c}]_\times$. Finally, therefore, $F = M^{-\top}[\mathbf{c}]_\times = M^{-\top} \times \mathbf{c}$. Thus, the matrix F is uniquely determined by P and P'.

Specific formulas for F in terms of P and P' were given in Proposition 2.2 for the case where both cameras were located at finite points. A formula for the fundamental matrix that holds independent of that assumption is given next.

Proposition 2.5. *The fundamental matrix corresponding to a pair of camera matrices* $P = [I \mid \mathbf{0}]$ *and* $P' = [M \mid \mathbf{t}]$ *is* $F = \mathbf{t} \times M$.

Proof. One simply verifies that

$$ {P'}^\top F P = \begin{pmatrix} M^\top \\ \mathbf{t}^\top \end{pmatrix} [\mathbf{t}]_\times M [I \mid \mathbf{0}] = \begin{bmatrix} M^\top [\mathbf{t}]_\times M & 0 \\ 0 & 0 \end{bmatrix} $$

is skew symmetric. □

This result may be used to compute the fundamental matrix for any pair of matrices by transforming the two camera matrices to the form given in the proposition by applying an appropriate projective transformation H.

Factorization of the fundamental matrix : Conversely, given a fundamental matrix, it is possible to determine a pair of camera matrices that give rise to that fundamental matrix by applying Proposition 2.5.

Suppose that the singular value decomposition ([1]) of F is given by $F = UDV^\top$, where D is the diagonal matrix $D = \mathrm{diag}(r, s, 0)$. The following factorization of F may now be verified by inspection.

$$F = SM \; ; \quad S = UZU^\top \; ; \quad M = UE\mathrm{diag}(r, s, \alpha)V^\top$$

where

$$E = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ; \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and α is an arbitrary number. The matrix S is skew-symmetric, $S = [\mathbf{t}]_\times$.

By factoring F in this way, as $F = \mathbf{t} \times M$, we can then apply Proposition 2.5 to get a pair of cameras matrices $P = [I \mid \mathbf{0}]$ and $P' = [M \mid \mathbf{t}]$ that correspond to F.

Uniqueness of Camera Matrix up to Projective Transformation. We are almost ready to prove that the fundamental matrix determines the two camera matrices up to a projective transformation. First, however, we need one more lemma.

Lemma 2.6. *Let the rank 2 matrix F factor in two different ways as $F = \mathbf{t}\times M = \mathbf{t}'\times M'$. Then $\mathbf{t} = \mathbf{t}'$ and $M' = M + \mathbf{t}\mathbf{a}^\top$ for some vector $\mathbf{a}$.*

Proof. First, note that $\mathbf{t}F = \mathbf{t}[\mathbf{t}]_\times M = 0$, and similarly, $\mathbf{t}'F = 0$. Since F has rank 2, it follows that $\mathbf{t} = \mathbf{t}'$ as required. Next, from $[\mathbf{t}]_\times M = [\mathbf{t}]_\times M' = F$ it follows that $[\mathbf{t}]_\times(M' - M) = 0$, and so $M' - M = \mathbf{t}\mathbf{a}^\top$ for some $\mathbf{a}$. Hence, $M' = M + \mathbf{t}\mathbf{a}^\top$ as required. □

We now answer the question when two pairs of camera matrices may correspond to the same fundamental matrix.

Theorem 2.7. *Let (P_1, P_1') and (P_2, P_2') be two pairs of camera transforms. Then (P_1, P_1') and (P_2, P_2') correspond to the same fundamental matrix F if and only if there exists a 4×4 non-singular matrix H^{-1} such that $P_1H^{-1} = P_2$ and $P_1'H^{-1} = P_2'$.*

Proof. The *if* part of this theorem has already been proven, so we turn to the *only if* part. Since each of the matrices P_1 and P_2 has rank 3, we can multiply them (on the right) by suitable matrices H_1 and H_2 to transform them each to the matrix $[I \mid \mathbf{0}]$. If the matrices P_1' and P_2' are also multiplied by H_1 and H_2 respectively, then the fundamental matrix corresponding to the camera matrix pairs are unchanged, as seen previously. Thus, we have reduced the problem to the case where $P_1 = P_2 = [I \mid \mathbf{0}]$.

Suppose therefore, that $P_1 = P_2 = [I \mid \mathbf{0}]$ and that $P_1' = [M_1' \mid \mathbf{t}_1']$ and $P_2' = [M_2' \mid \mathbf{t}_2']$. By proposition 2.5 we have $F = [\mathbf{t}_1']_\times M_1' = [\mathbf{t}_2']_\times M_2'$. According to Lemma 2.6 this implies that $\mathbf{t}_1' = \mathbf{t}_2' = \mathbf{t}$ and that $M_2' = M_1' + \mathbf{t}\mathbf{a}^\top$ for some vector $\mathbf{a}$. Let H^{-1} be the matrix $\begin{bmatrix} I & 0 \\ \mathbf{a}^\top & 1 \end{bmatrix}$. Then one verifies that $[I \mid \mathbf{0}] = [I \mid \mathbf{0}]H^{-1}$, so $P_2 = P_1H^{-1}$. Furthermore, $P_1'H^{-1} = [M_1' \mid \mathbf{t}]H^{-1} = [M_1' + \mathbf{t}\mathbf{a}^\top \mid \mathbf{t}] = [M_2' \mid \mathbf{t}] = P_2'$. Thus H^{-1} is the matrix required for the conclusion of theorem 2.7. □

2:2.3 Point set reconstruction

Given a pair of camera matrices P and P' and a pair of matched points $\mathbf{u} \leftrightarrow \mathbf{u}'$ it is evident that the space point $\mathbf{x}$ that gives rise to the two matching image points is uniquely defined, and may be obtained by intersecting two rays from the camera centres. Here is a simple way of computing the point $\mathbf{x}$.

Suppose that the fundamental matrix factors as as $F = \mathbf{t}' \times M'$, and let $P = [I \mid \mathbf{0}]$ and $P' = [M' \mid \mathbf{t}']$ be a realization of the matrix F. Let $\mathbf{u} \leftrightarrow \mathbf{u}'$ be a pair of matched points in the two images. We wish to find a point $\mathbf{x}$ in space such that $\mathbf{u} = P\mathbf{x}$ and $\mathbf{u}' = P'\mathbf{x}$. From the relation $\mathbf{u}'^\top F\mathbf{u} = \mathbf{u}'^\top [\mathbf{t}']_\times M'\mathbf{u} = \mathbf{u}'^\top (\mathbf{t}' \times M'\mathbf{u}) = 0$, it follows that $\mathbf{u}'$, $M'\mathbf{u}$ and $\mathbf{t}'$ are linearly dependent. If in particular $M'\mathbf{u} = \beta\mathbf{u}' - \alpha\mathbf{t}'$ then we define the corresponding object space point $\mathbf{x}$ to be the point $\begin{pmatrix} \mathbf{u} \\ \alpha \end{pmatrix}$. It is now easily verified that $P\mathbf{x} = [I \mid \mathbf{0}]\mathbf{x} = \mathbf{u}$ and $P'\mathbf{x} = [M' \mid \mathbf{t}']\mathbf{x} = M'\mathbf{u} + \alpha\mathbf{t}' = \mathbf{u}'$. This verifies that the given values of P, P' and $\mathbf{x}_i$ constitute a projective reconstruction of the data.

As shown, $\mathbf{x}$ is determined by the two camera matrices P and P' and the matched points $\mathbf{u} \leftrightarrow \mathbf{u}'$. If we choose a different pair of camera matrices PH and $P'H$ realizing the same fundamental matrix F, then in order to preserve the same pair of matched image points, the point $\mathbf{x}$ must be replaced by $H^{-1}\mathbf{x}$. Thus, changing to a different realization of F results in a projective transformation (namely H^{-1}) of the scene. This proves the following theorem

Theorem 2.8. (Faugeras [12], Hartley et al. [22]) *Given a set of image correspondences $\{\mathbf{u}_i\} \leftrightarrow \{\mathbf{u}'_i\}$ sufficient to determine the fundamental matrix, the corresponding object space coordinates $\{\mathbf{x}_i\}$ may be computed up to a collineation of projective 3-space $\mathcal{P}^3$.*

The foregoing discussion has effectively given an algorithm for doing projective reconstruction from a pair of uncalibrated cameras, given a set of matched points $\mathbf{u}'_i \leftrightarrow \mathbf{u}_i$ in the two images.

1. Compute the fundamental matrix using the formula $\mathbf{u}'^\top_i F\mathbf{u}_i = 0$ for all i.
2. Factor the fundamental matrix as $F = [\mathbf{t}]_\times M$.
3. Set the two camera matrices equal to $P = [I \mid 0]$ and $P' = [M \mid \mathbf{t}]$.
4. Reconstruct the points by triangulation.

The two main steps of this matrix are the determination of the fundamental matrix and the triangulation step. These two steps will be examined in detail in the following two sections of this report.

Further Reading

For methods of computing the Fundamental Matrix, see the work of Faugeras, Luong et al, as well as the original work of Longuet-Higgins [42, 13, 43]. For iterative methods of projective reconstruction, see [12, 47, 29]. A more geometric approach to reconstruction is taken by Ponce et.al ([53]) and Shashua ([63, 64]). Reconstruction from lines instead of points is considered in [31] and a later section of this report.

2:3 The 8-point Algorithm

The 8-point algorithm for computing the essential matrix was introduced by Longuet-Higgins in a now classic paper ([42]). In that paper the essential matrix is used to

compute the structure of a scene from two views with calibrated cameras. The great advantage of the 8-point algorithm is that it is linear, hence fast and easily implemented. If 8 point matches are known, then the solution of a set of linear equations is involved. With more than 8 points, a linear least squares minimization problem must be solved. The term 8-point algorithm will be used in this report to describe this method whether only 8 points, or more than 8 points are used.

The essential property of the essential matrix is that it conveniently encapsulates the epipolar geometry of the imaging configuration. As shown in section 2:2, the same method may be used to compute a matrix with this property from uncalibrated cameras. In this case of uncalibrated cameras it has become customary to refer to the matrix so derived as the *fundamental matrix*. Just as in the calibrated case, the fundamental matrix may be used to reconstruct the scene from two uncalibrated views, but in this case only up to a projective transformation. Apart from scene reconstruction, the fundamental matrix may also be used for many other tasks, such as image rectification ([28]), computation of projective invariants ([8]), outlier detection ([9]) and stereo matching ([79]);

Unfortunately, despite its simplicity the 8-point algorithm has often been criticized for being excessively sensitive to noise in the specification of the matched points. Indeed this belief has become the prevailing wisdom. Consequently, because of its importance, many alternative algorithms have been proposed for the computation of the fundamental matrix. See [44] for a description and comparison of several algorithms for finding the fundamental matrix. Without exception, these algorithms are considerably more complicated than the 8-point algorithm. Other iterative algorithms have been described (briefly) in [29, 4].

It is the purpose of this section to challenge the common view that the 8-point algorithm is inadequate and markedly inferior to the more complicated algorithms. The poor performance of the 8-point algorithm can probably be traced to implementations that does not take sufficient account of numerical considerations, most specifically the condition of the set of linear equations being solved. It is shown in this section that a simple transformation (translation and scaling) of the points in the image before formulating the linear equations leads to an enormous improvement in the condition of the problem and hence of the stability of the result. The added complexity of the algorithm necessary to do this transformation is insignificant.

It is not claimed here that this modified 8-point algorithm will perform quite as well as the best iterative algorithms. However it is shown by thousands of experiments on many images that the difference is not very great between the modified 8-point algorithm and iterative techniques. Indeed the 8-point algorithm does better than some of the iterative techniques.

2:3.1 Outline of the 8-point Algorithm

Linear solution for the fundamental matrix. The fundamental matrix is defined by the equation

$$\mathbf{u}'^{\top} F \mathbf{u} = \mathbf{0} \tag{2.14}$$

for any pair of matching points $\mathbf{u}' \leftrightarrow \mathbf{u}$ in two images. Given sufficiently many point matches $\mathbf{u}'_i \leftrightarrow \mathbf{u}_i$, (at least 8) this equation (2.14) can be used to compute the unknown matrix F. In particular, writing $\mathbf{u} = (u, v, 1)^{\top}$ and $\mathbf{u}' = (u', v', 1)^{\top}$ each point match gives rise to one linear equation in the unknown entries of F. The coefficients of this

equation are easily written in terms of the known coordinates $\mathbf{u}$ and $\mathbf{u}'$. Specifically, the equation corresponding to a pair of points $(u, v, 1)$ and $(u', v', 1)$ will be

$$uu'f_{11} + uv'f_{21} + uf_{31} + vu'f_{12} + vv'f_{22} + vf_{32} + u'f_{13} + v'f_{23} + f_{33} = 0 \ . \quad (2.15)$$

The row of the equation matrix may be represented as a vector

$$(uu', uv', u, vu', vv', v, u', v', 1) \ . \quad (2.16)$$

From all the point matches, we obtain a set of linear equations of the form

$$A\mathbf{f} = 0 \quad (2.17)$$

where $\mathbf{f}$ is a 9-vector containing the entries of the matrix F, and A is the equation matrix. The fundamental matrix F, and hence the solution vector $\mathbf{f}$ is defined only up to an unknown scale. For this reason, and to avoid the trivial solution $\mathbf{f}$, we make the additional constraint

$$||\mathbf{f}|| = 1 \quad (2.18)$$

where $||\mathbf{f}||$, is the norm of $\mathbf{f}$[1].

Under these conditions, it is possible to find a solution to the system (2.17) with as few as 8 point matches. With more than 8 point matches, we have an overspecified system of equations. Assuming the existence of a non-zero solution to this system of equations, we deduce that the matrix A must be rank-deficient. In other words, although A has 9 columns, the rank of A must be at most 8. In fact, except for exceptional configurations ([45]) the matrix A will have rank exactly 8, and there will be a unique solution for $\mathbf{f}$.

This previous discussion assumes that the data is perfect, and without noise. In fact, because of inaccuracies in the measurement or specification of the matched points, the matrix A will not be rank-deficient – it will have rank 9. In this case, we will not be able to find a non-zero solution to the equations $A\mathbf{f} = \mathbf{0}$. Instead, we seek a least-squares solution to this equation set. In particular, we seek the vector $\mathbf{f}$ that minimizes $||A\mathbf{f}||$ subject to the constraint $||\mathbf{f}|| = \mathbf{f}^\top\mathbf{f} = 1$. It is well known (and easily derived using Lagrange multipliers) that the solution to this problem is the unit eigenvector of $A^\top A$ corresponding to the smallest eigenvalue of A. Note that since $A^\top A$ is positive semi-definite and symmetric, all its eigenvectors are real and positive, or zero. For convenience, (though somewhat inexactly), we will call this eigenvector the *least eigenvector* of $A^\top A$. An appropriate algorithm for finding this eigenvector is the algorithm of Jacobi ([55]) or the Singular Value Decomposition ([55, 1]).

The singularity constraint. An important property of the fundamental matrix is that it is singular, in fact of rank 2. Furthermore, the left and right null-spaces of F are generated by the vectors representing (in homogeneous coordinates) the two epipoles in the two images. Most applications of the fundamental matrix rely on the fact that it has rank 2. The matrix F found by solving the set of linear equations (2.17) will not in general have rank 2, and we should take steps to enforce this constraint. The most convenient way to enforce this constraint is to correct the matrix F found by the solution of (2.17). Matrix F is replaced by the matrix F' that minimizes the Frobenius

[1] An alternative is to set $F_{33} = 1$ and solving a linear least squares minimization problem. The general conclusions of this section are equally valid for this version of the algorithm.

norm $||F - F'||$ subject to the condition $\det F' = 0$. A convenient method of doing this is to use the Singular Value Decomposition (SVD). In particular, let $F = UDV^\top$ be the SVD of F, where D is a diagonal matrix $D = \mathrm{diag}(r, s, t)$ satisfying $r \geq s \geq t$. We let $F' = U\mathrm{diag}(r, s, 0)V^\top$. This method was suggested by Tsai and Huang ([75]) and has been proven to minimize the Frobenius norm of $F - F'$, as required.

Thus, the 8-point algorithm for computation of the fundamental matrix may be formulated as consisting of two steps, as follows.

Linear solution. Given point matches $\mathbf{u}'_i \leftrightarrow \mathbf{u}_i$, solve the equations ${\mathbf{u}'_i}^\top F \mathbf{u}_i = 0$ to find F. The solution is the least eigenvector, $\mathbf{f}$ of $A^\top A$, where A is the equation matrix.

Constraint Enforcement. Replace F by F', the closest singular matrix to F under Frobenius norm. This is done using the Singular Value Decomposition.

The algorithm thus stated is extremely simple, and rapid to implement, assuming the availability of a suitable linear algebra library (for instance [55]).

2:3.2 Transformation of the Input

Image coordinates are sometimes given with the origin at the top-left of the image, and sometimes with the origin at the centre. The question immediately occurs whether this makes a difference to the results of the 8-point algorithm for computing the fundamental matrix. More generally, to what extent is the result of the 8-point algorithm dependent on the choice of coordinates in the image. Suppose, for instance the image coordinates were changed by some affine or even projective transformation before running the algorithm. Will this materially change the result? That is the question that we will now consider.

Suppose that coordinates $\mathbf{u}$ in one image are replaced by $\hat{\mathbf{u}} = T\mathbf{u}$, and coordinates $\mathbf{u}'$ in the other image are replaced by $\hat{\mathbf{u}}' = T'\mathbf{u}'$. Substituting in the equation ${\mathbf{u}'}^\top F \mathbf{u} = 0$, we derive the equation ${\hat{\mathbf{u}}'}^\top {T'}^{-\top} F T^{-1} \hat{\mathbf{u}} = 0$, where ${T'}^{-\top}$ is the inverse transpose of T'. This relation implies that ${T'}^{-\top} F T^{-1}$ is the fundamental matrix corresponding to the point correspondences $\hat{\mathbf{u}}' \leftrightarrow \hat{\mathbf{u}}$. An alternative method of finding the fundamental matrix is therefore suggested, as follows.

1. Transform the image coordinates according to transformations $\hat{\mathbf{u}}_i = T\mathbf{u}_i$ and $\hat{\mathbf{u}}'_i = T'\mathbf{u}'_i$.
2. Find the fundamental matrix $\hat{F}$ corresponding to the matches $\hat{\mathbf{u}}'_i \leftrightarrow \hat{\mathbf{u}}_i$.
3. Set $F = {T'}^\top \hat{F} T$.

The fundamental matrix found in this way corresponds to the original untransformed point correspondences $\mathbf{u}'_i \leftrightarrow \mathbf{u}_i$. What choice should be made for the transformations T and T' will be left unspecified for now. First, we need to determine whether carrying out this transformation has any effect whatever on the result.

As verified above, ${\mathbf{u}'}^\top F \mathbf{u} = {\hat{\mathbf{u}}'}^\top \hat{F} \hat{\mathbf{u}}$, where $\hat{F}$ is defined by $\hat{F} = {T'}^{-\top} F T^{-1}$. Thus, if ${\mathbf{u}'}^\top F \mathbf{u} = \epsilon$, then also ${\hat{\mathbf{u}}'}^\top \hat{F} \hat{\mathbf{u}} = \epsilon$. Thus, there is a one-to-one correspondence between F and $\hat{F}$ giving rise to the same error. It may appear therefore that the matrices F and $\hat{F}$ minimizing the error ϵ (or more exactly, the sum of squares of errors corresponding to all points) will be related by the formula $\hat{F} = {T'}^{-\top} F T^{-1}$, and hence one may retrieve

F as the product ${T'}^\top \hat{F} T$. This conclusion is **false** however. For, although F and $\hat{F}$ so defined give rise to the same error ϵ, the condition $||F|| = 1$, imposed as a constraint on the solution, is not equivalent to the condition $||\hat{F}|| = 1$. In particular, there is no one-to-one correspondence between F and $\hat{F}$ giving rise to the same error ϵ, subject to the constraint $||F|| = ||\hat{F}|| = 1$.

This is a crucial point, and so we will look at it from a different point of view. A set of point correspondences $\mathbf{u}'_i \leftrightarrow \mathbf{u}_i$ give rise to a set of equations of the form $A\mathbf{f} = 0$. If now we make the transformation $\hat{\mathbf{u}}_i = T\mathbf{u}_i$ and $\hat{\mathbf{u}}'_i = T'\mathbf{u}'_i$, then the set of equations will be replaced by a different set of equations of the form $\hat{A}\hat{\mathbf{f}} = 0$. One may verify, in particular that the matrix $\hat{A}$ may be written in the form $\hat{A} = AS$ where S is a 9×9 matrix that may be written explicitly in terms of the entries of T and T' (but it is not very important exactly how). Therefore one is led to consider the two sets of equations $A\mathbf{f} = 0$ and $AS\hat{\mathbf{f}} = 0$. One may guess that the least-squares solutions to these two sets of equations will be related according to $\hat{\mathbf{f}} = S^{-1}\mathbf{f}$. If this were so, then replacing $\hat{\mathbf{f}}$ by $S\hat{\mathbf{f}}$ one once more retrieves the original solution $\mathbf{f}$. The mapping $\hat{\mathbf{f}} \mapsto S\hat{\mathbf{f}}$ corresponds precisely to the matrix mapping $\hat{F} \mapsto {T'}^\top \hat{F} T$.

However, things are not that simple. Perhaps the least-squares solutions to the two sets of equations $A\mathbf{f} = 0$ and $AS\hat{\mathbf{f}} = 0$ are not so simply related. The solution $\mathbf{f}$ to the system $A\mathbf{f} = 0$ is the least eigenvector of the matrix $A^\top A$. Is it so that $\hat{\mathbf{f}} = S^{-1}\mathbf{f}$ is the least eigenvector of $(AS)^\top (AS)$? Letting λ be the least eigenvalue of $A^\top A$, we verify :

$$\begin{aligned} S^\top A^\top AS\hat{\mathbf{f}} &= S^\top A^\top AS S^{-1}\mathbf{f} \\ &= S^\top A^\top A\mathbf{f} \\ &= S^\top \lambda \mathbf{f} \\ &= \lambda S^\top S\hat{\mathbf{f}} \\ &\neq \lambda \hat{\mathbf{f}} \ . \end{aligned}$$

Thus, in fact, $S^{-1}\mathbf{f}$ is not the least eigenvector of $(AS)^\top AS$. In fact it is not an eigenvector at all.

Let us see how significant this effect is. We take the example that T and T' are simply scalings of the coordinates, in fact, multiplication of the coordinates by a factor of 10. These transformations are represented by diagonal matrices of the form $T = T' = \mathrm{diag}(10, 10, 1)$ acting on homogeneous coordinates. In this case, the matrix S is also a diagonal matrix of the form $S = \mathrm{diag}(10^2, 10^2, 10, 10^2, 10^2, 10, 10, 10, 1)$, assuming that the vector $\mathbf{f}$ represents the elements of F in the row-major order $f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}$. The matrix $S^\top S$ equals $\mathrm{diag}(10^4, 10^4, 10^2, 10^4, 10^4, 10^2, 10^2, 10^2, 1)$. In this case, we see $(AS)^\top AS\hat{\mathbf{f}} = \lambda S^\top S\hat{\mathbf{f}}$, and so $\hat{\mathbf{f}}$ is very far from being an eigenvector of $(AS)^\top AS$.

We conclude that the method of transformation leads to a different solution for the fundamental matrix. This is a rather undesirable feature of the 8-point algorithm as it stands, that the result is changed by a change of coordinates, or even simply a change of the origin of coordinates. To correct this, it seems advisable to normalize the coordinates of the points in some way by expressing them in some fixed canonical frame, as yet unspecified.

2:3.3 Condition of the System of Equations

The linear method consists in finding the least eigenvector of the matrix $A^\top A$. This may be done by expressing $A^\top A$ as a product $UDU^\top$ where U is orthogonal and D is diagonal. We assume that the diagonal entries of D are in non-increasing order. In this case, the least eigenvector of $A^\top A$ is the last column of U. Denote by κ the ratio d_1/d_8 (recalling that $A^\top A$ is a 9×9 matrix). The parameter κ is the condition number[2] of the matrix $A^\top A$, well known to be an important factor in the analysis of stability of linear problems ([15]). Its relevance to the problem of finding the least eigenvector is briefly explained next.

The bottom right hand 2×2 block of matrix D is of the form $\begin{bmatrix} d_8 & 0 \\ 0 & 0 \end{bmatrix}$, assuming that $d_9 = 0$, which ideally will be the case. Now, suppose that this block is perturbed by the addition of noise to become $\begin{bmatrix} d_8 & \epsilon \\ \epsilon & 0 \end{bmatrix}$. In order to restore this matrix to diagonal form we need to multiply left and right by $V^\top$ and V, where V is a rotation through an angle $\theta = \arctan(2\epsilon/d_8)$ (as the reader may verify). If ϵ is of the same order of magnitude as d_8 then this is a significant rotation. Looking at the full matrix, $A^\top A = UDU^\top$, we see that the perturbed matrix will be written in the form $U\bar{V}D'\bar{V}^\top U^\top$ where $\bar{V} = \begin{bmatrix} I_{7\times 7} & 0 \\ 0 & V \end{bmatrix}$. Multiplying by $\bar{V}$ replaces the last column of U by a combination of the last two columns. Since the last column of U is the least eigenvector of the matrix, this perturbation will drastically alter the least eigenvector of the matrix $A^\top A$. Thus, changes to $A^\top A$ of the order of magnitude of the eigenvalue d_8 cause significant changes to the least eigenvector. Since multiplication by an orthogonal matrix does not change the Frobenius norm of a matrix, we see that $||A^\top A|| = \left(\sum_{i=1}^{9} d_i^2\right)^{1/2}$. If the ratio $\kappa = d_1/d_8$ is very large, then d_8 represents a very small part of the Frobenius norm of the matrix. A perturbation of the order of d_8 will therefore cause a very small relative change to the matrix $A^\top A$, while at the same time causing a very significant change to the smallest eigenvalue. Since $A^\top A$ is written directly in terms of the coordinates of the points $\mathbf{u} \leftrightarrow \mathbf{u}'$, we see that if κ is large, then very small changes to the data can cause large changes to the solution. This is obviously very undesirable. The sensitivity of invariant subspaces is discussed in greater detail in [15], p413, where more specific conditions for the sensitivity of invariant subspaces are given.

We now consider how the condition number of the matrix $A^\top A$ may be made small. We consider two sorts of transformation, translation and scaling. These methods will be given only an intuitive justification, since a complete analysis of the condition number of the matrix is too complex to undertake here.

The major reason for the poor condition of the matrix $A^\top A$ is the lack of homogeneity in the image coordinates. In an image of dimension 200×200, a typical image point will be of the form $(100, 100, 1)$. If both $\mathbf{u}$ and $\mathbf{u}'$ are of this form, then the corresponding row of the equation matrix will be of the form $\mathbf{r}^\top = (10^4, 10^4, 10^2, 10^4, 10^4, 10^2, 10^2, 10^2, 1)$. The contribution to the matrix $A^\top A$ is of the form $\mathbf{r}\mathbf{r}^\top$, which will contain entries ranging between 10^8 and 1. For instance, the diagonal entries of $A^\top A$ will be $(10^8, 10^8, 10^4, 10^8, 10^8, 10^4, 10^4, 10^4, 1)$. Summing over all point correspondences will result in a matrix $A^\top A$ for which the diagonal entries are approximately in this proportion.

[2]Strictly speaking, d_1/d_9 is the condition number, but d_1/d_8 is the parameter of importance here

We may now use the *Interlacing Property* ([15], page 411) for the eigenvalues of a symmetric matrix to get a bound on the condition number of the matrix. Suppose that the diagonal entries of $X = A^\top A$ are equal to $(10^8, 10^8, 10^4, 10^8, 10^8, 10^4, 10^4, 10^4, 1)$. We denote by X_r the trailing $r \times r$ principal submatrix (that is the last r columns and rows) of the matrix $A^\top A$, and by $\lambda_i(X_r)$ its i-th largest eigenvalue. Thus, $X_9 = A^\top A$ and $\kappa = \lambda_1(X_9)/\lambda_8(X_9)$. First we consider the eigenvalues of X_2. Since the sum of the two eigenvalues is $\text{trace}(X_2) = 10^4 + 1$, we see that $\lambda_1(X_2) + \lambda_2(X_2) = 10^4 + 1$. Since the matrix is positive semi-definite, both eigenvalues are non-negative, so we may deduce that $\lambda_1(X_2) \leq 10^4 + 1$. From the interlacing property, we deduce that $\lambda_8(X_9) \leq \lambda_7(X_8) \leq \ldots \lambda_1(X_2) \leq 10^4 + 1$. On the other hand, also from the interlacing property, we know that the largest eigenvalue of $A^\top A$ is not less than the largest diagonal entry. Thus, $\lambda_1(X_9) \geq 10^8$. Therefore, the ratio $\kappa = \lambda_1(X_9)/\lambda_8(X_9) \geq 10^8/(10^4 + 1)$. Usually, in fact $\lambda_8(X_9)$ will be much smaller than $10^4 + 1$ and the condition number will be far greater.

This analysis shows that scaling the coordinate so that the homogeneous coordinates are on the average equal to unity will improve the condition of the matrix $A^\top A$.

Translation Consider a case where the origin of the image coordinates is at the top left hand corner of the image, so that all the image coordinates are positive. In this case, an improvement in the condition of the matrix may be achieved by translating the points so that the centroid of the points is at the origin. This claim will be verified by experimentation, but can also be explained informally by arguing as follows. Suppose that the first image coordinates (the u-coordinates) of a set of points are $\{1001.5, 1002.3, 998.7, \ldots\}$. By translating by 1000, these numbers may be changed to $\{1.5, 2.3, -1.3\}$. Thus, in the untranslated values, the significant values of the coordinates are obscured by the coordinate offset of 1000. The significant part of the coordinate values is found only in the third or fourth significant figure of the coordinates. This has a bad effect on the condition of the corresponding matrix $A^\top A$. A more detailed analysis of the effect of translation is not provided here.

2:3.4 Normalizing transformations

The previous sections concerned with the condition number of the matrix $A^\top A$ indicate that it is desirable to apply a transformation to the coordinates before carrying out the 8-point algorithm for finding the fundamental matrix. This normalization has been implemented as a prior step in the 8-point algorithm with excellent results.

Isotropic Scaling

As a first step, the coordinates in each image are translated (by a different translation for each image) so as to bring the centroid of the set of all points to the origin. The coordinates are also scaled. In the discussion of scaling, it was suggested that the best results will be obtained if the coordinates are scaled, so that on the average a point $\mathbf{u}$ is of the form $\mathbf{u} = (u, v, w)^\top$, with each of u, v and w having the same average magnitude. Rather than choose different scale factors for each point, an isotropic scaling factor is chosen so that the u and v coordinates of a point are scaled equally. To this end, we choose to scale the coordinates so that the average distance of a point $\mathbf{u}$ from the origin

is equal to $\sqrt{2}$. This means that the "average" point is equal to $(1,1,1)^\top$. In summary the transformation is as follows

1. The points are translated so that their centroid is at the origin.
2. The points are then scaled so that the average distance from the origin is equal to $\sqrt{2}$.
3. This transformation is applied to each of the two images independently.

Non-isotropic Scaling

In non-isotropic scaling, the centroid of the points is translated to the origin as before. After this translation the points form a cloud about the origin. Scaling is then carried out so that the two principal moments of the set of points are both equal to unity. Thus, the set of points will form an approximately symmetric circular cloud of points of radius one about the origin.

Both translation and scaling can be done in one step as follows. Let $\mathbf{u}_i = (u_i, v_i, 1)^\top$ for $i = 1, \ldots, N$ and form the matrix $\sum_i \mathbf{u}_i \mathbf{u}_i{}^\top$. Since this matrix is symmetric and positive definite, we may take its Choleski factorization ([1, 55]) to get $\sum_{i=1}^{N} \mathbf{u}_i \mathbf{u}_i{}^\top = NKK^\top$, where K is upper triangular. It follows that $\sum_i K^{-1}\mathbf{u}_i \mathbf{u}_i{}^\top K^{-\top} = NI$, where I is the identity matrix. Setting $\hat{\mathbf{u}}_i = K^{-1}\mathbf{u}_i$, we have $\sum_i \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i{}^\top = NI$. Consequently, the set of points $\hat{\mathbf{u}}_i$ have their centroid at the origin and the two principal moments are both equal to unity, as desired. Note that K^{-1} is upper triangular, and so it represents an affine transformation.

To summarize : the points are transformed so that

1. Their centroid is at the origin.
2. The principal moments are both equal to unity.

2:3.5 Scaling in Stage 2

So far we have discussed the effect of a normalizing transformation on the first stage of the 8-point algorithm, namely the solution of the set of linear equations to find F. The second step of the algorithm is to enforce the singularity constraint that $\det F = 0$.

The method described above of enforcing the singularity constraint gives the singular matrix $\hat{F}$ nearest to F in Frobenius norm. The trouble with this method is that it treats all entries of the matrix equally, regardless of their magnitude. Thus, entries of F small in absolute value may be expected to undergo a perturbation much greater relative to their magnitude than the large entries.

Suppose that a set of matched points is normalized so that on the average all three homogeneous coordinates have the same magnitude. Thus, a typical point will look like $(1,1,1)^\top$. The fundamental matrix computed from these normalized coordinates may be expected to have all its entries approximately of the same magnitude. This is an intuitive argument only, but it is borne out by experience, as will be seen below. It may be further justified by the following remark however.

A permutation of the three homogeneous coordinates in either or both the images will result in another set of realizable matched points. The corresponding fundamental matrix will be obtained from the original one by permuting the corresponding rows and/or columns of the matrix. In doing this, any entry of F may be moved to any other position. This means that no entry of the fundamental matrix is qualitatively different from any other, and hence on the average (over all possible sets of matched points) all entries of F will have the same average magnitude.

Now, consider what happens if we scale the coordinates of points $\mathbf{u}_i$ and $\mathbf{u}'_i$ by a factor which we will assume is equal to 100. Thus, a typical coordinate will be of the order of $(100, 100, 1)^\top$. The corresponding fundamental matrix F will be obtained from original one by multiplying the first two rows, and the first two columns by 10^-2. Entries in the the top left 2×2 block will be multiplied by 10^{-4}. We conclude that a typical fundamental matrix derived from coordinates of magnitude $(100, 100, 1)^\top$ will have entries of the following order of magnitude.

$$F = \begin{bmatrix} 10^{-4} & 10^{-4} & 10^{-2} \\ 10^{-4} & 10^{-4} & 10^{-2} \\ 10^{-2} & 10^{-2} & 1 \end{bmatrix} \tag{2.19}$$

To verify this conclusion, here is the fundamental matrix for the pair of house images in Fig 2.1[3].

$$F = \begin{bmatrix} -9.796e-08 & 1.473e-06 & -6.660e-04 \\ -6.346e-07 & 1.049e-08 & 7.536e-03 \\ 8.107e-04 & -7.739e-03 & -2.364e-02 \end{bmatrix} \tag{2.20}$$

In comparing (2.20) with (2.19), one must bear in mind that F is defined only up to nonzero scaling. The imbalance of the matrix (2.20) is even worse than predicted by (2.19) because the image has dimension 512×512. Now, in taking the closest singular matrix, all entries will tend to be perturbed by approximately the same amount. However, the relative perturbation will be greatest for the smallest entries. The question arises whether the small entries in the matrix F are important. Consider a typical point $\mathbf{u} \approx (100, 100, 1)^\top$. In computing the corresponding epipolar line $F\mathbf{u}$, we see that the largest entries in the vector $\mathbf{u}$ are multiplied by the smallest, and hence least relatively stable entries of the matrix F. Thus, for computation of the epipolar line, the smallest entries in F are the most important. We have the following undesirable condition :

> The most important entries in the fundamental matrix are precisely those that are subject to the largest relative perturbation when enforcing the singularity constraint without prior normalization.

This condition is corrected if normalization of the image coordinates is carried out first, for then all entries of the fundamental matrix will be treated approximately equally, and none is more important than another in computing epipolar lines.

[3]The notation -9.766e-08 means -9.766×10^{-8}.

2:3.6 Experimental Evaluation

The 8-point algorithm with prior transformation of the coordinates, as described here will be called the *normalized 8-point algorithm*. This algorithm was tested on a large number of real images to evaluate its performance. In carrying out these tests, the 8-point algorithm with pre-normalization as described above was compared with several other algorithms for finding the fundamental matrix. For the most part the implementations of these other algorithms were provided by other researchers, whom I will acknowledge later. In this way the results were not biased in any way by our possibly inefficient implementation of competing algorithms. In addition, the images and matched points that I have tested the algorithms on have been supplied to me. Methods of obtaining the matched points therefore varied from image to image, as did methods for eliminating bad matches (outliers). In all cases, however, the matched points were found by automatic means, and usually some sort of outlier detection and removal was carried out, based on least-median squares techniques (see [9]).

The general procedure for evaluation was as follows.

1. Matching points were computed by automatic techniques, and outliers were detected and removed.

2. The fundamental matrix was computed using a subset of all points.

3. In the case of algorithms, such as the 8-point algorithm, that do not automatically enforce the singularity constraint (that is the constraint that $\det F = 0$) this constraint constraint was enforced a posteriori by finding the nearest singular matrix to the computed fundamental matrix. This was done using the Singular Value Decomposition (as in [75, 23]).

4. For each point $\mathbf{u}_i$, the corresponding epipolar line $F\mathbf{u}_i$ was computed and distance from the line $F\mathbf{u}_i$ from the matching point $\mathbf{u}'_i$ was calculated. This was done in both directions, (that is starting from points $\mathbf{u}_i$ in the first image and also from $\mathbf{u}'_i$ in the second image). The average distance of the epipolar line from the corresponding point was computed, and used as a measure of quality of the computed Fundamental matrix. This evaluation was carried out using **all** matched points, and not just the ones that were used to compute F.

Other algorithms.

Here is a brief description of the algorithms tested.

The 8-point algorithm In this algorithm, the points were used as is, without pre-transformation to compute the fundamental matrix. The singularity constraint was enforced.

The 8-point algorithm with isotropic scaling The 8-point algorithm was used with the translation and isotropic scaling method described in section 2:3.4. The singularity constraint was enforced.

The 8-point algorithm with non-isotropic scaling This is the same as the previous method, except that the non-isotropic scaling method described in section 2:3.4 was used.

Minimizing the epipolar distances In implementation by Long Quan of an algorithm described by Luong ([44, 9]) was used. This is an iterative algorithm that uses a parametrization of the fundamental matrix with 7 parameters. Thus the singularity constraint is enforced as part of the algorithm. The cost function being minimized is the squared sum of distances of the points from epipolar lines. The point-line distances in both images are taken into account.

Minimizing point displacement This algorithm (our own implementation) is an iterative algorithm. It finds the fundamental matrix F, and points $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{u}}'_i$ such that $\hat{\mathbf{u}}_i'^{\top} F \hat{\mathbf{u}}_i = 0$ exactly, $\det F = 0$ and the squared pixel error $\sum_i d(\hat{\mathbf{u}}_i, \mathbf{u}_i)^2 + d(\hat{\mathbf{u}}'_i, \mathbf{u}'_i)^2$ is minimized. The details of how this is done are described in [29, 30]. Under the assumption of gaussian noise in the placement of the matched points (an approximation to the truth), this algorithm gives the fundamental matrix corresponding to the most likely true placement of the matched points (the estimated points $\hat{\mathbf{u}}_i \leftrightarrow \hat{\mathbf{u}}'_i$). For this reason, I have generally considered this algorithm to be the best available. The experiments generally bear out this belief, but it is not the purpose of this report to justify this point.

Approximate Calibration The results of an algorithm of Beardsley and Zisserman ([4]) were provided for comparison. This algorithm does an approximate normalization of the coordinates by selecting the origin of coordinates at the centre of the image, and by scaling by division by the approximate focal length of the camera (measured in pixels – that is, the scaling factor in the calibration matrix). Since this method employs a normalization similar to the isotropic scaling algorithm, one expects it to give similar results. It does, however rely on some approximate knowledge of camera calibration.

Iterative Linear Another algorithm provided by Beardsley and Zisserman is representative of a general approach to improving the performance of linear algorithms. This same approach can be applied to may different linear algorithms, such as camera pose and calibration estimation ([71]), projective reconstruction from lines ([31]) and reconstruction of point positions in space ([33]). In this approach, the 8-point algorithm is run a first time. From this initial solution a set of weights for the linear equations are computed. The set of linear equations are multiplied by these weights and the 8-point algorithm is run again. This may be repeated several times. The weights are chosen in such a way that the linear equations express a meaningful measurable quantity. To be specific, in the case of the 8-point algorithm each point correspondence $\mathbf{u}_i'^{\top} F \mathbf{u}_i = 0$ gives one linear equation in the entries of F. However, the quantity $\mathbf{u}_i'^{\top} F \mathbf{u}_i$ does not correspond to any meaningful geometric quantity, certainly not to distance between the point $\mathbf{u}'_i$ and the epipolar line $F\mathbf{u}_i$. Writing $F\mathbf{u}_i = (\lambda, \mu, \nu)^{\top}$, the distance $d(\mathbf{u}'_i, F\mathbf{u}_i)$ is equal to $\mathbf{u}_i'^{\top} F \mathbf{u}_i / \sqrt{\lambda^2 + \nu^2}$. Thus, weighting the equation $\mathbf{u}_i'^{\top} F \mathbf{u}_i = 0$ by the weight $\sqrt{\lambda^2 + \nu^2}$, where λ and μ are computed from the current estimate of F, one measures the distance from line $F\mathbf{u}_i$ to the point $\mathbf{u}_i'^{\top}$. In order to treat the two images symmetrically, one can choose to weight the equation so that it measures the value $\sqrt{d(\mathbf{u}'_i, F\mathbf{u}_i)^2 + d(\mathbf{u}_i, F^{\top}\mathbf{u}'_i)^2}$.

This involves multiplying by a weight

$$w_i = \left(\frac{\lambda^2 + \mu^2 + \lambda'^2 + \mu'^2}{(\lambda^2 + \mu^2)(\lambda'^2 + \mu'^2)} \right)^{1/2}$$

The advantage of this type of algorithm is that it is simple to implement compared with iterative parameter estimation methods, such as Levenberg-Marquardt ([55]).

The Images.

The various algorithms were tried with 5 different pairs of images. The images are presented in Figures 2.1 – 2.5 to show the diversity of image types, and the placement of the epipoles. A few of the epipolar lines are shown in the images. The intersection of the pencil of lines is the epipole. There was a wide variation in the accuracy of the matched points for the different images, as will be indicated later.

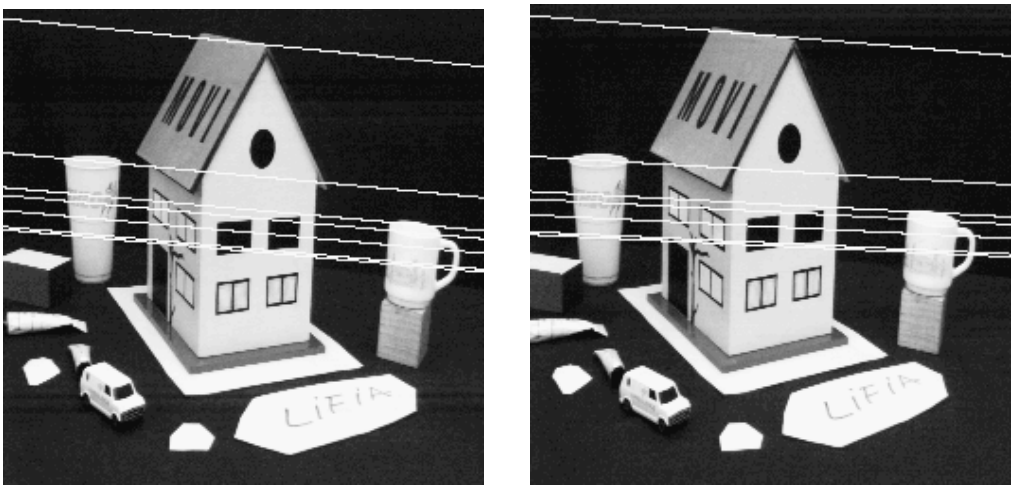


Figure 2.1: **Houses Images.** *The epipoles are a long way from the image centres.*

Graphical Presentation of the Results.

The following graphs show the results of several runs of the algorithms, with different numbers of points being used. The number of points used to compute the fundamental matrix ranged from 8 up to three-quarters of the total number of matched points. For each value of N, the algorithms were run 100 times using randomly selected sets of N matching points. The average error (point – epipolar line distance) was computed using all available matched points. The graphs show the average error over the 100 runs for each value of N. The error shown is the average point-epipolar line distance measured in pixels.

In the graph annotations the following notation is used.

method 0 represents the unnormalized 8-point algorithm

Figure 2.2: **Statue image** *An outdoor scene with the epipoles well away from the centre.*

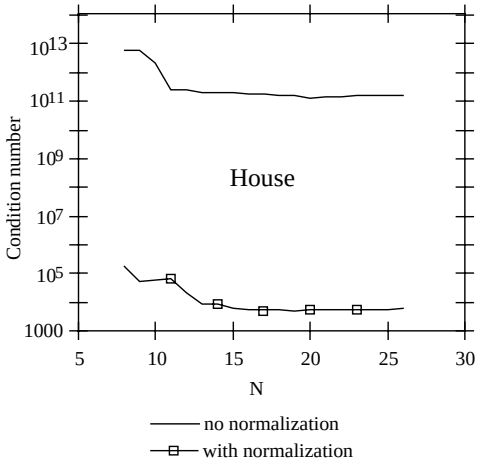
method 1 represents the 8-point algorithm with scaling in stage 1. (For an explanation, see below).

method 2 represents the 8-point algorithm with scaling in stage 2.

method 3 represents the normalized 8-point algorithm (normalization in both stages.

method 4 represents the "optimal" algorithm (minimization of point displacement).

Graph 1 : Effect of Normalization on the Condition Number.



This graph shows a plot of the base-10 logarithm of the condition number of the linear equation set in the case of the house images, for varying numbers of points (the x-axis). The upper curve is without normalization, the lower one with normalization. The improvement is approximately 10^8.

Graph 2 : Effect of normalization on the two stages of the algorithm.

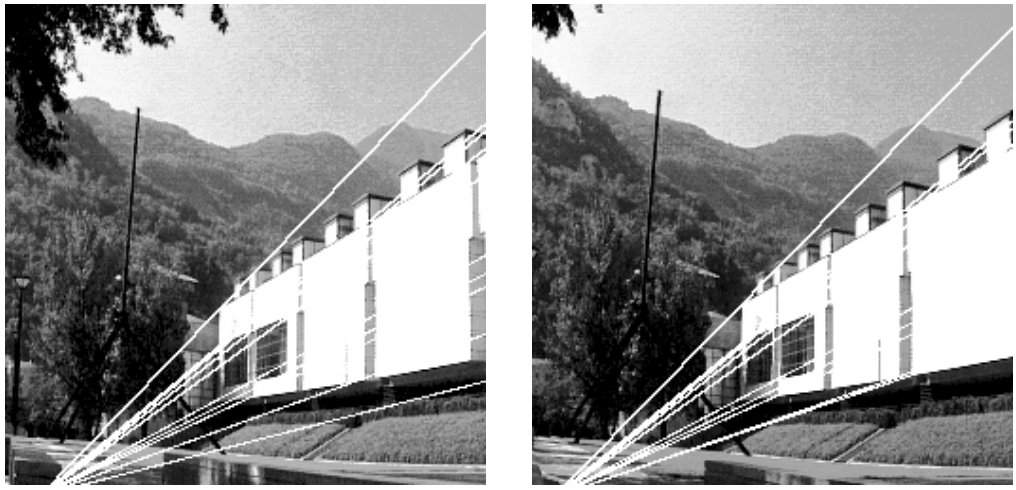
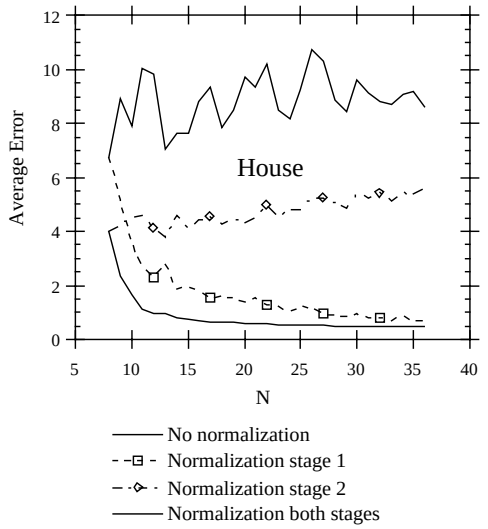


Figure 2.3: **Grenoble Museum** *The epipoles are close to the image.*



This plot shows the effect of normalization in the two stages of the 8-point algorithm. To explain this, four algorithmic steps may be identified :

Normalization Transformation of the image coordinates using transforms T and T'.

Solution Finding matrix F by solving a set of linear equations.

Constraint enforcement Replacing F by the closest singular matrix.

Denormalization Replacing F by ${T'}^{\top} F T$.

It is possible to take these steps in a different order to show the effect of normalization on the Solution (stage 1) and Constraint enforcement (stage 2) steps of the algorithm. Thus, the four curves shown correspond to the following algorithm steps.

Figure 2.4: **Corridor Scene** *In the corridor scene the epipoles are right in the image.*

1. No normalization : Solution – Constraint enforcement.

2. Stage 1 normalization : Normalization – Solution – Denormalization – Constraint enforcement.

3. Stage 2 normalization : Solution – Normalization – Constraint enforcement – Denormalization.

4. Both stages of normalization : Normalization – Solution – Constraint enforcement – Denormalization.

As may be seen, normalization has the greatest effect on stage 1 (the Solution stage), but normalization for stage 2 has a significant effect as well. The best results are had by doing normalization in both stages.

Note, how for $N = 8$ the normalization has no effect on stage 1, since in this case we are finding the solution to a set of equations, and not a least-squares solution to a redundant set. This explains why the two pairs of curves show the same results for $N = 8$.

For these experiments, the house images were used.

Graph 3 : Comparison of normalized and unnormalized 8-point algorithms.

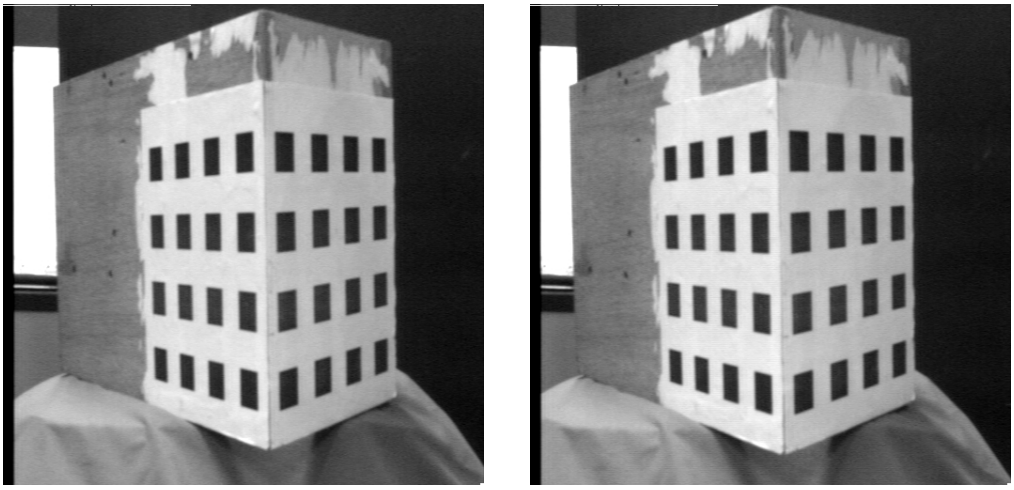
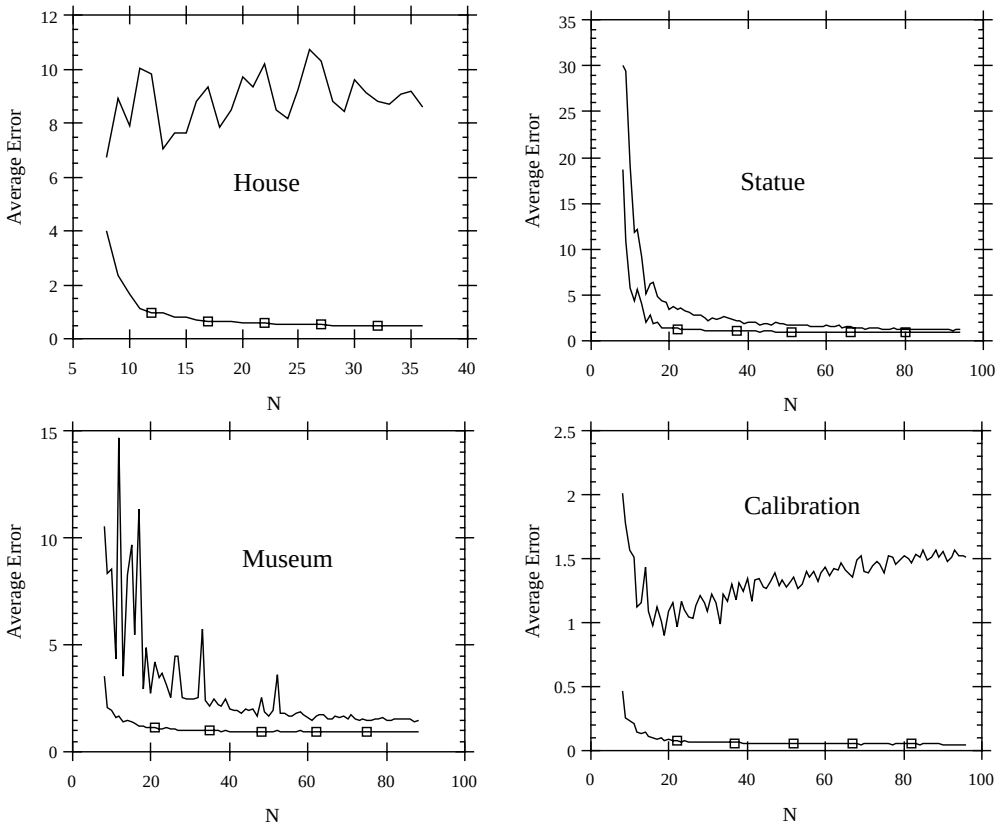
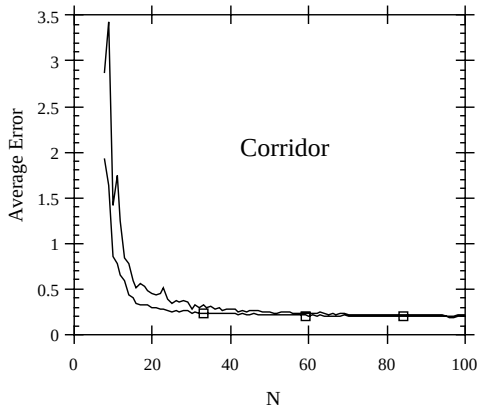


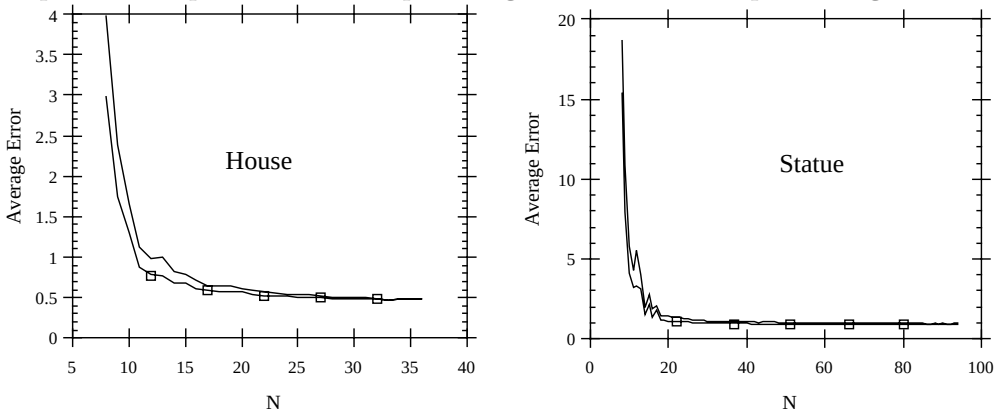
Figure 2.5: **Calibration Jig** *In this calibration jig, the matched points were known extremely accurately.*

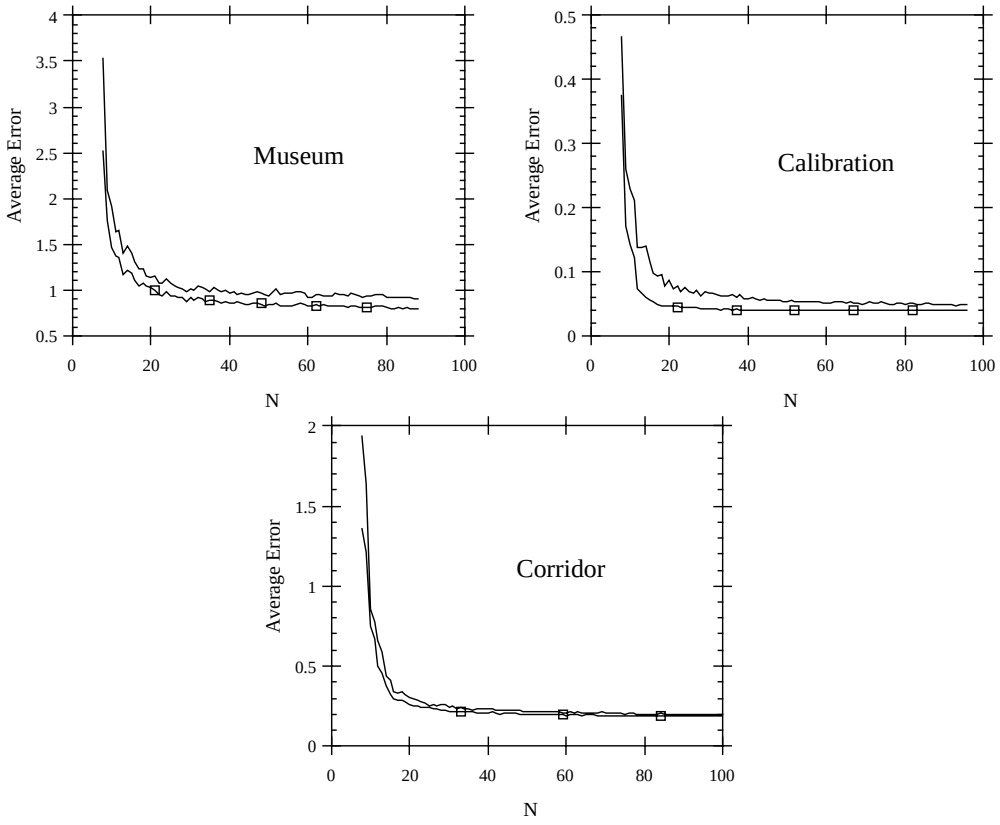




These set of graphs show the improvement achieved by normalization. The images used are from left-to-right and top-to-bottom : house, statue, museum, calibration, basement. Note the differences in Y-scale for the different plots. For some of the images the matched points were known with extreme accuracy (calibration image, basement scene), whereas for others, the matches were less accurate (museum image). In all cases the normalized algorithm performs better than the unnormalized algorithm. In the cases of the calibration and corridor images the effect is not so great. In the case of the images with less accurate matches, the advantage of normalization is dramatic.

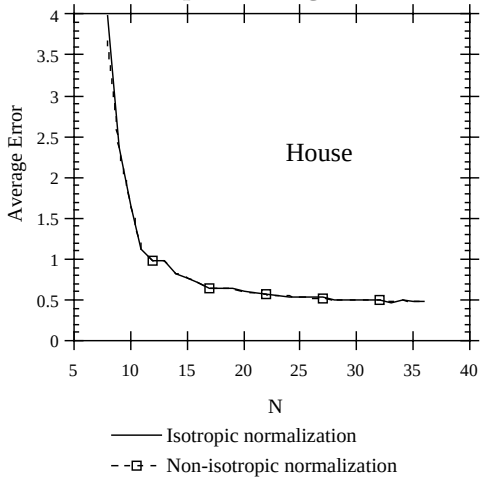
Graph 4 : Comparison of the 8-point algorithm with the optimal algorithm.





This is the same as the previous set of graphs, except that it compares the normalized 8-point algorithm with the optimal (minimized point displacement) algorithm. In all cases the normalized 8-point algorithm performs almost as well as the optimal algorithm.

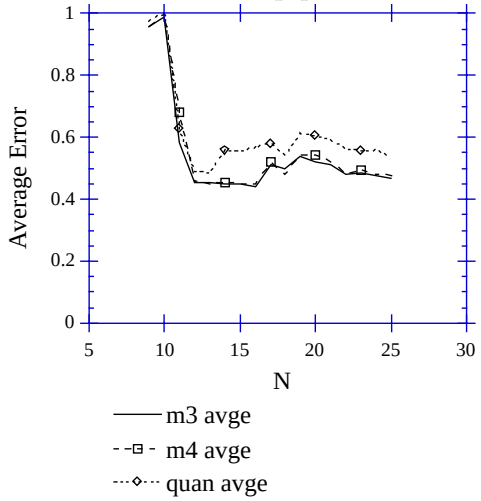
Graph 5 : Isotropic vs. non-isotropic scaling.



The 8-pt algorithm with isotropic and non-isotropic scaling was compared. The two

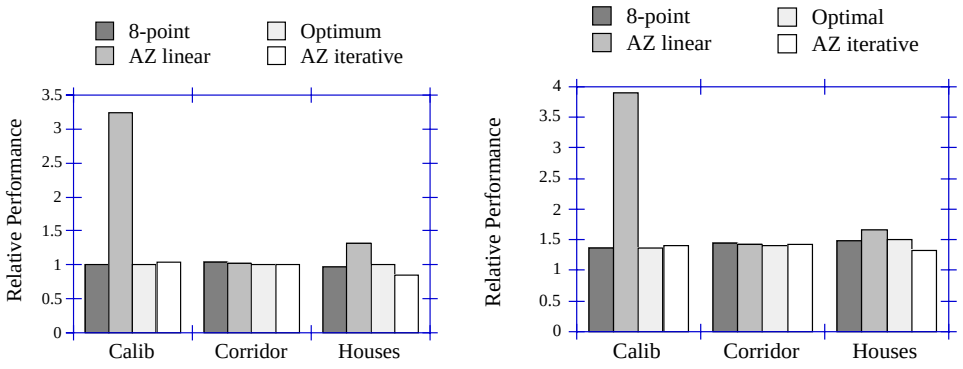
graphs are almost indistinguishable.

Graph 6 : Comparison with minimized epipolar distance



This graph compares the normalized 8-point algorithm, the optimal algorithm and Long Quan's implementation of an iterative algorithm to find minimize the point-epipolar line distance. In implementing this algorithm Quan was following Luong's description ([44, 9]). In this case data was gathered for only one run for each value of N. Nevertheless, the results seem to be consistent. The normalized 8-point and optimal algorithms perform best, and the point-epipolar distance algorithm slightly less well. The graphs start with 9 points. Only the optimal algorithm performed well with 8 points (1.2 pixels error), and the other algorithms were off the graph.

Graph 7 : Comparison with other algorithms

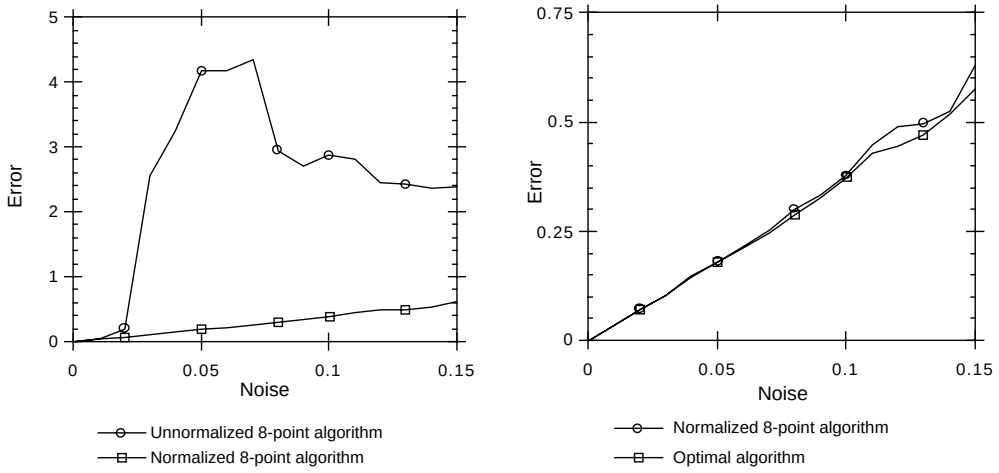


This graph compares the normalized 8-point algorithm and the Optimal algorithm with the results of two algorithms supplied by Andrew Zisserman and Paul Beardsley. These are respectively the algorithms referred to as "Approximate Calibration" and "Iterative Linear" in section 2:3.6. The left hand graph shows average error, and the right hand graph shows RMS error. With results supplied by other researchers, such thorough

testing was not possible. However, for each of the three data sets indicated one test was carried out with number of points $N > 20$, but less than half the points. The number of points N was different for each of the data sets. Except for one case, the results of the various algorithms were comparable.

Note that the algorithm was run on one set of points and the resulting matrix F evaluated with another set of points. For this reason by chance, the so-called "optimal" algorithm did not do so well on one of the data sets as the iterative linear test. However, there was no case in which the optimal algorithm was beaten when evaluated against the points used to run this algorithm. This justifies its use in these experiments as a benchmark algorithm, representing almost the best results possible.

Graph 8 : Reconstruction Error.



To test the performance of the various algorithms for reconstruction accuracy experiments were done to measure the degradation of accuracy as noise levels increase. The Calibration images (2.5) were used for this purpose. Since reconstruction error is most appropriately measured in an Euclidean frame, a Euclidean model was built for the calibration cube, initially by inspection and then by refinement using the image data. This model served as ground truth. Next, the image coordinates were corrected (by an average of 0.02 pixels) to agree exactly with the Euclidean model. Varying amounts of zero-mean gaussian noise were added to the image coordinates, a projective reconstruction was carried out, and a projective transformation was computed to bring the projective reconstruction most nearly into agreement with the model. The average 3D displacement of the reconstructed points from the model was measured. The plotted values are the result average over all points (128 in all) for 10 trials. The reconstruction error is measured in units equal to the length of the side of one of the black squares in the image.

At the left are the results of three algorithms : at the top is unnormalized 8-point algorithm, whereas at the bottom almost overlapped are the results of the normalized 8-point algorithm and the optimal algorithm. In the right hand graph, only the normalized 8-point and optimal algorithms are shown. The result shows that the results of the normalized 8-point algorithm is almost indistinguishable from the optimal algorithm, but that the unnormalized algorithm performs very much worse.

2:3.7 Conclusions

With normalization of the coordinates in order to improve the condition of the problem, the 8-point algorithm performs almost as well as the best iterative algorithms. On the other hand, it runs about 20 times faster and is far easier to code. There seems to be little advantage in choosing the non-isotropic scaling scheme for the normalization transform, since the simpler isotropic scaling performs just as well. Without normalization of the inputs, however, the 8-point algorithm performs quite badly, often with errors as large as 10 pixels, which makes it virtually useless. It would seem to follow that the reason that other researchers have had such poor results with the 8-point algorithm is that they have not carried out any preliminary normalization step as discussed here.

2:4 Triangulation

In this section, we consider the problem of finding the position of a point in space given its position in two images taken with cameras with known calibration and pose. This is the second main step of the projective reconstruction algorithm outlined in section 2:2, after the camera matrices have been found by factoring the fundamental matrix. Finding the point locations requires the intersection of two known rays in space, and is commonly known as triangulation. In the absence of noise, this problem is trivial. When noise is present, the two rays will not generally meet, in which case it is necessary to find the best point of intersection. This problem is especially critical in affine and projective reconstruction in which there is no meaningful metric information about the object space. It is desirable to find a triangulation method that is invariant to projective transformations of space. This section solves that problem by assuming a gaussian noise model for perturbation of the image coordinates. The triangulation problem then may be formulated as a least-squares minimization problem. In this section a non-iterative solution is given that finds a global minimum. It is shown that in certain configurations, local minima occur, which are avoided by the new method. Extensive comparisons of the new method with several other methods show that it consistently gives superior results.

We suppose that a point $\mathbf{x}$ in R^3 is visible in two images. The two camera matrices P and P' corresponding to the two images are supposed known. Let $\mathbf{u}$ and $\mathbf{u}'$ be projections of the point $\mathbf{x}$ in the two images. From this data, the two rays in space corresponding to the two image points may easily be computed. The triangulation problem is to find the intersection of the two lines in space. At first sight this is a trivial problem, since intersecting two lines in space does not present significant difficulties. Unfortunately, in the presence of noise these rays can not be guaranteed to cross, and we need to find the best solution under some assumed noise model.

A commonly suggested method ([4]) is to choose the mid-point of the common perpendicular to the two rays (the *mid-point method*). Perhaps a better choice would be to divide the common perpendicular in proportion to the distance from the two camera centres, since this would more closely equalize the angular error. Nevertheless, this method will not give optimal results, because of various approximations (for instance the angles will not be precisely equal in the two cases). In the case of projective reconstruction, or affine reconstruction however, the camera matrices, will be known in a projective frame of reference, in which concepts such as common perpendicular, or mid-point (in the projective case) have no sense. In this case, the simple mid-point method here will not work.

The importance of a good method for triangulation is clearly shown by Beardsley et. al. who demonstrate that the mid-point method gives bad results. In [4, 5] they suggest an alternative method based on "quasi-Euclidean" reconstruction. In this method, an approximation to the correct Euclidean frame is selected and the mid-point method is carried out in this frame. The disadvantage of this method is that an approximate calibration of the camera is needed. It is also clearly sub-optimal.

In this section a new algorithm is described that gives an optimal global solution to the triangulation problem, equally valid in both the affine and projective reconstruction cases. The solution relies on the concepts of epipolar correspondence and the fundamental matrix ([12]). The algorithm is non-iterative and simple in concept, relying on techniques of elementary calculus to minimize the chosen cost function. It is also moderate in computation requirements. In a series of experiments, the algorithm is extensively tested against many other methods of triangulation, and found to give consistent superior performance. No knowledge of camera calibration is needed.

The triangulation problem is a small cog in the machinery of computer vision, but in many applications of scene reconstruction it is a critical one, on which ultimate accuracy depends ([4]).

2:4.1 Transformational Invariance

In the last few years, there has been considerable interest in the subject of affine or projective reconstruction ([12, 22, 40, 48, 64, 54, 61]). In such reconstruction methods, a 3D scene is to be reconstructed up to an unknown transformation from the given class. Normally, in such a situation, instead of knowing the correct pair of camera matrices P and P', one has a pair PH^{-1} and $P'H^{-1}$ where H is an unknown transformation.

For instance, in the method of projective reconstruction given in section 2:2 one starts with a set of image point correspondences $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$. From these correspondences, one can compute the fundamental matrix F, and hence a pair of camera matrices $\hat{P}$ and $\hat{P}'$. The pair of camera matrices differ from the true ones by an unknown transformation H, and $\hat{P}$ is normalized so that $\hat{P} = [I \mid 0]$. Finally, the 3D space points can be computed by triangulation. If desired, the true Euclidean reconstruction of the scene may then be accomplished by the use of ground control points to determine the unknown transformation, H, and hence the true camera matrices, P and P'. Similarly, in a later section (section 5:2) of this report a projective reconstruction algorithm is given that does a projective reconstruction of points or lines seen in three views, normalized so that the first camera matrix has the form $[I \mid 0]$. In this case, an initial projective reconstruction may be transformed to a Euclidean reconstruction under the assumption that the images are taken all with the same camera, as described in section 6:3.1 of this report.

A desirable feature of the method of triangulation used is that it should be invariant under transformations of the appropriate class. Thus, denote by τ a triangulation method used to compute a 3D space point $\mathbf{x}$ from a point correspondence $\mathbf{u} \leftrightarrow \mathbf{u}'$ and a pair of camera matrices P and P'. We write

$$\mathbf{x} = \tau(\mathbf{u}, \mathbf{u}', P, P')$$

The triangulation is said to be invariant under a transformation H if

$$\tau(\mathbf{u}, \mathbf{u}', P, P') = H^{-1}\tau(\mathbf{u}, \mathbf{u}', PH^{-1}, P'H^{-1})$$

This means that triangulation using the transformed cameras results in the transformed point. If the camera matrices are known only up to an affine (or projective) transformation, then it is clearly desirable to use an affine (resp. projective) invariant triangulation method to compute the 3D space points.

2:4.2 The Minimization Criterion

We assume that the camera matrices, and hence the fundamental matrix, are know exactly, or at least with great accuracy compared with a pair of matching points in the two images. The two rays corresponding to a matching pair of points $\mathbf{u} \leftrightarrow \mathbf{u}'$ will meet in space if and only if the points satisfy the familiar relationship

$$\mathbf{u}'^{\top} F \mathbf{u} = 0 \ . \tag{2.21}$$

It is clear, particularly for projective reconstruction, that it is inappropriate to minimize errors in the 3D projective space, $\mathcal{P}^3$. For instance, the method that finds the midpoint of the common perpendicular to the two rays in space is not suitable for projective reconstruction, since concepts such as distance and perpendicularity are not valid in the context of projective geometry. In fact, in projective reconstruction, this method will give different results depending on which particular projective reconstruction is considered – the method is not projective-invariant.

Normally, errors occur not in placement of a feature in space, but in its location in the two images, due to digitization errors, or the exact identification of a feature in the image. It is common to assume that features in the images are subject to Gaussian noise which displaces the feature from its correct location in the image. We assume that noise model in this report.

A typical observation consists of a noisy point correspondence $\mathbf{u} \leftrightarrow \mathbf{u}'$ which does not in general satisfy the epipolar constraint (2.21). In reality, the correct values of the corresponding image points should be points $\hat{\mathbf{u}} \leftrightarrow \hat{\mathbf{u}}'$ lying close to the measured points $\mathbf{u} \leftrightarrow \mathbf{u}'$ and satisfying the equation $\hat{\mathbf{u}}'^{\top} F \hat{\mathbf{u}}$ exactly. We seek the points $\hat{\mathbf{u}}$ and $\hat{\mathbf{u}}'$ that minimize the function

$$d(\mathbf{u}, \hat{\mathbf{u}})^2 + d(\mathbf{u}', \hat{\mathbf{u}}')^2 \ , \tag{2.22}$$

where $d(*, *)$ represents Euclidean distance, subject to the epipolar constraint

$$\hat{\mathbf{u}}'^{\top} F \hat{\mathbf{u}} = 0 \ .$$

Assuming a Gaussian error distribution, the points $\hat{\mathbf{u}}'$ and $\hat{\mathbf{u}}$ are the most likely values for true image point correspondences. Once $\hat{\mathbf{u}}'$ and $\hat{\mathbf{u}}$ are found, the point $\mathbf{x}$ may be found by any triangulation method, since the corresponding rays will meet precisely in space.

2:4.3 An Optimal Method of Triangulation.

In this section, we describe a method of triangulation that finds the global minimum of the cost function (2.22) using a non-iterative algorithm. If the gaussian noise model can be assumed to be correct, this triangulation method is then provably optimal. This new method will be referred to as the **Polynomial** method, since it requires the solution of a sixth order polynomial.

Reformulation of the Minimization Problem

Given a measured correspondence $\mathbf{u} \leftrightarrow \mathbf{u}'$, we seek a pair of points $\hat{\mathbf{u}}'$ and $\hat{\mathbf{u}}$ that minimize the sum of squared distances (2.22) subject to the epipolar constraint $\hat{\mathbf{u}}'^{\top} F \hat{\mathbf{u}} = 0$.

Any pair of points satisfying the epipolar constraint must lie on a pair of corresponding epipolar lines in the two images. Thus, in particular, the optimum point $\hat{\mathbf{u}}$ lies on an epipolar line λ and $\hat{\mathbf{u}}'$ lies on the corresponding epipolar line λ'. On the other hand, any other pair of points lying on the lines λ' and λ' will also satisfy the epipolar constraint. This is true in particular for the point $\bar{\mathbf{u}}$ on λ lying closest to the measured point $\mathbf{u}$, and the correspondingly defined point $\bar{\mathbf{u}}'$ on λ'. Of all pairs of points on the lines λ and λ', the points $\bar{\mathbf{u}}$ and $\bar{\mathbf{u}}'$ minimize the squared distance sum (2.22). It follows that $\hat{\mathbf{u}}' = \bar{\mathbf{u}}'$ and $\hat{\mathbf{u}} = \bar{\mathbf{u}}$, where $\bar{\mathbf{u}}$ and $\bar{\mathbf{u}}'$ are defined with respect to a pair of matching epipolar lines λ and λ'. Consequently, we may write $d(\mathbf{u}, \hat{\mathbf{u}}) = d(\mathbf{u}, \lambda)$, where $d(\mathbf{u}, \lambda)$ represents the perpendicular distance from the point $\mathbf{u}$ to the line λ. A similar expression holds for $d(\mathbf{u}', \hat{\mathbf{u}}')$.

In view of the previous paragraph, we may formulate the minimization problem differently as follows. We seek to minimize

$$d(\mathbf{u}, \lambda)^2 + d(\mathbf{u}', \lambda')^2 \tag{2.23}$$

where λ and λ' range over all choices of corresponding epipolar lines. The point $\hat{\mathbf{u}}$ is then the closest point on the line λ to the point $\mathbf{u}$ and the point $\hat{\mathbf{u}}'$ is similarly defined.

Our strategy for minimizing (2.23) is as follows

1. Parametrize the pencil of epipolar lines in the first image by a parameter t. Thus an epipolar line in the first image may be written as $\lambda(t)$.
2. Using the fundamental matrix F, compute the corresponding epipolar line $\lambda'(t)$ in the second image.
3. Express the distance function $d(\mathbf{u}, \lambda(t))^2 + d(\mathbf{u}', \lambda'(t))^2$ explicitly as a function of t.
4. Find the value of t that minimizes this function.

In this way, the problem is reduced to that of finding the minimum of a function of a single variable, t. It will be seen that for a suitable parametrization of the pencil of epipolar lines the distance function is a rational polynomial function of t. Using techniques of elementary calculus, the minimization problem reduces to finding the real roots of a polynomial of degree 6.

Details of Minimization.

If both of the image points correspond with the epipoles, then the point in space lies on the line joining the camera centres. In this case it is impossible to determine the position of the point in space. If only one of the corresponding point lies at an epipole, then we conclude that the point in space must coincide with the other camera centre. Consequently, we assume that neither of the two image points $\mathbf{u}$ and $\mathbf{u}'$ corresponds with an epipole.

In this case, we may simplify the analysis by applying a rigid transformation to each image in order to place both points $\mathbf{u}$ and $\mathbf{u}'$ at the origin, $(0,0,1)^\top$ in homogeneous coordinates. Furthermore, the epipoles may be placed on the x-axis at points $(1,0,f)^\top$ and $(1,0,f')^\top$ respectively. A value f equal to 0 means that the epipole is at infinity. Applying these two rigid transforms has no effect on the sum-of-squares distance function (2.22), and hence does not change the minimization problem.

Thus, in future we assume that in homogeneous coordinates, $\mathbf{u} = \mathbf{u}' = (0,0,1)^\top$ and that the two epipoles are at points $(1,0,f)^\top$ and $(1,0,f')^\top$. In this case, since $F(1,0,f)^\top = (1,0,f')F = 0$, the fundamental matrix has a special form

$$F = \begin{bmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{bmatrix} .$$

Consider an epipolar line in the first image passing through the point $(0,t,1)^\top$ (still in homogeneous coordinates) and the epipole $(1,0,f)^\top$. We denote this epipolar line by $\lambda(t)$. The vector representing this line is given by the cross product $(0,t,1) \times (1,0,f) = (tf, 1, -t)$, so the sqared distance from the line to the origin is

$$d(\mathbf{u}, \lambda(t))^2 = \frac{t^2}{1+(tf)^2} .$$

Using the fundamental matrix to find the corresponding epipolar line in the other image, we see that

$$\lambda'(t) = F(0,t,1)^\top = (-f'(ct+d), at+b, ct+d)^\top .$$

This is the representation of the line $\lambda'(t)$ as a homogeneous vector. The squared distance of this line from the origin is equal to

$$d(\mathbf{u}', \lambda'(t))^2 = \frac{(ct+d)^2}{(at+b)^2 + f'^2(ct+d)^2} .$$

The total squared distance is therefore given by

$$s(t) = \frac{t^2}{1+f^2t^2} + \frac{(ct+d)^2}{(at+b)^2 + f'^2(ct+d)^2} . \tag{2.24}$$

Our task is to find the minimum of this function.

We may find the minimum using techniques of elementary calculus, as follows. We compute the derivative

$$s'(t) = \frac{2t}{(1+f^2t^2)^2} - \frac{2(ad-bc)(at+b)(ct+d)}{((at+b)^2 + f'^2(ct+d)^2)^2} . \tag{2.25}$$

Maxima and minima of $s(t)$ will occur when $s'(t) = 0$. Collecting the two terms in $s'(t)$ over a common denominator, and equating the numerator to 0 gives a condition

$$\begin{aligned} f(t) &= t((at+b)^2 + f'^2(ct+d)^2)^2 \\ &\quad -(ad-bc)(1+f^2t^2)^2(at+b)(ct+d) \\ &= 0 . \end{aligned} \tag{2.26}$$

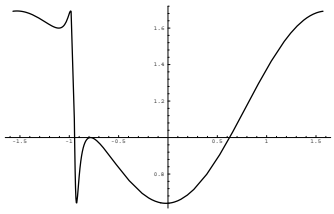


Figure 2.6: *Example of a cost function with three minima.*

The minima and maxima of $s(t)$ will occur at the roots of this polynomial. This is a polynomial of degree 6, which may have up to 6 real roots, corresponding to 3 minima and 3 maxima of the function $s(t)$. The absolute minimum of the function $s(t)$ may be found by finding the roots of $f(t)$ and evaluating the function $s(t)$ given by (2.24) at each of the real roots. More simply, one checks the value of $s(t)$ at the real part of each root (complex or real) of $f(t)$, which saves the trouble of determining if a root is real or complex. One should also check the asymptotic value of $s(t)$ as $t \to \infty$ to see if the minimum distance occurs when $t = \infty$, corresponding to an epipolar line $-fu = 1$ in the first image.

Local Minima

The fact that $f(t)$ in (2.26) has degree 6 means that $s(t)$ may have as many as three minima. In fact, this is indeed possible, as the following case shows. Setting $f = f' = 1$ and

$$F = \begin{bmatrix} 3 & -4 & -3 \\ -2 & 3 & 2 \\ -3 & 4 & 3 \end{bmatrix}$$

gives a function

$$s(t) = \frac{t^2}{1+t^2} + \frac{(4t+3)^2}{(3t+2)^2 + (4t+3)^2}$$

with graph as shown in Fig 2.6[4] The three minima are clearly shown.

As a second example, we consider the case where $f = f' = 1$, and

$$F = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 2 & -1 \\ 0 & 1 & 0 \end{bmatrix} .$$

[4]In this graph and also Fig 2.7 we make the substitution $t = tan(\theta)$ and plot for θ in the range $-\pi/2 \leq \theta \leq \pi/2$, so as to show the whole infinite range or t.

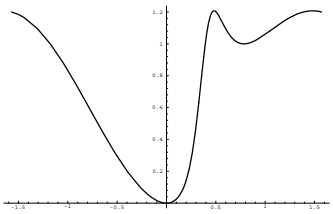


Figure 2.7: *This is the cost function for a perfect point match, which nevertheless has two minima*

In this case, the function $s(t)$ is given by

$$s(t) = \frac{t^2}{t^2+1} + \frac{t^2}{t^2+(2t-1)^2}$$

In this case, both terms of the cost function vanish for a value of $t = 0$, which means that the corresponding points $\mathbf{u}$ and $\mathbf{u}'$ exactly satisfy the epipolar constraint. This can be verified by observing that ${\mathbf{u}'}^\top F\mathbf{u} = 0$. Thus the two points are exactly matched. A graph of the cost function $s(t)$ is shown in Fig 2.7. One sees apart from the absolute minimum at $t = 0$ there is also a local minimum at $t = 1$. Thus, even in the case of perfect matches local minima may occur. This example shows that an algorithm that attempts to minimize the cost function (2.22), or equivalently (2.23) by an iterative search beginning from an arbitrary initial point is in danger of finding a local minimum, even in the case of perfect point matches.

2:4.4 Other Triangulation Methods

In this section, we discuss several other triangulation methods that will be compared with the polynomial method.

Linear Triangulation

The linear triangulation method
is the most common one, described for instance in [22]. Suppose $\mathbf{u} = P\mathbf{x}$. We write in homogeneous coordinates $\mathbf{u} = w(u, v, 1)^\top$, where (u, v) are the observed point coordinates and w is an unknown scale factor. Now, denoting by ${\mathbf{p}_i}^\top$ the i-th row of the matrix P, this equation may be written as follows :

$$wu = {\mathbf{p}_1}^\top\mathbf{x} \quad , \quad wv = {\mathbf{p}_2}^\top\mathbf{x} \quad , \quad w = {\mathbf{p}_3}^\top\mathbf{x} \ .$$

Eliminating w using the third equation, we arrive at

$$\begin{aligned} u\mathbf{p}_3{}^\top\mathbf{x} &= \mathbf{p}_1{}^\top\mathbf{x} \\ v\mathbf{p}_3{}^\top\mathbf{x} &= \mathbf{p}_2{}^\top\mathbf{x} \end{aligned} \tag{2.27}$$

From two views, we obtain a total of 4 linear equations in the coordinates of the $\mathbf{x}$, which may be written in the form $A\mathbf{x} = \mathbf{0}$ for a suitable 4×4 matrix, A. These equations define $\mathbf{x}$ only up to an indeterminant scale factor, and we seek a non-zero solution for $\mathbf{x}$. Of course, with noisy data, the equations will not satisfied precisely, and we seek a best solution.

The Linear-Eigen method. There are many ways to solve for $\mathbf{x}$ to satisfy $A\mathbf{x} = 0$. In one popular method, one finds $\mathbf{x}$ to minimize $||A\mathbf{x}||$ subject to the condition $||\mathbf{x}|| = 1$. The solution is the unit eigenvector corresponding to the smallest eigenvector of the matrix $A^\top A$. This problem may be solved using the Singular Value Decomposition, or Jacobi's method for finding eigenvalues of symmetric matrices ([55, 1]).

The Linear-LS method. By setting $\mathbf{x} = (x, y, z, 1)^\top$ one reduces the set of homogeneous equations, $A\mathbf{x} = 0$ to a set of 4 non- homogeneous equations in 3 unknowns. One can find a least-squares solution to this problem by the method of pseudo-inverses, or by using the Singular Value Decomposition [55, 1].

Discussion. These two methods are quite similar, but in fact have quite different properties in the presence of noise. The **Linear-LS** method assumes that the solution point $\mathbf{x}$ is not at infinity, for otherwise we could not assume that $\mathbf{x} = (x, y, z, 1)^\top$. This is a disadvantage of this method when we are seeking to carry out a projective reconstruction, when reconstructed points may lie on the plane at infinity. On the other hand, neither of these two linear methods is quite suitable for projective reconstruction, since they are non projective-invariant. To see this, suppose that camera matrices P and P' are replaced by PH^{-1} and $P'H^{-1}$. On sees that in this case the matrix of equations, A becomes AH^{-1}. A point $\mathbf{x}$ such that $A\mathbf{x} = \epsilon$ for the original problem corresponds to a point $H\mathbf{x}$ satisfying $(AH^{-1})(H\mathbf{x}) = \epsilon$ for the transformed problem. Thus, there is a one-to-one correspondence between points $\mathbf{x}$ and $H\mathbf{x}$ giving the same error. However, neither the condition $||\mathbf{x}|| = 1$ nor the condition $\mathbf{x} = (x, y, z, 1)^\top$ is invariant under application of the projective transformation H. Thus, in general the point $\mathbf{x}$ solving the original problem will not correspond to a solution $H\mathbf{x}$ for the transformed problem.

For affine transformations, on the other hand, the situation is different. In fact, although the condition $||\mathbf{x}|| = 1$ is not preserved under affine transformation, the condition $\mathbf{x} = (x, y, z, 1)^\top$ is preserved, since for an affine transformation, $H(x, y, z, 1)^\top = (x', y', z', 1)^\top$. This means that there is a one-to-one correspondence between a vector $\mathbf{x} = (x, y, z, 1)^\top$ such that $A(x, y, z, 1)^\top = \epsilon$ and the vector $H\mathbf{x} = (x', y', z', 1)^\top$ such that $(AH^{-1})(x', y', z', 1)^\top = \epsilon$. The error is the same for corresponding points. Thus, the points that minimize the error $||\epsilon||$ correspond as well. Hence, the method **Linear-LS** is affine-invariant, whereas the method **Linear-Eigen** is not. These conclusions are confirmed by the experimental results.

Iterative Linear Methods.

A cause of inaccuracy in the two methods **Linear-LS** and **Linear-Eigen** is that the value being minimized $||A\mathbf{x}||$ has no geometric meaning, and certainly does not correspond to the cost function (2.22). In addition, multiplying each of the equations (rows of A) by some weight will change the solution. The idea of the iterative linear method is to change the weights of the linear equations adaptively so that the weighted equations correspond to the errors in the image coordinate measurements.

In particular, consider the first of the equations (2.27). In general, the point $\mathbf{x}$ we find will not satisfy this equation exactly – rather, there will be an error $\epsilon = u\mathbf{p}_3{}^\top\mathbf{x} - \mathbf{p}_1{}^\top\mathbf{x}$. What we really want to minimize however, is the difference between the measured image coordinate value u and the projection of $\mathbf{x}$, which is given by $\mathbf{p}_1{}^\top\mathbf{x}/\mathbf{p}_3{}^\top\mathbf{x}$. Specifically, we wish to minimize $\epsilon' = \epsilon/\mathbf{p}_3{}^\top\mathbf{x} = u - \mathbf{p}_1{}^\top\mathbf{x}/\mathbf{p}_3{}^\top\mathbf{x}$. This means that if the equation had been weighted by the factor $1/w$ where $w = \mathbf{p}_3{}^\top\mathbf{x}$, then the resulting error would have been precisely what we wanted to minimize. The same weight $1/w$ is the correct one to apply to the second equation of (2.27). For a second image, the correct weight would be $1/w'$ where $w' = \mathbf{p}'_3{}^\top\mathbf{x}$. Of course, we can not weight the equations in this manner because the weights depend on the value of $\mathbf{x}$ which we do not know until after we have solved the equations. Therefore, we proceed iteratively to adapt the weights. We begin by setting $w_0 = w'_0 = 1$, and we solve the system of equations to find a solution $\mathbf{x}_0$. This is precisely the solution found by the linear method **Linear-Eigen** or **Linear-LS**, whichever is being used. Having found $\mathbf{x}_0$ we may compute the weights.

We repeat this process several times, at the i-th step multiplying the equations (2.27) for the first view by $1/w_i$ where $w_i = \mathbf{p}_3\mathbf{x}_{i-1}$ and the equations for the second view by $1/w'_i$ where $w'_i = \mathbf{p}'_3\mathbf{x}_{i-1}$ using the solution $\mathbf{x}_{i-1}$ found in the previous iteration. Within a few iterations this process will converge (one hopes) in which case we will have $\mathbf{x}_i = \mathbf{x}_{i-1}$ and so $w_i = \mathbf{p}_3{}^\top\mathbf{x}_i$. The error (for the first equation of (2.27) for example) will be $\epsilon_i = u - \mathbf{p}_1{}^\top\mathbf{x}_i/\mathbf{p}_3{}^\top\mathbf{x}_i$ which is precisely the error in image measurements as in (2.22).

This method may be applied to either the **Linear-Eigen** or **Linear-LS** method. The corresponding methods will be called **Iterative-Eigen** and **Iterative-LS** respectively. The advantage of this method over other iterative least-squares minimization methods such as a Levenberg-Marquardt (**LM**) iteration ([55]) is that it is very simple to program. In fact, they require only a trivial adaptation to the linear methods. There is no need for any separate initialization method, as is often required by **LM**. Furthermore the decision on when to stop iterating (convergence) is simple. One stops when the change in the weights is small. Exactly when to stop is not critical, since the change in the reconstructed points $\mathbf{x}$ is not very sensitive to small changes in the weights. The disadvantage of this method is that it sometimes fails to converge. In unstable situations, such as when the points are near the epipoles, this occurs sufficiently often to be a problem (perhaps for 5% of the time). If this method is to be used in such unstable circumstances, then a fall-back method is necessary. In the experiments, we have used the optimal **Polynomial** method as a backup in case convergence has not occurred within 10 iterations. In this way the statistics are not negatively biased by occasional very bad results, due to non-convergence.

Despite the similarities of the properties of the **Iterative-LS** method with an direct non-linear least squares minimization of the goal function 2.22, it is not identical. Because the **Iterative-LS** method separates the two steps of computing $\mathbf{x}$ and the weights w and w', the result is slightly different. In fact the three methods **Iterative-LS**, **Iterative-Eigen**

and **LM** are distinct. In particular, the methods **Iterative-LS** and **Iterative-Eigen** are not projective-invariant, though experiments show that they are quite insensitive to projective transformation. Of course, **Iterative-LS** is affine-invariant, just as **Linear-LS** is.

Experiments show that the iterative methods **Iterative-LS** and **Iterative-Eigen** perform substantially better than the corresponding non- iterative linear methods.

Mid-point method

A commonly suggested method for triangulation is to find the mid-point of the common perpendicular to the two rays corresponding to the matched points. This method is relatively easily to compute using a linear algorithm. However, ease of computation is almost its only virtue. This method is neither affine nor projective invariant, since concepts such as perpendicular or mid-point are not affine concepts. It is seen to behave very poorly indeed under projective and affine transformation, and is by far the worst of the methods considered here in this regard. For the record, we outline an algorithm to compute this mid-point. Let $P = [M \mid -M\mathbf{c}]$ be a decomposition of the first camera matrix. The centre of the camera is $\begin{pmatrix} \mathbf{c} \\ 1 \end{pmatrix}$ in homogeneous coordinates. Furthermore, the point at infinity that maps to a point $\mathbf{u}$ in the image is given by $\begin{pmatrix} M^{-1}\mathbf{u} \\ 0 \end{pmatrix}$. Therefore, any point on the ray mapping to $\mathbf{u}$ may be written in the form $\begin{pmatrix} \mathbf{c} + \alpha M^{-1}\mathbf{u} \\ 1 \end{pmatrix}$ or in non-homogeneous coordinates, $\mathbf{c} + \alpha M^{-1}\mathbf{u}$, for some α. Given two images, the two rays must meet in space, which leads to an equation $\alpha M^{-1}\mathbf{u} - \alpha' M'^{-1}\mathbf{u}' = \mathbf{c}' - \mathbf{c}$. This gives three equations in two unknowns (the values of α and α') which we may solve using linear least-squares methods. This minimizes the squared distance between the two rays. The mid point between the two rays is then given by $(\mathbf{c} + \alpha M^{-1}\mathbf{u} + \mathbf{c}' + \alpha' M'^{-1}\mathbf{u})/2$.

Minimizing the sum of the magnitudes of distances

Instead of minimizing the square sum of image errors, it is possible to adapt the polynomial method to minimize the sum of absolute values of the distances, instead of the squares of distances. This method will be called **Poly-Abs**.

The quantity to be minimized is $d(\mathbf{u}, \lambda) + d(\mathbf{u}', \lambda')$ which, as a function of t, is expressed by

$$s_2(t) = \frac{|t|}{\sqrt{1 + f^2 t^2}} + \frac{|ct + d|}{\sqrt{(at + b)^2 + f'^2(ct + d)^2}}.$$

The first derivative is of the form

$$\begin{aligned} s_2'(t) &= = \omega_1 \frac{1}{(1 + f^2 t^2)^{3/2}} \\ &\quad -\omega_2 \frac{(ad - bc)(at + b)}{((at + b)^2 + f'^2(ct + d)^2)^{3/2}} \end{aligned}$$

where ω_1 and ω_2 are equal to -1 or 1, depending on the signs of t and $ct + d$ respectively.

Setting the derivative equal to zero, separating the two terms on opposite sides of the equal sign and squaring to remove the square roots gives

$$\frac{1}{(1+f^2t^2)^3} = \frac{(ad-bc)^2(at+b)^2}{((at+b)^2+f'^2(ct+d)^2)^3}$$

which finally leads to a polynomial of degree 8 in t. We evaluate $s_2(t)$ at the roots of this polynomial to find the global minimum of $s_2(t)$.

2:4.5 Experimental Evaluation of Triangulation Methods

A large number of experiments were carried out to evaluate the different methods described above. We concentrated on two configurations.

Configuration 1 The first configuration was meant to simulate a situation similar to a robot moving down a corridor, looking straight ahead. This configuration is shown in the left part of Fig 2.8. In this case, the two epipoles are close to the centre of the images. For points lying on the line joining the camera centres depth can not be determined, and for points close to this line, reconstruction becomes difficult. Simulated experiments were carried out for points at several distances in front of the front camera.

Numerical values we used are as follows:

- The distance between the two cameras is 1 unit.
- The radius of the sphere of observed points is 0.05 units.
- The distance between the center of the point sphere and the projection center of the second camera is chosen as 0.15 or 0.55 units. The center of the sphere lies on the baseline of the two cameras.
- The cameras have the same calibration matrix

$$K = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Configuration 2 In the other configuration, the pair of cameras were almost parallel, as in an aerial imaging situation. The points were assumed to be approximately equidistant from both cameras, with several different distances being tried. This configuration is shown in the right-hand part of figure 2.8). This was a fairly benign configuration for which most of the methods worked relatively well

In each set of experiments, 50 points were chosen at random in the common field of view. For each of several noise levels varying from 1 to 10 pixels (in a 700×700 image), each point was reconstructed 100 times, with different instances of noise chosen from a gaussian random variable with the given standard deviation (noise level). For each reconstructed point both the 3D reconstruction error, and the 2D residual error (after reprojection of the point) were measured. The errors shown are the average errors. Median errors were also computed. In this latter case the graphs (not shown in this report) had the same

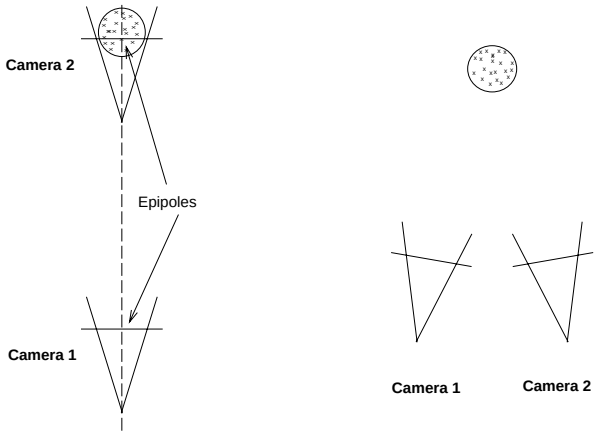
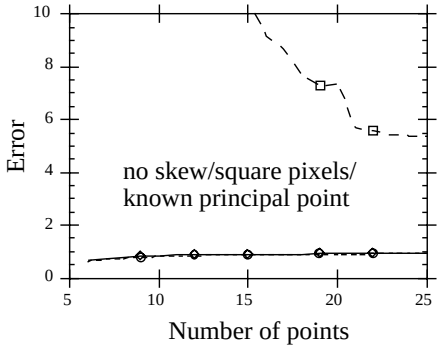


Figure 2.8: *The two simulation configurations.*

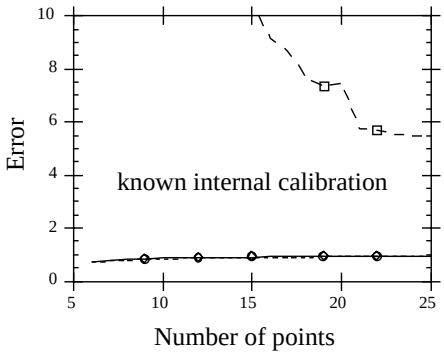
general form and led to the same conclusions. However, they were a little smoother than the graphs shown here, being less sensitive to the occasional gross error.

To measure the invariance to transformation, an affine or projective transformation was applied to each camera matrix. The projective and affine transformations were chosen so that one of the camera matrices was of the form $[I \mid 0]$. This is the normalized form of a camera matrix used in the projective reconstruction method of [22]. It represents a significant distortion, since the actual camera matrix was (by construction) of the form $[M \mid \mathbf{0}]$, where M was a diagonal matrix $\mathrm{diag}(700, 700, 1)$.

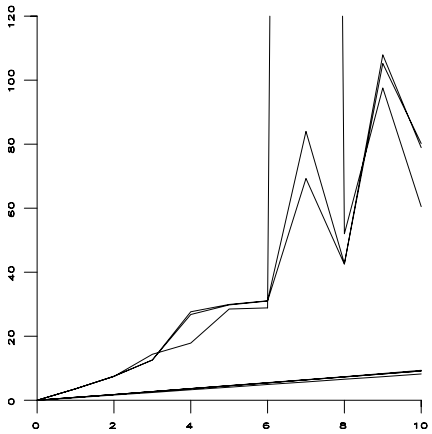
The most unstable situation is Configuration 1, in which the epipoles are in the centre of the two images, and points lie close to the epipoles. Since this situation gave the most severe test to the algorithms, we will give the results for that configuration. Results of two cases are presented. In one case the points are at a distance of 0.15 units in front of the first camera (near points case) and in the other case, they are at 0.55 units distance (far points case). The results will be presented in the form of graphs with a commentary for each graph. The measured error is denoted either as 2D error (meaning error of measured compared with the reprojected points), or 3D error, meaning the error compared with the correct values of the points in space. In addition, we talk of euclidean, affine and projective reconstruction errors. For affine or projective reconstruction, the camera matrices were transformed by a transformation of the given sort, the triangulation was carried out, and finally the reconstructed points were retransformed into the original frame to compare with the correct values. For euclidean reconstruction, no transformation was carried out. Every data point in the graph is the result of 5000 trials, and expresses the RMS or mean value over all the trials. The horizontal axis of each graph is the noise level (between 0 and 10 pixels RMS in each axial direction), and the vertical axis measures the error, in pixels for 2D error, or in space units for 3D error.



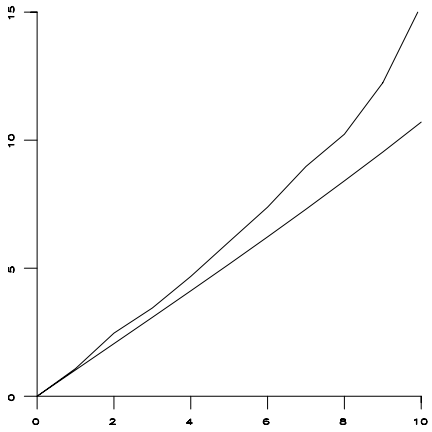
Graph 1 : **3D error for Euclidean reconstruction (near points).** *This graph shows all methods. All perform almost equally. Marginally the best results are given by Mid-point, Linear-LS and Iterative-LS, which are almost indistinguishable. The Polynomial method performs marginally worse than the others. It is designed to minimize 2D error, which explains why optimal in this regard, it is not quite optimal for 3D errors. Euclidean reconstruction is the only instance in which Mid-point performed even marginally well, and the only case in which Polynomial and Poly-Abs were beaten.*



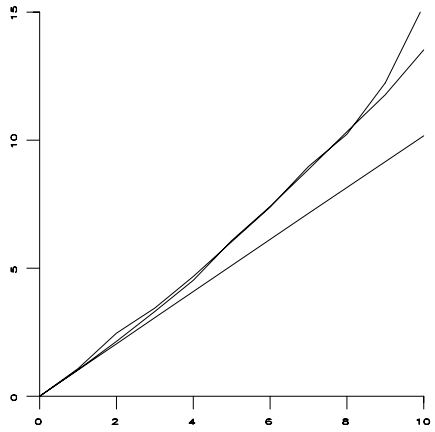
Graph 2 : 3D error for Euclidean reconstruction (far points). *The configuration is the same as for Graph 1 except that the points are further from both cameras. The curves from the bottom are Linear-LS, Poly-Abs and then Polynomial and Linear-Eigen which cross each other. The curves for Mid-point and Iterative-LS are identical with Linear-LS, and only one curve is shown. The same is true of Linear-Eigen and Iterative-Eigen.*



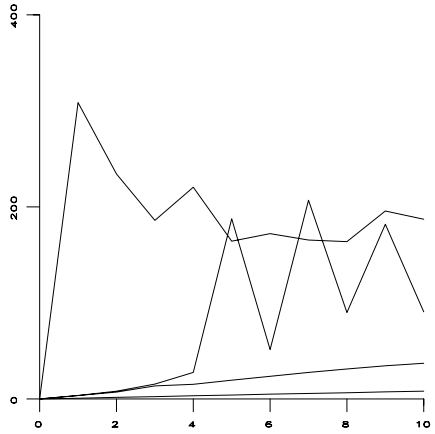
Graph 3 : **2D error for Euclidean reconstruction (near points)** *The configuration is the same as for Graph 1 , except that the average (not RMS) 2D error is measured. Of course Poly-Abs performs best (since it is optimized for this task) but Polynomial, Iterative-LS and Iterative-Eigen are almost indistinguishable. The three very bad performers are Linear-Eigen, Linear-LS and Mid-Point. The maximum Y-scale is 120 pixels. This graph shows that 2D error and 3D error are not well correlated, since despite large 2D errors, these methods perform well in terms of 3D error.*



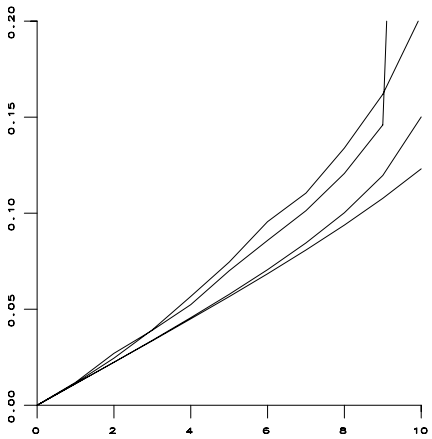
Graph 4 : **Comparison of Euclidean (lower curve) and Projective 2D errors.** *The method shown is Iterative-Eigen. The graph shows that this method is almost projective invariant (that is the two curves are almost the same). This would be an excellent method, except for its failure to converge in very unstable situations (about 1% of trials with noise above 2 pixels). The non-converging cases are ignored in this graph. In cases where the points are not near the epipoles non-convergence is not a problem. The method Iterative-LS (not shown) performs similarly, but just slightly worse, whereas Polynomial is exactly projective-invariant (the two curves are superimposed).*



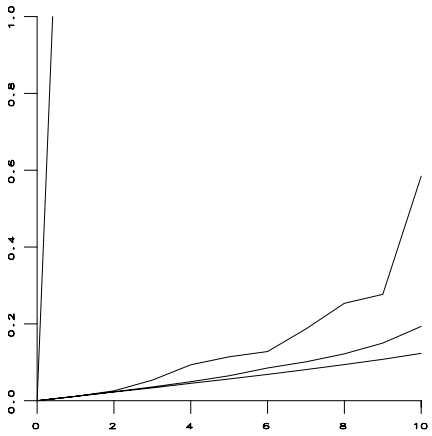
Graph 5 : **2D error for Projective reconstruction (near points).** *This is the case for which all methods performed well in the Euclidean case. This graph shows the results for methods (from the bottom) Polynomial, Iterative-Eigen, and Iterative-LS. The method Poly-Abs (not shown) performed almost identically with Polynomial. This graph shows that Polynomial, or Poly-Abs is the best method for projective reconstruction, whereas Iterative-Eigen and Iterative-LS (except for occasional non-convergence) perform almost as well. Full Y-scale is 20 pixels.*



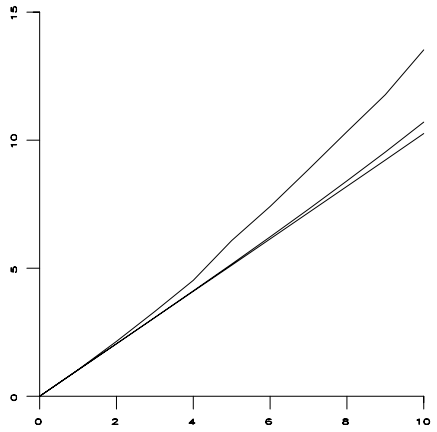
Graph 6 : **2D error for Projective reconstruction (near points), continued.** *This shows the average (not RMS) error, to mitigate the effect of outliers. The graphs shown are (from the bottom), Poly-Abs, Linear-Eigen, Linear-LS and Mid-point. Full Y-scale is 400 pixels. This shows how serious a problem non-invariance under transforms can be.*



Graph 7 : **3D error for Projective reconstruction (near points).** *This is the same as Graph 5 except that we show the 3D error. Poly-Abs performs marginally better than Polynomial. Then follow Iterative-Eigen (except that it fails for noise level of 10 pixels) and Iterative-LS. Full Y-scale is 0.5 units.*



Graph 8 : **3D error for Projective reconstruction (near points), continued.** *The same as Graph 7 for the less well performing methods. From the bottom, are shown Poly-Abs (for reference), Linear-Eigen, Linear-LS and Mid-point (going off scale for noise of 1 pixel). Full Y-scale is 1.0 units.*



Graph 9 : **Affine Invariance.** *The three curves shown are from the bottom Iterative-Eigen (Euclidean) Iterative-LS (Euclidean and Affine superimposed) and Iterative-Eigen (Affine). Thus, as predicted by theory, the Iterative-LS method is precisely affine-invariant, but Iterative-Eigen is not (but almost). Once more we remark that except for occasional non-convergence, these would be good methods.*

2:4.6 Evaluation with real images.

The algorithms were also carried out with the pair of real images shown in Figures 2.9. These images were the images used for one set of experiments in [4].

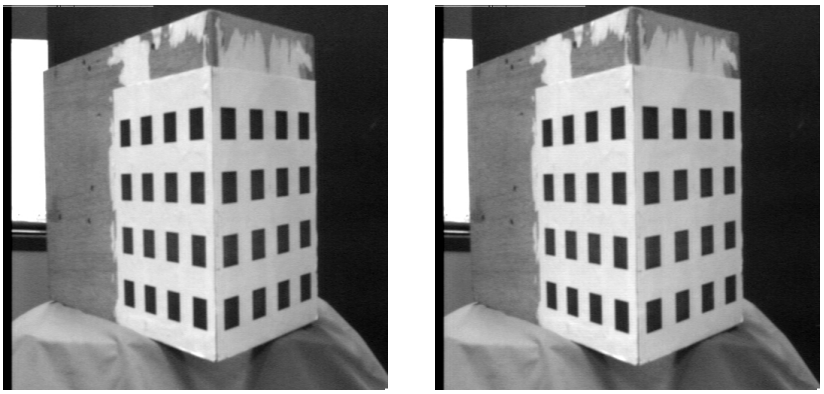
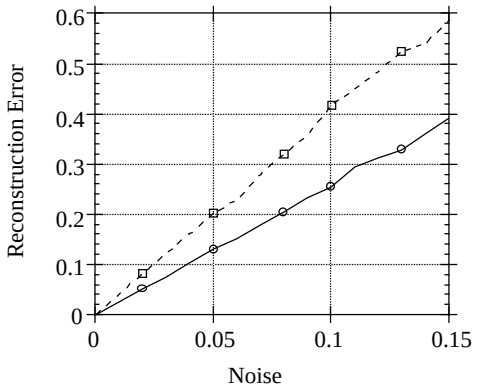


Figure 2.9: *Pair of images used for reconstruction experiments, showing matching epipolar lines.*

The goal of these experiments was to determine how the triangulation method effects the accuracy of reconstruction. Since it makes sense to measure the accuracy of reconstruction in a Euclidean frame where distance has a meaning, a close approximation to a correct Euclidean model for the object was estimated by eye and refined using the measured image locations of the corners of the dark squares. The Euclidean model so obtained was used as ground truth.

We desired to measure how the accuracy of the reconstruction varies with noise. For this reason, the measured pixel locations were corrected to correspond exactly to the Euclidean model. This involved correcting each point coordinate by an average of 0.02 pixels. The correction was so small, because of the very great accuracy of the provided matched points. At this stage we had a model and a set of matched points corresponding exactly to the model. Next, a projective reconstruction of the points was computed by the method of [22, 29], and a projective transform H was computed that brought the projective reconstruction into agreement with the Euclidean model. Next, controlled zero-mean Gaussian noise was introduced into the point coordinates, triangulation was carried out in the projective frame, the transformation H was applied, and the error was measured in the Euclidean frame. Graph 10 shows the results of this experiment for two triangulation methods. It clearly shows that the optimal method gives superior reconstruction results. Note that for these experiments, the projective frame was computed only once, with noiseless data, but triangulation was carried out for data with added noise. This was done to separate the effect of noise on the computation of the projective frame from the effect of noise in the triangulation process. The graph shows the average reconstruction error over all points in 10 separate runs at each chosen noise level.



Graph 10 : **Reconstruction Error** *This graph shows the reconstruction error for the Mid-point (above) and Polynomial methods. On the horizontal axis is the noise, on the vertical axis the reconstruction error. The units for reconstruction error are relative to a unit distance equal to the side of one of the dark squares in the image. The methods Linear-LS, Linear-Eigen, Iterative-LS and Iterative-Eigen gave results close to the Polynomial method. Even for the best method the error is large for higher noise levels, because there is little movement between the images. However, for the actual coordinate error in the original matched points (about 0.02 pixels), the error is small.*

In this pair of images, the two epipoles are distant from the image. For cases where the epipoles are close to the images, the results on synthetic images show that the advantage of the Polynomial methods will be more pronounced.

2:4.7 Timing

The following table shows approximate relative relative speeds for the different algorithms.

Poly	28
Linear-Eigen	6
Iterative-Eigen	10
Mid-point	4
Poly-Abs	60
Linear-LS	4
Iterative-LS	6

Since these are relative measurements only no units appear, but all these algorithms will process several thousands of points per second. In most applications, speed of computation will not be an issue, since it will be small compared with other parts of the computation, such as point matching, or camera model computation.

2:4.8 Discussion of Results

All the methods performed relatively for Euclidean reconstruction, as measured in terms of 3D error. In the case of 2D error, only the methods Polynomial, Poly-Abs, Iterative-LS and Iterative-Eigen perform acceptably, and the last two have the disadvantage of occasional non-convergence. The **Poly-Abs** method seems to give slightly better 3D error performance than **Polynomial** but both of these seem to be excellent methods, not susceptible to serious failure and giving the best overall 3D and 2D error performance. The only distinct disadvantage is that they are not especially easily generalizable to more than two images. They are a bit slower that the other methods, but by a factor of 2 or 3 only, which is probably not significant.

The **Iterative-LS** method is a good method, apart from the problem of occasional non-convergence. Its advantage is that it is about 3 times as fast as the polynomial method and is nearly projective-invariant. In general **Iterative-LS** seems to perform better than **Iterative-Eigen**, but not very significantly. The big problem, however, is non-convergence. This occurs frequently enough in unstable situations to be a definite problem. If this method is used, there must be a back-up method, such as the polynomial method to use in case of non-convergence.

We summarize the conclusions for the various methods.

Poly This is the method of choice when there are only two images and time is not an issue. It is clearly superior to all other methods, except perhaps **Poly-Abs**. In fact, it is optimum under the assumption of a gaussian noise model. It is affine and projective-invariant.

Poly-Abs This is guaranteed to find the global minimum of sum of magnitude of image error. This may be a better model for image noise, placing less emphasis on larger errors. It seems to give slightly better 3D error results. Otherwise it does not behave much differently from **Poly** and it is affine and projective-invariant.

Mid-point This is not a method that one could recommend in any circumstances. Even for Euclidean reconstruction it is no better than other linear methods, such as Linear-LS, which beats it in most other respects. It is neither affine nor projective invariant.

Linear-Eigen The main advantage is speed and simplicity. It is neither affine nor projective invariant.

Linear-LS This has the advantage of being affine invariant, but should not be used for projective reconstruction.

Iterative-Eigen This method gives very good results, markedly better than **Linear-Eigen**, but not quite as good as **Poly**. It may easily be generalized to several images, and is almost projective invariant. The big disadvantage is occasional non-convergence, which occurs often enough to be a problem. It must be used with a back-up method in case of non-convergence.

Iterative-LS This method is similar in performance and properties to **Linear-Eigen**, but should not be used for projective reconstruction, since it does not handle points at infinity well. On the other hand it is affine-invariant.

In summary, the **Polynomial** or **Poly-Abs** method is the method of choice for almost all applications. The **Poly-Abs** method seems to give slightly better 3D reconstruction results. Both these methods are stable, provably optimal, and relatively easy to code. For Euclidean reconstruction, the linear methods are a possible alternative choice, as long as 2D error is not important. However, for affine or projective reconstruction situations, they may be orders of magnitude inferior.

Acknowledgement

Thanks to Paul Beardsley and Andrew Zisserman for making the calibration images and data available to me.

Part III

3-Dimensional Projective Invariants

3:1 Invariant Configurations in $\mathcal{P}^3$

As has been shown in section 2:2.3, although it is impossible to determine the exact geometry of a scene from multiple views, it is in general possible to reconstruct the scene up to an unknown projective transformation of space. Then projective invariants of the 3D structure computed from a projective reconstruction of the scene will have the same value as if it were constructed from the actual scene. Such projective invariants do not include such scene properties as angles and length ratios, which are not invariant under projective transformations of the scene. However, projective invariants do exist for sets of points and lines in space, and these invariants may be computed from two or more views. The general strategy of computing these invariants is as follows.

1. Compute the projective reconstruction of the scene, using for instance the method of Section 2:2.3
2. Compute a projective invariant of the reconstructed scene in $\mathcal{P}^3$.

A lower bound on the number of invariants that exist can be obtained by a simple counting argument. A point in space is described by three parameters (the coordinates of the points), whereas a line is described by four parameters. On the other hand, a projective transformation of space is described by 15 parameters (the 15 independent entries in a 4×4 matrix, defined up to scale). A lower bound on the number of invariants of a set of points and lines is given by

$$\#\text{ invariants } \geq 3\#\text{ points } + 4\#\text{ lines } - 15 \tag{3.1}$$

Thus we see that invariants exist for configurations of 6 points, 4 points + a line, 3 points + 2 lines, 2 points + 3 lines or 4 lines.

We will begin by considering configurations of points and lines in 3-space that have a projective invariant.

3:1.1 Invariants of point sets in $\mathcal{P}^3$

In this section some of the projective invariants of point sets in $\mathcal{P}^3$will be investigated. In particular, a projective invariant of a set of six points $\{\mathbf{x}_i\}$ in $\mathcal{P}^3$will be described.

Given a set of six points $\{\mathbf{x}_i\}$ in $\mathcal{P}^3$, a coordinate system may be selected in which the first five points have coordinates $(1,0,0,0)^\top$, $(0,1,0,0)^\top$, $(0,0,1,0)^\top$, $(0,0,0,1)^\top$ and $(1,1,1,1)^\top$. The coordinates of the sixth point give rise to three independent projective invariants of the six points.

Another formulation of these invariants is given by selecting $\mathbf{x}_0$ and $\mathbf{x}_1$ as base points. Given any other point in $\mathcal{P}^3$, not collinear with $\mathbf{x}_0$ and $\mathbf{x}_1$, there exists a unique plane passing through that point and the two base points $\mathbf{x}_0$ and $\mathbf{x}_1$. In this way, the four points $\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ and $\mathbf{x}_5$ give rise to four planes all containing the line joining $\mathbf{x}_0$ to $\mathbf{x}_1$. From the four planes it is possible to define a cross ratio. In particular, if λ is any line in space, skew to the line passing through $\mathbf{x}_0$ and $\mathbf{x}_1$, then λ intersects the four planes at points $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ and $\mathbf{p}_5$. The cross ratio of these four points on the line λ is a projective invariant of the six original points in $\mathcal{P}^3$.[5]

[5]Both these definitions of invariants fail if three of the points happen to be collinear, however, this case will be ignored for the sake of simplicity.

This is the analogy one dimension higher of the well known invariant of 5 points in a plane. Given 5 points $\mathbf{x}_i$ in $\mathcal{P}^2$, an invariant may be defined by selecting one of the points $\mathbf{x}_0$ and joining it to each of the other points in the plane. The cross ratio of the set of four lines so formed is a projective invariant of the original five points.

There is another way in which invariants may be defined. Five points in general position in the plane may be used to define a unique conic. The conic may be parametrized by a parameter θ and this parametrization may be done in such a way that three of the points have fixed known parameter values, 0, 1 and ∞. The parameters for the other two points may be denoted by α and β, and these two values are independent invariants of the set of five points.

An analogous method of describing the invariants of six points in $\mathcal{P}^3$also holds. In particular, given 6 points in $\mathcal{P}^3$in general position, there exists a unique twisted cubic c that passes through the six points ([60]), and c may be parametrized by a parameter θ in such a way that three of the points receive parameters $0, 1$ and ∞. The parameters of the other three points will then be α, β and γ, and these values are projective invariants of the set of six points.

3:1.2 Line Invariants

In this section, invariants of lines in space will be described. It will be shown that four lines in the 3-dimensional projective plane, $\mathcal{P}^3$give rise to two independent invariants under collineations of $\mathcal{P}^3$. Two different ways of defining invariants will be described, one algebraic and one geometric.

Computing Lines in Space

To be able to compute invariants of lines in space, it is necessary to be able to compute the locations of the lines in $\mathcal{P}^3$from their images in two views. In general, this is impossible as remarked in [77] unless other information is available. Therefore, it will be assumed here that the fundamental matrix F corresponding to the two images is known. This may be derived from a sufficient number of point correspondences, or else from line correspondences, as shown in section 5:2. From the matrix F, two camera transformations P and P' realizing F can be computed as in section 2:2.3.

Lines in the image plane are represented as 3-vectors. For instance, a vector $\mathbf{l} = (l, m, n)^\top$ represents the line in the plane given by the equation $lu+mv+nw = 0$. Similarly, planes in 3-dimensional space are represented in homogeneous coordinates as a 4-dimensional vector $\pi = (p, q, r, s)^\top$.

The relationship between lines in the image space and the corresponding plane in object space is given by the following lemma.

Lemma 3.1. *Let λ be a line in $\mathcal{P}^3$and let the image of λ as taken by a camera with transformation matrix P be $\mathbf{l}$. The locus of points in $\mathcal{P}^3$that are mapped onto the image line $\mathbf{l}$ is a plane, π, passing through the camera centre and containing the line λ. It is given by the formula $\pi = P^\top \mathbf{l}$.*

Proof. A point $\mathbf{x}$ lies on π if and only if it is mapped to a point on the line $\mathbf{l}$ by the action

of the transformation matrix. This means that $P\mathbf{x}$ lies on the line $\mathbf{l}$, and so

$$\mathbf{l}^\top P\mathbf{x} = 0 \ . \tag{3.2}$$

On the other hand, a point $\mathbf{x}$ lies on the plane π if and only if $\pi^\top \mathbf{x} = 0$. Comparing this with (3.2) lead to the conclusion that $\pi^\top = \mathbf{l}^\top P$ or $\pi = P^\top \mathbf{l}$ as required. □

Now, given two images $\mathbf{l}$ and $\mathbf{l}'$ of a line λ in space as taken by two cameras with camera matrices P and P', the line λ is the intersection of the planes $P^\top \mathbf{l}$ and $P'^\top \mathbf{l}'$. This line was computed assuming a particular realization of the fundamental matrix F by P and P'. As with points, the choice of a different realization of F will correspond to a collineation of $\mathcal{P}^3$. The positions of a set of lines seen in the two images will be determined by F up to a collineation.

Algebraic Formulation of Line Invariants

Consider four lines λ_i in space. A line may be given by specifying either two points on the line or dually, two planes that meet in the line. It does not matter in which way the lines are described. For instance, in the formulae (3.4) and (3.5) below certain invariants of lines are defined in terms of pairs of points on each line. The same formulae could be used to define invariants in which lines are represented by specifying a pair of planes that meet along the line. Since the method of determining lines in space from two view given in section 3:1.2 gives a representation of the line as an intersection of two planes, the latter interpretation of the formulae is most useful.

Nevertheless, in the following description, of algebraic and geometric invariants of lines, lines will be represented by specifying two points, since this method seems to allow easier intuitive understanding. It should be borne in mind, however, that the dual approach could be taken with no change whatever to the algebra, or geometry.

In specifying lines, each of two points on the line will be given as a 4-tuple of homogeneous coordinates, and so each line λ_i is specified as a pair of 4-tuples

$$\lambda_i = \big((a_{i1}, a_{i2}, a_{i3}, a_{i4})(b_{i1}, b_{i2}, b_{i3}, b_{i4})\big)$$

Now, given two lines λ_i and λ_j, one can form a 4×4 determinant, denoted by

$$|\lambda_i \lambda_j| = \det \begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \\ a_{j1} & a_{j2} & a_{j3} & a_{j4} \\ b_{j1} & b_{j2} & b_{j3} & b_{j4} \end{bmatrix} \ . \tag{3.3}$$

Finally, it is possible to define two independent invariants of the four lines by

$$I_1(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \frac{|\lambda_1\lambda_2| \ |\lambda_3\lambda_4|}{|\lambda_1\lambda_3| \ |\lambda_2\lambda_4|} \tag{3.4}$$

and

$$I_2(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \frac{|\lambda_1\lambda_2| \ |\lambda_3\lambda_4|}{|\lambda_1\lambda_4| \ |\lambda_2\lambda_3|} \ . \tag{3.5}$$

It is necessary to prove that the two quantities so defined are indeed invariants under collineations of $\mathcal{P}^3$. First, it must be demonstrated that the expressions do not depend

on the specific formulation of the lines. That is, there are an infinite number of ways in which a line may be specified by designating two points lying on it, and it is necessary to demonstrate that choosing a different pair of points to specify a line does not change the value of the invariants. To this end, suppose that $(a_{i1}, a_{i2}, a_{i3}, a_{i4})^\top$ and $(b_{i1}, b_{i2}, b_{i3}, b_{i4})^\top$ are two distinct points lying on a line λ_i, and that $(a'_{i1}, a'_{i2}, a'_{i3}, a'_{i4})^\top$ and $(b'_{i1}, b'_{i2}, b'_{i3}, b'_{i4})^\top$ are another pair of points lying on the same line. Then, there exists a 2×2 matrix D_i such that

$$\begin{bmatrix} a'_{i1} & a'_{i2} & a'_{i3} & a'_{i4} \\ b'_{i1} & b'_{i2} & b'_{i3} & b'_{i4} \end{bmatrix} = D_i \begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \end{bmatrix} .$$

Consequently,

$$\begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \\ a_{j1} & a_{j2} & a_{j3} & a_{j4} \\ b_{j1} & b_{j2} & b_{j3} & b_{j4} \end{bmatrix} = \begin{bmatrix} D_i & 0 \\ 0 & D_j \end{bmatrix} \begin{bmatrix} a'_{i1} & a'_{i2} & a'_{i3} & a'_{i4} \\ b'_{i1} & b'_{i2} & b'_{i3} & b'_{i4} \\ a'_{j1} & a'_{j2} & a'_{j3} & a'_{j4} \\ b'_{j1} & b'_{j2} & b'_{j3} & b'_{j4} \end{bmatrix} .$$

Taking determinants, it is seen that the net result of choosing a different representation of the lines λ_i and λ_j is to multiply the value of $|\lambda_i \lambda_j|$ by a factor $\det(D_i)\det(D_j)$. Since each of the lines λ_i appears in both the numerator and denominator of the expressions (3.4) and (3.5), the factors will cancel and the values of the invariants will be unchanged.

Next, it is necessary to consider the effect of a change of projective coordinates. If H is a 4×4 invertible matrix representing a coordinate transformation of $\mathcal{P}^3$, then it may be applied to each of the points used to designate the four lines. The result of applying this transformation is to multiply the determinant $|\lambda_i \lambda_j|$ by a factor $\det(H)$. The factors on the top and bottom cancel, leaving the values of the invariants (3.4) and (3.5) unchanged. This completes the proof that I_1 and I_2 defined by (3.4) and (3.5) are indeed projective invariants of the set of four lines.

An alternative invariant may be defined by

$$I_3(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \frac{|\lambda_1 \lambda_4| \ |\lambda_2 \lambda_3|}{|\lambda_1 \lambda_3| \ |\lambda_2 \lambda_4|} . \tag{3.6}$$

It is easily seen, that $I_3 = I_1 / I_2$. However, if $|\lambda_1 \lambda_2|$ vanishes, then both I_1 and I_2 are zero, but I_3 is in general non-zero. This means that I_3 can not always be deduced from I_1 and I_2. A preferable way of defining the invariants of four lines is as a homogeneous vector

$$I(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (|\lambda_1 \lambda_2| \ |\lambda_3 \lambda_4| , |\lambda_1 \lambda_3| \ |\lambda_2 \lambda_4| , |\lambda_1 \lambda_4| \ |\lambda_2 \lambda_3|) . \tag{3.7}$$

Two such computed invariant values are deemed equal if they differ by a scalar factor. Note that this definition of the invariant avoids problems associated with vanishing or near-vanishing of the denominator in (3.4) or (3.5).

The definitions of I_1 and I_2 are similar to the definition of the cross-ratio of points on a line. It is well known that for four points on a line, there is only one independent invariant. It may be asked whether I_1 may be obtained from I_2 by some simple arithmetic combination. This is not the case, as will become clearer when the connection of these algebraic invariants with geometric invariants is shown.

Degenerate Cases

The determinant $|\lambda_i \lambda_j|$ as given in (3.3) will vanish if and only if the four points involved are coplanar, that is, exactly when the two lines are coincident (meet in space). If all three components of the vector $I(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ given by (3.7) vanish, then the invariant is undefined. Enumeration of cases indicates that there are two essentially different configurations of lines in which this occurs.

1. Three of the lines lie in a plane.

2. One of the lines meets all the other three.

The configuration where one line meets two of the other lines is not degenerate, but does not lead to very much useful information, since two of the components of the vector vanish. Up to scale, the last component may be assumed to equal 1, which means that two such configurations can not be distinguished. In fact any two such configurations are equivalent under collineation.

Geometric Formulation of Invariants of Lines

Consider four lines λ_i in general position (which means that they are not coincident) in $\mathcal{P}^3$. It will be shown that there exist exactly two further lines τ_1 and τ_2, called *transversals*, which meet each of the four lines. Once this is established, it is easy to define invariants.

The points of intersection of each of the four lines λ_i with one of the transversals τ_j constitute a set of four points on a line in $\mathcal{P}^3$. The cross ratio of these points is an invariant of the four lines λ_i. In this way, two invariants may be defined, one for each of the two transversals.

Invariants may be defined in a dual manner as follows. Given a transversal, τ_j, meeting each of the lines λ_i, there exists, for each λ_i a plane denoted $< \tau_j, \lambda_i >$, containing τ_j and λ_i. This gives rise to a set of four planes meeting in a common line τ_j. The cross-ratio of this set of planes is an invariant of the lines λ_i.

It is easy to see that this dual construction does not give rise to any new invariant. Specifically, consider the cross-ratio of the four planes meeting at τ_1. The cross-ratio of four planes meeting along a line is equal to the cross-ratio of the points of intersection of the planes with any other non-coincident line in space. The line τ_2 is such a line. Hence, the cross ratio of the planes $< \tau_1, \lambda_i >$ is equal to the cross-ratio of the points $< \tau_1, \lambda_i > \cap \tau_2$, where the symbol $\cap$ denotes the point of intersection. However, plane $< \tau_1, \lambda_i >$ meets τ_2 in the point $\lambda_i \cap \tau_2$. In other words, the cross-ratio of the four planes meeting along τ_1 is equal to the cross-ratio of the four points along τ_2, and vice-versa.

Existence of Transversals

Although the existence of transversals is well known, we include here a proof for completeness. To prove the existence of transversals, we start by considering three lines in space.

Lemma 3.2. *There exists a unique quadric surface containing three given lines λ_1, λ_2 and λ_3 in general position in $\mathcal{P}^3$.*

Proof. For a reference to properties of quadric surfaces, the reader is referred to [60]. It is shown there that a quadric surface is a doubly ruled surface containing two families of lines A and B. Two lines from the same set A or B do not meet, whereas any two lines chosen one from each set will always meet. Assuming that the lines λ_i lie on a quadric surface, since they do not meet, they must all come from the same family, which we assume to be A. Now consider any point $\mathbf{x}$ on the quadric surface. There is a unique line passing through $\mathbf{x}$ and belonging to the class B. This line must meet each of the lines λ_i, which belong to class A.

We are led therefore to consider the locus of all points $\mathbf{x}$ in $\mathcal{P}^3$for which there exists a line passing through $\mathbf{x}$ meeting all the lines λ_i. To this end, let $\mathbf{x} = (x, y, z, t)^\top$ be a point on this locus. For each of the lines λ_i we may define a plane π_i passing through $\mathbf{x}$ and λ_i. The condition that there exists a line passing through $\mathbf{x}$ meeting each λ_i means that the three planes π_i meet along that line.

Next, we formulate this last condition algebraically and give a method of computing the formula for the quadric surface. As before, letting $(a_{i1}, a_{i2}, a_{i3}, a_{i4})^\top$ and $(b_{i1}, b_{i2}, b_{i3}, b_{i4})^\top$ be two points on the line λ_i, the plane π_i passing through $\mathbf{x} = (x, y, z, t)^\top$ and the line λ_i may be computed as follows. Consider the matrix

$$\begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \\ x & y & z & t \end{bmatrix} \tag{3.8}$$

The plane π_i is given by the homogeneous vector $(p_{i1}, p_{i2}, p_{i3}, p_{i4})^\top$ where $(-1)^j p_{ij}$ is the determinant of the 3×3 matrix obtained by deleting the j-th column of (3.8). Consequently, each p_{ij} is a homogeneous linear expression in x, y, z and t. Furthermore, since point $(x, y, z, t)^\top$ lies on this plane it follows that

$$x p_{i1} + y p_{i2} + z p_{i3} + t p_{i4} = 0 \ . \tag{3.9}$$

Now the fact that the three planes π_j meet along a common line translates into the algebraic fact that the rank of the matrix

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

is 2. This is equivalent to the condition

$$\det\left(P^{(j)}\right) = 0 \qquad \text{for all } j \ , \tag{3.10}$$

where $P^{(j)}$ is the matrix obtained by removing the j-th column of P. Since each entry p_{ij} of P is a linear homogeneous expression in the variables x, y, z and t, the determinant $\det\left(P^{(j)}\right)$ is a cubic homogeneous polynomial. A point on the required locus must satisfy the condition $\det\left(P^{(j)}\right) = 0$ for $j = 1, \ldots, 4$. However, because of condition (3.9) these four equations are not independent. In particular, if $\mathbf{p}_j$ represents the j-th column of P, then (3.9) implies a relation

$$x\mathbf{p}_1 + y\mathbf{p}_2 + z\mathbf{p}_3 + t\mathbf{p}_4 = 0$$

Then

$$\begin{aligned} x\det\left(P^{(4)}\right) &= x\det\left(\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3\right) \\ &= \det\left(x\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3\right) \\ &= \det\left(-y\mathbf{p}_2 - z\mathbf{p}_3 - t\mathbf{p}_4 \quad \mathbf{p}_2 \quad \mathbf{p}_3\right) \\ &= \det\left(-t\mathbf{p}_4 \quad \mathbf{p}_2 \quad \mathbf{p}_3\right) \\ &= -t\det\left(\mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4\right) \\ &= -t\det\left(P^{(1)}\right) \ . \end{aligned} \tag{3.11}$$

This equation implies that x divides $\det(P^{(1)})$ and t divides $\det(P^{(4)})$. Furthermore, applying the same argument to other coordinates gives rise to an equation

$$\det(P^{(1)})/x = -\det(P^{(2)})/y = \det(P^{(3)})/z = -\det(P^{(4)})/t = R(x,y,z,t)$$

where $R(x,y,z,t)$ is some homogeneous degree-2 polynomial. Then the defining equations (3.10) of the locus become

$$xR(x,y,z,t) = yR(x,y,z,t) = zR(x,y,z,t) = tR(x,y,z,t) = 0 \ . \tag{3.12}$$

This implies that either $R(x,y,z,t) = 0$ or $x = y = z = t = 0$. The latter condition can be discounted, since $(0,0,0,0)$ is not a valid set of homogeneous coordinates. Consequently, the desired locus is described by the degree-2 polynomial equation $R(x,y,z,t) = 0$, and is therefore a quadric surface. Since it is easily verified that the four original lines λ_i lie on this surface, the proof of the lemma is complete. □

It is now a simple matter to prove the existence of transversals.

Theorem 3.3. *There exist exactly two transversals to four lines in general position in* $\mathcal{P}^3$.

Proof. We choose three of the lines λ_1, λ_2 and λ_3 and construct the quadric surface S that they all line on. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the two points of intersection of the fourth line λ_4 with the quadric surface. The construction of S in Lemma 3.2 shows that any transversal to lines λ_1, λ_2 and λ_3 must lie on S. Further, the lines λ_1, λ_2 and λ_3 all belong to one of the families, A, of ruled lines on the quadric surface, S. Let τ_1 and τ_2 be the lines in the other family B passing through $\mathbf{x}_1$ and $\mathbf{x}_2$. Then τ_1 and τ_2 are the two transversals to all four lines. □

Of course, it is possible that λ_4 does not meet the surface S in any real point, or is tangent to S. The statement of the theorem must be interpreted as allowing complex or double solutions. In the case of four real lines in space, there are either two real transversals or two conjugate complex traversals. In the case of complex traversals, there is no conceptual difficulty in defining the invariants as in the real case. The cross-ratio of points of intersections of the lines with the two conjugate transversals will result in two invariants which are complex conjugates of each other.

Various degenerate sets of lines also allow two transversals. For instance suppose that λ_1 and λ_2 are coincident, and so are λ_3 and λ_4. One transversal to the four lines passes through the two points of intersection of the pairs of lines. The other transversal is the line of intersection of the two planes defined by λ_1, λ_2 and by λ_3, λ_4. The cross-ratio invariant corresponding to the first transversal is zero, but the invariant corresponding to the second transversal is in general non-zero and is a useful invariant for this geometric configuration. This is similar to what happens for the algebraically defined invariants (see Section 3:1.2).

Independence and Completeness

We shall now show that the two geometrically defined invariants are independent and together completely characterize the set of four lines up to a collineation of $\mathcal{P}^3$.

To show independence, we start by selecting τ_1 and τ_2, two arbitrary non-intersecting lines in space to serve as transversals. Next, we mark off points $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ and $\mathbf{a}_4$ along τ_1 in such a way that their cross ratio is equal to any arbitrarily chosen invariant value. Similarly, mark off along τ_2 points $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ and $\mathbf{b}_4$ having another arbitrarily chosen cross-ratio invariant value. Now, joining $\mathbf{a}_i$ to $\mathbf{b}_i$ for each i gives a set of four lines having the two arbitrarily chosen invariants.

Next, it will be shown that the two invariants completely characterize the set of four lines up to a collineation. Consequently, let four lines in space have two given cross-ratio invariant values with respect to transversals τ_1 and τ_2 respectively. Let the points of intersection of the four lines with τ_1 be $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ and $\mathbf{a}_4$ and the intersection points with τ_2 be $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ and $\mathbf{b}_4$. Let a second set of lines with the same invariants be given, with transversals τ_j' and intersection points $\mathbf{a}_i'$ and $\mathbf{b}_i'$. Our goal is to demonstrate that there is a collineation taking τ_j to τ_j' for $j = 1, 2$, taking points $\mathbf{a}_i$ to $\mathbf{a}_i'$ and $\mathbf{b}_i$ to $\mathbf{b}_i'$ for $i = 1, \ldots 4$. It will follow that the collineation takes one set of lines λ_i onto the other set.

Choosing two points on each of τ_1 and τ_2, four points in all, and two points on each of τ_1' and τ_2' a further four points, there exists a collineation taking the first set of four points to the second set, and hence taking τ_1 to τ_1' and τ_2 to τ_2'. Suppose that this collineation takes $\mathbf{a}_i$ to $\mathbf{a}_i''$ and $\mathbf{b}_i$ to $\mathbf{b}_i''$, it remains to be shown that there exists a collineation preserving τ_1' and τ_2' and taking $\mathbf{a}_i''$ to $\mathbf{a}_i'$ and $\mathbf{b}_i''$ to $\mathbf{b}_i'$. Without loss of generality it may be assumed that τ_1' is the line $x = y = 0$ and that τ_2' is the line $z = t = 0$. With this choice, we see that a collineation of $\mathcal{P}^3$represented by a matrix of the form $\begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$, where each H_j is a 2×2 block, maps each τ_j' to itself. Furthermore each H_j represents a homography of the line τ_j'. Since the points $\mathbf{a}_i'$ and $\mathbf{a}_i''$ on τ_1' have the same cross-ratio, there is a homography of τ_1' taking $\mathbf{a}_i'$ to $\mathbf{a}_i''$ for $i = 1, \ldots, 4$, and the same can be said for the points $\mathbf{b}_i'$ and $\mathbf{b}_i''$ on τ_2. Hence by independent choice of the two 2×2 matrices H_1 and H_2, both mappings can be carried out simultaneously and the proof is complete.

Existence of an Isotropy

Four lines in $\mathcal{P}^3$can be represented by a total of 16 independent parameters. On the other hand, there are 15 degrees of freedom for collineations of $\mathcal{P}^3$. This suggests that there should be only one invariant for four lines in space, but we have seen that there are two. The discrepancy arises because of the existence of an isotropy ([52]). To understand this, we need to determine the subgroup of all collineations of $\mathcal{P}^3$that fix four given lines. Any such collineation will also fix the two transversals as well as the four points of intersection of the lines with each transversal. Since four points on each transversal are fixed, every point on the transversal must be fixed. This shows that a collineation of $\mathcal{P}^3$fixes four given lines if and only if it fixes the two transversals pointwise. Assuming as before that the two transversals are the lines $x = y = 0$ and $z = t = 0$, it is easily seen that a collineation fixes the transversals pointwise if and only if it is represented by a matrix of the form $\mathrm{diag}(k_1, k_1, k_2, k_2)$ where k_1 and k_2 are two independent constants. Allowing for an arbitrary scale factor in the matrix, this implies that there is a one-parameter subgroup of collineations fixing the four lines. This reduces the number of degrees of

freedom of the group action of collineations of $\mathcal{P}^3$on sets of four lines in space to 14, and explains why there are two independent invariants.

Relationship of Geometric to Algebraic Invariants

The fact that for real lines the algebraic invariants defined in Section 3:1.2 must be real whereas the geometric invariants may be complex indicates that they are not the same. However, since the geometric invariants completely determine the four lines up to collineation, it must be possible to determine the algebraic invariants given the values of the geometric ones. Consider four lines with geometric invariants α and β. We desire to determine the values of the algebraic invariants given by (3.7). To this end, we may assume that the transversals are the lines $x = y = 0$ and $z = t = 0$ and that the points of intersections of the four lines with the transversals have coordinates

$$\begin{aligned} \mathbf{a}_2 &= (0,0,0,1)^\top \\ \mathbf{a}_1 &= (0,0,\alpha,1)^\top \\ \mathbf{a}_3 &= (0,0,1,1)^\top \\ \mathbf{a}_4 &= (0,0,1,0)^\top \end{aligned}$$

and

$$\begin{aligned} \mathbf{b}_2 &= (0,1,0,0)^\top \\ \mathbf{b}_1 &= (\beta,1,0,0)^\top \\ \mathbf{b}_3 &= (1,1,0,0)^\top \\ \mathbf{b}_4 &= (1,0,0,0)^\top \ . \end{aligned}$$

These points have cross-ratio invariants α and β on the transversal lines $x = y = 0$ and $z = t = 0$ respectively.

From this it is easy to compute the value of the invariant (3.7) to be

$$I = (\alpha\beta, 1, 1 + \alpha\beta - \alpha - \beta) \ . \tag{3.13}$$

Hence, it is easy to compute the algebraic invariants from the geometric ones. Similarly, given I, it is easy to solve (3.13) for α and β, which indicates that the algebraic invariant (3.7) is complete.

3:2 Geometric Approach to Invariants

We have seen that projective invariants of scenes may be computed from two of more views by first computing a projective reconstruction of the scene and then computing the invariants directly in 3-dimensions.

An interesting alternative approach to invariant computation has been developed by Gros and Quan ([16, 17, 18]) based on Geometric construction. The following description of geometric construction methods is somewhat different from their approach, but is similar in spirit.

The Coplanarity Test Gros and Quan make use of a test for coplanarity of four points, that I have first seen referred to in [12], ascribed to Roger Mohr. We describe here a different test for coplanarity, that seems to be very slightly simpler. Consider

four points $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ for $i = 1, \ldots, 4$ appearing in two images. Let $\mathbf{x}_i$ be the points in $\mathcal{P}^3$corresponding to these image points. We assume that the epipoles $\mathbf{p}$ and $\mathbf{p}'$ in the two images are also known. The epipole $\mathbf{p}$ is the point where the camera centre of the second camera appears in the first image, and $\mathbf{p}'$ is symmetrically defined. We assume that none of the points $\mathbf{u}_i$ or $\mathbf{u}'_i$ corresponds with one of the epipoles.

Proposition 3.4. *The four points $\mathbf{x}_i$ lie in a plane if and only if there is a 2D projective transformation taking $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4, \mathbf{p}$ to $\mathbf{u}'_1, \mathbf{u}'_2, \mathbf{u}'_3, \mathbf{u}'_4, \mathbf{p}'$.*

The fact that the sets of points are in projective equivalence means of course that the cross-ratio invariants of the two point sets are equal.

Proof. Suppose that the four points $\mathbf{x}_i$ lie in a plane π and let $\mathbf{e}$ be the point where the line joining the two camera centres meets π. Then both sets of points $\mathbf{u}_1, \ldots, \mathbf{u}_4, \mathbf{p}$ and $\mathbf{u}'_1, \ldots, \mathbf{u}'_4, \mathbf{p}'$ are (2D) projectively equivalent to the set $\mathbf{x}_1, \ldots, \mathbf{x}_4, \mathbf{e}$, and hence to each other. To prove the converse, suppose that π is the plane containing $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ and as before, let $\mathbf{e}$ be the intersection of π with the line joining the camera centres.

There exists a unique 2D projective transformations T taking $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{p}$ to $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{e}$, since a 2D transform is uniquely defined by 4 points. A transform T' may be similarly defined, and $T'^{-1}T$ is the unique transform taking $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{p}$ to $\mathbf{u}'_1, \mathbf{u}'_2, \mathbf{u}'_3, \mathbf{p}'$. If the point $\mathbf{x}_4$ does not lie on the plane π, however, then its projections onto π from the two camera centres are different. This means that $T\mathbf{x}_4 \neq T'\mathbf{x}_4$, and so $T'^{-1}T\mathbf{u}_4 \neq \mathbf{u}'_4$, so the points are not in projective correspondence. □

Gros and Quan use a coplanarity criterion to allow them to compute the point of intersection of a line with a plane defined by three points. Using the above coplanarity criterion, this may be done as follows. Let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ be the three points and let L be a line. Let the images of the points and line be $\mathbf{u}_i$ and λ in one image, and the same with primes in the other image. Let $\mathbf{p}$ and $\mathbf{p}'$ be the epipoles. Consider the transformation T defined by $T\mathbf{u}_i = \mathbf{u}'_i$ and $T\mathbf{p} = \mathbf{p}'$. The line λ' is also transformed by T to a line which we may (somewhat loosely) denote $T\lambda$. The intersection of λ' and $T\lambda$ is the image of the point where the L meets the plane π. This follows immediately from Proposition 3.4, since this is the unique point on λ' that is in projective correspondence, via T, with a point on the line λ. This is illustrated in fig 3.1.

Three points and Two Lines. The idea put forward by Gros and Quan is to use this method to compute invariants. As an example, consider three points and two lines in 3D, and suppose that the epipoles are known. The three points define a plane. The intersection of the two lines with this plane, plus the three original points give five points in a plane, from which one may derive two invariants. By using the construction of the previous paragraph and applying it to both of the lines, one immediately finds the image of the three points and the two intersection points, as seen in the second (primed) image. This method is explained further in Fig 3.2.

Four Points and One Line. As another example, consider four points and one line. One may extract four subsets of three points from among the set of points. Each such subset defines a plane that meets the line in a single point. This provides four points on one line, and hence a single invariant. Using the above construction, one easily computes the four points, as seen in the primed image. This is illustrated in Fig 3.3.

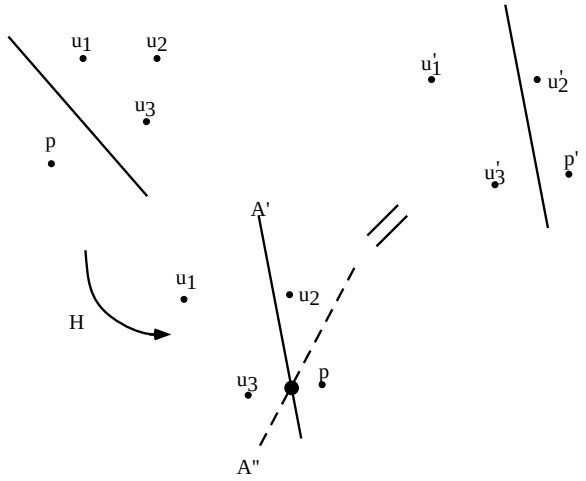


Figure 3.1: **Intersection of a line with a plane defined by three points.** *Consider points u_1, u_2, u_3 and the epipole p in one image and corresponding points u'_1, u'_2, u'_3 and p' in the second image. Compute the 2D projective mapping H that takes the points to their corresponding points in the second image. Transfer the line by H and compute its intersection with the line in the second image. This intersection point is the point where the line meets the plane of the three points, as seen in the second image.*

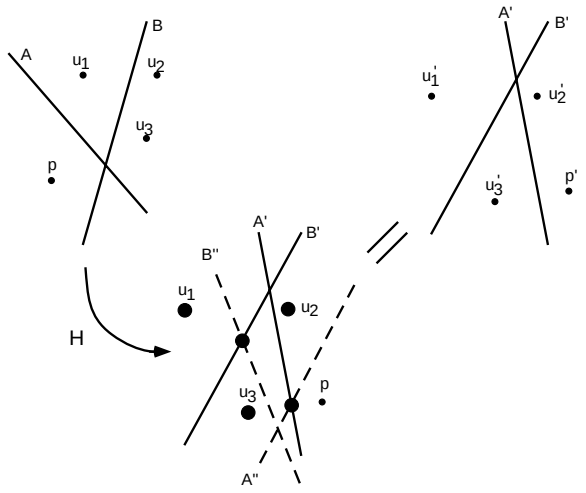


Figure 3.2: **Invariants from 3 points and 2 lines.** *At the top are two views of 3 points and 2 lines. It is assumed that the epipoles p and p' are also known. To compute the invariants of the set of points and lines, one finds a 2D projective transformation H that takes the points u_i to u'_i and p to p'. Since a 2D projectivity is determined by 4 points, the transform H is uniquely defined. Let $A \leftrightarrow A'$ and $B \leftrightarrow B'$ be the two matching lines. By transforming (warping) the first image by H, point u_i is mapped to u'_i, and the lines A and B are mapped to lines A'' and B''. The points u'_1, u'_2, u'_3 along with $u'_4 = A' \cap A''$ and $u'_5 = B' \cap B''$ form a set of 5 coplanar points (in the image plane of the primed camera). The two 2D projective invariants of these 5 points are invariants of the set of 3 points and 2 lines.*

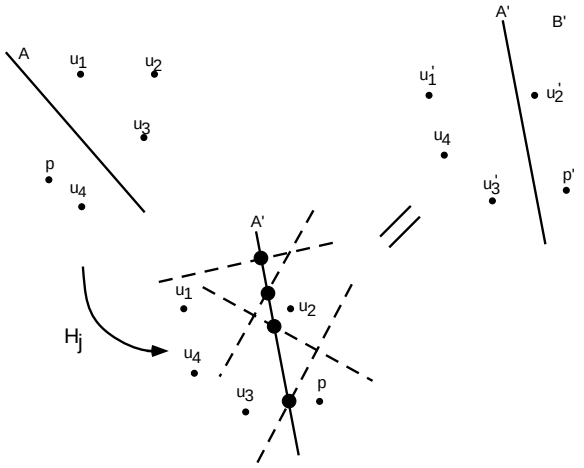


Figure 3.3: **Invariants from 4 points and 1 line.** *At the top are two views of 4 points and 1 lines. It is assumed that the epipoles p and p' are also known. For $j = 1, \ldots, 4$ one finds 2D projective transforms H_j defined as follows : H_j is the transform that takes p to p' and u_i to u'_i for each $i = 1, \ldots, 4$,* **except** *for $i = j$. By mapping the line A into the other image by transformations H_j one obtains four lines A''_j in the second image. The cross-ratio of the four points of intersection of A''_j with A' is an invariant of the four points and one line in space. This is because these four intersection points are in projective correspondence with the points of intersection of the line with the four planes defined by sets of three points.*

Six Points The construction used for four points and one line can be used for six points by selecting two of the points to define a line. One is thus reduced to the case of four points and one line. The difference, however is that one now has two extra points on the line, namely the two points used to define the line. Thus we have a total of six points on the line A', namely the four intersection points plus the two points defining A'. From these six points on a line, we may extract three independent cross ratios.

Further Reading

Other methods of geometric computation of invariants are given by Ponce ([53]) Quan ([56]) and in [30]. This includes invariants derived from smaller numbers of points (6 points in 3 views or 7 points in 2 views).

3:3 Algebraic Approach to Invariants

A very interesting algebraic method of computation of invariants was given by Carlsson ([8]). By completely algebraic techniques, involving the so-called *double algebra*, he derived explicit formulas for projective invariants of point and line sets as seen in a pair of images. The formulae express the invariants directly in terms of the image coordinates of the points as seen in the two views.

As an example, we consider the line invariants discussed in section 3:1.2. Thus, let $\ell_i \leftrightarrow \ell'_i$ for $i = 1, \ldots, 4$ be a set of four corresponding lines in two views, and let λ_i be the corresponding line in 3-space. Denote by $\mathbf{u}_{ij}$ the intersection of the lines ℓ_i and ℓ_j. Thus $\mathbf{u}_{ij} = \ell_i \times \ell_j$. Similarly, let $\mathbf{u}'_{ij} = \ell'_i \times \ell'_j$. Then, with $|\lambda_i \lambda_j|$ defined as in (3.3), Carlsson shows that

$$|\lambda_i \lambda_j| = k {\mathbf{u}'_{ij}}^\top F \mathbf{u}_{ij} \tag{3.14}$$

where k is a constant. From this it follows that the invariant (3.7) may be written entirely in terms of the image coordinates, and the fundamental matrix.

$$I = ({\mathbf{u}'_{12}}^\top F \mathbf{u}_{12} . {\mathbf{u}'_{34}}^\top F \mathbf{u}_{34},\ {\mathbf{u}'_{13}}^\top F \mathbf{u}_{13} . {\mathbf{u}'_{24}}^\top F \mathbf{u}_{24},\ {\mathbf{u}'_{14}}^\top F \mathbf{u}_{14} . {\mathbf{u}'_{23}}^\top F \mathbf{u}_{23}) \ .$$

This formula will be proven later in this report (section 3:3), when we return to a consideration of the fundamental matrix and the trilinear relations.

By consideration of the results of Carlsson, it may be seen that many of the invariants of mixed point and line sets in $\mathcal{P}^3$are actually just disguised invariants of sets of lines. In fact, replacing two points by the line that passes through them and then defining a line invariant gives the same result. This is certainly true of all the invariants considered in [8].

Invariants of Points Given a set of six points, $x_1, \ldots, x_6$, denote by λ_{ij} the line that passes through points $\mathbf{x}_i$ and $\mathbf{x}_j$. Then the expression

$$I_1(\lambda_{12}, \lambda_{45}, \lambda_{13}, \lambda_{46}) = \frac{|\lambda_{12}\lambda_{45}|\ |\lambda_{13}\lambda_{46}|}{|\lambda_{12}\lambda_{46}|\ |\lambda_{13}\lambda_{45}|} \tag{3.15}$$

is invariant. The lines λ_{ij} map to lines ℓ_{ij} and ℓ'_{ij} in the two images, and these lines may be computed easily in terms of the measured image coordinates $\mathbf{u}_i$ and $\mathbf{u}'_i$. Then,

applying (3.14), one easily obtains a formula for the six-point invariant in terms of the image coordinates $\mathbf{u}_i$ and $\mathbf{u}'_i$ and the fundamental matrix F.

$$I = (\mathbf{x}_1, \ldots, \mathbf{x}_6) = \frac{{\mathbf{u}'_{12:45}}^\top F \mathbf{u}_{12:45} . {\mathbf{u}'_{13:46}}^\top F \mathbf{u}_{13:46}}{{\mathbf{u}'_{12:46}}^\top F \mathbf{u}_{12:46} . {\mathbf{u}'_{13:45}}^\top F \mathbf{u}_{13:45}} \tag{3.16}$$

where $\mathbf{u}_{ij:kl}$ is the intersection of the line $\mathbf{u}_i \times \mathbf{u}_j$ with the line $\mathbf{u}_k \times \mathbf{u}_l$. Namely, $\mathbf{u}_{ij:kl} = (\mathbf{u}_i \times \mathbf{u}_j) \times (\mathbf{u}_k \times \mathbf{u}_l)$. This formula is formula (30) in [8]. The lines λ_{12} and λ_{13} meet in one point, $\mathbf{x}_1$, and similarly lines λ_{45} and λ_{46} meet in $\mathbf{x}_4$. Consequently, there is only one invariant associated with the set of four lines. By choosing other combinations of the points to construct lines, one obtains more invariants. In total, there are three independent invariants of a set of six points.

Invariants of Mixed Points and Lines Given a set of mixed points and lines in $\mathcal{P}^3$, one can define invariants in a similar manner by joining subsets of the points together to make lines. Then, one applies the 4-line invariant to obtain invariants of the set of points and lines.

3:4 Experimental Results

Three images of a pair of wooden blocks representing houses were acquired and vertices and edges were extracted. The images are shown in Fig 3.4. Corresponding edges and vertices were selected by hand from among those detected automatically. The chosen edges and vertices are also shown in Fig 3.4. There were 13 edges and 15 lines extracted from each of the images. The dotted edges were not visible in all images and were not chosen. Vertices are represented by numbers and edges by letters in the figure. Because of the way edges and vertices were found by the segmentation algorithm, the edges do not always pass precisely through the indicated vertices, but sometimes through a closely neighboring vertex. On other occasions, the full edge was not detected as a single, but was broken into several pieces. This is usual with most edge detection algorithms, and is a source of error in the computation of invariants.

The fundamental matrices F_{12} for the first and second images and F_{23} for the second and third images were computed from the point matches.

3:4.1 Comparison of Invariant Values

The invariants described in this section are represented as homogeneous vectors. Two such vectors are considered equivalent if they differ by a non-zero scale factor. Because of arithmetic error and image noise, two computed invariant values will rarely be exactly proportional. In order to compare two such computed invariant values (perhaps when attempting to match an object with a reference object), each homogeneous vector is multiplied by a scale factor chosen to normalize its length to 1. This normalization determines the vector up to a multiplication by a factor ± 1. Two such normalized homogeneous vector invariants $\mathbf{v}_1$ and $\mathbf{v}_2$ are deemed *close* if $\mathbf{v}_1$ is close to $\mathbf{v}_2$ or to $-\mathbf{v}_2$ using a Euclidean norm. Correspondingly, a metric may be defined by

$$d(\mathbf{v}_1, \mathbf{v}_2) = \left(1 - \left|\frac{\mathbf{v}_1 . \mathbf{v}_2}{||\mathbf{v}1||\,||\mathbf{v}_2||}\right|\right)^{1/2} . \tag{3.17}$$

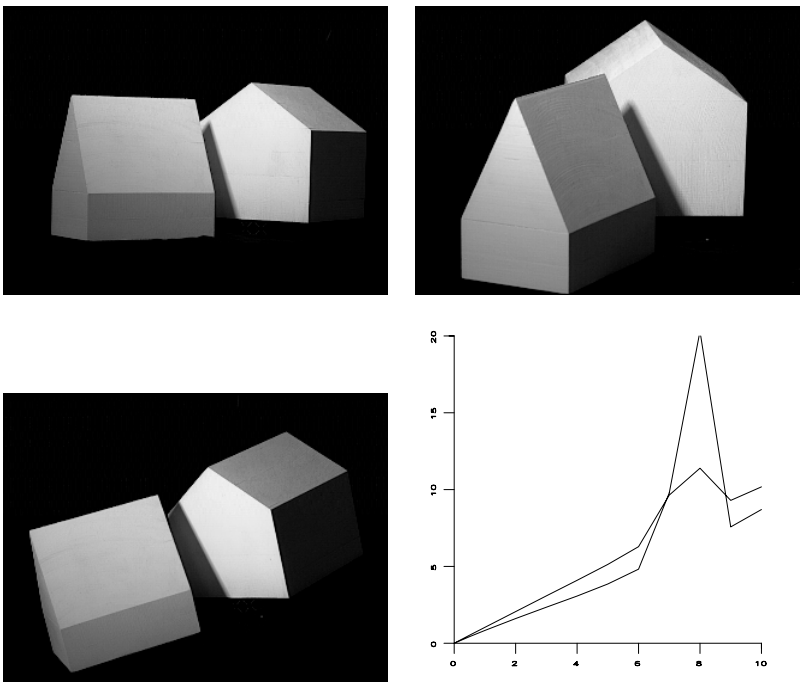


Figure 3.4: Three views of houses, and numbered selected vertices

For any $\mathbf{v}_1$ and $\mathbf{v}_2$, distance $d(\mathbf{v}_1, \mathbf{v}_2)$ lies between 0 and 1.

3:4.2 Invariants of 6 points

The invariants of six points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_6\}$ were computed by finding a projective coordinate frame in which the points $\mathbf{x}_1, \ldots, \mathbf{x}_5$ have coordinates $(1,0,0,0)^\top$, $(0,1,0,0)^\top$, $(0,0,1,0)^\top$, $(0,0,0,1)^\top$ and $(1,1,1,1)^\top$ respectively. The homogeneous coordinates or the sixth point, $\mathbf{x}_6$ in that frame are the desired invariants of the original set of points. Two points are compared using the metric (3.17). Six sets of six points were chosen for computation of invariants. The sets of points were chosen arbitrarily by hand. The six sets of six lines chosen as in the following table which shows the indices of the lines as given in Fig. 3.4.

$$\begin{array}{rcrrrrrr} S_1 & = & \{1, & 2, & 3, & 6, & 9, & 10\} & , \\ S_2 & = & \{2, & 4, & 6, & 8, & 10, & 12\} & , \\ S_3 & = & \{1, & 3, & 5, & 7, & 9, & 11\} & , \\ S_4 & = & \{1, & 2, & 3, & 6, & 7, & 8\} & , \\ S_5 & = & \{1, & 4, & 7, & 10, & 13, & 12\} & , \\ S_6 & = & \{2, & 5, & 8, & 11, & 12, & 13\} & \end{array}$$

Table (3.18) shows the invariant of the sets of six points as computed from the first and second and from the second and third images.

$$\begin{array}{llllll} 0.0266367 & 0.970462 & 0.975994 & 0.619897 & 0.847914 & 0.823575 \\ 0.995416 & 0.0155304 & 0.0648768 & 0.841029 & 0.252926 & 0.548214 \\ 0.967114 & 0.066834 & 0.0136234 & 0.863063 & 0.276384 & 0.516868 \\ 0.617346 & 0.830651 & 0.873538 & 0.0166752 & 0.704992 & 0.752215 \\ 0.861618 & 0.238502 & 0.289846 & 0.708237 & 0.00561718 & 0.590905 \\ 0.828638 & 0.54423 & 0.519272 & 0.719518 & 0.574651 & 0.0263892 \end{array} \tag{3.18}$$

The (i, j)-th entry of the table shows the distance according to the metric (3.17) between the invariant of set S_i as computed from images 1 and 2 with that of set S_j as computed from images 2 and 3. The diagonal entries of the matrix (in bold) should be close to 0.0, which indicates a match. The matrix should be approximately symmetric, which is in fact the case.

The off-diagonal entries are not close to zero, except for the $(2, 3)$ entry – but even that entry is greater than the diagonal entries. This indicates that the six-point invariant is very good at discriminating between sets of points with different geometrical structure. Evidently, sets of points S_2 and S_3 are quite similar in arrangement, at least up to collineation.

3:4.3 Invariants of 4 lines

The same experiment was carried out with six sets of four lines. First the fundamental matrices were computed using point matches and then the line invariant (3.7) was computed for each pair of line sets and compared using the metric (3.17).

The sets of lines chosen are given in the following table (refer to Fig. 3.4).

$$\begin{array}{rcl} S_1 & = & \{B, C, J, K\} \\ S_2 & = & \{B, G, J, N\} \\ S_3 & = & \{A, B, H, I\} \\ S_4 & = & \{B, D, E, G\} \\ S_5 & = & \{A, C, O, J\} \\ S_6 & = & \{B, I, L, N\} \end{array}$$

Table (3.19 shows the results. The only bad entry in this matrix is in the position (4, 4). This is because of the fact that the four lines chosen contained three coplanar lines (lines B, D and E). This causes the values of the invariant to be indeterminate (that is $(0, 0, 0)$), and shows that such instances must be detected and avoided.

$$\begin{array}{llllll} 0.0128906 & 0.674135 & 0.302728 & 0.688589 & 0.642501 & 0.449448 \\ 0.646976 & 0.0337898 & 0.741489 & 0.83827 & 0.706921 & 0.221636 \\ 0.0619738 & 0.691264 & 0.229193 & 0.707536 & 0.708276 & 0.461339 \\ 0.286604 & 0.607681 & 0.182331 & 0.890303 & 0.855833 & 0.383939 \\ 0.656635 & 0.72182 & 0.899625 & 0.718942 & 0.00349575 & 0.694361 \\ 0.473184 & 0.239022 & 0.555218 & 0.947915 & 0.719282 & 0.0332098 \end{array} \tag{3.19}$$

Once again, the four-line invariant is shown to be a powerful discriminator between sets of four lines.

Part IV

Quasi-Affine Reconstruction

We have seen that a set of points in 3 dimensions is determined up to projectivity from two views with uncalibrated cameras. It is shown in this section that this result may be improved by distinguishing between points in front of and behind the camera. Any point that lies in an image must lie in front of the camera producing that image. Using this idea, it is shown that the scene is determined from two views up to a more restricted class of mappings known as *quasi-affine transformations*, which are precisely those projectivities that preserve the convex hull of an object of interest. An invariant of quasi-affine transformation known as the *cheiral sequence* of a set of points is defined and it is shown how the cheiral sequence may be computed using two uncalibrated views. As demonstrated theoretically and by experiment the cheiral sequence may distinguish between sets of points that are projectively equivalent. These results lead to necessary and sufficient conditions for a set of corresponding pixels in two images to be realizable as the images of a set of points in 3 dimensions.

Using similar methods, a necessary and sufficient condition is given for the *orientation* of a set of points to be determined by two views. If the perspective centres are not separated from the point set by a plane, then the orientation of the set of points is determined from two views.

Consider a set of points $\{\mathbf{x}_i\}$ lying in a plane in space and let $\{\mathbf{u}_i\}$ and $\{\mathbf{u}'_i\}$ be two images of these points taken with arbitrary uncalibrated perspective (pinhole) cameras. It is well known that the points $\mathbf{u}_i$ and $\mathbf{u}'_i$ are related by a planar projectivity, that is, there exists h a projectivity of the plane such that $h\mathbf{u}_i = \mathbf{u}'_i$ for all i. This fact has been used for the recognition of planar objects. For instance in [59] planar projective invariants were used to define indexing functions allowing look-up of the objects in an object data-base. Since the indexing functions are invariant under plane projectivities, they provide the same value independent of the view of the object.

In a similar way, it has been shown in [12] and [22] that a set of points in 3-dimensions is determined up to a 3-dimensional projectivity by two images taken with uncalibrated cameras. Both these papers give a constructive method for determining the point configuration (up to projectivity). This permits the computation of projective invariants of sets of points seen in two views. An experimental verification of these results has been reported in [21] and is summarized in this section.

The papers just cited make no distinction between points that lie in front of the camera and those that lie behind. The property of a point that specifies that it lies in front of or behind a given camera will be termed the *cheirality* of the point with respect to the camera. This word is derived from the Greek word : $\chi\epsilon\iota\rho$ meaning *hand* or *side*. It is well know that cheirality is valuable in determining scene geometry for calibrated cameras. Longuet-Higgins [42] uses it to distinguish between four different possible scene reconstructions from two views. More recently, Robert and Faugeras ([58]) have used it for the construction of convex hulls of three-dimensional point sets. No systematic treatment of cheirality for uncalibrated cameras has previously appeared, however. Investigation of this phenomenon turns out to be quite fruitful, as is seen in the present section. Cheirality is valuable in distinguishing different point sets in space, especially in allowing projectively equivalent point sets to be distinguished.

Projective transforms have the property of swapping points from the front to the back of the camera. We will say that a transform is cheirality-reversing for a given point if it swaps the point from the front to the back of the camera, or vice-versa. Otherwise it is called cheirality-preserving. The use of the word cheirality differs slightly from the

conventional usage in topology where it refers to local spatial orientation. In topology, a cheirality reversing transform is one that reverses orientation, such as a mapping that takes a point set to its mirror image. For instance, knots that are the same as their mirror image are called *amphicheiral* ([27]). It will be seen in this report that for affine spatial transforms our definition of cheirality-preserving corresponds with the topological definition in that an orientation preserving transformation preserves the front and back of the cameras. For arbitrary projective transforms the two concepts are distinct.

Summary of major results of the section. In Definition 4.4 a class of projectivities called *quasi-affine transformations* is defined, consisting of those that preserve the convex hull of a set of points of interest. Theorem 4.13 strengthens the result of [12, 22] by showing that a 3-dimensional point set is determined up to quasi-affine transformation by its image in two uncalibrated views. This sharpening of the theorem of [12, 22] results from a consideration of the cheirality of the cameras. This result leads naturally to the concept of a quasi-affine reconstruction of a scene, which is one that differs by at most a quasi-affine transformation from the true geometry. A practical algorithm for computing a quasi-affine reconstruction of a scene seen in two (or more) views is given in section 4:7.

Consideration of cheirality leads to a necessary and sufficient condition for a set of image correspondences to be derived as projections of points in a real scene. This is discussed in section 4:5.

In section 4:6 the concept of quasi-affine transformation is applied to orientation of point sets, explaining why some point sets allow two differently oriented quasi-affine reconstructions from two views, whereas some do not. The relationship of this result to human visual perception of 3D scenes is briefly mentioned, noting that the brain is able to reconstruct differently oriented quasi-affine models of a scene.

Sections 4:8 and 4:9 consider the application of cheirality to view synthesis in which a new view of a scene is constructed from a set of given images.

In section 4:10.1 a quasi-affine invariant is defined – the cheiral sequence. In section 4:11 an example of computation of the cheiral sequence for 3D point sets shows that it is useful in distinguishing between non-equivalent point sets. This invariant may be seen as formalizing and extending to three dimensions the thesis and paper of Morin [50, 51] on distinguishing planar shapes.

4:1 Notation

In this section we will consider object space to be the 3-dimensional Euclidean space R^3 and represent points in R^3 as 3-vectors. Similarly, image space is the 2-dimensional Euclidean space R^2 and points are represented as 2-vectors. Euclidean space, R^3 is embedded in a natural way in projective 3-space $\mathcal{P}^3$by the addition of a plane at infinity. Similarly, R^2 may be embedded in the projective 2-space $\mathcal{P}^2$by the addition of a line at infinity. The simplicity of considering projections between $\mathcal{P}^3$and $\mathcal{P}^2$has makes it convenient in many instances to identify $\mathcal{P}^3$and $\mathcal{P}^2$as the object and images space. This sometimes leads one to forget that real points and cameras lie in Euclidean and not in projective space. Where convenient we will consider points in R^2 and R^3 as lying in

$\mathcal{P}^2$and $\mathcal{P}^3$respectively, via the natural embedding. However, in this case the line (or plane) at infinity will be considered to be a special distinguished line (or plane).

Vectors will be represented as usual as bold-face lower case letters, such as $\mathbf{x}$. The notation $\tilde{\mathbf{x}}$ represents a non-homogeneous 3-vector representing an element of R^3. Similarly, $\tilde{\mathbf{u}}$ is a non-homogeneous 2-vector. The notation $\hat{\mathbf{x}}$ represents a vector with final coordinate equal to 1, sometimes meant implicitly to represent the same point as a homogeneous vector $\mathbf{x}$. Similarly $\hat{\mathbf{u}}$ represents a vector of the form $(u, v, 1)^\top$.

The notation $a \doteq b$ means that a and b have the same sign. For instance $a \doteq 1$ has the same meaning as $a > 0$.

4:2 Projections in $\mathcal{P}^3$

We recall some facts from from section 6:5.1. A projection from $\mathcal{P}^3$into $\mathcal{P}^2$is represented as usual by a 3×4 matrix P, whereby a point $\mathbf{x}$ maps to the point $\mathbf{u} = P\mathbf{x}$. If P is a camera transform matrix for a camera with perspective centre not at infinity, then P can be written as $P = [M \mid -M\tilde{\mathbf{c}}]$ where M is a non-singular 3×3 matrix and $\tilde{\mathbf{c}}$ represents the perspective centre in R^3.

There exist points in $\mathcal{P}^3$that are mapped to points at infinity in the image. To find what they are, we suppose that $\mathbf{u} = (u, v, 0)^\top = P\mathbf{x}$. Letting ${\mathbf{p}_1}^\top$, ${\mathbf{p}_2}^\top$ and ${\mathbf{p}_3}^\top$ be the rows of P, we see that ${\mathbf{p}_3}^\top\mathbf{x} = 0$. In other words, a point $\mathbf{x}$ in $\mathcal{P}^3$that maps to a point at infinity in the image must satisfy the equation $\mathbf{x}^\top\mathbf{p}_3 = 0$. Looked at another way, if $\mathbf{p}_3$ is taken as representing a plane in $\mathcal{P}^3$, then it represents the plane consisting of all points mapping to infinity in the image. Since $P\mathbf{c} = 0$, we see in particular that ${\mathbf{p}_3}^\top\mathbf{c} = 0$ and so $\mathbf{c}$ lies on the plane $\mathbf{p}_3$. To summarize this paragraph, the set of points in $\mathcal{P}^3$mapping to a point at infinity in the image is a plane passing through the perspective centre and represented by $\mathbf{p}_3$, where ${\mathbf{p}_3}^\top$ is the last row of P. In conformity with standard terminology, this plane will be called the *principal plane* of the camera.

Restricting now to R^3, consider a point $\mathbf{x}$ in space, not lying on the principal plane. It is projected by the camera with matrix P onto a point $\mathbf{u}$ where $w\hat{\mathbf{u}} = P\hat{\mathbf{x}}$ for some scale factor w. The value of w will vary continuously with $\mathbf{x}$ and the set of points where it vanishes is precisely the principal plane. It follows that on one side of the principal plane $w > 0$ and on the other side, $w < 0$.

In a Euclidean context, the value of w can be given a precise metric interpretation as explained next. The line perpendicular to the principal plane through the perspective centre is called the *principal ray*. In general, the normal vector to a plane $(q, r, s, t)^\top$ is given in non-homogeneous coordinates as the vector $(q, r, s)^\top$. Thus, if $P = [M \mid -M\tilde{\mathbf{c}}]$, then the principal ray is represented by the last row of M, denoted ${\mathbf{m}_3}^\top$.

For a point $\mathbf{x}$ in space, we see that

$$\begin{aligned} w\hat{\mathbf{u}} &= P\hat{\mathbf{x}} \\ &= [M \mid -M\tilde{\mathbf{c}}]\begin{pmatrix} \tilde{\mathbf{x}} \\ 1 \end{pmatrix} \\ &= M\tilde{\mathbf{x}} - M\tilde{\mathbf{c}} \\ &= M(\tilde{\mathbf{x}} - \tilde{\mathbf{c}}) \ , \end{aligned}$$

and so $w = {\mathbf{m}_3}^\top(\tilde{\mathbf{x}} - \tilde{\mathbf{c}})$ As just remarked, $\mathbf{m}_3$ represents the direction of the principal

ray, and $\tilde{\mathbf{x}} - \tilde{\mathbf{c}}$ is the vector from the camera centre to the point $\mathbf{x}$. If P is scaled by multiplication by an appropriate factor so that $||\mathbf{m}_3|| = 1$ then, w is equal to the depth of the point $\mathbf{x}$ from the camera perspective centre in the direction of the principal ray. This metric interpretation of w, though useful in some applications, such as depth recovery ([69]) will not be used further in this report.

Any real camera can only view points on one side of the principal plane, those points that are "in front of" the camera. Points on the other side will not be visible. In order to distinguish the front of the camera from the back, a convention is necessary.

Definition 4.1. A camera matrix $P = [M \mid \mathbf{v}]$ is said to be *normalized* if $\det(M) > 0$. If P is a normalized camera matrix, a point $\mathbf{x}$ is said to lie *in front of* the camera if $P\hat{\mathbf{x}} = w\hat{\mathbf{u}}$ with $w > 0$. Points $\mathbf{x}$ for which $w < 0$ are said to be *behind* the camera.

Any camera matrix may be normalized by multiplying it by -1 if necessary. The selection of which side of the camera is the front is simply a convention, consistent with the assumption that for a camera with matrix $[I \mid 0]$, points with positive z-coordinate lie in front of the camera. This is the usual convention in computer vision literature, used for instance in [42].

To avoid having always to deal with normalized camera matrices, we define the following parameter χ.

Definition 4.2. Suppose a point $\mathbf{x} = (x, y, z, t)^\top$ maps to an image point $\mathbf{u} = (u, v, w)^\top$ by a camera with matrix $P = [M \mid \mathbf{v}]$. Thus, let $(u, v, w)^\top = P(x, y, z, t)^\top$. We define

$$\chi(\mathbf{x}; P) = (\det M)^{1/3} t/w$$

□

Note that the value of χ is unchanged if the point $\mathbf{x}$ is multiplied by a non-zero scale, since the value of w is multiplied by the same scale. Similarly, if P is multiplied by a constant scale k, then both $\det M^{1/3}$ and w are multiplied by k, and the value of χ is unchanged. Thus, $\chi(\mathbf{x}; P)$ is independent of the particular homogeneous representation of $\mathbf{x}$ and P. If P is normalized and $t = 1$ so that $\mathbf{x} = \hat{\mathbf{x}}$, then $\chi(\mathbf{x}; P) \doteq w$. Thus, corresponding to Definition 4.1 we have

Proposition 4.3. *The point* $\mathbf{x}$ *lies in front of the camera* P *if and only if* $\chi(\mathbf{x}; P) > 0$.

In fact, χ is positive for points in front of the camera, negative for points behind the camera, zero on the plane at infinity and infinite on the principal plane of the camera. If the camera centre or the point $\mathbf{x}$ is at infinity, then χ is still defined but is equal to zero. In this case, it is not well defined whether the point lies behind or in front of the camera.

In general, we will only be concerned with the sign of χ and not its magnitude. We may then write $\chi(\mathbf{x}; P) \doteq t \det M/w$ (remember that the symbol $\doteq$ indicates equality of sign). The quantity $\text{sign}(\chi(\mathbf{x}; P))$ will be referred to as the *cheirality* of the point $\mathbf{x}$ with respect to the camera P. The cheirality of a point is said to be reversed by a transformation if it is swapped from 1 to -1 or vice versa.

Note on the figures. In the figures included in this section, a non-standard representation of cameras is used. A camera is denoted by a line representing its principal plane, along with an arrow pointing in the direction of the front of the camera. The tail of the arrow lies at the centre of projection, on the principal plane. Generally, the figures contain one or two cameras. The diagrams may be thought of as representing the projection of R^3 along the direction of the common line of intersection of the two cameras' principal planes. Thus, each principal plane projects to a line, and their line of intersection projects to a point.

4:3 Quasi-Affine Transformations

A subset B of R^n is called convex if the line joining any two points in B also lies entirely within B. The convex hull of B, denoted $\bar{B}$ is the smallest convex set containing B. We denote by L_∞ the $(n-1)$-dimensional subspace (line, plane, etc) at infinity in $\mathcal{P}^n$. For simplicity, we will refer to it as the plane at infinity, except where we are specifically considering $\mathcal{P}^2$. The inverse image of L_∞ under a projective transformation h is denoted $\pi_\infty = h^{-1}(L_\infty)$.

Definition 4.4. Let B be a subset of R^n and let h be a projectivity of $\mathcal{P}^n$. The projectivity h is said to be "quasi-affine" with respect to the set B if $h^{-1}(L_\infty)$ does not meet $\bar{B}$, where L_∞ is the plane at infinity.

A projectivity that is quasi-affine with respect to B is precisely one that preserves the convex hull of B (as will be seen later).

It may be verified that if h is quasi-affine with respect to B, then h^{-1} is quasi-affine with respect to $h(B)$. Furthermore, if h is quasi-affine with respect to B and g is quasi-affine with respect to $h(B)$, then $g \circ h$ is quasi-affine with respect to B. Thus, quasi-affine projectivities may be composed in this fashion. Strictly speaking, however, quasi-affine projectivities with respect to a given fixed set of points do not form a group.

We will be considering sets of points $\{\mathbf{x}_i\}$ and $\{\mathbf{x}'_i\}$ that correspond via a projectivity. When we speak of the projectivity being *quasi-affine*, we will mean with respect to the set $\{\mathbf{x}_i\}$.

An alternative characterization of quasi-affine transformations is given in the following theorem.

Theorem 4.5. *A projectivity $h : \mathcal{P}^n \to \mathcal{P}^n$ represented by a matrix H is quasi-affine with respect to a set $B = \{\mathbf{x}_i\} \subset R^n - h^{-1}(L_\infty)$ if an only if there exist constants w_i,* all of the same sign*, such that $H\hat{\mathbf{x}}_i = w_i\hat{\mathbf{x}}'_i$*

Proof. To prove the forward implication, we assume that h is a quasi-affine transformation. By definition, constants w_i exist such that $H\hat{\mathbf{x}}_i = w_i\hat{\mathbf{x}}'_i$. What needs proof is that they all have the same sign. The value of w in the mapping $w\hat{\mathbf{x}}' = H\hat{\mathbf{x}}$ is a continuous function of the point $\mathbf{x}$. If $w_i > 0$ for some point $\mathbf{x}_i$, and $w_j < 0$ for another point $\mathbf{x}_j$, then there must exist some point $\mathbf{x}_\infty$ on the line segment joining $\mathbf{x}_i$ to $\mathbf{x}_j$ for which $w = 0$. This means that $\mathbf{x}_\infty$ lies in $\bar{B}$, but $h(\mathbf{x}_\infty)$ lies on the line at infinity, contrary to hypothesis.

Next, to prove the converse, we assume that there exist such constants w_i all of the same sign. We need to show that $h^{-1}(L_\infty)$ does not meet $\bar{B}$. Let S be the subset of R^n consisting of all points $\mathbf{x}$ satisfying the condition $H\hat{\mathbf{x}} = w\hat{\mathbf{x}}'$ such that w has the same sign as all w_i. The set S contains B and it is clear that $S \cap h^{-1}(L_\infty) = \emptyset$. All that remains to show is that S is convex, for then S must contain the convex hull of B. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are points in S with corresponding constants w_i and w_j, then any intermediate point $\mathbf{x}$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ must have w value intermediate between w_i and w_j. To see this, consider a point $\hat{\mathbf{x}} = \alpha\hat{\mathbf{x}}_i + (1-\alpha)\hat{\mathbf{x}}_j$ where $0 \leq \alpha \leq 1$. This point lies between $\mathbf{x}_i$ and $\mathbf{x}_j$. Denote by ${\mathbf{h}_4}^\top$ the last row of H. Then,

$$\begin{aligned} w &= {\mathbf{h}_4}^\top \hat{\mathbf{x}} \\ &= {\mathbf{h}_4}^\top(\alpha\hat{\mathbf{x}}_i + (1-\alpha)\hat{\mathbf{x}}_j) \\ &= \alpha{\mathbf{h}_4}^\top\hat{\mathbf{x}}_i + (1-\alpha){\mathbf{h}_4}^\top\hat{\mathbf{x}}_j \\ &= \alpha w_i + (1-\alpha)w_j \end{aligned}$$

which lies between w_i and w_j as claimed. Consequently, the value of w must have the same sign as w_i and w_j, and so $\mathbf{x}$ lies in S also. This completes the proof. □

This theorem gives an effective method of identifying quasi-affine mappings. The question remains whether quasi-affine mappings form a useful class. This question will be answered by the following theorem.

Theorem 4.6. *If B is a point set in a plane (the "object plane") in R^3 and B lies entirely in front of a projective camera, then the mapping from the object plane to the image plane defined by the camera is quasi-affine with respect to B.*

Proof. That there is a projectivity h mapping the object plane to the image plane is well known. What is to be proven is that the projectivity is quasi-affine with respect to B. Let L be the line in which the principal plane of the camera meets the object plane. Since B lies entirely in front of the camera, L does not meet the convex hull of B. However, by definition of the principal plane $L = h^{-1}(L_\infty)$, where L_∞ is the line at infinity in the image plane. Therefore, h is a quasi-affine with respect to B. □

As an example to illustrate the difference between projective and quasi affine mapping, consider Fig. 4.1. This figure shows an image of a comb and the image resampled according to a projective mapping. Most people will agree that the resampled image is unlike any view of a comb seen by camera or human eye. Nevertheless, the two images are projectively equivalent and will have the same projective invariants. The projective mapping is **not**, however, quasi-affine with respect to the comb.

Note that if points $\mathbf{u}_i$ are visible in an image, then the corresponding object points must lie in front of the camera. Applying Theorem 4.6 to a sequence of imaging operations (for instance, a picture of a picture of a picture, etc), it follows that the original and final images in the sequence are connected by a planar projectivity which is quasi-affine with respect to any point set in the object plane visible in the final image.

Similarly, if two images are taken of a set of points $\{\mathbf{x}_i\}$ in a plane, $\{\mathbf{u}_i\}$ and $\{\mathbf{u}'_i\}$ being corresponding points in the two images, then there is a quasi-affine mapping (with respect to the $\mathbf{u}_i$) mapping each $\mathbf{u}_i$ to $\mathbf{u}'_i$, and so Theorem 4.5 applies, yielding the following proposition.

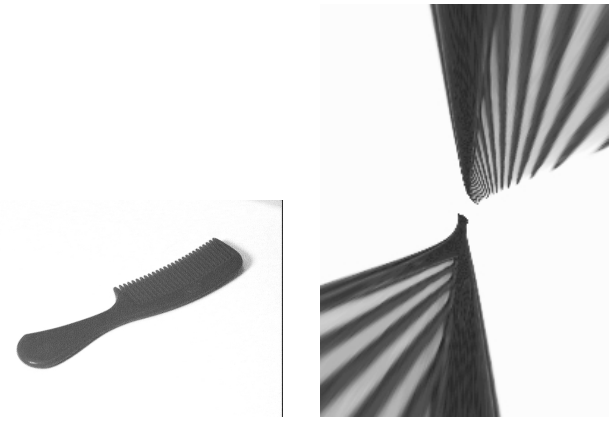


Figure 4.1: *Picture of a comb and a non-quasi-affine resampling of the comb*

Proposition 4.7. *If* $\{\mathbf{u}_i\}$ *and* $\{\mathbf{u}'_i\}$ *are corresponding points in two views of a set of object points* $\{\mathbf{x}_i\}$ *lying in a plane, then there is a matrix* H *representing a planar projectivity such that* $H\hat{\mathbf{u}}_i = w_i\hat{\mathbf{u}}'_i$ *and all* w_i *have the same sign.*

This fact was previously discovered and exploited by Andrew Zisserman and Charles Rothwell (private communication) and served as a starting point for the current investigation. They derived this result using the methods of [69].

4:4 Three dimensional point sets

We now consider three-dimensional point sets seen in a pair of images. The 3D locations of the points will be assumed unknown, but image point matches $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ will be known. It will be assumed that sufficiently many point matches knowf for the matrix F to be determined unambiguously, that is at least 8 points ([42]). Under these conditions as shown in [22] and [12] it is possible to determine the location of points $\mathbf{x}_i$ and cameras P and P' such that $\mathbf{u}_i = P\mathbf{x}_i$ and $\mathbf{u}'_i = P'\mathbf{x}_i$, and furthermore, the choice is unique up to projectivity of $\mathcal{P}^3$. Recalling the definition of χ (definition 4.2) and Proposition 4.3, if $\chi(\mathbf{x}_i; P)$ and $\chi(\mathbf{x}_i; P')$ are both positive, then the point $\mathbf{x}_i$ lies in front of both cameras, and maps to points $\mathbf{u}_i$ and $\mathbf{u}'_i$ in the two images. Normally, this will not be the case. It is possible, however, that another choice of P, P' and $\mathbf{x}_i$ exists with the desired property.

We introduce some new terminology. A triplet $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ is called an *epipolar configuration* if F is a rank 2 matrix satisfying the epipolar constraint equation $\mathbf{u}_i'^{\top} F\mathbf{u}_i = 0$ for all i. A *weak realization* of $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ is a triplet $(P, P', \{\mathbf{x}_i\})$, where P and P' are a choice of camera matrices corresponding to the fundamental matrix F and the points $\{\mathbf{x}_i\}$ are object points satisfying the equations $\mathbf{u}_i = P\mathbf{x}_i$ and $\mathbf{u}'_i = P'\mathbf{x}_i$ for each i. A *strong realization* is such a triplet satisfying the additional condition that $\chi(\mathbf{x}_i; P) > 0$ and $\chi(\mathbf{x}_i; P) > 0$ for all i. This condition implies that the points and the camera centres are at finite points. The triplet $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ is called a *feasible configuration* if a strong realization exists. The purpose of considering epipolar configurations, rather than simply a set of point correspondences $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ is to avoid the problem of having insufficiently many points, or critical configurations of points that make unique determination of the

fundamental matrix impossible. The fundamental matrix will be assumed known. Another common terminology that expresses the same thing is that the cameras are "weakly calibrated".

At this point, it is desirable to derive a slightly different form of the definition of the function χ defined in Definition 4.2. In this definition, and henceforth, we allow the possibility that the camera is located at infinity. Let P be a camera matrix. The centre of P is the unique point $\mathbf{c}$ such that $P\mathbf{c} = 0$. One can write an explicit formula for $\mathbf{c}$ as follows.

Definition 4.8. Given a camera matrix P, we define ${\mathbf{c}_P}^\top$ to be the vector (c_1, c_2, c_3, c_4), where

$$c_i = (-1)^i \det \hat{P}^{(i)}$$

and $\hat{P}^{(i)}$ is the matrix obtained by removing the i-th column of P. □

For convenience of typesetting, we introduce the notation $(P/\mathbf{v}^\top)$ to represent a 4×4 matrix made up of a 3×4 camera matrix P augmented with an final row $\mathbf{v}^\top$. Definition 4.8 leads to a simple formula for $\det(P/\mathbf{v}^\top)$. Cofactor expansion of the determinant along the last row gives $\det(P/\mathbf{v}^\top) = \mathbf{v}^\top \mathbf{c}_P$ for any row vector $\mathbf{v}^\top$. As a special case, if ${\mathbf{p}_i}^\top$ is the i-th row of P, then

$${\mathbf{p}_i}^\top \mathbf{c}_P = \det(P/{\mathbf{p}_i}^\top) = 0$$

where the last equality is true because the matrix has a repeated row. Since this is true for all i, it follows that $P\mathbf{c}_P = 0$, and so $\mathbf{c}_P$ is the camera centre, as claimed.

Note that submatrix $\hat{P}^{(4)}$ is the same as matrix M in the decomposition $P = [M \mid \mathbf{v}]$, and so $\det M = c_4$. This allows us to reformulate the definition of χ as given in Definition 4.2, as follows.

$$\chi(\mathbf{x}; P) \doteq ({\mathbf{e}_4}^\top \mathbf{x})({\mathbf{e}_4}^\top \mathbf{c})/w \tag{4.1}$$

where $\mathbf{c} = \mathbf{c}_P$ as defined in Definition 4.8, and ${\mathbf{e}_4}^\top$ is the vector $(0, 0, 0, 1)$. It is significant to note here that $\mathbf{e}_4$ is the vector representing the plane at infinity – a point $\mathbf{x}$ lies on the plane at infinity if and only if ${\mathbf{e}_4}^\top \mathbf{x} = 0$.

4:4.1 Effect of Transformations on Cheirality

We now consider a projective transformation represented by matrix H. Writing $P' = PH^{-1}$ and $\mathbf{x}' = H\mathbf{x}$ one sees that $P\mathbf{x} = P'\mathbf{x}'$. So if $\mathbf{u} = P\mathbf{x}$ then $\mathbf{u} = P'\mathbf{x}'$. Thus, the image correspondences are preserved by this transformation. When speaking of a projective transformation being applied to a set of points and to a camera, it is meant that a point $\mathbf{x}$ is transformed to $H\mathbf{x}$ and the camera matrix is transformed to PH^{-1}.

In this section we will consider such projective transformations and their effect on the cheirality of points with respect to a camera. First, we wish to determine what happens to $\mathbf{c}_P$ when P is transformed to PH^{-1}. To answer that question, consider an arbitrary 4-vector $\mathbf{v}$. We see that

$$\mathbf{v}^\top H^{-1} \mathbf{c}_{PH^{-1}} = \det(PH^{-1}/\mathbf{v}^\top H^{-1}) = \det(P/\mathbf{v}^\top) \det H^{-1} = \mathbf{v}^\top \mathbf{c}_P \det H^{-1} \ .$$

Since this is true for all vectors $\mathbf{v}$, it follows that $H^{-1}\mathbf{c}_{PH^{-1}} = \mathbf{c}_P \det H^{-1}$, or

$$\mathbf{c}_{PH^{-1}} = H\mathbf{c}_P \det H^{-1} \tag{4.2}$$

At one level, this formula is saying that the transformation H takes the camera centre $\mathbf{c} = \mathbf{c}_P$ to the new location $\mathbf{c}_{PH^{-1}} \approx H\mathbf{c}$. However, we are interested in the exact coordinates of $\mathbf{c}_{PH^{-1}}$ especially the sign of the last coordinate c_4 which appears in the formula (4.1). Thus, the factor H^{-1} is significant.

Now, applying (4.2) to (4.1) gives

$$\begin{aligned}\chi(H\mathbf{x}; PH^{-1}) &\doteq (\mathbf{e}_4{}^\top H\mathbf{x})(\mathbf{e}_4{}^\top \mathbf{c}_{PH^{-1}})/w \\ &\doteq (\mathbf{e}_4{}^\top H\mathbf{x})(\mathbf{e}_4{}^\top H\mathbf{c}) \det H^{-1}/w\end{aligned}$$

where $\mathbf{c} = \mathbf{c}_P$. Finally, denoting the fourth row of the transformation matrix H by $\mathbf{h}_4{}^\top$, and $\text{sign}(\det H)$ by δ, we obtain

$$\chi(H\mathbf{x}; PH^{-1}) \doteq \delta(\mathbf{h}_4{}^\top \mathbf{x})(\mathbf{h}_4{}^\top \mathbf{c})/w \ . \tag{4.3}$$

This equation will be used extensively. Note that it may be considered to be a generalization of (4.1) as will now be explained. A point $\mathbf{x}$ is mapped to the plane at infinity by H if and only if $\mathbf{h}_4{}^\top \mathbf{x} = 0$. Interpreting $\mathbf{h}_4$ as the coordinates of a plane, this condition means that $\mathbf{h}_4$ represents the plane mapped to infinity by H. The factor $\delta \doteq \det H^{-1}$ represents the change of spatial orientation effected by the transformation H, in that H is orientation-preserving if $\det H > 0$ and orientation-reversing if $\det H < 0$. This point will be explained more fully in section 4:6. Thus, the terms in (4.3) may be interpreted as follows : $\mathbf{x}$ are the point coordinates; $\mathbf{c}$ are the coordinates of the camera centre, as in Definition 4.8; $\mathbf{h}_4$ are the coordinates of the plane at infinity and δ is the spatial orientation. Compare this with (4.1) in which $\mathbf{e}_4$ represents the plane at infinity.

We now consider the effect of different transformations on the cheirality of points with respect to a camera. An affine transformation is one represented by a matrix H for which $\mathbf{h}_4{}^\top = \mathbf{e}_4{}^\top = (0, 0, 0, 1)$. The effect of an affine transformation may now be described.

Proposition 4.9. *An affine transformation with positive determinant preserves the cheirality of any point with respect to a camera. An affine transformation with negative determinant reverses cheirality.*

Proof. From (4.1) and (4.3) we see that $\chi(\mathbf{x}; P) \doteq \chi(H\mathbf{x}; PH^{-1}) \det H$ from which the result follows. □

We now determine how an arbitrary projective transformation affects cheirality.

Proposition 4.10. *Let H represent a projective transformation with positive determinant, and let π_∞ be the plane in space mapped to infinity by H. The cheirality of a point $\mathbf{x}$ is preserved by H if and only if $\mathbf{x}$ lies on the same side of the plane π_∞ as the camera centre.*

Proof. Since $\det H > 0$, we see from (4.1) and (4.3) that $\chi(\mathbf{x}; P) \doteq \chi(H\mathbf{x}; PH^{-1})$ if and only if $(\mathbf{h}_4{}^\top \mathbf{x})(\mathbf{h}_4{}^\top \mathbf{c}) \doteq (\mathbf{e}_4{}^\top \mathbf{x})(\mathbf{e}_4{}^\top \mathbf{c})$. Suppose the point $\mathbf{x}$ and the camera P are located at finite points so that the cheirality is well defined, and let them be scaled so that $\mathbf{x} = \hat{\mathbf{x}}$ and $\mathbf{c} = \hat{\mathbf{c}}$. In this case, $(\mathbf{e}_4{}^\top \mathbf{x})(\mathbf{e}_4{}^\top \mathbf{c}) = 1$ and we see that cheirality is preserved, if and only if $(\mathbf{h}_4{}^\top \hat{\mathbf{x}})(\mathbf{h}_4{}^\top \hat{\mathbf{c}}) \doteq 1$, or otherwise expressed $\mathbf{h}_4{}^\top \hat{\mathbf{x}} \doteq \mathbf{h}_4{}^\top \hat{\mathbf{c}}$. Since $\mathbf{h}_4$ represents the plane π_∞, this condition may be interpreted as meaning that the points $\mathbf{c}$ and $\mathbf{x}$ both lie on the same side of the plane π_∞. Hence, the cheirality of a point $\mathbf{x}$ is preserved by a transformation H, if and only if it lies on the same side of the plane π_∞ as the camera centre. □

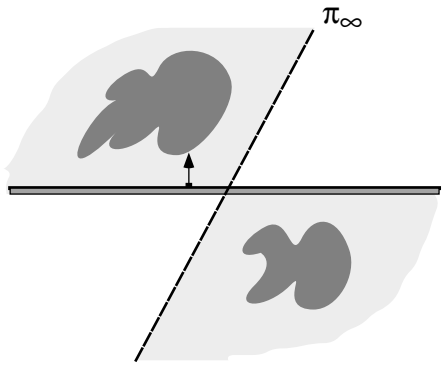


Figure 4.2: **Effect of a projective transform with positive determinant.** *The principal plane of the camera and the plane π_∞ divide R^3 into four segments. One pair of opposite segments (shown shaded) are transformed to points in front of the camera. The opposite pair of segments are transformed to points behind the camera. In the local neighbourhood of the camera centre the front and back of the camera are preserved. This consideration determines which pair of segments become the front of the camera. Thus the two dark shaded sets of points lie in front of the camera after transformation. For a transform with negative determinant the opposite pair of segments become the front of the camera.*

Points $\mathbf{x}$ close to the camera centre will lie on the same side of π_∞ as the camera centre, and hence, their cheirality will be preserved. Thus, Proposition 4.10 implies that cheirality is preserved in a local neighbourhood of the camera centre. This is illustrated in Fig 4.2.

4:4.2 Quasi-affine invariance of strong realizations

For planar object sets, Theorem 4.6 established the existence of a quasi-affine mapping between the object plane and the image plane. For *non-planar* objects seen in two views, strong realizations of the epipolar configuration take the rôle played by sets of image points in the two dimensional case.

Theorem 4.11. *Let $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ be an epipolar configuration and let $(P, P', \{\mathbf{x}_i\})$ and $(\bar{P}, \bar{P}', \{\bar{\mathbf{x}}_i\})$ be two separate strong realizations of the configuration. Then the projectivity h mapping each point $\mathbf{x}_i$ to $\bar{\mathbf{x}}_i$ is quasi-affine.*

Proof. If the projectivity is not quasi-affine, then there are points on both sides of $\pi_\infty = h^{-1}(L_\infty)$. Since h preserves the cheirality of points lying on only one side of π_∞ it follows that h does not preserve the cheirality of all points, Therefore at least one of the realizations can not be a strong realization, and so the hypothesis that h is not quasi-affine is not tenable. □

The particular case where one of the two realizations is the "correct" realization is of interest. It is the analogue in three dimensions of Proposition 4.6.

Corollary 4.12. *If $\{\mathbf{x}_i\}$ are points in R^3, image coordinates $\{\mathbf{u}_i\}$ and $\{\mathbf{u}'_i\}$ are corresponding image points in two uncalibrated views from which the fundamental matrix F is determined uniquely, and $(P, P', \{\bar{\mathbf{x}}_i\})$ is a strong realization of the triple $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$, then there is a quasi-affine mapping taking each $\mathbf{x}_i$ to $\bar{\mathbf{x}}_i$.*

From this corollary, we can deduce one of the main results of this section.

Theorem 4.13. *Let $(P, P', \{\mathbf{x}_i\})$ and $(\bar{P}, \bar{P}', \{\bar{\mathbf{x}}_i\})$ be two different reconstructions of 3D scene geometry derived as strong realizations of possibly different epipolar configurations corresponding to possibly different pairs of images of a 3D point set. Then there is a quasi-affine transformation mapping each point $\mathbf{x}_i$ to $\bar{\mathbf{x}}_i$.*

What this theorem is saying is that if a point set in R^3 is reconstructed as a strong realization from two separate pairs of views, then the two results are the same up to a quasi-affine transformation.

Proof. By corollary 4.12 there exist quasi-affine transformations mapping each of the sets of reconstructed points $\{\mathbf{x}_i\}$ and $\{\bar{\mathbf{x}}_i\}$ to the actual 3D locations of the points. The result follows by composing one of these projectivities with the inverse of the other. □

4:5 When are a Set of Image Correspondences Realizable ?

Given a set of image correspondences $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ one may ask under what conditions these correspondences may arise from projection of points in a real scene into the two images. A well known constraint is the epipolar constraint $\mathbf{u}_i'^{\top} F \mathbf{u}_i = 0$ for some rank-2 matrix, the fundamental matrix. It is shown here that that condition is not sufficient, and a necessary and sufficient condition will be given.

As usual, we avoid the problem of critical point configurations, or insufficiently many point correspondences by assuming that the images are "weakly calibrated" meaning that the fundamental matrix is given. In the terminology already introduced, we assume that we have an epipolar configuration $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$. It has been shown in [22, 12] that a realization $(P, P', \{\mathbf{x}_i\})$ of this configuration exists, and that further, all realizations may be reached from this realization by applying a projective transformation.

Given a realization $(P, P', \{\mathbf{x}_i\})$ we write $P\mathbf{x}_i = w_i \hat{\mathbf{u}}_i$ and $P'\mathbf{x}_i = w'_i \hat{\mathbf{u}}'_i$. Suppose that there is a transformation H that transforms this to a strong realization. This means that $\chi(H\mathbf{x}_i; PH^{-1}) > 0$ and $\chi(H\mathbf{x}_i; P'H^{-1}) > 0$ for all i, from which it follows that $\chi(H\mathbf{x}_i; PH^{-1}) \doteq \chi(H\mathbf{x}_i; P'H^{-1})$ for all i. Substituting the formula (4.3) gives

$$(\mathbf{h}_4{}^{\top}\mathbf{x}_i)(\mathbf{h}_4{}^{\top}\mathbf{c})\delta/w_i \doteq (\mathbf{h}_4{}^{\top}\mathbf{x}_i)(\mathbf{h}_4{}^{\top}\mathbf{c}')\delta/w'_i \ .$$

Cancelling common terms from both sides gives

$$(\mathbf{h}_4{}^{\top}\mathbf{c})/w_i \doteq (\mathbf{h}_4{}^{\top}\mathbf{c}')/w'_i \ .$$

Now $(\mathbf{h}_4{}^{\top}\mathbf{c})$ and $(\mathbf{h}_4{}^{\top}\mathbf{c}')$ must be non-zero, since $\chi(H\mathbf{x}_i; PH^{-1})$ and $\chi(H\mathbf{x}_i; P'H^{-1})$ are non-zero. Rearranging terms leads to $w_i w'_i \doteq (\mathbf{h}_4{}^{\top}\mathbf{c})(\mathbf{h}_4{}^{\top}\mathbf{c}')$. Since the right side does not depend on i, this means that $w_i w'_i$ has constant sign for all i, which proves the following proposition.

Proposition 4.14. *Let $(P, P', \{\mathbf{x}_i\})$ be a realization of a feasible epipolar configuration. Write $P\mathbf{x}_i = w_i\hat{\mathbf{u}}_i$ and $P'\mathbf{x}_i = w'_i\hat{\mathbf{u}}'_i$. Then $w_iw'_i$ has the same sign for all i.*

Proposition 4.14 has a geometric interpretation as follows. The principal plane of a camera separates R^3 into two regions. For points on one side of the principal plane $P\mathbf{x}_i = w_i\hat{\mathbf{u}}_i$ with $w_i > 0$, whereas on the other side, $w_i < 0$. The two principal planes divide up R^3 into four quadrants. The condition that $\text{sign}(w_iw'_i)$ is constant corresponds to the geometric condition that the points $\mathbf{x}_i$ all lie in a pair of opposite quadrants.

A Sufficient Condition Proposition 4.14 gives a necessary condition for an epipolar configuration to be feasible. It will next be shown that this condition is also sufficient. This will be done by explicitly showing how the weak realization may be transformed to a strong realization. To ensure that this is possible, we need two extra conditions.

Condition 4.15.

1. The image coordinates of the points $\mathbf{x}_i$ as seen by two cameras are bounded.
2. At least one of the camera centres is not a limit point of the point set X.

Since image coordinates are unchanged under transformation, the first condition is independent of the particular weak realization considered. The second condition concerning limit points is unchanged under continuous transformations. Since the transformations we consider are continuous in a neighbourhood of the camera centres, this condition is also independent of the particular weak realization considered. In any reasonable imaging situation, both these conditions will hold. For finite point sets the two conditions are trivially satisfied. For infinite point sets, the image coordinates of the points will still be limited by the extent of the image, so the first condition will hold. For a topologically closed point set, the second condition will hold, since a point that coincides with the camera centre can not be imaged. In general, for arbitrary point sets, it will not normally be the case that the points can lie arbitrarily close to the camera centre.

This condition may be illustrated graphically as in Fig 4.3.

Now, we proceed to transform an arbitrary weak realization into a strong realization. We proceed in steps. As a preliminary step, we need to ensure that neither of the two camera centres lies on the plane at infinity. If this were to occur, then we would choose a new weak realization for which the camera centres do not lie on the plane at infinity.

The principal planes of the two cameras must now meet in a line in space. Consider a plane π_∞ containing that line, but not equal to either of the two principal planes. This plane will be contained in two opposite quadrants of R^3, except where it meets the two principal planes. Let this plane also be chosen so that it passes through the two quadrants of space that do not contain any of the points $\mathbf{x}_i$. This situation is shown in Fig 4.4. In this case the plane π_∞ separates the two point sets X_+ and X_- lying in opposite quadrants of space. Now consider the effect of a transformation mapping the plane π_∞ to infinity. According to Proposition 4.10, the cheirality of one of the two sets X_+ and X_- (with respect to say the first camera) will be reversed and the cheirality of the other will be preserved by this transformation. Since originally X_+ and X_- have opposite cheirality, after the transformation they will have the same cheirality. In other

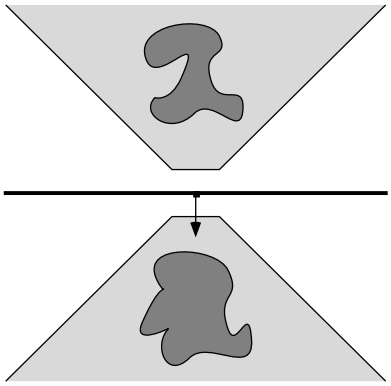


Figure 4.3: *The point set X (dark shading) must lie inside a truncated cone (dark shading). The cone represents the bounding of the image coordinates. The cone is truncated near the camera centre c since points in X can not lie arbitrarily close to the camera centre. In the general case, points may lie both behind and in front of the camera.*

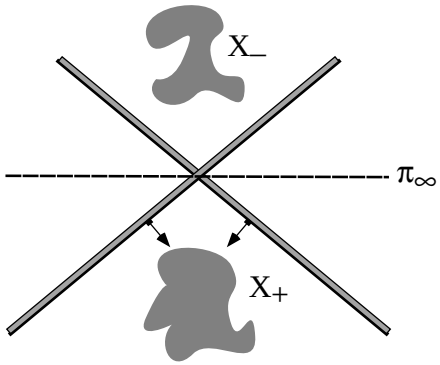


Figure 4.4: **Step 1 of transformation.** *We choose the plane at infinity to pass through the two quadrants that do not contain the point set. After this transformation, all points will lie on one side of each camera.*

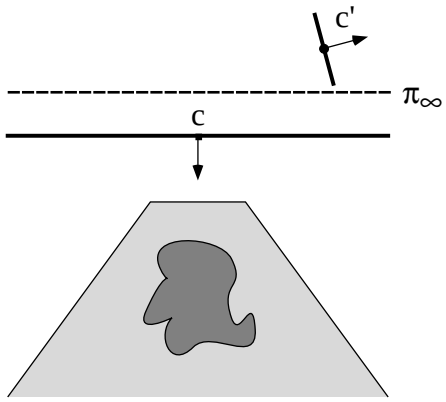


Figure 4.5: **Second camera behind the first camera** *We can separate the two camera centres* $\mathbf{c}$ *and* $\mathbf{c}'$ *with a plane* π_∞ *lying just behind the principal plane of the first camera. Since all the points lie in front of the camera, plane* π_∞ *does not separate the point set* X*.*

words, the whole set $X = X_+ \cup X_-$ will lie on the same side of the first camera. The same argument holds for the other camera.

In invoking Proposition 4.10, it was assumed that neither of the camera centres lay on the line of intersection of the two principal planes, and hence on the plane π_∞ chosen. If this were to occur, then we would choose instead a plane π_∞ slightly displaced from this intersection line but still separating the two sets X_+ and X_-. This is possible since conditions 4.15 ensure that the point set X does not approach the line of intersection of the principal planes.

The case where the two principal planes are identical must also be handled specially. In this case, the plane π_∞ is chosen slightly displaced from the cameras' common principal plane, and separating X_+ from X_-.

If after this first transformation step, the set X lies in front of both cameras, then we are done. If on the other hand it lies behind both cameras, then applying an affine transformation with negative determinant (for instance $H = \mathrm{diag}(-1, -1, -1, 1)$) will swap the set X to the front of both cameras. There remains the possibility that X lies in front of one camera and behind the other.

To handle this remaining case, we need a further transformation. We wish to find a plane π_∞ that separates the two camera centres, but does not separate the point set X. Assuming this is possible, X will then lie on the opposite side of π_∞ from one of the camera centres (but not the other). Now we apply a transformation that takes π_∞ to infinity. According to Proposition 4.10 the cheirality of X will be reversed with respect to one of the cameras, but not the other. Originally the cheirality of X was opposite with respect to the two cameras, and so after the transformation the cheirality will be the same. This means that X will lie on the same side of both cameras. By applying, if necessary, a cheirality-reversing affine transformation it may be assured that X lies in front of both cameras, and we are done.

It remains to explain how the required plane π_∞ is to be found. We suppose that the points X lie in front of the first camera and behind the second camera. We wish to find

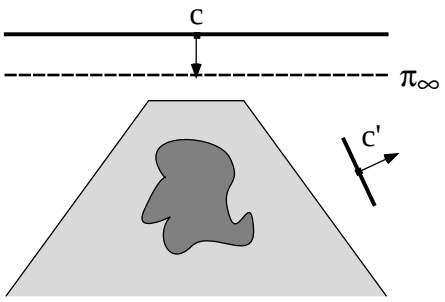


Figure 4.6: **Second camera in front of the first camera** *We can separate the two camera centres* $\mathbf{c}$ *and* $\mathbf{c}'$ *with a plane* π_∞ *lying just in front of the principal plane of the first camera. The point set* X *lies entirely inside the truncated cone (lightly shaded). The plane* π_∞ *can be chosen sufficiently close to* $\mathbf{c}$ *so as not to meet this cone. Consequently, it will not separate the point set* X*.*

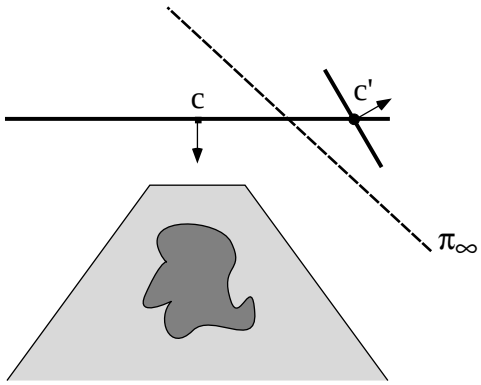


Figure 4.7: **Second camera lies on the principal plane of the first camera.** *We can separate the two camera centres* $\mathbf{c}$ *and* $\mathbf{c}'$ *with an oblique plane* π_∞ *which crosses the principal plane of the first camera. Plane* π_∞ *can be chosen so as not to meet the cone containing* X*, and consequently will not separate* X*.*

a plane that separates the two camera centers, but does not separate the point set X. The method for constructing this plane is given in Figures 4.5, 4.6 and 4.7 corresponding to whether the second camera lies behind, in front of, or on the principal plane of the first camera. Details of the construction are given in the captions of the figures.

We can summarize this discussion in the following theorem.

Theorem 4.16. *Let $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ be an epipolar configuration and let $(P, P', \{\mathbf{x}_i\})$ be a realization of that configuration. Suppose that conditions (4.15) are satisfied. Let $P\mathbf{x}_i = w_i\hat{\mathbf{u}}_i$ and $P'\mathbf{x}_i = w'_i\hat{\mathbf{u}}_i$. Then $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ is a feasible configuration if and only if $w_i w'_i$ has the same sign for all i.*

Since an epipolar configuration always possesses a weak realization ([22]), Theorem 4.16 gives a necessary and sufficient condition for an epipolar configuration to be realizable as a three dimensional scene.

4:6 Orientation

We now consider the question of image orientation. A mapping h from R^n to itself is called orientation-preserving at points $\mathbf{x}$ where the Jacobian of h (the determinant of the matrix of partial derivatives) is positive and orientation-reversing at points where the Jacobian is negative. Reflection of points of R^n with respect to a hyperplane (that is mirror imaging) is an example of an orientation reversing mapping. A projectivity h from $\mathcal{P}^n$ to itself restricts to a mapping from $R^n - h^{-1}(L_\infty)$ to R^n, where L_∞ is the hyperplane (line, plane) at infinity. Consider the case $n = 3$ and let H be a 4×4 matrix representing the projectivity h. We wish to determine at which points $\mathbf{x}$ in $R - h^{-1}(L_\infty)$ the map h is orientation preserving. It may be verified (quite easily using Mathematica [78]) that if $H\hat{\mathbf{x}} = w\hat{\mathbf{x}}'$ and J is the matrix of partial derivatives of h evaluated at $\mathbf{x}$, then $\det(J) = \det(H)/w^4$. This gives the following result.

Proposition 4.17. *A projectivity h of $\mathcal{P}^3$represented by a matrix H is orientation preserving at any point in $R^3 - h^{-1}(L_\infty)$ if and only if* $\det(H) > 0$.

Of course, the concept of orientability may be extended to the whole of $\mathcal{P}^3$, and it may be shown that h is orientation-preserving on the whole of $\mathcal{P}^3$if and only if $\det(H) > 0$. The essential feature here is that as a topological manifold, $\mathcal{P}^3$is orientable. The situation is somewhat different for $\mathcal{P}^2$, which is not orientable as a topological space. In this case, with notation similar to that used above, it may be verified that $\det(J) = \det(H)/w^3$. Therefore, the following proposition is true.

Proposition 4.18. *A projectivity h of $\mathcal{P}^2$is orientation preserving at a point $\mathbf{u}$ in $R^2 - h^{-1}(L_\infty)$ if and only if $w\det(H) > 0$, where $H\hat{\mathbf{u}} = w\hat{\mathbf{u}}'$.*

This theorem allows us to strengthen the statement of Theorem 4.5 somewhat.

Corollary 4.19. *If h is a quasi-affine transformation of $\mathcal{P}^2$with respect to a set of points $\{\mathbf{u}_i\}$ in R^2, then h is either orientation-preserving or orientation-reversing at all points $\mathbf{u}_i$. Suppose the matrix H corresponding to h is normalized to have positive determinant (by possible multiplication by -1) and let $H\hat{\mathbf{u}}_i = w_i\hat{\mathbf{u}}'_i$. Then h is orientation-preserving if and only if $w_i > 0$ for all i.*

An example where Corollary 4.19 applies is in the case where two images of a planar object are taken from the same side of the object plane. In this case, an orientation-preserving quasi-affine projectivity will exist between the two images. Consequently, all the w_i defined with respect to a matrix H will be positive, provided that H is normalized to have positive determinant.

The situation in 3-dimensions is rather more involved and more interesting. Two sets of points $\{\mathbf{x}_i\}$ and $\{\bar{\mathbf{x}}_i\}$ that correspond via a quasi-affine transformation are said to be *oppositely oriented* if the projectivity is orientation-reversing. This definition extends also to two strong realizations $(P, P', \{\mathbf{x}_i\})$ and $(\bar{P}, \bar{P}', \{\bar{\mathbf{x}}_i\})$ of a common epipolar configuration $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$, since in view of Theorem 4.11 the point sets are related via a quasi-affine transformation. Whether or not oppositely oriented strong realizations exist depends on the imaging geometry. Common experience provides some clues here. In particular a stereo pair may be viewed by presenting one image to one eye and the other image to the other eye. If this is done correctly, then the brain perceives a 3-D reconstruction of the scene (a strong realization of the image pair). If, however, the two images are swapped and presented to the opposite eyes, then the perspective will be reversed – hills become valleys and vice versa. In effect, the brain is able to compute two oppositely oriented reconstructions of the image pair. It seems, therefore, that in certain circumstances, two oppositely oriented realizations of an image pair exist. It may be surprising to discover that this is not always the case, as is shown in the following theorem.

Theorem 4.20. *Let $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ be an epipolar configuration and let $(P, P', \{\mathbf{x}_i\})$ be a strong realization of $(F, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$. There exists a different oppositely oriented strong realization $(\bar{P}, \bar{P}', \{\bar{\mathbf{x}}_i\})$ if and only if there exists a plane in R^3 such that the perspective centres of both cameras P and P' lie on one side of the plane, and the points $\mathbf{x}_i$ lie on the other side.*

Proof. Consider one strong realization of the configuration. By definition, all the points lie in front of both cameras. Suppose that there exists a plane separating the points from the two camera centres. Let G be a projective transformation mapping the given plane to infinity and let A be an affine transformation. Suppose further that $\det G > 0$ and $\det A < 0$. Let H be the composition $H = AG$. According to Proposition 4.10 the transformation H is cheirality reversing for the points, since the points are on the opposite side of the plane from the camera centres. According to Proposition 4.9 A is also cheirality reversing, since $\det A < 0$. The composition H must therefore be cheirality preserving, and it transforms the strong configuration to a different strong configuration. Since H has negative determinant, however, it is orientation reversing, so the two strong realizations have opposite orientations.

Conversely, suppose that two strong oppositely oriented realizations exist and let H be the transformation taking one to the other. Since H is orientation reversing, $\det H < 0$. The mapping H is by definition cheirality preserving on all points, with respect to both cameras. If π_∞ is the plane mapped to infinity by H, then according to Propositions 4.10 the points X must lie on the opposite side of the plane π_∞ from both camera centres. □

4:7 The Cheiral Inequalities

Several methods ([12, 22, 48]) have been proposed for computing a projective reconstruction (in our terminology a weak realization) of a scene from a set of point matches. In section 4:5 a constructive method was given for transforming a weak realization into a strong one. That method was not very suitable for computer computation. Accordingly, in this section a straight-forward algorithm will be given for computing a strong realization of an epipolar configuration. This will be done by transforming a weak realization into a strong realization by finding an appropriate transformation.

We start with a weak realization $(P, P', \{\mathbf{x}_i\})$ of an epipolar configuration. Let $w_i\hat{\mathbf{u}}_i = P\mathbf{x}_i$ and $w'_i\hat{\mathbf{u}}_i = P'\mathbf{x}_i$. We assume that $w_i w'_i$ has the same sign for all i. By multiplying the matrix P by -1 if necessary, we may ensure that $w_i w'_i > 0$ for all i. Furthermore, by multiplying $\mathbf{x}_i$ by -1 if necessary, we may ensure that $w_i > 0$ and hence $w'_i > 0$ for all i. We will assume that this has been done.

Now, we seek a transformation H that will transform the weak realization to a strong realization. After this transformation, all points will lie in front of both cameras. According to (4.3) this condition may be written (for camera P)

$$\chi(\mathbf{x}_i; P) \doteq (\mathbf{h}_4{}^\top \mathbf{x}_i)(\mathbf{h}_4{}^\top \mathbf{c})\delta > 0$$

where $\delta = \text{sign}(\det H)$. Similarly, for the other camera, we have

$$\chi(\mathbf{x}_i; P') \doteq (\mathbf{h}_4{}^\top \mathbf{x}_i)(\mathbf{h}_4{}^\top \mathbf{c}')\delta > 0 \ .$$

Since we are free to multiply $\mathbf{h}_4$ by -1 if necessary, we may assume that $(\mathbf{h}_4{}^\top \mathbf{c})\delta > 0$. From this it follows from the first inequality that $\mathbf{h}_4{}^\top \mathbf{x}_i > 0$ for all i. Then, from the second inequality, we have $(\mathbf{h}_4{}^\top \mathbf{c}')\delta > 0$. The total set of inequalities may now be written :

$$\begin{aligned} \mathbf{x}_i{}^\top \mathbf{h}_4 &> 0 \\ \delta \mathbf{c}^\top \mathbf{h}_4 &> 0 \\ \delta \mathbf{c}'^\top \mathbf{h}_4 &> 0 \end{aligned} \tag{4.4}$$

These equations (4.4) may be called the *cheiral inequalities*. Since the values of each $\mathbf{x}_i$, $\mathbf{c}$ and $\mathbf{c}'$ are known, they form a set of inequalities in the entries of $\mathbf{h}_4$. The value of δ is not known *a priori*, and so it is necessary to seek a solution for each of the two cases $\delta = 1$ and $\delta = -1$.

To find the required transformation H, first of all we solve the cheiral inequalities to find a value of $\mathbf{h}_4$, either for $\delta = 1$ or $\delta = -1$. The required matrix H is any matrix having $\mathbf{h}_4{}^\top$ as its last row and satisfying the condition $\det H \doteq \delta$. If the last component of $\mathbf{h}_4$ is non-zero, then H can be chosen to have the simple form in which the first three rows are of the form $\pm[I \mid \mathbf{0}]$.

Theorem 4.16 guarantees that there will be a solution either for $\delta = 1$ or $\delta = -1$. In some cases there will exist solutions of the cheiral inequalities for both $\delta = 1$ and $\delta = -1$. This will mean that two oppositely oriented strong realizations exist. The conditions under which this may occur were discussed in section 4:6.

Solving the Cheiral Inequalities Naturally, the cheiral inequalities may be solved using techniques of linear programming. As they stand however, if $\mathbf{h}_4$ is a solution, then so is $\alpha\mathbf{h}_4$ for any positive factor α. In order to restrict the solution domain to be bounded, one may add additional inequalities. For instance, if $\mathbf{h}_4 = (v_1, v_2, v_3, v_4)^\top$, then the inequalities $-1 < v_i < 1$ serve to restrict the solution domain to be a bounded polyhedron.

To achieve a unique solution we need to specify some goal function to be linearized. An appropriate strategy is to seek to maximize the extent by which each of the inequalities is satisfied. To do this, we introduce one further variable, d. Each of the inequalities $\mathbf{a}^\top\mathbf{h}_4$ of the form (4.4) for appropriate $\mathbf{a}$ is replaced by an inequality $\mathbf{a}^\top\mathbf{h}_4 > d$. We seek to maximize d while satisfying all the inequalities. This is a standard linear programming problem, for which many methods of solution exist, such as the simplex method ([55])[6]. If a solution is found for which $d > 0$ then this will be a desired solution.

4:7.1 Quasi-affine reconstruction

A strong realization of an epipolar configuration is a *quasi affine* reconstruction, since it differs from the true scene by a quasi-affine transformation (Corollary 4.12). Quasi-affine reconstructions of a scene have useful properties such as preservation of complex hull. Furthermore, computing a quasi-affine reconstruction has been used in [29] as a preliminary step towards computing a Euclidean reconstruction of a scene from three views with the same camera. A strong realization of an epipolar reconstruction is a slightly more restrictive than a general quasi-affine reconstruction, however, as will be shown now.

The inequalities (4.4) are seen to be of two types. The first inequality involves the points (one inequality for each i) and the other two involve the camera centres. One sees that if only the first inequality is satisfied (for all i), but possibly not the ones involving the camera centres, then the solution is less constrained. Instead of all points lying in front of both cameras, all points will lie on the same side of each camera. Thus, if $\delta\mathbf{c}^\top{\mathbf{h}_4}^\top < 0$, then all points will lie behind the first camera, since $\chi(\mathbf{x}_i; P) < 0$. Thus, solving the first inequality for all i is equivalent to the first step of the construction given in section 4:5. Adding the other two inequalities as well is equivalent to carrying out the second step of section 4:5. Note now that the transformation carried out in the second step is itself quasi-affine. In fact, referring to Figs 4.5, 4.6 and 4.7 one sees that the plane π_∞ does not separate the point set X. Thus, just by solving the first inequality of (4.4) one obtains a quasi-affine reconstruction of the point set. However, including the two inequalities for the camera locations further constrains the reconstruction to bring it closer to the true Euclidean reconstruction, and so is recommended in most cases.

If one is content with any quasi-affine reconstruction, however, then one can ignore the two last inequalities in (4.4). An example of when this may be sufficient is when one is computing the cheiral sequence of a set of points, to be described in section 4:10. In this case, there is a very simple means of solution. The inequalities that we need to solve are of the form ${\mathbf{h}_4}^\top\mathbf{x}_i > 0$ for all i. Recall that we are assuming that each $w_i > 0$ and $w'_i > 0$. This being so, we see that $w_i = {\mathbf{p}_3}^\top\mathbf{x}_i > 0$, where ${\mathbf{p}_3}^\top$ is the third row of the camera matrix P. Thus, we may choose $\mathbf{h}_4 = \mathbf{p}_3$ as the solution to the inequalities.

[6]The Simplex algorithm given in [55] is not suitable for use as stands, since it makes the unnecessary assumption that all variables are non-negative. It needs to be modified to be used for this problem

More generally, for any α between 0 and 1, we may choose $\mathbf{h}_4 = \alpha\mathbf{p}_3 + (1-\alpha)\mathbf{p}'_3$, where ${\mathbf{p}'_3}^\top$ is the third row of the other camera matrix P'. This corresponds precisely to the construction of Fig 4.4.

In the case where the weak realization is carried out in a way such that $P = [I \mid 0]$ (for instance, see the method of [22]), then we have a very easy way to obtain a quasi-affine reconstruction. In this case we choose $\mathbf{h}_4 = \mathbf{p}_3 = (0, 0, 1, 0)^\top$, and

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} .$$

Such an H simply swaps the two last components of any point $\mathbf{x}_i$, and the last two columns of each camera matrix. This gives a very simple way of computing a quasi-affine reconstruction.

1. Carry out a projective reconstruction of the scene for which the first camera has matrix $P = [I \mid 0]$.

2. Swap the last two coordinates of each point $\mathbf{x}_i$ and the last two columns of each camera matrix.

Quasi-affine reconstruction using the cheiral inequalities or the simple algorithm just given extends naturally to reconstruction from several views. There is no analogue of Theorem 4.16 to ensure a solution in the multi-view case, but of course if the input data is derived from real data of a real scene, then a solution will exist.

4:8 Which Points are Visible in a Third View

Consider a scene reconstructed from two views. We consider now the question of determining which points are visible in a third view. Such a question arises when one is given two uncalibrated views of a scene and one seeks to synthesize a third view. This can be done by carrying out a projective reconstruction of the scene from the first two views and then projecting into the third view. In this case, it is important to determine if a point lies in front of the third camera and is hence visible, or not.

If the third view is given simply by specifying the camera matrix with respect to the frame of reference of some given reconstruction, then it may be impossible to determine whether points are in front of the third camera or behind it in the true scene. The basic ambiguity is illustrated in Fig 4.8.

Knowledge of a single point known to be visible in the third view serves to break the ambiguity, however, as the following proposition shows. By applying Proposition 4.14 to the first and third views, one obtains the following criterion.

Proposition 4.21. *Let points $(P^1, P^2, \{\mathbf{x}_i\})$ be a realization of a set of correspondences $\mathbf{u}_i^1 \leftrightarrow \mathbf{u}_i^2$. Let P^3 be the camera matrix of a third view and suppose that $w_j^i \hat{\mathbf{u}}_i = P^i \mathbf{x}_j$ for $i = 1, \ldots, 3$. Then $w_j^1 w_j^3$ has the same sign for all points $\mathbf{x}_j$ visible in the third view.*

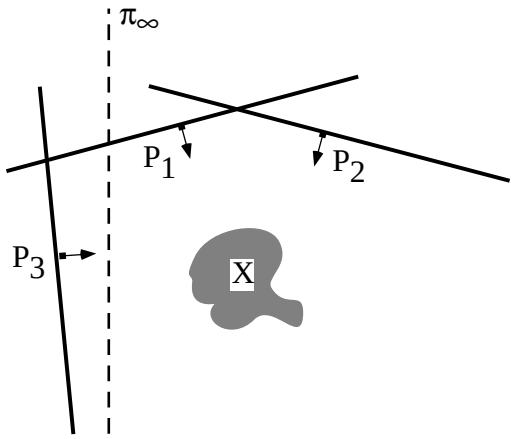


Figure 4.8: **Visibility.** *In the reconstruction as shown, the point set X lies entirely in front of the first two cameras. Thus, this represents a strong realization of the scene with respect to the first two cameras. As shown, the point set X lies in front of the third camera. However, if the configuration is subjected to a projective transformation so that plane π_∞ becomes the plane at infinity, then according to Theorem 4.10 the set X will remain in front of the first two cameras, but will be switched to lie behind the third camera. With no way of knowing where the plane at infinity lies, one can not determine whether X lies in front of or behind the third camera.*

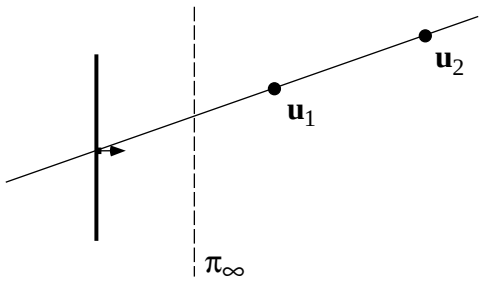


Figure 4.9: **Which points are in front.** *In the reconstruction shown, point* $\mathbf{u}_1$ *is closer to the third camera than* $\mathbf{u}_2$*. If, however, we apply an orientation-reversing projective transformation that maps the plane* π_∞ *to infinity, then the two points will still lie in front of both cameras, but now point* $\mathbf{u}_2$ *will lie closer to the third camera. This is because locally the front and back of the cameras will be reversed by the orientation-reversing transformation. In order to reach* $\mathbf{u}_1$ *from the centre of the third camera, without crossing* π_∞ *it is necessary to pass through* $\mathbf{u}_2$ *first.*

In practice, it will usually be the case that one knows at least on point visible in the third view. For instance, once a projective reconstruction has been carried out using two views, the camera matrix of the third camera may be determined from the images of six or more points by solving directly for the matrix P_3 given the correspondences $\mathbf{u}_i^3 = P_3\mathbf{x}_i$ where points $\mathbf{x}_i$ are the reconstructed points. This may be done by linear means ([71]).

4:9 Which Points are in Front of Which

When we are attempting to synthesize a new view of a scene that has been reconstructed from two or more uncalibrated views it is sometimes necessary to consider the possibility of points being obscured by other points. This leads to the question, given two points that project to the same point in the new view, which one is closer to the camera, and hence obscures the other. In the case where the possibility exists of oppositely oriented quasi-affine reconstructions it may once again be impossible to determine which of a pair of points is closer to the new camera. This is illustrated in Fig 4.9. If a plane exists, separating the camera centres from the point set, then two oppositely oriented reconstructions exist, and one can not determine which points are in front of which.

The sort of ambiguity shown in Fig 4.9 can only occur in the case where there exists a plane π_∞ that separates the camera centres from the set of all visible points. If this is not the case, then one can compute a quasi-affine reconstruction and the problem is easily solved. To avoid the effort of computing a quasi-affine reconstruction, however, we would like to solve this problem using only a projective reconstruction of the scene. How this may be done is explained next.

The parameter χ defined in Definition 4.2 is used to distinguish the front from the back of the camera in a Euclidean or quasi-affine frame. It is also useful for determining which points lie in front of which, as will be seen now. Recall that χ is zero for points $\mathbf{x}$ on the plane at infinity, infinite for points on the principal plane of the camera, positive for points in front of the camera and negative for points behind the camera. Furthermore,

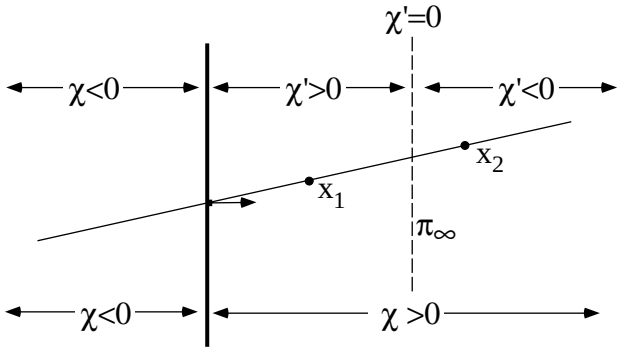


Figure 4.10: **Preservation of order of points.** *This shows the effect of a transformation with positive determinant taking the plane π_∞ to infinity. Both χ (before the transformation) and χ' (after the transformation) decrease monotonically along any ray through the camera centre. We find that $\chi'(\mathbf{x}_1) > \chi'(\mathbf{x}_2)$ if and only if $\chi(\mathbf{x}_1) > \chi(\mathbf{x}_2)$.*

given two points in front of the camera, projecting to the same point in the image , the point with the greater value of χ lies closer to the front of the camera.

The value of χ can be used to parametrize any line in $\mathcal{P}^3$through the camera centre. As one proceeds along the line in the direction of the front of the camera, the value of χ decreases continuously from infinity at the camera centre, through positive values. It reaches zero at the plane at infinity, and continues to decrease through negative values eventually reaching $-\infty$ when the line returns to the camera centre from the rear of the camera. This is illustrated in Fig 4.10.

Now, if the configuration undergoes a projective transformation H with positive determinant taking the plane π_∞ to infinity, then the parameter χ will be replaced by a new parameter χ' defined by $\chi'(\mathbf{x}) = \chi(H\mathbf{x}; PH^{-1})$. Since the transformation is assumed to have positive determinant, it will preserve the front of the camera locally near the camera centre (by Theorem 4.10). Now, as one proceeds along the line in the same direction as before, the parameter χ' will decrease continuously through positive values from infinity at the camera centre, reaching zero where the line crosses the plane π_∞ and then continuing to decrease through negative values until the line returns to the camera centre. Since both χ and χ' decrease monotonically as one proceeds along the line, one sees that if $\mathbf{x}_1$ and $\mathbf{x}_2$ are two points on the line, then $\chi'(\mathbf{x}_1) > \chi'(\mathbf{x}_2)$ if and only if $\chi(\mathbf{x}_1) > \chi(\mathbf{x}_2)$.

In the case where the projective transformation has negative determinant, then the front and back of the camera are reversed locally. In this case the direction of increase of the parameter χ' will be reversed. In this case $\chi'(\mathbf{x}_1) > \chi'(\mathbf{x}_2)$ if and only if $\chi(\mathbf{x}_1) < \chi(\mathbf{x}_2)$.

If the case where the projective transformation transforms the scene to the "true" scene, of two points that project to the same point in the image, the one with the higher value of χ' is closer to the camera. This leads to the following proposition that allows us to determine from an arbitrary projective reconstruction which of two points is closer to the front of the camera.

Proposition 4.22. *Suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ are two points that map to the same point in an image. Consider a projective reconstruction of the scene and let the parameter χ be*

defined (by formula (4.1)) in the frame of the projective reconstruction. If the projective reconstruction has the same orientation as the true scene, then the point that lies closer to the front of the camera in the true scene is the one that has the greater value of χ. On the other hand, if the projective transformation has the opposite orientation, then the point with smaller value of χ will lie closer to the front of the camera in the true scene.

As remarked previously, unless there exists a plane separating the point set from the cameras used for the reconstruction, the orientation of the scene is uniquely determined, and one can determine whether the projective transformation of theorem 4.22 has positive or negative determinant. However, to do this may require one to compute a strong realization of the configuration by the linear programming method as described in section 4:7. If differently oriented strong realizations exist, then as illustrated by Fig 4.9, there is an essential ambiguity. However, this ambiguity may be resolved by knowledge of the relative distance from the camera of a single pair of points.

4:10 3D quasi-affine invariants

One of the important properties of quasi-affine transformations is that they preserve separation by planes as will be explained next.

Proposition 4.23. *Let $\mathbf{x}_0$ and $\mathbf{x}_1$ be two points in space and let π be a plane not passing through either of the points. Let h be a quasi-affine transformation with respect to the two points taking $\mathbf{x}_i$ to $\mathbf{x}'_i$ and mapping π to a plane π'. Then $\mathbf{x}_0$ and $\mathbf{x}_1$ lie on the same side of π if and only if $\mathbf{x}_0$ and $\mathbf{x}_1$ lie on the same side of π'.*

Proof. Let π be represented by a 4-vector $\mathbf{v}$. The points lie on the same side of π if and only if $\mathbf{v}^\top \hat{\mathbf{x}}_0 \doteq \mathbf{v}^\top \hat{\mathbf{x}}_1$. Let H represent the projective transformation. Since H is a quasi-affine we have $\hat{\mathbf{x}}'_i = w_i H \hat{\mathbf{x}}_i$ where w_i has the same sign for $i = 0, 1$. The plane represented by $\mathbf{v}$ is mapped to the plane represented by $\mathbf{v}'$ such that $\mathbf{v}'^\top = \mathbf{v}^\top H^{-1}$. Then $\mathbf{v}'^\top \hat{\mathbf{x}}'_i = (\mathbf{v}^\top H^{-1})(w_i H \hat{\mathbf{x}}_i) = w_i \mathbf{v}^\top \hat{\mathbf{x}}_i$. Since all w_i have the same sign, it follows that $\mathbf{v}^\top \hat{\mathbf{x}}_0 \doteq \mathbf{v}^\top \hat{\mathbf{x}}_1$ if and only if $\mathbf{v}'^\top \hat{\mathbf{x}}'_0 \doteq \mathbf{v}'^\top \hat{\mathbf{x}}'_1$, whence the result. □

Given a point set $\{\mathbf{x}_i\}$ it results from this proposition that the set of planes that do not separate the point set is preserved under quasi-affine transformations. Consequently, the convex hull of a set of points is preserved by quasi-affine transformations as was claimed in section 4:3.

Proposition 4.23 may be used to define quasi-affine invariant properties of point sets. Let π be a plane partitioning the point set into two subsets X_+ and X_-. Applying a quasi-affine mapping the transformed point set will be partitioned into the same two subsets by the transformed plane. Thus for each plane π there exists an invariant partitioning of the set of points. If the partitioning plane is defined in terms of the point set itself (such as a plane passing through three specified points), then the resulting partition is invariant under quasi-affine transformation, and may be used for indexing purposes.

4:10.1 An invariant sequence

A way of finding a better invariant plane than the one defined by three points in the set is now described. We describe this method in general n-dimensional space.

Suppose we are given a set of $N \geq n+2$ points $\{\mathbf{x}_i\}$, $i = 1, \ldots, N$ in R^n. Let $\mathbf{e}_1, \ldots \mathbf{e}_{n+2}$ be points in R^n such that $\{\mathbf{e}_i\}$ form a canonical projective basis for $\mathcal{P}^n$. For $n = 2$, the points $(0,0)^\top$, $(1,0)^\top$, $(0,1)^\top$ and $(1,1)^\top$ will do. Assume that the points $\mathbf{x}_i$ are numbered in such a way that the first $n+2$ of them are in general position (meaning that no $n+1$ of them lie in a codimension 1 hyperplane). In this case, there exists a projectivity g (not in general quasi-affine) such that $g(\mathbf{x}_i) = \mathbf{e}_i$ for $i = 1, \ldots, n+2$. Let $\pi_\infty = g^{-1}(L_\infty)$ be the plane in R^n that is mapped to the plane at infinity by this mapping, g. The invariant partition that we are interested in is the one defined by the plane π_∞.

We can define the partition more specifically as follows. Let G be a matrix representing the projective transformation g. For each i we may define points $\mathbf{e}_i$ such that $G\hat{\mathbf{x}}_i = \eta_i \hat{\mathbf{e}}_i$ where $\mathbf{x}'_i$ is the image of $\mathbf{x}_i$ under g. In particular for $i = 1, \ldots, n+2$ the points $\mathbf{e}_i$ are our canonical projective basis. In this way, the set $\{\mathbf{x}_i\}$ is partitioned into those points for which $\eta_i > 0$ and those for which $\eta_i < 0$. In exceptional cases the point $\mathbf{e}_i = g(\mathbf{x}_i)$ may lie on the plane at infinity, in which case we set $\eta_i = 0$. This invariant partitioning is of course dependent on the choice of canonical basis $\{\mathbf{e}_i\}$.

The cheiral sequence. We define $\text{sign}(\eta_i)$ to be $+1$, -1 or 0 according to whether η_i is positive, negative or zero. The sequence of values $\text{sign}(\eta_i)$ for $i = 1, \ldots, N$ is called the *cheiral sequence* of the points $\mathbf{x}_i$. Except for a simultaneous change of sign of all η_i, the cheiral sequence is invariant under quasi-affine transformations.

If desired, it is possible to code the values η_i into a single number according to the formula

$$\chi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \left| \sum_{i=1}^{N} \text{sign}(\eta_i) 3^{i-1} \right| \tag{4.5}$$

The value $\chi(\mathbf{x}_i)$ is invariant under quasi-affine transformation of the ordered set of points $\mathbf{x}_i$.

We now make the assumption that $\eta_i \neq 0$. In this case the cheiral sequence, along with the projective invariants of the point configuration, constitute a complete quasi-affine invariant. This may be stated as follows.

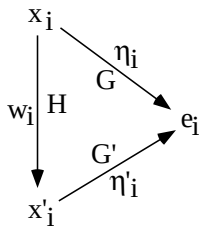
Theorem 4.24. *Let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be a set of points in R^n, where $N \geq n+2$. Suppose that the first $n+2$ of these points form a basis for $\mathcal{P}^n \supset R^n$, so that the cheiral sequence* $\text{sign}(\eta_i)$ *may be defined as above. Suppose further that for each i we have $\eta_i \neq 0$. Let $\mathbf{x}'_1, \ldots, \mathbf{x}'_N$ be another set of points in R^n, projectively equivalent to the points $\{\mathbf{x}_i\}$ via a projective transformation h. Then h is a quasi-affine mapping if and only $\eta_i \doteq \eta'_i \epsilon$ for some constant $\epsilon = \pm 1$.*

Proof. Let points $\mathbf{e}_i$ be defined as in the definition of the cheiral sequence. Further, let g be a projective transform represented by a matrix G and let η_i be defined by the equation $G\hat{\mathbf{x}}_i = \eta_i \hat{\mathbf{e}}_i$. Similarly, we may define projective transformation g' represented by matrix G' and values η'_i such that $G'\hat{\mathbf{x}}'_i = \eta'_i \hat{\mathbf{e}}_i$.

Since the transformation g is defined uniquely by its action on the basis set $\mathbf{x}_1, \ldots, \mathbf{x}_{n+2}$ we see that $g = g'h$. Let h be represented by a matrix H, which may be chosen with the correct sign such that $G = G'H$. We define constants w_i such that $H\hat{\mathbf{x}}_i = w_i \hat{\mathbf{x}}'_i$. It follows that $\eta_i = \eta'_i w_i$, since

$$\eta_i \hat{\mathbf{e}}_i = G\hat{\mathbf{x}}_i = G'H\hat{\mathbf{x}}_i = w_i G'\hat{\mathbf{x}}'_i = w_i \eta'_i \hat{\mathbf{e}}_i \quad .$$

This situation is represented by the following commutative diagram.



Now, if H represents a quasi-affine transformation, then all w_i have the same sign by Proposition 4.5. We may write $w_i \doteq \epsilon$ from which one sees that $\eta_i \doteq \epsilon\eta_i'$ for all i, and the cheiral sequences of the points $\mathbf{x}_i$ and $\mathbf{x}_i'$ differ at most by a sign change.

Conversely, suppose that $\eta_i \doteq \epsilon\eta_i'$. Then $\epsilon \doteq \eta_i/\eta_i'$, since by hypothesis $\eta_i \neq 0$, and so $\eta_i' \neq 0$. On the other hand, from $\eta_i = w_i\eta_i'$ we deduce that $w_i = \eta_i/\eta_i' \doteq \epsilon$ and the w_i all have the same sign, as required. □

This theorem is not true without the assumption that $\eta_i \neq 0$, as the reader is left to discover. In practice, because of measurement inaccuracies, it will (virtually) never be the case that a computed value of η_i will equal exactly 0. Therefore, for readability in displaying cheiral sequences the practice will be adopted of writing 0 instead of -1, so that the cheiral sequence becomes a sequence of 0 and 1 values, and may be interpreted as a binary integer if desired.

4:10.2 The cheiral sequence in two dimensions

To illustrate the principle of the cheiral sequence, we illustrate it for sets of 4 points in the plane. The interpretation of the cheiral sequence in this way for 2-dimensional sets was suggested by Charles Rothwell. We assume that no three of the points are collinear. Let the points be $\mathbf{u}_1, \ldots, \mathbf{u}_4$. We define a particular line in the plane as follows. Denote the line through two points $\mathbf{u}_i$ and $\mathbf{u}_j$ by $< \mathbf{u}_i, \mathbf{u}_j >$. Furthermore, denote the intersection of two lines by the symbol $\times$. Thus $< \mathbf{u}_1, \mathbf{u}_2 > \times < \mathbf{u}_3, \mathbf{u}_4 >$ is the intersection of the line through $\mathbf{u}_1$ and $\mathbf{u}_2$ with the line through the points $\mathbf{u}_3$ and $\mathbf{u}_4$.

Now, construct the points $\mathbf{p}_{1234} =< \mathbf{u}_1, \mathbf{u}_2 > \times < \mathbf{u}_3, \mathbf{u}_4 >$ and $\mathbf{p}_{1324} =< \mathbf{u}_1, \mathbf{u}_3 > \times < \mathbf{u}_2, \mathbf{u}_4 >$. Then construct the line $\pi =< \mathbf{p}_{1234}, \mathbf{p}_{1324} >$ joining these two points. This construction is shown in Fig 4.11 for several configurations of four points.

If points $\mathbf{u}_i$ are the points of a canonical basis with homogeneous coordinates $(0, 0, 1)$, $(1, 0, 1)$, $(0, 1, 1)$ and $(1, 1, 1)$, then points $\mathbf{p}_{1234}$ and $\mathbf{p}_{1324}$ are two points on the line at infinity, and so the line π is the line at infinity, denoted L_∞. If on the other hand, the points $\mathbf{u}_i$ are not the points of this canonical basis, but are mapped to that basis by a projective transformation h, then the line π is mapped to the line at infinity. Thus, we have $\pi = \pi_\infty = h^{-1}(L_\infty)$, and so π is the line defined in the definition of the cheiral sequence. If we choose ξ_i to be ± 1 according to which side of π the point $\mathbf{u}_i$ lies. The sequence of values ξ_i is the cheiral sequence. It is invariant up to simultaneous reversal of all signs. The invariant values are shown in Fig 4.11, where for readability the digit 0 is used instead of -1. The values of ξ_i are normalized in all cases so that $\xi_1 = 0$.

As seen in the diagram (and proven by Theorem 4.24) the cheiral sequence distinguishes all non-equivalent configurations of four points. These seven configurations of points

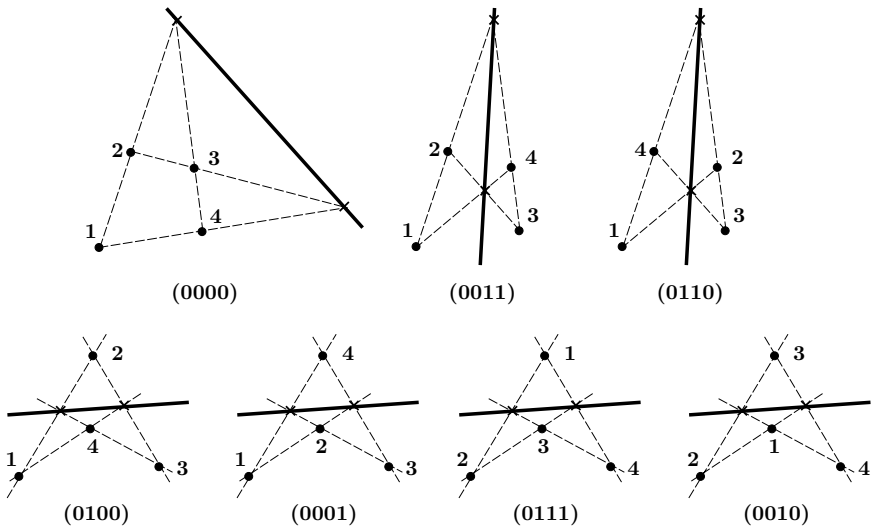


Figure 4.11: **Cheiral sequence in two dimensions.** *The cheiral sequnece is the sequence ξ_i for $i = 1, \ldots, 4$ where ξ_i is 0 or 1 according to whether the point $\mathbf{u}_i$ lies on the same side or the opposite side of π_∞ from $\mathbf{u}_1$. Shown are the 7 distinct arrangements of 4 points in the plane.*

in the plane were also considered by Morin (/citemorin93a,morin94a) who found them very useful for helping distinguish point sets in the plane using projective invariants. In that work it was shown that considering the quasi-affine structure (using the present terminology) of the set of points significantly increased the capability of distinguishing point sets in the plane as compared with using only projective geometric techniques.

4:10.3 Computation of 3D invariants

Computation of the cheiral sequence of a set of points seen in a set of views is relatively straight-forward. It takes place in four steps

1. Compute a projective reconstruction of the point set from the images.
2. Transform the projective reconstruction to a quasi-affine reconstruction.
3. Determine the mapping that maps the first five points to the canonical basis $\mathbf{e}_i$.
4. Project each point and compute the coefficients η_i.

Many ways ([12, 22, 48]) have been given for carrying out the first step of projective reconstruction. It will be easiest if one uses a method (for example [22]) that results in one of the cameras having matrix $[I \mid 0]$. Then one carries out the second step of quasi-affine reconstruction simply by swapping the last two coordinates of each point. Otherwise, the method of section 4:3 is still fairly straight-forward.

One may ask how many quasi-affinely distinct configurations of five points in space exist, analogous to the seven configurations of four points in the plane. We ignore configurations in which four points lie in a plane. In this case, the cheiral sequence of five points is of length five. Up to a common sign change, there are therefore 16 distinct cheiral sequences for five points. This gives an upper bound on the number of distinct configurations.

One may get an exact count by enumerating the different possible geometries of the convex hull of the points. As in two dimensions, there are two different types of configuration – those in which all five points lie on the convex hull, and those in which only 4 points lie on the convex hull. In this second case the convex hull is a tetrahedron containing the fifth point in the interior. Corresponding to the five possible choices of which point is in the interior, there are five possible such configurations.

We now analyze the configurations in which all five points lie on the convex hull. The convex hull is a polyhedron, bounded by triangular faces, since no four points are coplanar. Let n be the number of faces. Since each face has three edges, and each edge belongs to two faces, we see that there are $3n/2$ edges, and so the Euler characteristic of the polyhedron is $5-3n/2+n = 2$, since the boundary of the convex hull is topologically a sphere. From this it follows that there are $n = 6$ faces and 9 edges. Since each edge meets two vertices, the sum of degrees of the vertices must equal 18. Since no vertex can have degree 5 (there are only five vertices in total), the only possibility is that there are three vertices with degree 4 and two vertices with degree 3. The polyhedron must have the shape of two tetrahedra joined along one face. There are 10 possible such configurations corresponding to the 10 different ways of choosing the two vertices with degree 3.

In total therefore there are $15 = 5 + 10$ quasi-affinely distinct configurations of five (numbered) points in three dimensions. Proposition 4.24 shows that these configurations

may be distinguished by their cheiral sequences. Curiously enough, 15 is one less than the upper bound of 16 distinct cheiral sequences. Just as in the two dimensional case, there is one cheiral sequence which can not occur. Does this observation hold in higher dimensions also ? This question is left for the interested reader to resolve.

4:11 Experimental results

In considering real images of 3-D configurations it is necessary to take into account the effects of noise. In some cases, a value of η_i used in computing the cheiral sequence will lie so close to 0 that variations due to noise can swap its sign. For robust evaluation of a cheiral sequence value, it is necessary to select a noise model and determine how errors in the input data affect the sign of each η_i. In the following discussion, noise effects are ignored, however. As usual, cheiral sequences are written using the digit 0 instead of $\bar{1}$, for readability.

In section 3:1 the projective invariants of several point sets associated with three wooden block house images were computed. The images and results were given in section 3:1.

Although the projective invariants given in table (3.18) in section 3:1 there were quite effective at discriminating between different point sets, indicated by the fact that most off-diagonal entries are not close to zero, entries $(2,3)$ and $(3,2)$ are small indicating that the point sets numbered 2 and 3 are close to being equivalent up to projectivity.

Next, the cheiral sequence for each of the point sets were computed from the weak realization using the method described here. The computed values for each of the six point sets were as follows. The binary integer interpretation of the cheiral sequence is given in brackets.

$$\begin{aligned}
\chi(S_1) &= 011100 = (28)_{10} \\
\chi(S_2) &= 110000 = (60)_{10} \\
\chi(S_3) &= 000100 = (4)_{10} \\
\chi(S_4) &= 111100 = (60)_{10} \\
\chi(S_5) &= 101010 = (42)_{10} \\
\chi(S_6) &= 100100 = (36)_{10}
\end{aligned}$$

As expected these invariant values were the same whether computed using the first pair of views or the second pair. Note that the cheirality invariant clearly distinguishes point sets 2 and 3. Point sets S_2 and S_4 have the same cheiral sequence, but these are well distinguished by their projective invariants.

Conclusions : These results show that the cheiral sequence is quite effective at distinguishing between arbitrary sets of points. Given the relative ease with which the cheiral sequence may be computed, it may be extremely useful in grouping points. In addition, it may conveniently be used as an indexing function in an object recognition system. It has been demonstrated that the cheiral sequence gives supplementary information that is not available in projective invariants. As a theoretical tool, the cheiral sequence provides conditions under which image point matches may be realized by real point configurations.

Part V

Reconstruction from Three Views

In this part of the report, we turn to consideration of reconstruction from three images. Though it may be thought that this will be simply an extension of the techniques used for two views, it turns out that three views give rise to a new and interesting mathematical theory based around the trifocal tensor. In addition, using three views instead of two brings in the possibility of using images of lines in the reconstruction instead of only points.

The trifocal tensor is in some respects analogous to the fundamental matrix which we have argued is the basis for consideration of pairs of images. It is natural and convenient to make use of some of the conventions of tensor notation in dealing with the trifocal tensor, and we will therefore begin with a brief introduction to tensors.

5:1 Tensor notation

Consider a set of basis vectors $\mathbf{e}_i$ for a vector space. For reasons to become clear, we will write the indices as subscripts. In what follows, we will be using both upper and lower indices, and we will make use of the common summation convention for tensors. *When a given index appears as both upper and lower indices in a given expression, summation over all values of the index is implied.*

A point in the vector space spanned by the $\mathbf{e}_i$ is represented by a set of coordinates u^i. We write the coordinates with an upper index, as shown. Coordinates u^i represent the point $\mathbf{u} = \sum_i u^i \mathbf{e}_i$. Using the summation convention, we may write $\mathbf{u} = u^i \mathbf{e}_i$. Now, consider a change of coordinate axes in which the basis vectors $\mathbf{e}_i$ are replaced by a new basis set $\hat{\mathbf{e}}_j$. If h^i_j represent the entries of the transformation matrix, then we may write this transformation as

$$\hat{\mathbf{e}}_j = h^i_j \mathbf{e}_i \ , \tag{5.1}$$

where we have used the summation convention to imply a summation over the repeated index i.

We may write the vector $\mathbf{u}$ in the new coordinate axes as $\mathbf{u} = \hat{u}^j \hat{\mathbf{e}}_j$. The question arises as to how the new coordinates $\hat{u}^j$ are related to the old coordinates u^i. From (5.1) one sees that $\mathbf{e}_i = (h^{-1})^j_i \hat{\mathbf{e}}_j$, where h^{-1} is the inverse transform to h. Using this relation, one easily computes that

$$\begin{aligned} \mathbf{u} &= u^i \mathbf{e}_i \\ &= u^i (h^{-1})^j_i \hat{\mathbf{e}}_j \\ &= ((h^{-1})^j_i u^i) \hat{\mathbf{e}}_j \end{aligned}$$

from which we deduce that $\hat{u}^j = (h^{-1})^j_i u^i$. Thus, the coordinates u^i transform according to the inverse transform h^{-1}. This fact is expressed by stating that u^i transforms *contravariantly*. According to (5.1) however, the basis vectors themselves transform according to the transformation h. We say that the basis vectors transform *covariantly*.

As a further example, consider a hyperplane represented by a vector $\lambda = (\lambda_i)$. Here we use a lower index, since it will be seen soon that λ_i transforms covariantly with respect to a change of basis. A point $\mathbf{u} = (u^i)$ lies on the line $\lambda = (\lambda_i)$ if and only if $\lambda_i u^i = 0$. Consider a change of coordinates in which λ_i is transformed to $\hat{\lambda}_j$ and u^i is transformed to $\hat{u}^j$. Since incidence is preserved, we see that $\hat{\lambda}_j \hat{u}^j = 0$. However from $\lambda_i u^i = 0$ we deduce

$\lambda_i h^i_j \hat{u}^j = 0$ by the transformation rule for u^i. From this we deduce that $\hat{\lambda}_j = \lambda_i h^i_j$, and so λ_i indeed transforms covariantly.

Any indexed set that transforms according to the covariant or contravariant rule given here is known as a *tensor*. We distinguish contravariant from covariant tensors by using either a lower (for covariant) or upper (contravariant) index.

Doubly indexed quantities may be tensors also. As an example, consider a camera matrix M mapping points x^i in world coordinates to points u^j in image coordinates. The indices are written as superscripts, since as we have seen, points are represented as contravariant vectors. The position of the indices suggests that the camera mapping should be expressed as

$$u^j = m^j_i x^i \ .$$

Where $M = (m^j_i)$. One may in fact verify that m^j_i transforms contravariantly in the index j and covariantly in the index i. More specifically, if $\mathbf{g}$ and $\mathbf{h}$ represent changes of basis in the world and image spaces respectively, and $\hat{m}^k_l$ represents the transformation with respect to the new bases, then

$$\hat{m}^k_l = (h^{-1})^k_j m^j_i g^i_l \ . \tag{5.2}$$

Thus, the camera matrix is an example of a 2-dimensional tensor that is covariant in one index, and contravariant in the other.

Note that the order of the terms in the product is not important in (5.2) since the expression represents a sum over repeated indices, each term in the summation being a product of scalar quantities. Similarly, in tensors having an upper and lower index, such as m^j_i, there is no concept of an order of the two indices. Thus, we do not consider the index j as *preceding* the index i. This contrasts with what occurs with matrix notation. If the entry of a matrix M is denoted as M_{ij}, then i represents the row index and j is the column index. In this case the order of the two indices is important to distinguish between row and column index. In the tensor notation m^j_i the two indices are distinguished by the fact that one is covariant and the other is contravariant, and not by their order.

It is possible to have tensors with multiple indices – we will see three and four-index tensors later on. In such cases the order of the set of covariant indices is important, as is the order of the contravariant indices. There is no order relationship, however, between covariant and contravariant indices.

5:2 Reconstruction from Three Views

This section discusses the basic role of the trifocal tensor in scene reconstruction from three views. This $3 \times 3 \times 3$ tensor plays a role in the analysis of scenes from three views analogous to the role played by the fundamental matrix in the two-view case. In particular, the trifocal tensor may be computed by a linear algorithm from a set of 13 line correspondences in three views. It is further shown in this section, that the trifocal tensor is essentially identical to a set of coefficients introduced by Shashua to effect point transfer in the three view case. This observation means that the 13-line algorithm may be extended to allow for the computation of the trifocal tensor given any mixture of sufficiently many line and point correspondences. From the trifocal tensor the camera matrices of image may be computed, and the scene may be reconstructed. For unrelated uncalibrated cameras, this reconstruction will be unique up to projectivity.

Thus, projective reconstruction of a set of lines and points may be carried out linearly from three views.

This section gives an effective algorithm for the projective reconstruction of a scene consisting of lines and points in space as seen in three views with uncalibrated cameras. The placement of the cameras with respect to the scene is also determined. This algorithm is unique in the literature in that it gives a unified linear approach that can deal with a mixture of points and lines. For instance, previous algorithms have been specific to points ([42, 36, 37]) or lines ([70, 77]), but could not handle both. True, one could always use pairs of matching points to determine line matches, which can then be used in an algorithm for reconstruction from lines. This stratagem, however, achieves a unified approach for lines and points at considerable expense, since a pair of point matches contains much more information than a single line match (as will be made clear quantitatively in this section). The restraint of using only points or lines forces one to ignore important information available in most images, particularly of man-made objects, since typically, one can find both distinguished points and lines in matching images.

Points are important in that they give much more information than lines. For instance although one can do relative reconstruction from only two views of a set of points ([42]), for lines at least three views are necessary ([77]). On the other hand, lines have several advantages. Firstly, they can normally be determined more accurately than points, often with an accuracy of better than a tenth of a pixel. Secondly, line matches may be used in cases where occlusions occur. Often end points of lines are not visible in the images. For instance, in Fig 5.1, the left hand slanting roof edge of the rear house may be matched in the three images, although its end point is occluded behind the front house. On the other hand, if we are to use line matches, then at least three views must be used, since no information whatever about camera placements may be derived from any number of line-to-line correspondences in fewer than three views.

Outline. The key observation of this section is the connection (see equation (5.9)) between Shashua's 3-view trilinearity relationships ([65]) and 7-point algorithm and the previously published ([24, 31]) algorithm for projective reconstruction from lines. It immediately follows from that observation that one can amalgamate the two algorithms into one. This means that one can non-iteratively carry out projective reconstruction from three views of 7 points, or 13 lines or anything in between (so to speak). Projective reconstruction from 7 lines in three views was lurking behind Shashua's work, but never explicit previously.

In order to derive this key observation, we rederive Shashua's trilinearity relationships and place them in the standard context of projection using camera matrices. The hope hope is that this rederivation has the merit of throwing further light on the meaning of those relationships.

We take the opportunity in section 5:2.7 to provide a better method of determining the camera matrices than the one that previously published in [31]).

5:2.1 The Trifocal Tensor

A basic tool in the analysis of this section is an entity called, here for the first time, the *trifocal tensor*. Since this entity has appeared previously in the literature in different guises, and it is therefore appropriate to discuss its history. With hindsight, we may

attribute the discovery of the trifocal tensor to Spetsakis and Aloimonis ([70] and Weng, Huang and Ahuja ([77]), where it was used for scene reconstruction from lines in the case of calibrated cameras. It was later shown by the present author in [21, 24, 31] to be equally applicable to projective scene reconstruction from 13 lines in the uncalibrated case. Those papers form the basis for part of this article. In all of the above papers, the entity referred to here as the trifocal tensor was not considered as a tensor, but rather as a set of three 3×3 matrices. Perhaps the first author to refer to it as a tensor was Vieville ([76]) who continued the work on scene reconstruction from lines.

Meanwhile in independent work, Shashua introduced a set of 27 coefficients for a set of four independent tri-linearity conditions relating the coordinates of corresponding points in three views with uncalibrated cameras ([65]). Subsequently ([66]) Shashua gave a linear method for computation of the coefficients from only 7 point matches in three views.

A key result of this section is that the set of coefficients introduced by Shashua ([65]) are exactly the same as the entries of the three 3×3 matrices of ([77, 24]), except for a change of sign[7] and rearrangement of the indices. The importance of this result is that it allows an amalgamation of the linear algorithms for points (Shashua [66]) and for lines ([31]). This results in an algorithm of significantly greater applicability and power than either of the line or point algorithms alone.

Whereas the line papers [77, 24, 31] consider three 3×3 matrices, Shashua's paper defines his coefficients as the entries of nine 3-vectors. In fact, essentially, we are dealing with a triply indexed $3 \times 3 \times 3$ array of values, which it is natural to treat as a tensor, as suggested by Vieville. Therefore, in this report, we refer to this three-dimensional array as a tensor, though without making significant use of tensor notation or machinery. In recent unpublished work, Shashua has also considered his set of coefficients as a tensor. As for the name, we suggest the words *trifocal tensor* in an attempt to establish a standard terminology. Despite the potentially fundamental role played by this tensor in three-view stereo, we believe that the word *fundamental* is too often used for us to adopt the term *fundamental tensor*.

5:2.2 Notation and Basics

We adopt the tensor summation convention, which we repeat here for emphasis.

> Any repeated index in a product of vectors, matrices and tensors implies a summation over the range of index values. If the index range is not the same in both instances of the repeated index (as happens occasionally), then summation over the intersection of the two ranges is implied. Any formula involving indices is intended to hold for any choice of values of the free indices (which means those indices that are not repeated).

The three-dimensional space containing the scene will be considered to be the 3-dimensional projective space $\mathcal{P}^3$ and points in space will be represented by homogeneous 4-vectors $\mathbf{x}$. Similarly, image space will be regarded as the 2-dimensional projective space

[7] In order to avoid the sign discrepancy, Shashua's coefficients will be defined with opposite sign in this report

$\mathcal{P}^2$ and points in an image will be represented by homogeneous 3-vectors $\mathbf{u}$. Homogeneous quantities (vectors, matrices or tensors) that differ by a non-zero scale factor are considered to be equal. In this section, we use the symbol $\approx$ to indicate equality up to a constant non-zero scale. This is necessary, since we want to keep clear the distinction between quantities that are equal and those that are equal up to a constant factor.

The space-image mapping induced by a projective camera may be represented by a 3×4 matrix $M = [m^i_j]$ of rank 3, such that if $\mathbf{x}$ and $\mathbf{u}$ are corresponding object and image points then $\mathbf{u} \approx M\mathbf{x}$, or in tensor notation, $u^i \approx m^i_j x_j$. Such a matrix will be called a camera matrix. One special camera matrix is denoted by $[I \mid 0]$, made up of a 3×3 identity matrix I and a final zero column.

Normal Form for Camera Matrices. Consider a set of lines and points in space viewed by several cameras. We use the word *feature* to represent either a line or a point. We suppose that the image coordinates of each feature as seen in each image are given, but the actual positions of the features in space are unknown. The task of projective reconstruction is to find a set of camera matrices M_j and 3D-lines and points so that each such 3D feature is indeed mapped to the given image feature in each of the images. If the camera matrices are allowed to be arbitrary, then we have seen that the scene can not be reconstructed more precisely than up to an arbitrary 3D projective transformation.

Consider now a reconstruction from three views, and let the three camera matrices be M, M' and M''. We make the assumption that no two of the cameras are located at the same point in space. Let H be formed by adding one extra row to M to make a non-singular 4×4 matrix. Then since $HH^{-1} = I_{4\times 4}$, it follows that $MH^{-1} = (I|0)$. Since M may be transformed to $[I \mid 0]$, by applying transformation H to the reconstruction we may assume without loss of generality that $M = [I \mid 0]$.

To save the reader the trouble of having to count primes, we denote the entries of the camera matrices M' and M'' by a^i_j and b^i_j respectively, instead of by m'^i_j and m''^i_j. Thus, the three camera matrices M, M' and M'' may be written in the form $M = [I \mid 0]$, $M' = [a^i_j]$ and $M'' = [b^i_j]$

5:2.3 Transferring lines

In this section we investigate the relationship between the images of a line as taken by three separate cameras.

Given a general camera matrix (for instance M) and a line in the image (let it be λ), the projection of the line from the camera centre forms a plane in space consisting of all points in space that will map onto the given image line. We need to derive a formula for that plane. The simple answer is as follow.

Proposition 5.1. *The plane in space consisting of all points that are mapped to a line λ by a camera with matrix M is equal to $\pi = M^\top \lambda$.*

Proof. By definition, a point $\mathbf{x}$ is on the plane π if and only if $M\mathbf{x}$ lies on the line λ. This latter condition can be expressed as $\mathbf{x}^\top M^\top \lambda = 0$. On the other hand, $\mathbf{x}$ lies on π if and only if $\mathbf{x}^\top \pi = 0$. Hence, we see that $\mathbf{x}^\top = 0$ if and only if $\mathbf{x}^\top M^\top \lambda = 0$, and from this we deduce that $\pi = M^\top \lambda$ as required. □

In tensor notation we may write this expression as

$$\pi_j = m_j^i \lambda_i \ .$$

Now, given three cameras with matrices $M = [I \mid 0]$, $M' = [a_j^i]$ and $M'' = [b_j^i]$, and three lines λ, λ and λ'' in the three images, we seek a relationship between the coordinates of the three lines. Since the three image lines are derived from a single line in space, it follows that the planes corresponding to the three image lines must meet at this line in space. In particular, the three planes $M^\top\lambda$, $M'^\top\lambda'$ and $M''^\top\lambda''$ meet in a line. Writing the coordinate vectors of these three planes as the rows of a matrix we obtain

$$\begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & 0 \\ a_1^j\lambda'_j & a_1^j\lambda'_j & a_3^j\lambda'_j & a_4^j\lambda'_j \\ b_1^k\lambda''_k & b_2^k\lambda''_k & b_3^k\lambda''_k & b_4^k\lambda''_k \end{bmatrix} \ . \tag{5.3}$$

In writing this matrix, we have taken particular note of the simple form of the matrix $M = [I \mid 0]$. Since the three planes meet in space, there is a linear dependency between the rows of this matrix (5.3). Because of the zero entry in the top row, we deduce that

$$\begin{aligned} \lambda_i &\approx a_i^j\lambda'_j b_4^k\lambda''_k - a_4^j\lambda'_j b_i^k\lambda''_k \\ &= \lambda'_j\lambda''_k(a_i^j b_4^k - a_4^j b_i^k) \end{aligned}$$

Now, we define a $3 \times 3 \times 3$ tensor T_i^{jk} by the expression

$$T_i^{jk} = a_i^j b_4^k - a_4^j b_i^k \ . \tag{5.4}$$

Then we have the following formula.

$$\lambda_i \approx \lambda'_j\lambda''_k T_i^{jk} \ . \tag{5.5}$$

The tensor T_i^{jk} is the *trifocal tensor*, which is the basic entity investigated in this section. Formula (5.5) has been derived previously in [77, 70] for the special case of calibrated cameras. The above derivation shows how generalization to the case of uncalibrated cameras leads to a simplification of the derivation.

Given T_i^{jk} and the coordinates of matching lines λ' and λ'', expression (5.4) may be used to compute the line in the other image. The application of this process, know as *line transfer* will not be investigated in this report. Point transfer, the analogous process for points has been explored in [2, 65] and will be mentioned taken up later in this report. Once again, the trifocal tensor will be shown to be the fundamental entity for point transfer.

We describe now a first method for determining the fundamental tensor. If at least 13 line matches $\lambda \leftrightarrow \lambda' \leftrightarrow \lambda''$ are known, it is possible to solve for the entries of the tensor T_i^{jk}, since each line match provides two linear equations in the 27 unknown tensor entries. In particular, if the line λ is presented by specifying the two endpoints, then each endpoint $\mathbf{u} = (u^1, u^2, u^3)$ gives rise to an equation

$$u^i\lambda'_j\lambda''_k T_i^{jk} = 0 \tag{5.6}$$

To normalize these equations, the line λ' (and similarly λ'') should be scaled so that $\lambda_1'^2 + \lambda_2'^2 = 1$, and each end point $\mathbf{u}$ should be scaled so that $u^3 = 1$.

5:2.4 Transferring Points.

In this section we will investigate the relationship of the trifocal tensor with point-transfer methods, and in particular with the trilinearity relationships of Shashua.

Suppose that a point $\mathbf{x}$ in space is seen in three images, and that as usual the three cameras are given in the normalized form $M = [I \mid 0]$, $M' = (a^i_j)$ and $M'' = (b^i_j)$.

We suppose that the point $\mathbf{x}$ is seen at positions $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$ in the three images, where $\mathbf{u}$ (and similarly $\mathbf{u}'$ and $\mathbf{u}''$) is a 3-vector $\mathbf{u} = (u^1, u^2, u^3)$, the representation of the point in homogeneous coordinates. The coordinates $(u^1/u^3, u^2/u^3)$ are the coordinates actually seen in the image. We wish to find a relationship between the coordinates of the points $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$. At any point in the following derivation, we may set u^3, u'^3 or u''^3 to 1 to obtain equations relating to measured image coordinates.

Because of the form of the matrix $M = [I \mid 0]$, it is extremely simple to give a formula for the position of the point in space. In particular, since $[I \mid 0]\mathbf{x} \approx \mathbf{u}$, we may write $\mathbf{x} = \begin{pmatrix} \mathbf{u} \\ t \end{pmatrix}$ for some t, yet to be determined. It may be verified that t is the same as the "relative affine invariant", k, considered by Shashua ([65]). Now, projecting this point into the second image by the usual formula $u'^i \approx a^i_j x_j$ gives

$$u'^i \approx a^i_k u^k + a^i_4 t$$

The notation $\approx$ denotes equality up to an unknown scale factor. We may eliminate this scale factor to obtain equations

$$u'^i(a^j_k u^k + a^j_4 t) = u'^j(a^i_k u^k + a^i_4 t)$$

where each choice of the free indices i and j gives a separate equation. Of the three resulting equations, only two are independent. From each of these equations independently, one may compute the value of t. We obtain three separate estimates for t.

$$t = u^k(u'^i a^j_k - u'^j a^i_k)/(u'^j a^i_4 - u'^i a^j_4) \tag{5.7}$$

Substituting the value of t from (5.7) we see that the point $\mathbf{x}$ may be written as

$$\begin{aligned} \mathbf{x} &= \begin{pmatrix} \mathbf{u} \\ u^k(u'^i a^j_k - u'^j a^i_k)/(u'^j a^i_4 - u'^i a^j_4) \end{pmatrix} \\ &\approx \begin{pmatrix} (u'^j a^i_4 - u'^i a^j_4)\mathbf{u} \\ u^k(u'^i a^j_k - u'^j a^i_k) \end{pmatrix} \end{aligned}$$

Now, projecting this point via the third camera, $u''^l \approx b^l_k x_k$ we find that

$$\begin{aligned} u''^l &\approx b^l_k u^k(u'^j a^i_4 - u'^i a^j_4) \\ &+ b^l_4 u^k(u'^i a^j_k - u'^j a^i_k) \\ &\approx u^k u'^i(a^j_k b^l_4 - a^j_4 b^l_k) \\ &- u^k u'^j(a^i_k b^l_4 - a^i_4 b^l_k) \end{aligned} \tag{5.8}$$

Now, referring to (5.4), we recognize the tensor coefficients T^{jk}_i in this expression :

$$u''^{l} \approx u^{k}(u'^{i}T_{k}^{jl} - u'^{j}T_{k}^{il}) \tag{5.9}$$

As before we may eliminate the unknown scale factor implied by the $\approx$ sign to get (after some slight rearranging) the equations

$$\begin{aligned} u^{k}(u'^{i}u''^{m}T_{k}^{jl} - u'^{j}u''^{m}T_{k}^{il}) = \\ u^{k}(u'^{i}u''^{l}T_{k}^{jm} - u'^{j}u''^{l}T_{k}^{im}) \ . \end{aligned} \tag{5.10}$$

These are the trilinearity relationships of Shashua ([65]). In these equations, the indices i, j, l and m are free variables, and there is one equation for each choice of indices with $i \neq j$ and $l \neq m$. Since we get the same relation by interchanging i and l, or l and m, we may assume that $i < j$ and $l < m$. There are therefore 9 different equations defined by this expression. However, only two of the three choices of pair (i, j) given independent equations, and the same is true for pairs (l, m). Hence, there are 4 independent equations.

One choice of the four independent equations is obtained by setting $j = m = 3$, and letting i and l range freely. As stated previously, we may set u^3, u'^3 and u''^3 to 1 to obtain a relationship between observed image coordinates. The equations then become

$$u^{k}(u'^{i}u''^{l}T_{k}^{33} - u''^{l}T_{k}^{i3} - u'^{i}T_{k}^{3l} + T_{k}^{il}) = 0 \ . \tag{5.11}$$

The four different choices of $i, l = 1, 2$ give four different equations in terms of the observed image coordinates.

Given 7 point correspondences, we have 28 equations, which is enough to solve for the tensor values T_i^{jk}. Shashua states that better results are obtained by including 6 equations for each point match. The experiments reported in a later section use all 9 equations, but it is not clear how much advantage (if any) this gives.

5:2.5 Solving using lines and points.

We have seen that the entries of the trifocal tensor, T occur in equations involving both points (5.11) and lines (5.6). This has the significant implication that we may amalgamate the line and point algorithms into one algorithm. In particular, each line correspondence $\lambda \leftrightarrow \lambda' \leftrightarrow \lambda''$ gives two linear equations in the entries T_i^{jk}, whereas each point correspondence gives four linear equations. Therefore, provided that $2\#lines + 4\#points \geq 26$ we have sufficiently many matches to solve for the T_i^{jk}. The tensor entries T_i^{jk} are found up to a common scale, and we find a solution such that sum of squares of the entries is one, in symbols $T_i^{jk}T_i^{jk} = 1$. In the case where there are more than 26 equations, we find a least-squares solution satisfying this constraint.

The set of equations we construct are of the form, $A\mathbf{t} = \mathbf{0}$, where A is the equation matrix and $\mathbf{t}$ is a vector containing the elements of T_i^{jm} to be found. We are not interested in the solution $\mathbf{t} = 0$, and to avoid ambiguity, we impose the constraint $||t|| = 1$. Since we do not expect an exact solution, our task is to minimize the quantity $||A\mathbf{t}||$ subject to this constraint. The solution to this problem is easily seen (using Langrange multipliers, for instance) to be the unit eigenvector corresponding to the least eigenvalue of $A^\top A$. Being symmetric and positive definite, $A^\top A$ has only real positive eigenvalues. A good algorithm for finding this eigenvector is the method of Jacobi ([55]).

Normalization. Before setting out to write and solve the equations, it is a very good idea to normalize the data by scaling and translating the points. The algorithm does not do well if all points are of the form (u^1, u^2, u^3) in homogeneous coordinates with u^1 and u^2 very much larger than 1. A heuristic that works well is to translate the points in each image so that the centroid of all measured points is at the origin of the image coordinates, and then scaling so that the average distance of a point from the origin is $\sqrt{2}$ units. In this way the average point will be something like $(1, 1, 1)$ in homogeneous coordinates, and each of the homogeneous coordinates will be approximately of equal weight. This transformation improves the condition of the matrix of equations, and leads to a much better solution. Despite the seemingly harmless nature of this transformation, this is an essential step in the algorithm.

This normalization step has also been shown (in unpublished work) to be effective in the two-camera case for determining the fundamental matrix.

5:2.6 Retrieving the Camera Matrices

Formula (5.4) gives a formula for the trifocal tensor T_i^{jk} in terms of the camera matrices. It is possible to go the other way and retrieve the camera matrices, M' and M'' from the tensor T_i^{jk}.

This is done in two stages. In the first stage, one finds the vectors a_4^i and b_4^i (that is, the last columns of M' and M''). As shown in [24, 31], vectors a_4^i and b_4^i may be found as the common perpendiculars to the left (respectively, right) null-spaces of the three matrices, T_1^{ij}, T_2^{ij} and T_3^{ij}. In the second stage, one finds the remaining entries.

Closed form solution. In [31] closed form formulas were given for $M' = (a_j^i)$ and $M'' = (b_j^i)$. For $i, j = 1, \ldots, 3$ we have

$$\begin{array}{ll} a_j^i & = \sum_l (T_j^{il} - a_4^i a_4^k T_j^{kl}) b_4^l \\ b_j^i & = \sum_k -T_j^{ki} a_4^k \end{array} \tag{5.12}$$

This closed form method of determining M' and M'' was evaluated in [31]. By carefully examining the results of that method, it has subsequently been found that using these formulae to determine M' and M'' is very unstable in the presence of noise, and hence is **not** recommended.

The direct method If a_4^i and b_4^i and T_i^{jk} are all known, the equations (5.4) form a redundant set of linear equations in the remaining entries of the matrices M' and M''. We may solve these equations using linear least-squares methods ([55, 1]) to find $M' = (a_j^i)$ and $M'' = (b_j^i)$. Unfortunately, this method does not seem to give any better results than the closed form solution. In is also **not** recommended. The reason that this method gives poor results is that the trifocal tensor, T_i^{jk} contains small as well as large entries, differing by several orders of magnitude. The small entries are however important, and a changes in the large entries are less significant than changes of equal magnitude in the small entries. Thus, a method that weights changes to all the entries of T_i^{jk} equally (as the direct method does) gives bad results.

The recomputation method. A third method is to go right back to the equations (5.6) and (5.11) and substitute the formula (5.4) to get equations in terms of a^i_j and b^i_j. Now that a^j_4 and b^j_4 are known these equations are linear in the remaining a^i_j and b^i_j. Since we have already gone to a lot of trouble to find T^{jk}_i we do not want to carry out an equivalent amount of work (in terms of run time, and also coding time) to do much the same computation again. Therefore, one can take advantage of the previous computation of T^{jk}_i as follows.

The tensor T^{jk}_i was found by solving a system of equations $A\mathbf{t} = 0$ where $\mathbf{t}$ was a vector comprising the desired elements of T^{jk}_i. In solving these equations using the method of Jacobi (see section 5:2.5) one computed the symmetric positive definite matrix $A^\top A$. This matrix $A^\top A$ will be required for the current step, and hence should have been retained. Now, the equations (5.4) express $\mathbf{t}$ as a linear function of the values a^i_j and b^i_j for $i, j = 1, \ldots, 3$. This can be expressed as an equation $\mathbf{t} = H\mathbf{y}$ where $\mathbf{y}$ is the vector of the entries a^i_j and b^i_j that we are seeking, and H is the linear relationship expressed by (5.4). To find $\mathbf{y}$, we need to solve the equations $AH\mathbf{y} = 0$. More exactly, our task is to minimize $||AH\mathbf{y}||$ subject to the constraint $||\mathbf{y}|| = 1$. The solution is the eigenvector corresponding to the least eigenvalue of $H^\top A^\top AH$. Since we already have the matrix $A^\top A$ we do not need to recompute it. The required eigenvector may be computed using the method of Jacobi. In this way we compute the remaining value of the camera matrices.

Unfortunately, there is one small problem which causes an instability in this method as just described. In particular, it may be verified that the values of a^i_j and b^i_j are not uniquely determined by T^{jk}_i. In particular, if a^i_j is replaced by $a^i_j + \alpha_j a^i_4$ for any vector α_j, and similarly b^i_j is replaced by $b^i_j + \alpha_j b^i_4$ for the same α_j, then the value of T^{jk}_i defined by equation (5.4) is unchanged. This means that the matrix H described above is not of full rank, and consequently the matrix $H^\top A^\top AH$ will have a multidimensional eigenspace corresponding to the eigenvalue 0. This means that the solution found will be unstable. This may not be a significant problem, since any solution found by this method will be a valid solution, giving one solution for the camera matrices. Nevertheless, I prefer to constrain the solution by adding constraints on the entries a^i_j and b^i_j so as to ensure a stable solution.

One method of constraining a^i_j is by specifying that $a^i_j a^i_4 = 0$. This gives three constraints, one for each value of $j = 1, \ldots, 3$, which may be interpreted as meaning that 4-th column of $M' = (a^i_j)$ is orthogonal to all the other columns. One may verify that this condition is achieved by a suitable choice of the vector α_j in the last paragraph (in particular, setting $\alpha_j = -a^i_j a^i_4$). The condition $a^i_j a^i_4 = 0$ gives a set of three linear constraints.

The task of solving for a^i_j and b^i_j is therefore a constrained minimization problem of the form: minimize $||AH\mathbf{y}||$ subject to $||\mathbf{y}|| = 1$ and $C\mathbf{y} = 0$, where C is a matrix of linear constraints. This problem may be solved as follows. Let $C = UDV^\top$ be the Singular Value Decomposition of C, written with non-zero singular values (the diagonal entries of D) occurring before the zero ones. Let V' be the matrix obtained from V by eliminating the first 3 columns (or more generally r columns where r is the rank of C). Let $\mathbf{y}'$ be the eigenvector corresponding to the minimum eigenvalue of $V'^\top H^\top A^\top AHV$, and let $\mathbf{y} = V'\mathbf{y}'$. Then $\mathbf{y}$ is the required vector solving the constrained minimization problem, as may be verified. Since this is a critical step in reconstruction, we summarize the

algorithm :

Algorithm for computing camera-matrices from T_i^{jk}.

1. Retain the matrix $A^\top A$ used to solve the set of equations $A\mathbf{t} = \mathbf{0}$ where $\mathbf{t}$ is the vector containing the entries of T_i^{jk}
2. Compute the set of equations $\mathbf{t} = H\mathbf{y}$ from (5.4), where $\mathbf{y}$ is the vector of still unknown entries of a_j^i and b_j^i.
3. Compute the matrix C such that $C\mathbf{y} = 0$ expresses the three constraints $a_j^i a_4^i = 0$.
4. Compute the Singular Value Decomposition $C = UDV^\top$, and let V' be the matrix obtained from V by eliminating the first three columns.
5. Find
   the eigenvector $\mathbf{y}'$ corresponding to the smallest eigenvalue of $V'^\top H^\top A^\top AHV'$.
6. The required set of 18 values a_j^i and b_j^i are contained in the vector $\mathbf{y} = V'\mathbf{y}'$.

Since this algorithm is more complicated than either of the two previous algorithms (closed-form solution, and direct linear solution) it is comforting to verify that it performs at least 10 times better (in terms of measured errors) than they do. Therefore, the two earlier algorithms may be consigned to the scrap heap.

5:2.7 Reconstruction

Once the camera matrices are computed, it is a simple task to compute the positions of the points and lines in space. Different ways in which this may be done for points are described in [33]. For lines, a method was described in [24, 31]

5:2.8 Algorithm Outline

To tie together all the threads of the reconstruction algorithm, an outline will now be given. As input to this algorithm, we assume as set of point-to-point image correspondences, each one of the form $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}''$, and a set of line correspondences of the form $\lambda \leftrightarrow \lambda' \leftrightarrow \lambda''$, where $\#$ lines $+ 2 * \#$ points ≥ 13. The lines are assumed to be specified by giving the two end points of each line. The steps of the algorithm are as follows.

1. **Coordinate scaling and translation.** For each image separately, translate and scale the coordinates of the points such that the centroid of all given coordinates is at the origin, and the average distance of a point from the origin is $\sqrt{2}$.
2. **Computing and normalizing the lines.** Each line λ' and λ'' is computed from its endpoints, and normalized.
3. **Construct the equations.** For each line correspondence, construct a pair of equations of the form (5.5) in the entries of the tensor T_i^{jk}. Similarly, for each point correspondence, construct four equations of the form (5.11) also in the entries of tensor T_i^{jk}.

4. **Finding the trifocal tensor.** Solve the set of equations to find an estimate of T_i^{jk}.

5. **Computation of the epipoles.** Find a_4^i and b_4^i as the common normal to the left (respectively, right) null spaces of the three matrices T_1^{ij}, T_2^{ij} and T_3^{ij}.

6. **Computation of the Camera Matrices.** Solve for the remaining entries a_j^i and b_j^i using the method given in section 5:2.6

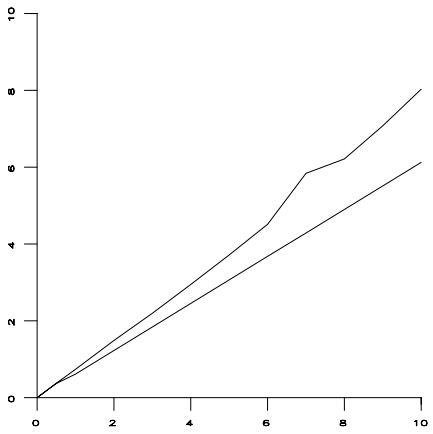
7. **Reconstruction.** Given the camera matrices, the points may be reconstructed using the triangulation methods of [33]. The lines may be reconstructed as described in section 5:2.7.

8. **Unscaling** The effect of the initial scaling and translation of the images may be undone by replacing the image coordinates by their original values, and making a corresponding adjustment to the computed camera matrices.

5:2.9 Experimental Evaluation of the Algorithm

This algorithm has been tested with synthetic and real data. First I describe the tests with synthetic data. A program was written to generate synthetic data, approximating the sort of data that would be taken with a 35mm camera with a standard 50mm lens. The images measured about 600×600 pixels. Between 0 and 10 pixels of noise was added to all three images.

Evaluation of the performance of this algorithm is still continuing, so we give here only a sample of the results obtained. The findings of the experiments will be given in the captions of the following graphs. The error is given in terms of residual 2D error, that is the RMS distance between the original points and the projection of the points obtained by reconstruction. Although this gives only an indirect measurement of the quality of the reconstruction, if the error is of the same order as the injected noise, then this is a good indication that the reconstruction is very good. The graphs all show injected noise on the horizontal axis, and residual error on the vertical axis.



Graph 1 : Residual error for 10 points. *This algorithm shows the results of reconstruction with ten points in three views. The upper curve is the linear algorithm, and*

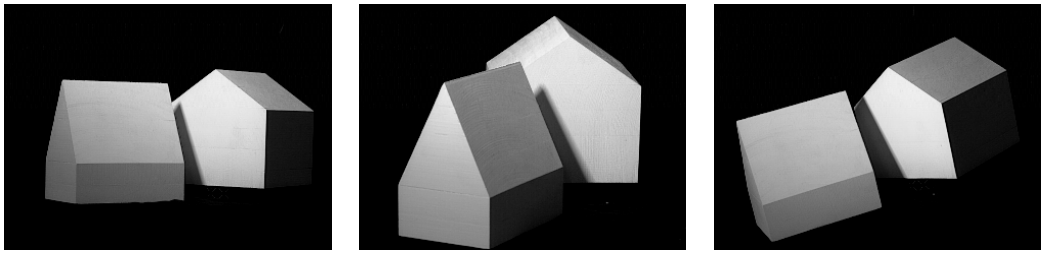
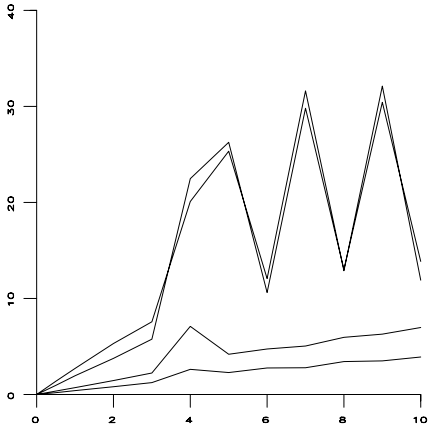


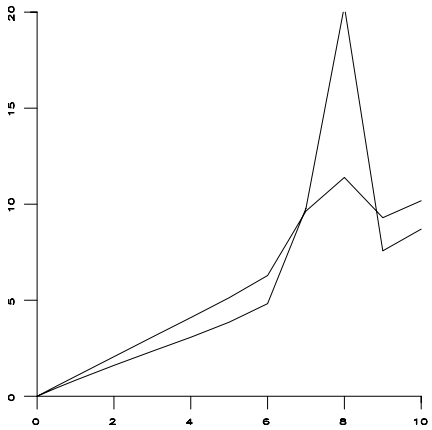
Figure 5.1: **Figure 1:** *Three photos of houses*

the lower curve is the iterative refinement. The residual errors are substantially smaller than the injected noise. The algorithm works well with as much as 10 pixels noise in each axial direction.

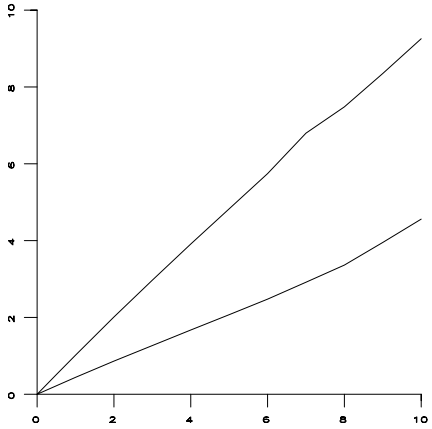


Graph 2 : 7 points and 10 lines. *The curves show the results both of the iterative algorithm and those obtained after iterative refinement of the results, using a Levenberg-Marquardt iterative least-squares algorithm. The graphs from the bottom are iterative algorithm line errors, iterative algorithm point errors and then linear algorithm line errors and point errors overlapping. The results after refinement are excellent, with the residual error being substantially less than the injected noise. The linear algorithm does not do quite so well, but the results are good, especially for noise levels less than 4 pixels.*

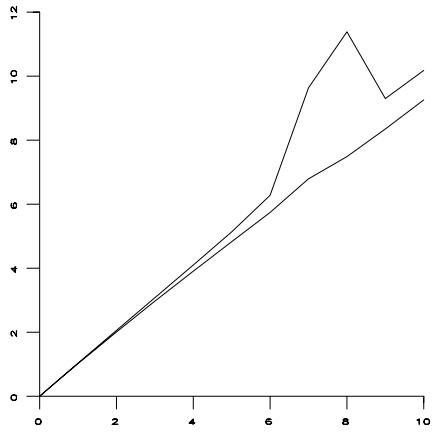
Next, we consider data obtained from a set of three images of a pair of wooden houses, shown in Fig 5.1. The dimension of the images was 640×480 pixels. The points and edges were extracted automatically and then matched between the images by hand. The coordinates of the image features were then corrected so that the match corresponded precisely with the projection model. This was so that noise could subsequently be injected in a controlled manner. There were a set of 15 lines and 13 points identifiable in the two images.



Graph 3 : Synthetic house data - linear. *This shows the residual error resulting from injection of data into the image measurements. The bottom curve shows the line error and the top curve, the point error. For the case of 8 pixels noise, the algorithm evidently hiccuped.*



Graph 4 : Synthetic house data - iterative *This shows the results obtained after refining the linear result with using Levenberg-Marquardt. Once more, the bottom curve represents the errors for lines. The results are very good, and the glitch is removed.*



Graph 5 : Synthetic house data - comparison *This compares the iterative (lower) and linear methods. Except for the case of 8 pixels, where the linear algorithm deviated slightly, there is very little difference. This indicates that the linear algorithm is very nearly as good as the optimum least-squares solution.*

Finally, the algorithm was run with the real uncorrected image data for the houses. The resulting errors were

1. For the linear algorithm : 1.05 pixels error for points, and 1.06 pixels for lines.
2. For the iterative refinement : 0.87 pixels for points, and 0.67 pixels for lines.

Thus, at least expressed in terms of residual error, we have achieved a very good reconstruction using the linear-algorithm, not very significantly improved using the iterative least-squares methods.

5:2.10 Lines specified by several points

In describing the reconstruction algorithm from lines, we have considered the case where lines are specified by their two end-points. Another common way that lines may be specified in an image is as the best line fit to several points. It will be shown now how that case may easily be reduced to the case of a line defined by two end points. Consider a set of points $\mathbf{u}_i$ in an image, and let $\lambda = (\lambda_1, \lambda_2, \lambda_3)^\top$ be a line, which we suppose is normalized such that $\lambda_1^2 + \lambda_2^2 = 1$. In this case, the distance from a point $\mathbf{u}_i$ to the line λ is equal to $\mathbf{u}_i{}^\top\lambda$. The squared distance may be written as $d^2 = \lambda^\top \mathbf{u}_i \mathbf{u}_i{}^\top \lambda$, and the sum-of-squares of all distances is

$$\sum_i \lambda^\top \mathbf{u}_i \mathbf{u}_i{}^\top \lambda = \lambda^\top (\sum_i \mathbf{u}_i \mathbf{u}_i{}^\top)\lambda \ .$$

The matrix $E = (\sum_i \mathbf{u}_i \mathbf{u}_i{}^\top)$ is positive-definite and symmetric.

As in the standard method for line-fitting, we may compute the line that minimizes the sum-of-squares distance $\lambda^\top E \lambda$ as follows. The line in question will be the one that minimizes $\lambda^\top E \lambda$ subject to the condition $\lambda_1^2 + \lambda_2^2 = 1$. Using the method of Lagrange

multipliers, this comes down to minimizing $\lambda^\top E\lambda - \xi(\lambda_1^2 + \lambda_2^2)$, where (since the conventional symbol λ is already in use) we denote the Lagrange coefficient by ξ. Now, taking the derivative with respect to λ and setting it to zero, we find that

$$2E\lambda - \xi \begin{pmatrix} 2\lambda_1 \\ 2\lambda_2 \\ 0 \end{pmatrix} = 0 \ .$$

This may be written as $(E - \xi J)\lambda = 0$. The minimizing solution is the line λ corresponding to the minimum root of the equation $\det(E - \xi J) = 0$. Let this minimum root be ξ_0.

Now, $E - \xi_0 J$ being symmetric and positive semi-definite may be written in the form $E - \xi_0 J = V\mathrm{diag}(r, s, 0)V^\top$ where V is an orthogonal matrix and r and s are positive. It follows that

$$\begin{aligned} E - \xi_0 J &= V\mathrm{diag}(r, 0, 0)V^\top + V\mathrm{diag}(s, 0, 0)V^\top \\ &= r\mathbf{v}_1{\mathbf{v}_1}^\top + s\mathbf{v}_2{\mathbf{v}_2}^\top \end{aligned}$$

where $\mathbf{v}_i$ is the i-th column of V. Therefore $E = \xi_0 J + r\mathbf{v}_1{\mathbf{v}_1}^\top + s\mathbf{v}_2{\mathbf{v}_2}^\top$. If (as we have assumed) $\lambda_1^2 + \lambda_2^2 = 1$, then

$$\begin{aligned} \sum_i ({\mathbf{u}_i}^\top\lambda)^2 &= \lambda^\top E\lambda \\ &= \xi_0 + (\sqrt{r}{\mathbf{v}_1}^\top\lambda)^2 + (\sqrt{s}{\mathbf{v}_2}^\top\lambda)^2 \ . \end{aligned}$$

Thus, we have replaced the sum-of-squares of several points by a constant value ξ_0, which is not capable of being minimized, plus the weighted sum-of-squares of the distances to two points $\mathbf{v}_1$ and $\mathbf{v}_2$.

In constructing the equations (5.6) for a line defined by a set of point, we use the pair of equations

$$\begin{aligned} \sqrt{r}{\mathbf{v}_1}^\top\lambda = 0 \\ \sqrt{s}{\mathbf{v}_2}^\top\lambda = 0 \end{aligned} \tag{5.13}$$

where λ defined by (5.5) is the transferred line. Similarly, in a full LM minimization, we may consider the line to be defined by the two weighted end points, $\sqrt{r}\mathbf{v}_1$ and $\sqrt{s}\mathbf{v}_2$.

5:2.11 Conclusions

The algorithm described in this section is unique in that no other linear method is known for handling both lines and points in a single non-iterative approach. The results obtained show indicate that this algorithm behaves very well in the presence of moderate amounts of noise. Compared with the two-view case, in which linear methods (such as the 8-point algorithm of Longuet-Higgins [42]) are quite sensitive to noise, the present algorithm gives superior results. This improvement may be attributed to the stabilizing influence of a third image. Shashua has made a similar observation ([66] and conversations). The advantage that this algorithm has over two view algorithms is that it applies to lines as well. Lines in images may often be computed with great precision. On the other hand, the algorithm benefits from the presence of points in the input data. The results

obtained using this algorithm are much better than those that I obtained previously with just lines. This is due to the use of mixed point and line data, and also to the improved algorithm for extracting the camera matrices, reported in section 5:2.6.

There are a few points that merit further investigation regarding this algorithm. The method for extracting the epipoles is probably the weak point of the algorithm, in that one must solve three consecutive linear equation sets in a row in order to find the epipoles. This is probably susceptible to noise degradation. Possibly better methods of finding the epipoles given three views may be found.

A second point regards the relative weighting of the line and point equations. The method of normalization of the data give some degree of standardization here, but since the two types of equations are derived separately, there is no obvious way to weight them so as to give equal emphasis to each sort of equation. At present, the algorithm seems to favour the lines, since residual line errors typically are smaller. Experiments with an adaptive weighting scheme are in progress.

The trifocal tensor seems to be a basic object in the analysis of the three-view situation. Apart from the use here described in projective scene reconstruction, it has also been used for point transfer and object recognition ([65]), and is suitable for line transfer as well, as shown in this section. Just as the fundamental matrix neatly encapsulates the geometry of the two-view case, the trifocal tensor serves a similar purpose for three views.

Finally, the perception of T_i^{jk} as a tensor (properly due to Vieville [76] and Shashua) though perhaps only a notational device, eases the formalism involved in the analysis.

5:3 Multilinear Relations

The fundamental matrix F and the trifocal tensor may be derived in a different manner that gives greater insight into their relationship. The following derivation is based on the work of Faugeras and Mourrain [14] and Triggs [73, 74].

5:3.1 Bilinear Relations

We consider first the relationship that holds between the coordinates of a point seen in two separate views. Thus, let $\mathbf{u} \leftrightarrow \mathbf{u}'$ be a pair of corresponding points as seen in two separate images. It will be convenient, for clarity of notation, to represent the two camera matrices by A and B, instead of the usual notation, P and P'. Both the points $\mathbf{u}$ and $\mathbf{u}'$ are images of the same point $\mathbf{x}$ in space. For convenience, we write

$$\mathbf{u} = \begin{pmatrix} u^1 \\ u^2 \\ u^3 \end{pmatrix} \quad ; \quad \mathbf{u}' = \begin{pmatrix} u'^1 \\ u'^2 \\ u'^3 \end{pmatrix} \quad ; \quad \mathbf{x} = \begin{pmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \end{pmatrix} \tag{5.14}$$

The projection from space to image can now be expressed as follows.

$$k \begin{pmatrix} u^1 \\ u^2 \\ u^3 \end{pmatrix} = A\mathbf{x}$$

$$k'\begin{pmatrix} u'^1 \\ u'^2 \\ u'^3 \end{pmatrix} = B\mathbf{x} \tag{5.15}$$

where k and k' are two undetermined constants.

These equations may be written down in one matrix equation. In order to do this, we denote the i-th row of the matrix A by $\mathbf{a}^i_{\cdot}$, and similarly the i-th row of matrix B by $\mathbf{b}^i_{\cdot}$. The projection due to the first camera may then be written as

$$\begin{bmatrix} \mathbf{a}^1_{\cdot} & u^1 \\ \mathbf{a}^2_{\cdot} & u^2 \\ \mathbf{a}^3_{\cdot} & u^3 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ -k \end{pmatrix} = 0 \ . \tag{5.16}$$

This expression may be compared with (5.15) which is is just another way of writing the same thing.

The projections of the point $\mathbf{x}$ into both views may be expressed as a single matrix equation by writing the equations derived from one view and derived from the other view in the same equation. This gives a set of equations

$$\left[\begin{array}{cc} \mathbf{a}^1_{\cdot} & u^1 \\ \mathbf{a}^2_{\cdot} & u^2 \\ \mathbf{a}^3_{\cdot} & u^3 \\ \hline \mathbf{b}^1_{\cdot} & u'^1 \\ \mathbf{b}^2_{\cdot} & u'^2 \\ \mathbf{b}^3_{\cdot} & u'^3 \end{array}\right] \begin{pmatrix} \mathbf{x} \\ -k \\ -k' \end{pmatrix} = 0 \tag{5.17}$$

Now, this is a 6×6 set of equations which by hypothesis has a non-zero solution, the vector $(\mathbf{x}, -k, -k')^\top$. It follows that the matrix of coefficients in (5.17) must have zero determinant. This condition leads to a bilinear relationship between the entries of the vectors $\mathbf{u}$ and $\mathbf{u}'$ expressed by the fundamental matrix F. We will now look specifically at the form of this relationship.

Consider the matrix appearing in (5.17). Denote it by X. The determinant of X may be written as an expression in terms of the quantities u^i and u'^i. Notice that the entries u^i and u'^i appear in only two columns of X. This implies that the determinant of X may be expressed as a quadratic expression in terms of the u^i and u'^i. In fact, since all the entries u^i appear in the same column, there can be no terms of the form $u^i u^j$ or $u'^i u'^j$. Briefly, as an expression in terms of the u^i and u'^i, the determinant of X is a bilinear expression. The fact that the determinant is zero may be written as an equation

$$(u'^1, u'^2, u'^3)F(u^1, u^2, u^3)^\top = u^i u'^j F_{ji} = 0 \tag{5.18}$$

where F is a 3×3 matrix, the well-known fundamental matrix[8].

We may compute a specific formula for the entries of the matrix F as follows. The entry F_{ij} of F is the coefficient of the term $u'^i u^j$ in the expansion of the determinant of X. In order to find this coefficient, we must eliminate the rows and columns of the matrix

[8]Here and elsewhere we use the tensor summation convention that an index repeated in upper (contravariant) and lower (covariant) positions implies summation over the range of indices. It may be more sensible to define F_{ij} by the formula $u^i u'^j F_{ij} = 0$, but the formula (5.18) is conventional.

containing u'^i and u^j, take the determinant of the resulting matrix and multiply by ± 1 as appropriate. For instance, the coefficient of $u'^1 u^1$ is obtained by eliminating two rows and the last two columns of the matrix X as shown in (5.17). The remaining matrix is

$$\begin{pmatrix} \mathbf{a}^2_\cdot \\ \mathbf{a}^3_\cdot \\ \mathbf{b}^2_\cdot \\ \mathbf{b}^3_\cdot \end{pmatrix}$$

and the coefficient of $u'^1 u^1$ is equal to the determinant of this 4×4 matrix. In general, we may write

$$F_{ji} = (-1)^{i+j} \det \begin{bmatrix} \sim \mathbf{a}^i_\cdot \\ \sim \mathbf{b}^j_\cdot \end{bmatrix} \quad . \tag{5.19}$$

In this expression, the notation $\sim \mathbf{a}^i_\cdot$ has been used to denote the matrix obtained from A by **omitting** the row $\mathbf{a}^i_\cdot$. Thus the symbol $\sim$ may be read as *omit*, and $\sim \mathbf{a}^i_\cdot$ represents two rows of A. The determinant appearing on the right side of (5.19) is therefore a 4×4 determinant. This expression for the fundamental matrix was pointed out to me by Rajiv Gupta, and is also noted by Carlsson ([8]).

A different way of writing the expression for F_{ji} makes use of the tensor ϵ_{rst} which is defined to be 0 unless all of r, s and t are different, and is ± 1 depending on whether the indices (r, s, t) constitute an even or odd permutation of $(1, 2, 3)$. 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.

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

Using this notation, one may derive the following formula.

$$F_{ji} = \left(\frac{1}{4}\right) \epsilon_{ipq} \epsilon_{jrs} \det \begin{bmatrix} \mathbf{a}^p_\cdot \\ \mathbf{a}^q_\cdot \\ \mathbf{b}^r_\cdot \\ \mathbf{b}^s_\cdot \end{bmatrix} \quad . \tag{5.20}$$

To see this, note that F_{ji} is defined in (5.20) in terms of a sum of determinants over all values of p, q, r and s. However for a given value of i, the tensor ϵ_{ipq} is zero unless p and q are different from i and from each other. This leaves only two remaining choices of p and q (for example if $i = 1$, then we may choose $p = 2, q = 3$ or $p = 3, q = 2$). Similarly, there are only two different choices of r and s giving rise to non-zero terms. Thus the sum consists of 4 non-zero terms only. Furthermore, the determinants appearing in these four terms consists of the same four rows of the matrices A and B and hence have equal values, except for sign. However, the value of $\epsilon_{ipq}\epsilon_{jrs}$ is such that the four terms all have the same sign and are equal. Thus, the sum (5.20) is equal to the single term appearing in (5.19).

A similar formula involving the fundamental matrix is

$$F_{ji} \epsilon^{ipq} \epsilon^{jrs} = \det \begin{bmatrix} \mathbf{a}^p_\cdot \\ \mathbf{a}^q_\cdot \\ \mathbf{b}^r_\cdot \\ \mathbf{b}^s_\cdot \end{bmatrix} \quad . \tag{5.21}$$

This formula may be derived in a straight-forward manner from (5.20).

Invariants of Lines

In this brief section it will be shown how the fundamental matrix may be used to define invariants of spatial objects (in this particular case, lines) in terms of the images of those objects in a pair of images. This method was discovered by Carlsson ([8]). Given two lines λ and μ in one image, and the corresponding lines λ' and μ' in the other image. From (5.21) we may see that

$$\begin{aligned}(\lambda' \times \mu')^\top F(\lambda \times \mu) &= F_{ji}(\lambda_p \mu_q \epsilon^{ipq})(\lambda'_r \mu'_s \epsilon^{jrs}) \\ &= \lambda_p \mu_q \lambda'_r \mu'_s \det \begin{bmatrix} \mathbf{a}^p_{\cdot} \\ \mathbf{a}^q_{\cdot} \\ \mathbf{b}^r_{\cdot} \\ \mathbf{b}^s_{\cdot} \end{bmatrix} \\ &= \det \begin{bmatrix} \lambda_p \mathbf{a}^p_{\cdot} \\ \mu_q \mathbf{a}^q_{\cdot} \\ \lambda'_r \mathbf{b}^r_{\cdot} \\ \mu'_s \mathbf{b}^s_{\cdot} \end{bmatrix} \\ &= \det \left[A^\top \lambda, A^\top \mu, B^\top \lambda', B^\top \mu' \right] \end{aligned} \tag{5.22}$$

The cross-products on the left side of this sequence of equations represent the point of intersection of the lines in the two images. A term such as $A^\top \lambda$ on the right represents a plane in space that projects via camera matrix A onto the line λ. We now consider four line correspondences in two views. For $i = 1, \ldots, 4$ let $\lambda^{(i)} \leftrightarrow \lambda'^{(i)}$ be the i-th line correspondence, where the upper index indicating the line number is put in parentheses to emphasize that it is not a tensorial index. We denote $\det \left[A^\top \lambda^{(i)}, A^\top \lambda^{(j)}, B^\top \lambda'^{(i)}, B^\top \lambda'^{(j)} \right]$ by I_{ij}. It was shown in [25] that the homogeneous vector

$$I = (I_{12} I_{34}, I_{13} I_{24}, I_{14} I_{23}) \tag{5.23}$$

is a complete projective invariant of the four lines in space corresponding to the matched lines in the images. According to (5.22), we may write $I_{ij} = (\lambda'^{(i)} \times \lambda'^{(j)})^\top F(\lambda^{(i)} \times \lambda^{(j)})$ Substituting this formula into (5.23) yields a neat formula due to Carlsson ([8]) for the projective invariants of four lines in space, in terms of their projections in two views.

5:3.2 Trilinear relations

The basic idea behind the derivation of the fundamental matrix can be used to derive relationships between the coordinates of points seen in three views. This analysis results in the definition of a triply-indexed tensor, known as the trifocal tensor, with one covariant and two contravariant indices. Unlike the Fundamental Matrix, this trifocal tensor relates both lines and points in the three images. We begin by describing the way matching points are related by the trifocal tensor.

Point relations

Consider a point correspondence across three views : $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}''$. Let the third camera matrix be C and let $\mathbf{c}^i_{\cdot}$ be its i-th row. Analogous to (5.17) we can write an equation

describing the projection of a point $\mathbf{x}$ into the three images.

$$\left[\begin{array}{cccc} \mathbf{a}_{\cdot}^1 & u^1 & & \\ \mathbf{a}_{\cdot}^2 & u^2 & & \\ \mathbf{a}_{\cdot}^3 & u^3 & & \\ \hline \mathbf{b}_{\cdot}^1 & & u'^1 & \\ \mathbf{b}_{\cdot}^2 & & u'^2 & \\ \mathbf{b}_{\cdot}^3 & & u'^3 & \\ \hline \mathbf{c}_{\cdot}^1 & & & u''^1 \\ \mathbf{c}_{\cdot}^2 & & & u''^2 \\ \mathbf{c}_{\cdot}^3 & & & u''^3 \end{array}\right] \begin{pmatrix} \mathbf{x} \\ -k \\ -k' \\ -k'' \end{pmatrix} = 0 \ . \tag{5.24}$$

This matrix, which as before we will call X, has 9 rows and 7 columns. From the existence of a solution to this set of equations, we deduce that its rank must be at most 6. Hence any 7×7 minor has zero determinant. This fact gives rise to the trilinear relationships that hold between the coordinates of the points $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$.

There are essentially two different types of 7×7 minors of X. In choosing 7 rows of X, we may choose either

1. Three rows from each of two camera matrices and one row from the third, or
2. Three rows from one camera matrix and two rows from each of the two others.

Let us consider the first type. A typical such 7×7 minor of X is of the form

$$\left[\begin{array}{cccc} \mathbf{a}_{\cdot}^1 & u^1 & & \\ \mathbf{a}_{\cdot}^2 & u^2 & & \\ \mathbf{a}_{\cdot}^3 & u^3 & & \\ \hline \mathbf{b}_{\cdot}^1 & & u'^1 & \\ \mathbf{b}_{\cdot}^2 & & u'^2 & \\ \mathbf{b}_{\cdot}^3 & & u'^3 & \\ \hline \mathbf{c}_{\cdot}^i & & & u''^i \end{array}\right] \ . \tag{5.25}$$

Note that this matrix contains only one entry in the last column, namely u''^i. Expanding the determinant by cofactors down this last column reveals that the determinant is equal to

$$u''^i \det \left[\begin{array}{cccc} \mathbf{a}_{\cdot}^1 & u^1 & & \\ \mathbf{a}_{\cdot}^2 & u^2 & & \\ \mathbf{a}_{\cdot}^3 & u^3 & & \\ \hline \mathbf{b}_{\cdot}^1 & & u'^1 & \\ \mathbf{b}_{\cdot}^2 & & u'^2 & \\ \mathbf{b}_{\cdot}^3 & & u'^3 & \end{array}\right] \ .$$

Apart from the factor u''^i, this just leads to the bilinear relationship expressed by the fundamental matrix, as discussed in section 5:3.1.

The other sort of 7×7 minor is of more interest. An example of such a determinant is of the form

$$\det \left[\begin{array}{cc|cc|cc} \mathbf{a}_{\cdot}^{1} & u^{1} & & & & \\ \mathbf{a}_{\cdot}^{2} & u^{2} & & & & \\ \mathbf{a}_{\cdot}^{3} & u^{3} & & & & \\ \hline \mathbf{b}_{\cdot}^{j} & & u'^{j} & & & \\ \mathbf{b}_{\cdot}^{l} & & u'^{l} & & & \\ \hline \mathbf{c}_{\cdot}^{k} & & & & u''^{k} & \\ \mathbf{c}_{\cdot}^{m} & & & & u''^{m} & \end{array}\right] . \tag{5.26}$$

By the same sort of argument as with the bilinear relations one sees that this leads to a trilinear relation of the form $\det X = f(\mathbf{u}, \mathbf{u}', \mathbf{u}'') = 0$. By expanding this determinant down the column containing u^i, one can find a specific formula for $\det X$, namely

$$\det X = \pm\frac{1}{2} u^i u'^j u''^k \epsilon_{ilm}\epsilon_{jqx}\epsilon_{kry} \det \left[\begin{array}{c} \mathbf{a}_{\cdot}^{l} \\ \mathbf{a}_{\cdot}^{m} \\ \mathbf{b}_{\cdot}^{q} \\ \mathbf{c}_{\cdot}^{r} \end{array}\right] = 0_{xy} \tag{5.27}$$

where x and y are free indices corresponding to the rows omitted from the matrices B and C to produce (5.26). We introduce the tensor

$$T_i^{qr} = \frac{1}{2}\epsilon_{ilm} \det \left[\begin{array}{c} \mathbf{a}_{\cdot}^{l} \\ \mathbf{a}_{\cdot}^{m} \\ \mathbf{b}_{\cdot}^{q} \\ \mathbf{c}_{\cdot}^{r} \end{array}\right] \tag{5.28}$$

The trilinear relationship (5.27) may then be written

$$u^i u'^j u''^k \epsilon_{jqx}\epsilon_{kry} T_i^{qr} = 0_{xy} \ . \tag{5.29}$$

The tensor T_i^{qr} is the trifocal tensor, and (5.29) is Shashua's trilinear relation. The indices x and y are free indices, and each choice of x and y leads to a different trilinear relation.

Just as in the case of the fundamental matrix, one may write the formula for the tensor T_i^{qr} in a slightly different way :

$$T_i^{qr} = (-1)^{i+1} \det \left[\begin{array}{c} \sim \mathbf{a}_{\cdot}^{i} \\ \mathbf{b}_{\cdot}^{q} \\ \mathbf{c}_{\cdot}^{r} \end{array}\right] \ . \tag{5.30}$$

As in section 5:3.1, the expression $\sim \mathbf{a}^i$ means the matrix A with row i omitted. Note that we omit row i from the first camera matrix, but *include* rows q and r from the other two camera matrices.

In the often-considered case where the first camera matrix A has the canonical form $[I \mid 0]$, the expression (5.30) for the trifocal tensor may be written simply ([26, 67]) :

$$T_i^{qr} = b_i^q c_4^r - b_4^q c_i^r \ . \tag{5.31}$$

Note that there are in fact 27 possible trilinear relations that may be formed in this way (refer to (5.26)). Specifically, note that each relation arises from taking all three rows

from one camera matrix along with two rows from each of the other two matrices. This gives the following computation.

- 3 ways to choose the first camera matrix from which to take all three rows.
- 3 ways to choose the row to omit from the second camera matrix.
- 3 ways to choose the row to omit from the third camera matrix.

This gives a total of 27 trilinear relations. However, among the 9 ways of choosing two rows from the second and third camera matrices, only 4 are linearly independent. This means that there are a total of 12 linearly independent trilinear relations.

It is important to distinguish between the number of trilinear relations, however, and the number of different trifocal tensors. As is shown by (5.29), several different trilinear relations may be expressed in terms of just one trifocal tensor. In (5.29) each distinct choice of the free indices x and y gives rise to a different trilinear relation, all of which are expressible in terms of the same trifocal tensor T_i^{qr}. On the other hand, in the definition of the trifocal tensor given in (5.28), the camera matrix A is treated differently from the other two, in that A contributes two rows (after omitting row i) to the determinant defining any given entry of T_i^{qr}, whereas the other two camera matrices contribute just one row. This means that there are in fact three different trifocal tensors corresponding to the choice of which of the three camera matrices contributes two rows.

Line relations

A line in an image is represented by a covariant vector λ_i, and the condition for a point $\mathbf{u}$ to lie on the line is that $\lambda_i u^i = 0$. Let x^j represent a point in space, and a^i_j represent a camera matrix. The 3D point x^j is mapped to the image point $u^i = a^i_j x^j$. It follows that the condition for the point x^j to project to a point on the line λ_i is that $\lambda_i a^i_j x^j = 0$. Another way of looking at this is that $\lambda_i a^i_j$ represents a plane consisting of all points that project onto the line λ_i. Once more, the condition for the point x^j to line on this plane is that $\lambda_i a^i_j x^j = 0$.

Consider the situation where a point x^j maps to a point u^i in one image and to some point on lines λ'_q and λ''_r in two other images. This may be expressed by equations

$$\begin{aligned} u^i &= k a^i_j x^j \\ \lambda'_q b^q_j x^j &= 0 \\ \lambda''_r c^r_j x^j &= 0 \end{aligned}$$

This may be written as a single matrix equation of the form

$$\left[\begin{array}{cc} \mathbf{a}^1_{\cdot} & u^1 \\ \mathbf{a}^2_{\cdot} & u^2 \\ \mathbf{a}^3_{\cdot} & u^3 \\ \hline \lambda'_q \mathbf{b}^q_{\cdot} & 0 \\ \hline \lambda''_r \mathbf{c}^r_{\cdot} & 0 \end{array}\right] \begin{pmatrix} \mathbf{x} \\ -k \end{pmatrix} = 0 \ . \tag{5.32}$$

Since this set of equations has a solution, one deduces that $\det X = 0$, where X is the matrix on the left of the equation. Expanding this determinant down the last column gives

$$\begin{aligned} 0 = \det X &= \frac{1}{2} u^i \epsilon_{ilm} \det \begin{bmatrix} \mathbf{a}_\cdot^l \\ \mathbf{a}_\cdot^m \\ \lambda'_q \mathbf{b}_\cdot^q \\ \lambda''_r \mathbf{c}_\cdot^r \end{bmatrix} \\ &= \frac{1}{2} u^i \lambda'_q \lambda''_r \epsilon_{ilm} \det \begin{bmatrix} \mathbf{a}_\cdot^l \\ \mathbf{a}_\cdot^m \\ \mathbf{b}_\cdot^q \\ \mathbf{c}_\cdot^r \end{bmatrix} \\ &= u^i \lambda'_q \lambda''_r T_i^{qr} \end{aligned} \tag{5.33}$$

This shows the connection of the trifocal tensor with sets of lines. The two lines λ'_q and λ''_r back project to planes meeting in a line in space. The image of this line in the first image is a line, which may be represented by λ_i. For any point u^i on that line the relation (5.33) holds. It follows that $\lambda'_q \lambda''_r T_i^{qr}$ is the representation of the line λ_i. Thus, we see that for three corresponding lines in the three images:

$$\lambda_p \approx \lambda'_q \lambda''_r T_p^{qr} \tag{5.34}$$

The symbol $\approx$ means that the two sides are equal up to a scale factor. Since the two sides of the relation (5.34) are vectors, this may be interpreted as meaning that the vector product of the two sides vanishes. Expressing this vector product using the tensor ϵ^{ijk}, we arrive at an equation

$$\lambda_p \lambda'_q \lambda''_r \epsilon^{ipw} T_i^{qr} = 0^w \quad . \tag{5.35}$$

In an analogous manner to the derivation of (5.29) and (5.33) one may derive a relationship between corresponding points in two images and a line in a third image. In particular, if a point x^j in space maps to points u^i and u'^i in the first two images, and to some point on a line λ''_r in the third image, then one may derive a relation

$$u^i u'^j \lambda''_r \epsilon_{jqx} T_i^{qr} = 0_x \quad . \tag{5.36}$$

In this relation, the index x is free, and there is one such relation for each choice of $x = 1, \ldots, 3$, of which two are linearly independent.

We can summarize the results of this section in the following table, in which the final column denotes the number of linearly independent equations.

Correspondence	Relation	number of equations
three points	$u^i u'^j u''^k \epsilon_{jqx} \epsilon_{kry} T_i^{qr} = 0_{xy}$	4
two points, one line	$u^i u'^j \lambda''_r \epsilon_{jqx} T_i^{qr} = 0_x$	2
one point, two lines	$u^i \lambda'_q \lambda''_r T_i^{qr} = 0$	1
three lines	$\lambda_p \lambda'_q \lambda''_r \epsilon^{piw} T_i^{qr} = 0^w$	2

Table 1. Trilinear Relations

Note how the different equation sets are related to each other. For instance, the second line of the table is derived from the first by replacing $u'^k\epsilon_{kry}$ by the line λ''_r and deleting the free index y.

Point relation as a special case of Line relations

It will now be shown that the trilinear relation (5.29) is in fact nothing but a special case of the trilinear relation (5.33) for lines. In the trilinear relation $u^i u'^j u''^k \epsilon_{jqx}\epsilon_{kry} T_i^{qr} = 0_{xy}$ for points, we may write $\lambda'_{qx} = u'^j\epsilon_{jqx}$ and $\lambda''_{ry} = u''^k\epsilon_{kry}$. The trilinear relation then becomes

$$u^i \lambda'_{qx}\lambda''_{ry} T_i^{qr} = 0_{xy} \tag{5.37}$$

which is beginning to looking much like the trilinear relation (5.33) for lines. Observe as before that the indices x and y are free variables in this expression. We will now show that for any choice of the free variables x and y, the terms λ'_{qx} and λ''_{ry} do represent geometrically meaningful lines. We concentrate on λ'_{qx}, since the analysis for λ''_{ry} is identical.

Consider the case $x = 1$. Then $\lambda'_{q1} = u'^j\epsilon_{jq1}$, and expanding this out, we see that $(\lambda'_{11}, \lambda'_{21}, \lambda'_{31}) = (0, -u'^3, u'^2)$. We see that $\lambda'_{.1}$ is a line passing through the point $u'^j = (u'^1, u'^2, u'^3)$ and parallel with the first coordinate axis. A similar thing occurs for the other choices $x = 2, 3$. Specifically, $\lambda'_{.2}$ is the line through u'^j parallel with the second coordinate axis, and $\lambda'_{.3}$ is the line through u'^j passing also through the coordinate origin, $(0, 0, 1)$.

What is all this saying ? Let $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}''$ be corresponding points in three image. If we choose any lines λ' and λ'' that pass respectively through the two points $\mathbf{u}'$ and $\mathbf{u}''$ then from (5.33) we obtain a relation $u^i\lambda'_j\lambda''_k T_i^{jk} = 0$. For any arbitrary choice of the lines λ' and λ'' we can write down such a relation. In the particular case where the lines λ' and λ'' are chosen to be lines through $\mathbf{u}'$ and $\mathbf{u}''$ either horizontal, vertical or passing through the origin, then one obtains Shashua's trilinear relation for points (5.29). One may choose two lines through each of the points $\mathbf{u}'$ and $\mathbf{u}''$, resulting in four independent trilinear relations.

This interpretation of the point relationships has been previously observed in [62] and [14].

5:3.3 Quadrilinear Relations

Similar arguments work in the case of four views. Once more, consider a point correspondence across 4 views : $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}'' \leftrightarrow \mathbf{u}'''$. With camera matrices A, B, C and D,

the projection equations may be written as

$$\left[\begin{array}{cccc} \mathbf{a}_{\cdot}^1 & u^1 & & \\ \mathbf{a}_{\cdot}^2 & u^2 & & \\ \mathbf{a}_{\cdot}^3 & u^3 & & \\ \hline \mathbf{b}_{\cdot}^1 & & u'^1 & \\ \mathbf{b}_{\cdot}^2 & & u'^2 & \\ \mathbf{b}_{\cdot}^3 & & u'^3 & \\ \hline \mathbf{c}_{\cdot}^1 & & & u''^1 \\ \mathbf{c}_{\cdot}^2 & & & u''^2 \\ \mathbf{c}_{\cdot}^3 & & & u''^3 \\ \hline \mathbf{d}_{\cdot}^1 & & & u'''^1 \\ \mathbf{d}_{\cdot}^2 & & & u'''^2 \\ \mathbf{d}_{\cdot}^3 & & & u'''^3 \end{array}\right] \left(\begin{array}{c} \mathbf{x} \\ -k \\ -k' \\ -k'' \\ k''' \end{array}\right) = 0 \ . \tag{5.38}$$

Since this equation has a solution, the matrix X on the left has rank at most 7, and so all 8×8 determinants are zero. As in the trilinear case, any determinants containing only one row from one of the camera matrices gives rise to a trilinear or bilinear relation between the remaining cameras. A different case occurs when we consider 8×8 determinants containing two rows from each of the camera matrices. Such a determinant leads to a new quadrilinear relationship of the form

$$u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} Q^{pqrs} = 0_{wxyz} \tag{5.39}$$

where each choice of the free variables w, x, y and z gives a different equation, and the 4-dimensional tensor Q^{pqrs} is defined by

$$Q^{pqrs} = \det \left[\begin{array}{c} \mathbf{a}_{\cdot}^p \\ \mathbf{b}_{\cdot}^q \\ \mathbf{c}_{\cdot}^r \\ \mathbf{d}_{\cdot}^s \end{array}\right] \tag{5.40}$$

Note that the four indices of the four-view tensor are contravariant, and there is no distinguished view as there is in the case of the trifocal tensor. There is only one four-view tensor corresponding to four given views, and this one tensor gives rise to 81 different quadrilinear relationships, of which 16 are linearly independent.

As in the case of the trifocal tensor, there are also relations between fixed lines and points in the case of the four-view tensor. This may be summarized in the following table.

Correspondence	Relation	number of equations
4 points	$u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} Q^{pqrs} = 0_{wxyz}$	16
3 points, 1 line	$u^i u'^j u''^k \lambda'''_s \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry} Q^{pqrs} = 0_{wxy}$	8
2 points, 2 lines	$u^i u'^j \lambda''_r \lambda'''_s \epsilon_{ipw}\epsilon_{jqx} Q^{pqrs} = 0_{wx}$	4
1 points, 3 lines	$u^i \epsilon_{ipw} \lambda'_q \lambda''_r \lambda'''_s Q^{pqrs} = 0_w$	2
4 lines	$\lambda_p \lambda'_q \lambda''_r \lambda'''_s Q^{pqrs} = 0$	1

Table 2. Quadrilinear Relations.

As in the case of the trilinear relationships, equations relating points are really just special cases of the relationship for lines.

5:3.4 Number of Independent Equations

It was asserted in considering the definition of the quadrifocal tensor Q^{pqrsl} that each point correspondence gives rise to 16 linearly independent equations. Similarly each point correspondence across three views gives rise to four linearly independent equations in the entries of the trifocal tensor T_i^{qr}. We now examine this point more closely. We begin with the four view case.

Four View Case

Given sufficiently many point matches across four views, one may solve for the tensor Q^{pqrs}. Subsequently, one may retrieve the camera matrices and carry out projective reconstruction. Details of how the step of retrieving the camera matrices is done are omitted here, but are given by Heyden ([35, 34]). A curious phenomenon occurs however, when one counts the number of point matches necessary to do this. As indicated above, it appears to be the case that each point match gives 16 linearly independent equations in the entries of the tensor Q^{pqrs}. On the other hand, it seems unlikely that the equations derived from two totally unrelated sets of point correspondences could have any dependencies. It would therefore appear that from 5 point correspondences one obtains 80 equations, which is enough to solve for the entries of Q^{pqrs} up to scale. From this argument it would appear that it is possible to solve for the tensor from only 5 point matches across 4 views, and thence one may solve for the camera matrices, up to the usual projective ambiguity. This conclusion however is contradicted by the following remark.

Proposition 5.2. *It is not possible to determine the positions of 4 (or any number of) cameras from the images of 5 points.*

Proof. Since any two sets of five points in $\mathcal{P}^3$are projectively equivalent (barring the case where 3 points are in a plane), we may assume that the five points form a projective basis for $\mathcal{P}^3$. Consider the first camera. The situation is that each point $\mathbf{x}_i$ is known for $i = 1, \ldots, 5$, and the images $\mathbf{u}_i$ of the points are also known. However, each such 3D to 2D correspondence gives two linear equations in the entries of the camera matrix M, a total of 10 equations in all from the 5 points. Since M has 11 degrees of freedom, it can not be determined uniquely from 10 equations. This is in agreement with the observation of Sutherland ([72]) that $5\frac{1}{2}$ such 3D to 2D correspondences are required to determine M. This means that the camera matrix M is not determined uniquely with respect to a fixed projective basis. The same applies to the other cameras, and thus the proposition is demonstrated. □

Obviously there is some error in our counting of equations. In fact, Heyden states ([35]) that six point correspondences are necessary to compute Q^{pqrs}. The truth is that our counting argument is false, as is shown by the following two propositions.

Proposition 5.3. *Consider a single point correspondence* $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}'' \leftrightarrow \mathbf{u}'''$ *across four views. Letting the four free indices* w, x, y *and* z *in (5.39) vary from 1 to 3 one obtains from this correspondence a set of 81 equations in the entries of* Q^{pqrs}. *The rank of this set of equations is 16. Furthermore, let the equations be written as* $A\mathbf{q} = 0$ *where* A *is an* 81×81 *matrix and* $\mathbf{q}$ *is a vector containing the entries of* Q^{pqrs}. *Then the 16 non-zero singular values of* A *are all equal.*

What this result is saying is that indeed as expected one obtains 16 linearly independent equations from one point correspondence, and in fact it is possible to reduce this set of equations by an orthogonal transform (multiplication of the equation matrix A on the left by an orthogonal matrix U) to a set of orthogonal equations. The rank of the set of equations is a "very solid" 16. This is a very favourable result as far as the conditioning of the problem is concerned.

The key point in the proof of Proposition 5.3 concerns the singular values of a skew-symmetric matrix.

Lemma 5.4. *A* 3×3 *skew-symmetric matrix has two equal non-zero singular values.*

Although this is well known, a brief proof is given.

Proof. Defining matrices

$$E = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ; \quad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad Z = ED = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

the Schurr normal form ([15]) of a 3×3 skew-symmetric matrix S can be written $S = kVZV^\top$ where k is a scalar. Thus the SVD of S is given by $S = kUDV^\top$ where $U = VE$. □

The rest of the proof of Proposition 5.3 is quite straight-forward as long as one does not get lost in notation.

Proof. (Proposition 5.3) The full set of 81 equations derived from a single point correspondence is of the form $u^i \epsilon_{ipw} u'^j \epsilon_{jqx} u''^k \epsilon_{kryv} u'''^l \epsilon_{lsz} Q^{pqrs} = 0_{wxyz}$. A total of 81 equations are generated by varying w, x, y, z over the range $1, \ldots, 3$. Thus, the equation matrix A may be written as

$$A_{(wxyz)(pqrs)} = u^i \epsilon_{ipw} u'^j \epsilon_{jqx} u''^k \epsilon_{kry} u'''^l \epsilon_{lsz} \tag{5.41}$$

where the indices $(wxyz)$ index the row and $(pqrs)$ index the column of A. We will have occasion frequently to consider a set of indices, such as $(wxyz)$ in this case, as a single index for the row or column of a matrix. This situation will be indicated by enclosing the indices in parentheses as here, and referring to them as a *combined index*.

We consider now the expression $u^i \epsilon_{ipw}$. This may be considered as a matrix indexed by the free indices p and w. Furthermore, since $u^i \epsilon_{ipw} = -u^i \epsilon_{iwp}$ we see that it is a skew-symmetric matrix, and hence has equal singular values. We denote this matrix by S_{wp}. Writing the result of Lemma 5.4, using tensor notation we have

$$U_a^w S_{wp} V_e^p = k D_{ae} \tag{5.42}$$

where the matrix D is as in Lemma 5.4. Now, the matrix A in (5.41) may be written as $A_{(wxyz)(pqrs)} = S_{wp}S'_{xq}S''_{yr}S'''_{zs}$. Consequently, applying (5.42) we may write

$$U_a^w U_b'^x U_c''^y U_d'''^z A_{(wxyz)(pqrs)} V_e^p V_f'^q V_g''^r V_h'''^s = kk'k''k''' D_{ae} D_{bf} D_{cg} D_{dh} \ . \tag{5.43}$$

Now, writing

$$\hat{U}^{(wxyz)}_{(abcd)} = U_a^w U_b'^x U_c''^y U_d'''^z$$

$$\hat{V}^{(pqrs)}_{(efgh)} = V_e^p V_f'^q V_g''^r V_h'''^s$$

$$\hat{D}_{(abcd)(efgh)} = D_{ae} D_{bf} D_{cg} D_{dh}$$

and

$$\hat{k} = kk'k''k'''$$

we see that (5.43) may be written as

$$\hat{U}^{(wxyz)}_{(abcd)} A_{(wxyz)(pqrs)} \hat{V}^{(pqrs)}_{(efgh)} = \hat{k}\hat{D}_{(abcd)(efgh)} \ . \tag{5.44}$$

As a matrix, $D_{(abcd)(efgh)}$ is diagonal with 16 non-zero diagonal entries, all equal to unity. To show that (5.44) is the SVD of the matrix $A_{(pqrs)(tuvw)}$, and hence to complete the proof, it remains only to show that $U^{(wxyz)}_{(abcd)}$ and $V^{(pqrs)}_{(efgh)}$ are orthogonal matrices. To this end, we show that $U^{(wxyz)}_{(abcd)}$ has unit norm orthogonal columns. Thus, for two columns with combined indices $(pqrs)$ and $(tuvw)$ respectively, we verify

$$\begin{aligned}
\sum_{i,j,k,l} \hat{U}^{(ijkl)}_{(pqrs)} \hat{U}^{(ijkl)}_{(tuvw)} &= \sum_{i,j,k,l} (U_p^i U_q'^j U_r''^k U_s'''^l)(U_t^i U_u'^j U_v''^k U_w'''^l) \\
&= \sum_i U_p^i U_t^i \sum_j U_q'^j U_u'^j \sum_k U_r''^k U_v''^k \sum_l U_s'''^l U_w'''^l \\
&= \delta_{pt}\delta_{qu}\delta_{rv}\delta_{sw} \\
&= \delta_{(pqrs)(tuvw)}
\end{aligned}$$

and so $\hat{U}$ is orthogonal, as required. A similar argument shows that $\hat{V}$ is orthogonal as well. This completes the proof that A has rank 16, and all non-zero singular values are equal. □

Thus, each point correspondence gives 16 equations. The surprising fact however is that the equation sets corresponding to two unrelated point correspondences have a dependency, as stated in the following proposition.

Proposition 5.5. *The set of equations (5.39) derived from a set of n general point correspondences across four views has rank $16n - \binom{n}{2}$, for $n \leq 5$.*

The notation $\binom{n}{2}$ means the number of choices of 2 among n, specifically, $\binom{n}{2} = n(n-1)/2$. Thus for 5 points there are only 70 independent equations, not enough to solve for Q^{pqrs}. For $n = 6$ points, $16n - \binom{n}{2} = 81$, and we have enough equations to solve for the 81 entries of Q^{pqrs}. These propositions will be proven below.

Proof. We consider two point correspondences across four views, namely $u^i \leftrightarrow u'^i \leftrightarrow u''^i \leftrightarrow u'''^i$ and $v^i \leftrightarrow v'^i \leftrightarrow v''^i \leftrightarrow v'''^i$. The first correspondence gives rise to a set of equations : $u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz}Q^{pqrs} = 0_{wxyz}$ where there is a different equation for each choice of w, x, y and z. There are a total of 81 equations in the 81 entries of the tensor Q^{pqrs}. The coefficients of each equation may be considered as a vector in the Euclidean space R^{81}. According to Proposition 5.3, however, the 81 such vectors span a subspace $S_{\mathbf{u}}$ of dimension 16 in R^{81}.

A similar set of equations may be derived from the second correspondence, and these equations span a second 16-dimensional subspace $S_{\mathbf{v}}$ of R^{81}. If the two subspaces $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$ intersect only in the zero vector, then together the two subspaces generate a subspace of dimension 32 of R^{81}. The proposition we are proving asserts that this is not so, however. Therefore, it is our goal to show that these two subspaces have non-trivial intersection.

The vectors generating $S_{\mathbf{u}}$ may be denoted by $\hat{\mathbf{u}}_{(wxyz)}$ where $(wxyz)$ is a combined index for the vector, and each $\hat{\mathbf{u}}_{(wxyz)}$ is a vector with components

$$\hat{u}_{(wxyz)(pqrs)} = u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz}$$

and $(pqrs)$ is a combined index for the component of the vector. We consider a specific linear combination of the vectors $\hat{\mathbf{u}}_{(wxyz)}$ given by $\mathbf{s} = v^w v'^x v''^y v'''^z \hat{\mathbf{u}}_{(wxyz)}$. This is a vector with components

$$s_{(pqrs)} = v^w v'^x v''^y v'''^z u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} \ . \tag{5.45}$$

In a similar fashion, the subspace $S_{\mathbf{v}}$ is generated by vectors $\hat{\mathbf{v}}_{(ijkl)}$, which have components

$$\hat{v}_{(ijkl)(pqrs)} = v^w v'^x v''^y v'''^z \epsilon_{wpi}\epsilon_{xqj}\epsilon_{yrk}\epsilon_{zsl} = v^w v'^x v''^y v'''^z \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} \ .$$

The last expression is obtained from the previous one by swapping the first and last indices of each ϵ.

Now, one forms the linear combination $\mathbf{s} = u^i u'^j u''^k u'''^l \hat{\mathbf{v}}_{(ijkl)}$, which when expanded turns out to have the same components as the vector $\mathbf{s}_{(pqrs)}$ in (5.45). In brief, we have

$$\mathbf{s} = v^w v'^x v''^y v'''^z \hat{\mathbf{u}}_{(wxyz)} = u^i u'^j u''^k u'''^l \hat{\mathbf{v}}_{(ijkl)} \ . \tag{5.46}$$

This shows that the subspaces $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$ or R^{81} intersect in a subspace generated by the vector $\mathbf{s}$. Thus, the dimension of the subspace generated by vectors $\hat{\mathbf{u}}_{(wxyz)}$ and $\hat{\mathbf{v}}_{(ijkl)}$ is at most 31, provided that $\mathbf{s}$ is non-zero. To consider the possibility that $\mathbf{s} = 0$, we rearrange (5.45) to get

$$\begin{aligned} s_{(pqrs)} &= (u^i v^w \epsilon_{ipw})(u'^j v'^x \epsilon_{jqx})(u''^k v''^y \epsilon_{kry})(u'''^l v'''^z \epsilon_{lsz}) \\ &= (\mathbf{u} \times \mathbf{v})_p (\mathbf{u}' \times \mathbf{v}')_q (\mathbf{u}'' \times \mathbf{v}'')_r (\mathbf{u}''' \times \mathbf{v}''')_s \end{aligned}$$

where $(\mathbf{u} \times \mathbf{v})$ represents the vector product. Such a vector product is nonzero unless the $\mathbf{u}$ and $\mathbf{v}$ are equal, up to scale, and hence represent the same point. Thus, $\mathbf{s}$ is non-zero unless $\mathbf{u}$ and $\mathbf{v}$ (or $\mathbf{u}'$ and $\mathbf{v}'$, etc) represent the same point. To take care of the case where for instance $\mathbf{u} = \mathbf{v}$, we note that then

$$v'^x v''^y v'''^z \hat{\mathbf{u}}_{(wxyz)} = -u'^j u''^k u'''^l \hat{\mathbf{v}}_{(wjkl)} \tag{5.47}$$

for each value of w. To verify this, note that the components of the vectors on each side of (5.47) are

$$v'^x v''^y v'''^z u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} = -u'^j u''^k u'''^l v^i v'^x v''^y v'''^z \epsilon_{ipw}\epsilon_{xqj}\epsilon_{yrk}\epsilon_{zsl} \ .$$

This means that $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$ intersect in at least a 3-dimensional subspace. Thus, in all cases, we have shown that the subspace generated by $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$ has dimension at most 31.

We now consider the possibility that the dimension of the subspace is less than 31. In such a case, all 31×31 sub-determinants of the matrix having as rows the vectors $\hat{\mathbf{u}}_{(wxyz)}$ and $\hat{\mathbf{v}}_{(ijkl)}$ must vanish. These subdeterminants may be expressed as polynomial expressions in the coefficients of the points $\mathbf{u}$, $\mathbf{u}'$, $\mathbf{u}''$, $\mathbf{u}'''$, $\mathbf{v}$, $\mathbf{v}'$, $\mathbf{v}''$ and $\mathbf{v}'''$. These coefficients together make up a 24-dimensional space. Thus, there is a function $f : R^{24} \to R^N$ for some N (equal to the number of such 31×31 subdeterminants), such that the equation matrix has rank less than 31 only on the set of zeros of the function f. Any arbitrarily chosen example may be used to show that the function f is not identically zero. It follows, that the set of point correspondences for which the set of equations has rank less than 31 is a variety in R^{24}, and hence is nowhere dense. Thus, for a general pair of point correspondences, the set of equations generated by a pair of point correspondences across 4 views has rank 31.

We now turn to the general case of n point correspondences across all 4 views. Note that the linear relationship (5.46) that holds for two point correspondences is non-generic, but depends on the pair of correspondences. In general, therefore, given n point correspondences, there will be $\binom{n}{2}$ such relationships. This reduces the dimension of the space spanned by the set of equations to $16n - \binom{n}{2}$ as required. □

Three View Case

In this section, we consider the set of equations relating the entries of the trifocal tensor T_i^{jk} generated by a single point correspondence across three views. We find the following favourable situation holds.

Proposition 5.6. *Consider a single point correspondence $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}''$ across three views. Letting the two free indices x and y in (5.29) vary from 1 to 3 one obtains from this correspondence a set of 9 equations in the entries of T_i^{qr}. The rank of this set of equations is 4. Furthermore, let the equations be written as $A\mathbf{t} = 0$ where A is a 9×27 matrix and $\mathbf{t}$ is a vector containing the entries of T_i^{qr}. Then the 4 non-zero singular values of A are equal.*

Proof. This proposition is similar to Proposition 5.3, and is proven in much the same way. The full set of 9 equations derived from a single point correspondence is of the form $u^i u'^j \epsilon_{jqx} u''^k \epsilon_{kry} T_i^{qr} = 0_{xy}$. A total of 9 equations are generated by varying x and y over the range $1, \ldots, 3$. Thus, the equation matrix A may be written as

$$A_{(xy)\binom{i}{qr}} = u^i u'^j u''^k \epsilon_{jqx}\epsilon_{kry} \tag{5.48}$$

where the indices (xy) index the row and $\binom{i}{qr}$ index the column of A. As in the proof of Proposition 5.3, we may write $u'^j \epsilon_{jqx} = S'_{xq}$ and $u''^k \epsilon_{kry} = S''_{yr}$. Then the matrix A

in (5.48) may be written as $A_{(xy)(^{i}_{qr})} = u^i S'_{xq} S''_{yr}$. Consequently, applying (5.42) we may write

$$U'^{x}_{a} U''^{y}_{b} A_{(xy)(^{i}_{qr})} V'^{q}_{e} V''^{r}_{f} = k'k'' u^i D_{ae} D_{bf} \ . \tag{5.49}$$

Next, introducing a vector u_i with covariant (lower) index, defined such that $u_i = u^i$ for all i, we have

$$U'^{x}_{a} U''^{y}_{b} A_{(xy)(^{i}_{qr})} u_i V'^{q}_{e} V''^{r}_{f} = k'k'' u_i u^i D_{ae} D_{bf} = k'k'' ||\mathbf{u}||^2 D_{ae} D_{bf} \ .$$

Now, writing

$$\hat{U}^{(xy)}_{(ab)} = U'^{x}_{a} U''^{y}_{b}$$

$$\hat{V}^{(^{qr}_{i})}_{(ef)} = u_i V'^{q}_{e} V''^{r}_{f}$$

$$\hat{D}_{(ab)(ef)} = D_{ae} D_{bf} \ .$$

and

$$\hat{k} = k'k'' ||\mathbf{u}||^2$$

we see that (5.49) may be written as

$$\hat{U}^{(xy)}_{(ab)} A_{(xy)(^{i}_{qr})} \hat{V}^{(^{qr}_{i})}_{(ef)} = \hat{k} \hat{D}_{(ab)(ef)} \tag{5.50}$$

The matrix $D_{(ab)(ef)}$ is diagonal with 4 unit diagonal entries. As before, to complete the proof, we need only show that $\hat{U}$ and $\hat{V}$ are orthogonal, The matrix $\hat{U}$ is orthogonal as is shown using the same argument as before. Since the matrix $\hat{V}^{(^{qr}_{i})}_{(ef)}$ is not square (it has dimension 27×9), we need to show that is has orthogonal columns. Details are as follows.

$$\begin{aligned}
\sum_{q,r,i} \hat{V}^{(^{qr}_{i})}_{(ef)} \hat{V}^{(^{qr}_{i})}_{(e'f')} &= \sum_{i,p,q} (V'^{q}_{e} V''^{r}_{f} u_i)(V'^{q}_{e'} V''^{r}_{f'} u_i) \\
&= \sum_{q} V'^{q}_{e} V'^{q}_{e'} \sum_{r} V''^{r}_{f} V''^{r}_{f'} \sum_{i} (u_i^2) \\
&= \delta_{ee'} \delta_{ff'} ||\mathbf{u}||^2 \\
&= \delta_{(ef)(e'f')} ||\mathbf{u}||^2 \ .
\end{aligned}$$

Thus, in fact, the rows of $\hat{V}$ are orthogonal, and each one has the same norm equal to $||\mathbf{u}||$. This completes the proof. □

Choosing equations

In the previous two sections, proofs have been given that the singular values of the full set of equations derived from three or four point equations are all equal. The key point in the argument is that the two non-zero singular values of a 3×3 skew-symmetric matrix are equal. This proof may clearly be extended to apply to any of the other sets of equations derived from line or point correspondences given in sections 5:3.2 and 5:3.3.

Consider once more the case of three point correspondences in three views. The results on singular values show that it is in general advisable to include all 9 equations derived from this correspondence, rather than selecting just four independent equations. This

will avoid difficulties with near singular situations. This conclusion is supported by experimental observation. Indeed, numerical examples show that the condition of a set of equations derived from a set several point correspondences is substantially better when all 9 equations are included for each point correspondence. In this context, the condition of the equation set is given by the ratio of the first (largest) to the n-th singular value, where n is the number of linearly independent equations.

Including all 9 equations rather than just 4 means that the set of equations is larger, leading to increased complexity of solution. However, whether the equations are solved using the Singular Value Decomposition, or the method of normal equations, the increase in complexity needs only to be linear. This point is explained in [32]. For formulae about the complexity of the SVD, see [15].

An alternative to including all 9 equations (or all 81 in the 4-view case) is to include the minimum number (4 or 16 respectively) of correctly chosen equations. This notion will be illustrated for the three-view case. As we saw in section 5:3.2, the equations $u^i u'^j u''^k \epsilon_{jqx} \epsilon_{kry} T_i^{qr} = 0_{xy}$ derived from a point correspondence across three views may be considered as a set of line equations $u^i \lambda'_{qx} \lambda''_{ry} T_i^{qr} = 0_{xy}$ by writing $\lambda'_{qx} = u'^j \epsilon_{jqx}$ and $\lambda''_{ry} = u''^k \epsilon_{kry}$. Each choice of x or y gives a different line through the points u'^j and u''^k, a total of 9 choices.

Now, as a matrix, λ'_{qx} is skew-symmetric, and hence has two equal singular values. As remarked, this is the basis for the set of all 9 equations having rank 4, and 4 equal singular values. On the other hand, if we select just two lines passing through u'^j by making a choice of two values for the index x, then the two columns of the matrix λ'_{qx} are not orthogonal, and the resulting 3×2 matrix does not have equal singular values, and in fact may be nearly rank-deficient. As previously seen, one remedy is to include the equations derived from all 3 choices of the index x. The corresponding three lines λ'_{qx} are the lines parallel with the coordinate axes and through the origin and passing through the point u'^j. An alternative is to select two other lines $\hat{\lambda}'_{qx}$, for $x = 1, 2$, passing through u'^j and represented by an orthonormal pair of vectors. In this case, the matrix $\hat{\lambda}'_{qx}$ of dimension 3×2 will have two equal singular values. If this is done also for the point u''^k, then arguments of section 5:3.4 apply, and the resulting set of 4 equations $u^i \hat{\lambda}'_{qx} \hat{\lambda}''_{ry} T_i^{qr} = 0_{xy}$ for $x, y = 1, 2$ will be independent and orthonormal. Note that the condition that the vectors $\hat{\lambda}'_{q1}$ and $\hat{\lambda}'_{q2}$ are orthonormal has nothing to do with geometric orthogonality of the lines. A simple way of finding a pair of orthonormal vectors $\hat{\lambda}'_{qx}$ such that $u'^q \hat{\lambda}'_{qx} = 0_x$ is using Householder transforms ([15]). A Householder matrix h_{qx} is an orthogonal matrix such that $u'^q h_{qx} = \delta_{3x} = (0, 0, 1)$. Setting $\hat{\lambda}_{qx} = h_{qx}$ for $x = 1, 2$ gives the required pair of lines passing through the point u'^q.

We summarize this discussion as follows.

Recommended method for formulating point equations.

Given a point correspondence $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}''$ across three views :

1. Find Householder matrix h'_{qx} and h''_{ry} such that $u'^q h'_{qx} = \delta_{3x}$ and $u''^r h'_{ry} = \delta_{3y}$.
2. For $x, y = 1, 2$ set $\hat{\lambda}'_{qx} = h'_{qx}$ and $\hat{\lambda}''_{ry} = h''_{ry}$.
3. The formula $u^i \hat{\lambda}'_{qx} \hat{\lambda}''_{ry} T_i^{qr} = 0_{xy}$ for $x, y = 1, 2$ gives a set of four orthonormal

equations in the entries of T_i^{qr}.

Once more, it is evident that essentially this method will work for all the types of equations summarized in Tables 1 and 2.

5:3.5 Summary and Other Work

Although using a slightly different approach, this chapter summarized previous results of Triggs ([73]) and Faugeras and Mourrain ([14]) on the derivation of multilinear relationships between corresponding image coordinates. The formulae for relations between mixed point and line correspondences are extensions of the result of [26, 32]. This chapter suggests that the most basic relations are the point-line-line correspondence equation $u^i \lambda'_q \lambda''_r T_i^{qr} = 0$ in the three-view case, and the line-correspondence equation $\lambda_p \lambda'_q \lambda''_r \lambda'''_s Q^{pqrs} = 0$ for four views. Indeed numerical robustness may be enhanced by reducing other correspondences to this type of correspondence, for carefully selected lines.

Part VI

Euclidean Reconstruction and Calibration

To this point, we have been chiefly concerned with the problem of projective reconstruction. In considering the projective reconstruction problem it has been assumed that the individual cameras involved were unrelated. In many cases, however one may know that all the cameras involved in reconstruction of a scene from several views have the same calibration. This will particularly be true in the case of several views taken with the same camera for which the calibration does not change. In this case one knows that the internal calibration for all the cameras is the same, even if one does not know exactly what the calibration is. Under these circumstances it turns out to be possible to carry out affine and even Euclidean reconstruction of the scene.

It has been shown by the work of Maybank and Faugeras and Luong ([46, 13]) that calibration of a camera is possible from three or more views taken with the same camera. If this is so, then the reconstruction problem can be reduced to the problem of reconstruction from calibrated cameras. It is well know (see for instance [42]) that for calibrated cameras, reconstruction up to scale is possible. More precisely, scene reconstruction is possible, up to a similarity transformation, namely up to unknown absolute position and orientation of the scene, and overall scale. We will refer to such reconstruction in this report (a little inexactly) as Euclidean reconstruction. Although the paper [46] opened up the theoretical possibility of doing Euclidean reconstruction, no definitive or robust algorithm for this problem was given. In this section we investigate the problem of Euclidean reconstruction and give an algorithm that works quite well on three or more views.

According to the result of Maybank and Faugeras ([46]) if all the cameras have the same calibration then the calibration matrix K may be determined, and is at least locally unique. (Whether this is true for exactly three views was left somewhat ambiguous in [46] but was clarified by Luong ([43])). Consequently, we assume in this section that all cameras have the same calibration, so that $P_i = K[R_i \mid -R_i\mathbf{t}_i]$, where each R_i is a rotation matrix and K is an upper-triangular matrix, the common calibration matrix of all the cameras. We attempt to retrieve P_i and $\mathbf{x}_j$ from a set of image correspondences $\mathbf{u}^i_j$. The points $\mathbf{x}_j$ and the camera matrices, P_i can not be determined absolutely. Instead it is required to determine them up to a Euclidean transformation. In order to constrain the solution, it may be assumed that $R_0 = I$ and $\mathbf{t}_0 = \mathbf{0}$. The solution may then be determined up to scaling.

Because of Maybank and Faugeras's result, with more than three views any reconstruction for which all the camera matrices P_i have the same calibration is virtually assured of being the true reconstruction, or at least differing by at most a Euclidean transformation – it is a Euclidean reconstruction.

This part of the report gives an algorithm for computing a Euclidean reconstruction of a scene based only on image correspondence data from uncalibrated cameras. An alternative method for Euclidean reconstruction that uses extra Euclidean constraints is reported in [7].

6:1 The DIAC and calibration

As was pointed out by Maybank and Faugeras ([46]), one of the basic concepts related to calibration of a camera, or Euclidean reconstruction is the absolute conic. The connection between the absolute conic and camera calibration is demonstrated next. It is seen that

knowledge of the image of the absolute conic in an image is equivalent to calibration of the camera that took the image.

The absolute conic is a conic lying on the plane at infinity, having equation $x^2+y^2+z^2 = 0; \quad t = 0$, where $(x, y, z, t)^\top$ are coordinates of a point in 3-dimensional space. The absolute conic does not contain any points with real coordinates – it is composed entirely of complex points. This does not, however, diminish its usefulness. The image of the absolute conic in an image is representable by a real symmetric 3×3 matrix, as will be seen next.

Define a vector $\mathbf{x} = (x, y, z)^\top$. A point $(x, y, z, 0)^\top$ is on the absolute conic if and only if $\mathbf{x}^\top\mathbf{x} = 0$. Consider a camera matrix $P = K[R \mid -R\mathbf{t}]$. A point $(x, y, z, 0)^\top$ on the absolute conic maps to $\mathbf{u} = P(x, y, z, 0)^\top = KR\mathbf{x}$. Thus, $\mathbf{x} = R^\top K^{-1}\mathbf{u}$, and the condition $\mathbf{x}^\top\mathbf{x}$ becomes $\mathbf{u}^\top K^{-\top} RR^\top K^{-1}\mathbf{u} = \mathbf{u}^\top K^{-\top} K^{-1}\mathbf{u} = 0$. Thus, a point $\mathbf{u}$ is on the image of the absolute conic if and only if it lies on the conic represented by the matrix $K^{-\top}K^{-1}$. In other words, $K^{-\top}K^{-1}$ is the matrix representing the image of the absolute conic. Taking inverses (dual conics) reveals that $KK^\top$ is the dual of the image of the absolute conic. We will denote $KK^\top$ by C. If C is known then the calibration matrix K may be retrieved by Choleski factorization. Specifically, any symmetric positive-definite matrix (such as C) may be uniquely factored as a product $KK^\top$ such that K is an upper triangular matrix with positive diagonal entries. For an algorithm for Choleski factorization, see [55].

Since we will be considering the absolute conic in subsequent pages, we adopt the following abbreviations :

AC means the absolute conic.

IAC means the image of the absolute conic.

DIAC means the dual of the image of the absolute conic.

We have shown how the calibration matrix K may be retrieved if the matrix C representing the DIAC is known. Conversely, if K is known, then $C = KK/tr$ is determined. This shows the intimate link between the IAC and calibration. An important point to note is that the formula $C = KK^\top$ for the DIAC depends only on the calibration matrix, and not on the orientation R or the position $\mathbf{t}$ of the camera. The IAC is fixed under Euclidean motions of the camera.

6:2 Kruppa's Equations

Given two views of a scene taken with two different cameras, one may compute a projective reconstruction of the scene. If one has additional information that the cameras used to image the two scenes have the same internal calibration, then this implies a certain restriction on the class of possible reconstruction. This in turn implies a restriction on the internal calibration of the camera. This restriction may be expressed by Kruppa's equations, which will be derived in this section. Kruppa's equations will be formulated here in terms of simple conditions on the DIAC, which we have seen to be equivalent to the internal calibration.

The purpose of this section is to give a specific form for Kruppa's equations in terms of the Fundamental matrix. Kruppa's equations can be written explicitly in terms of the singular value decomposition (SVD) of the fundamental matrix.

Consider two camera matrices P and P' with the same calibration. Let C be the DIAC as imaged by these two cameras. As was shown in section 6:6.5 the image of the absolute conic (IAC) is independent of the pose of the camera, and so it is the same for the two cameras in question.

Let F be the fundamental matrix for the pair of cameras. For reasons to become apparent later, we wish to apply projective transformations represented by 3×3 transformation matrices A and A' to the two images. After the transformations, the effective camera matrices will be AP and $A'P'$, corresponding to the camera projection followed by projective transformation of the image. This will of course change the DIAC to some new conic envelopes, which we will call D and D'. Since A and A' may be different, we can no longer assume that $D = D'$.

Suppose that A and A' are chosen so that the fundamental matrix for the two new camera matrices AP and $A'P'$ has the special form

$$E = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} . \tag{6.1}$$

This is a very special fundamental matrix having the property that the two epipoles are at the origin and that corresponding epipolar lines are identical in the two images.

Now, consider a plane passing through the two camera centres, tangent to the absolute conic. Such a plane will project to a pair of corresponding epipolar lines in the two images, and these two lines will be tangent to the IAC. Since there are two such tangential planes, there are two pairs of corresponding epipolar tangents.

We recall that corresponding epipolar lines in the two images are identical. Let $(\lambda, \mu, 0)^\top$ be a tangent to the IAC. Since D is the DIAC in the first image, this tangential relationship may be written as

$$(\lambda, \mu, 0)D(\lambda, \mu, 0)^\top = 0$$

and similarly, $(\lambda, \mu, 0)D'(\lambda, \mu, 0)^\top = 0$. Writing these two equations out explicitly gives

$$\lambda^2 d_{11} + 2\lambda\mu d_{12} + \mu^2 d_{22} = 0$$

and

$$\lambda^2 d'_{11} + 2\lambda\mu d'_{12} + \mu^2 d'_{22} = 0$$

where $D = (d_{ij}$ and $D' = (d'_{ij})$, both of which are symmetric.

Since the two tangent lines to the IAC must be the same two lines in the two images, these two equations must have the same pair of solutions for λ and μ. This means that they must be identical equations (up to scale), and so

$$\frac{d_{11}}{d'_{11}} = \frac{d_{12}}{d'_{12}} = \frac{d_{22}}{d'_{22}} . \tag{6.2}$$

These are the Kruppa equations.

We now repeat this argument, this time being more precise about specific values. The purpose is to find explicit expressions for the matrices D and D' in terms of the fundamental matrix F.

Let the Singular Value Decomposition of the fundamental matrix be $F = UDV^\top$, where U and V are orthogonal, and $D = \mathrm{diag}(r, s, 0)$ is a diagonal matrix. We may write this as follows:

$$F = U \begin{bmatrix} r & & \\ & s & \\ & & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^\top \ .$$

We write

$$A'^\top = U \begin{bmatrix} r & & \\ & s & \\ & & 1 \end{bmatrix}$$

and

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^\top \ .$$

Then, we see that $F = A'^\top E A$ with A and A' non-singular. For a pair of matching image points $\mathbf{u}' \leftrightarrow \mathbf{u}$ we have $\mathbf{u}'^\top F \mathbf{u} = 0$. Thus, $\mathbf{u}'^\top A'^\top E A \mathbf{u} = 0$. Setting $\hat{\mathbf{u}} = A\mathbf{u}$ and $\hat{\mathbf{u}}' = A'\mathbf{u}'$, we see that $\hat{\mathbf{u}}'^\top E \hat{\mathbf{u}} = 0$. Thus, A and A' are the two transforms that we require.

Next, we investigate the effect of this transformation of the DIAC. Consider a transformation A. How does this transformation transform lines ? Well, a point $\mathbf{u}$ lies on a line λ if and only if $\lambda^\top \mathbf{u} = 0$. This can be written as $\lambda^\top A^{-1} A \mathbf{u} = 0$. Thus, $\mathbf{u}$ lies on λ if and only if $A\mathbf{u}$ lies on $A^{-\top}\lambda$. Thus, $A^{-\top}\lambda$ is the transformed line. Now, a line λ belongs to a conic envelope C if and only if $\lambda^\top C \lambda = 0$. This can be written as $(\lambda A^{-1})(A C A^\top)(A^{-\top}\lambda) = 0$. Thus, the transformation A maps the conic envelope C to a $D = ACA^\top$, and similarly A' maps C to $D' = A'CA'^\top$.

Now, we want to compute d_{ij}, where $D = (d_{ij})$. Let

$$A = \begin{pmatrix} \mathbf{a}_1{}^\top \\ \mathbf{a}_2{}^\top \\ \mathbf{a}_3{}^\top \end{pmatrix}$$

where $\mathbf{a}_i$ is the i-th row of A. Then, from $D = ACA^\top$ we compute $d_{ij} = \mathbf{a}_i{}^\top C \mathbf{a}_j$.

Then (6.2) leads to the following explicit form for the Kruppa equations :

$$\frac{\mathbf{a}_1{}^\top C \mathbf{a}_1}{\mathbf{a}_1'{}^\top C \mathbf{a}_1'} = \frac{\mathbf{a}_1{}^\top C \mathbf{a}_2}{\mathbf{a}_1'{}^\top C \mathbf{a}_2'} = \frac{\mathbf{a}_2{}^\top C \mathbf{a}_2}{\mathbf{a}_2'{}^\top C \mathbf{a}_2'} \tag{6.3}$$

We can write these equations directly in terms of the SVD of the fundamental matrix $F = U\mathrm{diag}(r, s, 0)V^\top$. Specifically, we have

$$A' = \begin{bmatrix} r & & \\ & s & \\ & & 1 \end{bmatrix} U^\top$$

from which we have

$$A' = \begin{pmatrix} \mathbf{a}'_1{}^\top \\ \mathbf{a}'_2{}^\top \\ \mathbf{a}'_3{}^\top \end{pmatrix} = \begin{pmatrix} r\mathbf{u}_1{}^\top \\ s\mathbf{u}_2{}^\top \\ \mathbf{u}_3{}^\top \end{pmatrix} .$$

where $\mathbf{u}_i$ is the i-th column of U.

For A we have

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^\top$$

and so

$$A = \begin{pmatrix} \mathbf{a}_1{}^\top \\ \mathbf{a}_2{}^\top \\ \mathbf{a}_3{}^\top \end{pmatrix} = \begin{pmatrix} \mathbf{v}_2{}^\top \\ -\mathbf{v}_1{}^\top \\ \mathbf{v}_3{}^\top \end{pmatrix} .$$

where $\mathbf{v}_i$ is the i-th column of V.

From (6.3) we obtain

$$\frac{\mathbf{v}_2{}^\top C \mathbf{v}_2}{r^2 \mathbf{u}_1{}^\top C \mathbf{u}_1} = \frac{-\mathbf{v}_2{}^\top C \mathbf{v}_1}{rs \mathbf{u}_1{}^\top C \mathbf{u}_2} = \frac{\mathbf{v}_1{}^\top C \mathbf{v}_1}{s^2 \mathbf{u}_2{}^\top C \mathbf{u}_2} \tag{6.4}$$

6:2.1 Consequences of Kruppa's Equations

Since the numerator and denominator of (6.3) or (6.4) are linear expression in the entries of the matrix C, the resulting equations are quadratic. From (6.4) one obtains two independent quadratic equations in the entries of C. Since C is symmetric, and defined only up to a scale, it has 5 degrees of freedom. A single pair of views is clearly not sufficient to determine C. However from three views, one obtains three fundamental matrices, one for each of the pairs of views. Now we have six quadratic equations in five unknowns. It is not clear, of course that the six equations are independent. However, as demonstrated empirically by Luong ([43]) they are sufficient (at least with noise-free data) to determine C. This will also be demonstrated in this report by Euclidean reconstruction from three views.

Special cases of the Kruppa equations are of interest, as will be seen now.

Calibrated Cameras. If we assume that the calibration matrix of the cameras is the identity matrix $K = I$ in both cases, then the matrix $C = KK^\top$ representing the DIAC is also equal to the identity. In this case, Kruppa's equations (6.4) reduce to the form

$$1/r^2 = 0/0 = 1/s^2$$

where r and s are the singular values of the fundamental (or essential) matrix. This implies $r = s$. In other words if the calibration matrix is the identity, then the two non-zero singular values of the fundamental matrix are equal. This is an easy proof of a result of Faugeras and Huang ([39]).

Camera with known principal point. If the principal point of the camera is known, then by a suitable change of image coordinates one may assume that the principal point is at the origin. If it is further assumed that there is no *skew* parameter, then the calibration matrix is a diagonal matrix of the form $\mathrm{diag}(k_u, k_v, 1)$. In this case, $C = \mathrm{diag}(k_u^{-2}, k_v^{-2}, 1)$ and the kruppa equations (6.4) are quadratic in the variables k_u^{-2} and k_v^{-2}. We have a pair of quadratic equations in two variables – sufficient to solve for k_u^{-2} and k_v^{-2} in a relatively straight-forward manner. Thus under assumption of known principal point, and no skew one may solve for k_u^{-2} and k_v^{-2}, and hence for k_u and k_v. There may be up to four solutions, but only positive solutions for k_u^{-2} and k_v^{-2} need be considered.

Translatory motion. In the case of pure translatory motion of the camera, the fundamental matrix F is skew-symmetric. This may be seen in a variety of ways. For instance, if $K[I \mid 0]$ and $K[I \mid \mathbf{t}]$ are the two cameras, then according to Proposition (2.2) the fundamental matrix is given by $F = K^{-\top}[\mathbf{t}]_\times K^{-}1$. Since F is skew-symmetric, it factors as $F = UEU^\top$ where U is orthogonal and E is as given in (6.1). The two matrices A and A' required to transform F to the simple form (6.1) are both equal. Thus both numerator and denominator are equal in the form (6.3) of the Kruppa equations. Then equations (6.3) takes the simple $1 = 1 = 1$. Although this is a significant mathematical fact, it is not a useful constraint on camera calibration. Thus, a translatory motion of the cameras does not impose any constraint on camera calibration. It is interesting that affine scene reconstruction is possible from translatory motions, however ([49]).

Kruppa's equations for three of more views. For three views or more, one has sufficiently many equations to solve for the calibration. Direct solution of simultaneous quadratics given by Kruppa's equations, tried by Luong ([43]) is difficult because of the existence of multiple solutions to non-linear equations, and the difficulties involved with solving redundant systems. Luong's method was to solve subsets of five equations, giving up to $2^5 = 32$ solutions, and then selecting solutions common to all subsets of equations. This method is somewhat inexact and prone to noise. Consequently, in the work reported here different methods were tried.

6:3 Least-Squares Method for Euclidean Reconstruction

6:3.1 Reconstruction by Direct Levenberg-Marquardt Iteration

A direct approach to the Euclidean reconstruction problem is to solve directly for the unknown camera matrices, $P_i = K[R_i \mid -R_i\mathbf{t}_i]$ and points $\mathbf{x}_j$. In particular, we search for P_i of the required form, and $\mathbf{x}_j$ such that $\hat{\mathbf{u}}_j^i = P_i\mathbf{x}_j$ and such that the squared error sum

$$\sum_{i,j} d(\hat{\mathbf{u}}_j^i, \mathbf{u}_j^i)^2$$

is minimized, where $d(*,*)$ represents Euclidean distance. Using this minimization criterion is optimal under the assumption that measurement errors are caused by errors in measurement of the pixel locations of the $\mathbf{u}_j^i$, and that these errors are independent and gaussian. This problem may be formulated in the form $\mathbf{y} = f(\mathbf{x})$, where the independent

variables $\mathbf{x}$ comprise the 3D coordinates of each of the points $\mathbf{x}$ in space, the rotations R_i of each of the cameras and the common calibration matrix K. The dependent variables $\mathbf{y}$ comprise the image coordinates $\mathbf{u}^i_j$.

There are various methods of parametrizing the rotations. Horn ([36, 37]) uses quaternions to do this. We prefer to parametrize rotations using Eulerian angles. This has the advantage that a rotation is parametrized by the minimum of three parameters, instead of four using quaternions. To avoid problems of singularities in the representation of rotations by Eulerian angles, rotations are parametrized as incremental rotations with respect to the present "base rotation". Thus, each R_i is represented as a product $R_i = X_i\Delta(\theta_i, \phi_i, \kappa_i)$, where $\Delta(\theta_i, \phi_i, \kappa_i)$ is the rotation represented by Eulerian angles θ_i, ϕ_i and κ_i. Initially, X_i is set to the initial estimate of the rotation, and θ_i, ϕ_i and κ_i are all set to zero (and hence Δ is the identity mapping). At the end of each Levenberg-Marquardt (LM) iteration X_i is set to the product $X_i\Delta(\theta_i, \phi_i, \kappa_i)$, and θ_i, ϕ_i and κ_i are reset to zero. Since the incremental rotation adjustment applied at each step of iteration is small, the Eulerian angles used to represent it are small. Consequently the difficulty of singularities in the representation of rotations by Eulerian angles does not arise. In this way, only three parameters are used to represent the incremental adjustment in the estimation step, rather than four using quaternions. This provides a speed advantage when large numbers of rotations are being estimated.

Such an approach to Euclidean scene reconstruction will work perfectly well, *provided the initial estimate is sufficiently close*. With arbitrary or random guesses at initial values of the parameters it usually fails dismally. The problem as posed is similar to the relative placement problem. This problem was given a robust solution by Horn ([36, 37]) In fact the algorithm given in [36] amounts essentially to Newton iteration by solving the normal equations, using the method of back-substitution mentioned in Section 2:1.4, and parametrizing rotations as quaternions. Horn avoids the need for an informed initial guess by iterating from each of a number of equally spaced or random rotations and selecting the best solution. The problem considered by Horn differs from the problem considered here in that we are considering uncalibrated cameras, and we wish to be able to solve for a large number of cameras at once. Thus, there is an unknown calibration matrix that must be estimated. Furthermore, instead of one rotation, we have several. With more than a small number of cameras the idea of sampling the rotation space is unworkable.

In short, direct iteration may be used to refine a solution found by other techniques, but can not be used on its own.

6:4 Converting Projective to Euclidean Reconstruction

Instead of attempting a direct reconstruction, calibration and pose estimation as in the previous section, we use a two-step approach. In the first step, a projective reconstruction of the scene is computed, dropping the assumption that the images are all taken with the same camera. The scene configuration obtained in this manner will differ from the true configuration by a 3D projective transformation. In the second step, this projective transform is estimated. The advantage of proceeding in this manner is that projective reconstruction is relatively straight-forward. Then step two, the estimation of the correct

3D transformation, comes down to solving an 8-parameter estimation problem, which is far more tractable than the original problem.

For the present, we drop the assumption that all the cameras have the same calibration. A projective reconstruction of the scene is carried out as described in this report (section 5:2.7). Once we have a projective reconstruction of the imaging geometry any other reconstruction (including a desired Euclidean reconstruction) may be obtained by applying a 3D projective transformation. In particular, if $(\{P_i\}, \{\mathbf{x}_j\})$ is a projective reconstruction, then any other reconstruction is of the form $(\{P_i H^{-1}\}, \{H\mathbf{x}_j\})$ where H is a 4×4 non-singular matrix. We seek such a matrix H such that the transformed camera matrices $P_i H^{-1}$ all have the same (yet to be determined) calibration matrix, K. In other words, we seek H such that $P_i H^{-1} = K[R_i \mid -R_i \mathbf{t}_i]$ for all i, where each R_i is a rotation matrix and K is the common upper-triangular calibration matrix.

Without loss of generality, we may make the additional restriction that the zeroeth camera remains located at the origin and that R_0 is the identity. Since in the original projective reconstruction $P_0 = [I \mid 0]$, it follows that H^{-1} may be assumed to have the restricted form

$$H^{-1} = \begin{bmatrix} K & \mathbf{0} \\ \mathbf{v}^\top & \alpha \end{bmatrix} \ .$$

Since the constant α represents scaling in 3-space, we may further assume that $\alpha = 1$. Equivalently, since K is non-singular, we may (and shall) rather assume that H^{-1} has the form

$$H^{-1} = \begin{bmatrix} K & 0 \\ -\mathbf{v}^\top K & 1 \end{bmatrix} = \begin{bmatrix} I & \mathbf{0} \\ -\mathbf{v}^\top & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1 \end{bmatrix} \tag{6.5}$$

Now, writing each $P_i = [A_i \mid -A_i \mathbf{t}_i]$ and multiplying out leads to a requirement that

$$A_i(I + \mathbf{t}_i \mathbf{v}^\top)K \approx K R_i \tag{6.6}$$

for some rotation matrix R_i. Our goal is to find K and $\mathbf{v}$ to satisfy this set of conditions. Recall that K is upper triangular, and further assume that K_{33} equals 1, hence K contains five unknown entries. The vector $\mathbf{v}$ has a further three unknown entries. In total, it is required to estimate these eight unknown parameters.

Of course, for inexact data, the equations (6.6) will not be satisfied exactly, and so we will cast this problem as a least-squares minimization problem that may be solved using LM. In particular, given values for K and $\mathbf{v}$, we compute the expression $A_i(I + \mathbf{t}_i \mathbf{v}^\top)K$ for each i (remembering that A_i and $\mathbf{t}_i$ are known). Taking the QR decomposition of this matrix, we obtain upper-triangular matrices K_i' such that

$$A_i(I + \mathbf{t}_i \mathbf{v}^\top)K = K_i' R_i \ . \tag{6.7}$$

Subsequently, we compute the matrices $X_i = K^{-1} K_i'$ for all i. Since we have assumed that $P_0 = [A_0 \mid -A_i \mathbf{t}_i] = [I \mid 0]$, it follows that $X_0 = I$. Furthermore, if K and $\mathbf{v}$ satisfy the desired condition (6.6) then $K_i' \approx K$ for all $i > 0$, and so $X_i \approx I$. Accordingly, we seek to minimize the extent by which X_i differs from the identity matrix. Consequently, we multiply each X_i by a normalizing factor α_i chosen so that the sum of squares of diagonal entries of $\alpha_i X_i$ equals 3, and so that $\det \alpha_i X_i > 0$. Now, we seek K and $\mathbf{v}$ to minimize the expression

$$\sum_{i>0} ||\alpha_i X_i - I||^2 \tag{6.8}$$

Note that each $\alpha_i X_i - I$ is an upper-triangular matrix. This minimization problem fits the general form of LM estimation of a function $f : R^8 \mapsto R^{6(N-1)}$ where N is the total number of cameras. The function f maps the eight [9] variable entries of K and $\mathbf{v}$ to the diagonal and above-diagonal entries of $\alpha_i X_i - I$ for $i > 0$. Since this minimization problem involves the estimation of 8 parameters only, it is obviously a great improvement over the original problem as stated in Section 6:3.1 that required the simultaneous estimation of the matrix K, the $N-1$ rotation matrices R_i for $i > 0$ and the 3D point coordinates of all points $\mathbf{x}_j$.

It turns out still to be impractical to solve this minimization problem without a good initial guess at K and $\mathbf{v}$. It is possible to take a good prior guess at K if some knowledge of the camera is available. On the other hand, it is difficult to guess the vector $\mathbf{v}$, so it will be necessary to find some way to obtain an initial estimate for $\mathbf{v}$. It will turn out that if $\mathbf{v}$ is known, then the calibration matrix K can be computed by a straight-forward non-iterative algorithm, so there is no need to guess K.

6:4.1 Euclidean From Affine Reconstruction

With H^{-1} of the form (6.5) matrix H may be written as

$$H = \begin{bmatrix} K^{-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathbf{v}^\top & 1 \end{bmatrix} .$$

The right-hand one of these two matrices represents a transformation that moves the plane at infinity, whereas the second one is an affine transformation, not moving the plane at infinity. In fact, if $\mathbf{x}$ is a point being mapped to infinity by the transformation H, then $(\mathbf{v}^\top 1)\mathbf{x} = 0$. So $(\mathbf{v}^\top 1)$ represents the plane that is mapped to the plane at infinity by H.

We will now suppose that by some magic we have been able to determine $\mathbf{v}$. This means, in effect that we know the position of the plane at infinity in the reconstruction. Otherwise stated, we have been able to determine the structure up to an affine transformation. We will now present a simple non-iterative algorithm for the determination of K, and hence of the Euclidean structure.

Equation (6.6) may be written as $B_i K = K R_i$ where $B_i = \alpha_i A_i (I + \mathbf{t}_i \mathbf{v}^\top)$, and the constant factor α_i is chosen so that $\det B_i = 1$. Matrix B_i is known since A_i, $\mathbf{t}_i$ and $\mathbf{v}$ are assumed known. The equation $B_i K = K R_i$ may be written as $K^{-1} B_i K = R_i$. In other words, each B_i is the conjugate of a rotation matrix, the conjugating element being the same in each case – the calibration matrix K. For any non-singular matrix X, let $X^{-\top}$ be the inverse transpose of X. For a rotation matrix R, we have $R = R^{-\top}$. From the equation $R_i = K^{-1} B_i K$ it follows by taking inverse transposes that $R_i = K^\top {B_i}^{-\top} K^{-\top}$. Equating these two expressions for R_i we get $K^\top {B_i}^{-\top} K^{-\top} = K^{-1} B_i K$, from which it follows that

$$(K K^\top) {B_i}^{-\top} = B_i (K K^\top) \tag{6.9}$$

Given sufficiently many views and corresponding matrices B_i equation 6.9 may be used to solve for the entries of the matrix $K K^\top$. In particular, denoting $K K^\top$ by C and

[9] It is possible to assume certain restrictions on the entries of K, such as that skew is zero and that the pixels are square, thereby diminishing the number of variable parameters

writing

$$C = KK^\top = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

the equation (6.9) gives rise to a set of nine linear equations in the six independent entries of C. It may be seen that multiplying C by a constant factor does not have any effect on the equation (6.9). Consequently, C can only be solved up to a constant factor. It turns out that because of redundancy, the nine equations derived from (6.9) for a single known transformation B_i are not sufficient to solve for C. However, if two or more such B_i are known, then we may solve for C. In particular, for each view and corresponding B_i for $i = 1, \ldots, N-1$ we have nine equations in the entries of C. This overconstrained system of equations may be written in the form $X\mathbf{a} = 0$, where X is a matrix of dimension $9(N-1) \times 6$ and the vector $\mathbf{a}$ contains the independent entries of C. The least-squares solution $\mathbf{a}$ is the eigenvector corresponding to the least eigenvalue of $X^\top X$. This is easily found using the Jacobi method for finding the eigenvalues of a symmetric matrix ([55]). Note that the views are numbered starting at 0, so we need three views to provide two independent transforms B_i, and hence to solve for C.

Once $C = KK^\top$ is found it is an easy matter to solve for K using the Choleski factorization ([1, 55]). A solution for K is only possible when C is positive-definite. This is guaranteed for noise-free data, since by construction, C possesses such a factorization. If we insist that the diagonal entries or K are positive, then the Choleski factorization $C = KK^\top$ is unique.

In cases where the input data is defective, or the plane at infinity is not accurately known it is possible that the matrix C turns out not to be positive-definite, and so the calibration matrix can not be found. In practice however, the algorithm works extremely well, provided the plane at infinity is accurately placed and there are no gross inaccuracies (mistaken matched points) in the data.

As has been remarked previously, the matrix $C = KK^\top$ has a geometric interpretation. It is the dual of the image of the absolute conic. The condition that $C = BCB^\top$ is related to the fact that C is invariant under translation and rotation of the camera.

Euclidean reconstruction from Affine Constraints

If certain collateral data is given that allows the affine structure of the scene to be determined, then this algorithm can be used to determine the Euclidean structure. For instance, if three independent pairs of parallel lines are known, then these can be used to determine where the true plane at infinity lies in a projective reconstruction. In particular, the points of intersection of the parallel lines must all lie on the plane at infinity. Given three pairs of lines, and hence three points on the plane at infinity the plane at infinity is determined. This determines the affine structure of the scene. The above algorithm then may be used to determine the Euclidean reconstruction of the scene.

Another affine constraint that may be used is a known ratio of distances of points on a line. For instance, suppose collinear points O, A and B are given and the ratio of distances $OA/OB = a/b$ is known. The line OAB in a projective reconstruction may be parametrized such that O, A and B have parameter values 0, a and b. The point with parameter ∞ on this line must lie on the plane at infinity.

Another method using Euclidean constraints to get the Euclidean reconstruction of a scene is reported by Boufama [7]. On the other hand, Sparr ([69]) gives a method of computing affine structure given a single view, and Koenderink and van Doorn [40] give a method for computing affine structure from pairs of orthographic views. Quan [57] gives a method of affine construction from two views given affine constraints.

6:4.2 Quasi-affine Reconstruction

We have seen that once an affine reconstruction of the scene is known, it may be transformed by linear means into an Euclidean reconstruction. One way of finding an affine reconstruction is to rely on known geometric information about the scene. We are interested, however, in finding the plane at infinity without any extra given information. We are unaware of any direct method, generally applicable of finding the plane at infinity in the scene, equivalent to performing affine reconstruction. On the other hand, in part IV of this report we described a method of finding a quasi-affine reconstruction of the scene. This algorithm does not precisely locate the plane at infinity, but it finds an approximation to the plane at infinity, at least placing it correctly with respect to the convex hull of the point set in question. Our strategy is, rather than to find an affine reconstruction of the scene, to be content with a quasi-affine reconstruction, and proceed by iteration from there to find the Euclidean structure.

The first step will be to get an approximation to the plane at infinity. This can be done by solving the cheiral inequalities as in section 4:7 to find the plane at infinity. By solving these cheiral inequalities, we find a candidate value for $\mathbf{v}$, defined as in (6.6). By the method of Section 6:4.1 we can now compute the corresponding value of K. This estimate may then be refined using the method described in Section 6:3. There is one flaw in this scheme, namely that it may not be possible to find K corresponding to the estimated $\mathbf{v}$, because the matrix C, which should equal $KK^\top$, is not positive definite. In this case, it is necessary to select a different $\mathbf{v}$. This may be done by carrying out a random search over the convex region of 3-space defined by the cheirality inequalities. In fact, a reasonable approach is to find several candidate vectors $\mathbf{v}$ and iterate from each of them, finally selecting the best solution. This is what has been done in practice.

6:4.3 Algorithm Outline

Since the details of the outline have been obscured by the necessary mathematical analysis, the complete algorithm for Euclidean reconstruction will now be given. To understand the details of the steps of the algorithm, the reader must refer to the relevant section of the previous text.

1. Compute a projective reconstruction of the scene (Section 2:1.5)
   (a) Compute the fundamental matrix F for a pair of images and use this to parametrize the first two cameras, and reconstruct the points
   (b) Use LM iteration to refine this initial projective reconstruction.
   (c) Parametrize the other cameras by the DLT method. Compute new point locations as appropriate.
   (d) Refine the complete projective reconstruction using LM iteration.

2. Compute a quasi-affine reconstruction of the scene (Section 6:4.2)
   (a) Formulate the cheiral inequalities for the projective reconstruction
   (b) Use LP to solve the inequalities to find a vector $\mathbf{v}$.
   (c) Use the transformation matrix $H = \begin{bmatrix} I & 0 \\ \mathbf{v}^\top & 1 \end{bmatrix}$ to transform the projective reconstruction to a quasi-affine reconstruction.
3. Search for a quasi-affine reconstruction from which the calibration matrix K may be computed (Section 6:4.2)
   (a) For a randomly selected set of vectors $\mathbf{v}$ contained within the region determined by the cheiral inequalities solve the equations ${CB_i}^{-\top} = B_i C$ as described in Section 6:4.1 until we find a $\mathbf{v}$ such that the solution C is positive-definite.
   (b) Determine K by Choleski factorization of $C = KK^\top$.
4. Carry out LM iteration using the method of Section 6:3 to find a Euclidean reconstruction.
5. Using the values of K, R_i and $\mathbf{x}_j$ that come out of the previous step, do a complete LM iteration to find the optimal solution minimizing the image-coordinate error, using the method described in Section 6:3.1.

Various comments are in order here. First of all, some of the steps in this algorithm may not be necessary. Step 1(b) of the algorithm may not be needed, but it is easy to include and ensures an accurate starting point for the computation of the other camera parameters. The second step (determination of a specific quasi-affine reconstruction) may not be necessary either, since the third step does a search for a modified quasi-affine reconstruction. However, it is included, since it provides a point of reference for the subsequent search. The vector $\mathbf{v}$ found in the third step of the algorithm should be small, so that the modified quasi-affine reconstruction is close to the original one. In fact, as mentioned previously it is possible to use the cheiral inequalities to give bounds on the individual entries in the vector $\mathbf{v}$. Finally, it has been found that the last step of the algorithm, the final iteration is scarcely necessary, and does not make a very large difference to the solution. It commonly decreases the value of the image coordinate error by no more that about 10%, at least when there are many views. In addition, this last step is relatively costly in terms of computation time.

6:4.4 Experimental Evaluation

This algorithm has been evaluated on both real and synthetic data.

Solution with Three Cameras

Since three cameras are the minimum number needed for Euclidean reconstruction the algorithm was tested on synthetic data with three views. The algorithm was found to converge without difficulty for noise-free data, and for data with added gaussian noise of 0.1 and 0.5 pixels in an image of size approximately 700×600 pixels. The degradation

Noise	p_u	p_v	k_v	$skew$	k_u/k_v	Δ
–	3.0000e+02	3.5000e+02	2.5000e+03	2.0000e+01	9.0000e-01	–
0.0	3.0008e+02	3.5003e+02	2.4999e+03	2.0013e+01	8.9999e-01	0.0
0.1	2.7604e+02	3.3369e+02	2.5590e+03	1.7532e+01	8.9947e-01	0.09
0.5	1.2937e+02	2.3553e+02	2.9044e+03	3.2273e+00	8.9715e-01	0.50
1.0	-2.5284e+02	-1.1118e+01	3.5934e+03	4.6454e+01	8.7611e-01	5.67
2.0	2.3709e+02	2.7905e+02	2.3448e+03	6.6483e+01	8.7752e-01	5.22

Table 6.1: Reconstruction from Three Views

Noise	p_u	p_v	k_v	$skew$	k_u/k_v	Δ_1	Δ_2	Δ_3
–	5.00e+02	4.00e+02	1.0000e+03	-5.0000e+01	9.0000e-01	0.0	0.0	0.0
0.0	5.00e+02	4.00e+02	9.9999e+02	-5.0000e+01	9.0000e-01	9.805e-08	0.0	0.0
0.5	4.99e+02	3.98e+02	9.9959e+02	-4.9857e+01	9.0045e-01	8.359e-04	0.95	0.88
1.0	4.99e+02	3.97e+02	9.9911e+02	-4.9722e+01	9.0091e-01	1.678e-03	1.91	1.76
2.0	4.98e+02	3.95e+02	9.9792e+02	-4.9472e+01	9.0185e-01	3.386e-03	3.82	3.52
4.0	4.97e+02	3.90e+02	9.9463e+02	-4.9062e+01	9.0376e-01	6.911e-03	7.64	7.04
8.0	4.93e+02	3.81e+02	9.8455e+02	-4.8618e+01	9.0768e-01	1.454e-02	15.25	14.00
16.0	4.84e+02	3.67e+02	9.5125e+02	-4.9325e+01	9.1536e-01	3.314e-02	30.10	27.05

Table 6.2: Reconstruction from 15 Views

becomes progressively worse for greater degrees of noise, however the ratio k_u/k_v remains relatively stable. These results are shown in Table 1. The first line gives the correct values for the camera parameters. Subsequent lines show greater degrees of noise. The final column marked Δ gives the residual RMS pixel error, that is, the difference between the measured image coordinates and the ones derived from the reconstruction. This error should be of magnitude comparable with the noise level.

Solution with Large Numbers of Views

The algorithm was then carried out on synthetic data with 15 views of 50 points. The 50 points were randomly scattered in a sphere of radius 1 unit. The cameras were given random orientations and were placed at varying distances from the centre of the sphere at a mean distance from the centre of 2.5 units with a standard deviation of 0.25 units. They were placed in such a way that the principal rays of the cameras passed through randomly selected points on a sphere of radius 0.1 units. The calibration matrix was given a known value. In order to assess the quality of the Euclidean reconstruction the positions of the reconstructed points were compared with the known locations of the 3D points. Since the reconstructed points and the original points are not known in the same coordinate frame, it is necessary to align the two sets of points first. Then the RMS error was computed and used as a measure of quality of the reconstruction. The algorithm of Horn ([38]) was used to compute a rotation, translation and scaling that bring the reconstructed points into closest-possible alignment with the original point locations.

The results are shown in Table 2. The first line gives the correct values of the camera

parameters and subsequent lines show the computed values with added noise. The last three columns have the following meaning.

Δ_1 The error in reconstruction, namely the distance between the actual and the reconstructed point locations.

Δ_2 The residual pixel error after step 4 of the algorithm in Section 6:4.3.

Δ_3 The residual pixel error after step 5 of the algorithm. This shows only a 10% reduction compared with Δ_2.

As can be seen from the Table 2, the results of the reconstruction are extremely good and immune to noise, both as regards the extracted camera calibration parameters and the quality of the point reconstruction. Even for gaussian noise as high as 16 pixels standard deviation in an image of size approximately 600×600 (far greater levels of noise than will be encountered in practice) the camera parameters are reasonably accurate, and the reconstruction is accurate to within 0.033 units, or 3.3 centimetres in a sphere of radius 1 metre. Note that the three error estimates show extraordinary linearity in terms of the added noise.

Solution with Real Data

The algorithm was evaluated on a set of image coordinate correspondences kindly supplied by Boubakeur Boufama and Roger Mohr. The object in question was a wooden house, for which 9 views were used and a total of 73 points were tracked, not all points being visible in all views. This is the same image set as used in the paper [48]. The image coordinates were integer numbers ranging between 0 and 500. Figure 6.1 shows one of the views of the house. The algorithm converged very successfully on this data. The measured residual RMS pixel error was found to be 0.6 pixels per point, which is about as good as can be expected, since the image correspondences were not supplied with sub-pixel accuracy. Not having any ground truth information, I was unable to compare the reconstruction against the correct points. The right side of Figure 6.1 shows a reconstructed view of the set of 73 points looking directly down the edge of the house. Clearly visible is the corner of the house, showing a right-angled corner. This indicates the success of the Euclidean reconstruction, since angles are a Euclidean attribute of the scene.

There is, however, one reason to suspect the accuracy of the reconstruction. In cases where all the camera rotations are about a common axis (as occurs when the camera is stationary and the image rotates), it appears that the problem is not well posed, for the scene may be expanded in the direction of the rotation axis at will. This is possibly the case in this present case, since the the computed camera parameters showed non-square pixels, which seems to be unlikely.

6:5 Retrieving Focal Lengths

In section 6:4 of this report it was shown that complete camera calibration and Euclidean reconstruction is possible from three views with the same camera. The question of what is possible with just two views naturally arises. If the calibration of the cameras is

Figure 6.1: On the left one of the views of a house. On the right a view of the reconstructed house.

also known in advance, then complete Euclidean reconstruction is possible from just two views, as shown by Longuet-Higgins ([42]). In the present section it is shown that Euclidean reconstruction is possible from two views under a weaker assumption than complete known camera calibration. To be precise, the fundamental matrix may be used to solve the relative orientation problem for a pair of pinhole cameras with known principal points, but unknown (possibly different) focal lengths. A simple non-iterative algorithm is given for finding the two focal lengths. Once this is done, known techniques (Longuet-Higgins ([42])) may be used to find the relative orientation of the cameras, and to reconstruct the scene. Note that in this section, we are allowing the possibility that the focal lengths of the two cameras are different.

The fundamental matrix is a 3×3 matrix with 7 degrees of freedom. On the other hand, 5 parameters are sufficient to specify the relative placement of the two cameras. We will show in this section how the two extra degrees of freedom of the fundamental matrix may be used to compute the focal lengths of the two cameras, provided all other internal parameters of the two cameras are known. A very simple algorithm is given for the computation of the two focal lengths, based on projective geometry. This approach allows one to gain an intuitive understanding of the method, as well as allowing the identification of critical configurations where the algorithm will fail.

Experiments are carried out to evaluate the performance. It is shown that the algorithm performs relatively well, and is quite robust in the presence of noisy data, provided the camera geometry is sufficiently far from one of the critical configurations.

6:5.1 The Camera Model

In this section, we will be concerned with the pinhole camera model defined in section 6:5.1. That is, we assume no skew and equal scale factors in both axial directions. In this case the camera projection may be written as

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{bmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} .$$

Given two images, it is not possible to estimate the focal lengths of the two cameras as well as the principal point offset. In this section, it is assumed that the principal point of the camera is known. It then proves possible to compute the focal lengths of the two cameras. The principal point of a camera is rarely known precisely, but for practical purposes, it is sufficient to assume that the principal point is at the centre of the image, unless the image has been cropped. For instance, in [53] good scene reconstruction is achieved under this assumption. The most evident application of this work is for computing the focal length of a camera fitted with a zoom or changeable lens. It is reasonable to assume that the principal point will not move during zooming, though this is dependent on the quality of the zoom lens.

6:5.2 Computation of the Scale Factors

As observed in section 6:6.5, the key to camera calibration is the absolute conic, which is a conic on the plane at infinity consisting of points $(x, y, z, t)^\top$ such that $t = 0$ and $x^2 + y^2 + z^2 = 0$. As was shown in section 6:6.5, the image of the absolute conic as viewed by a camera with matrix $K[R \mid -R\mathbf{t}]$ is the plane conic represented by the matrix $(KK^\top)^{-1}$.

Consider now the special case of two cameras with calibration matrices diagonal, of the form $\mathrm{diag}(k, k, 1)$ and $\mathrm{diag}(k', k', 1)$. In other words, it is assumed that the principal point is known, and is equal to the origin $(0, 0)$ of image coordinates. Under these assumptions, the image of the absolute conic has matrix $\mathrm{diag}(k^2, k^2, 1)^{-1} \approx diag(1, 1, k^2)$ for one camera and $\mathrm{diag}(k'^2, k'^2, 1)^{-1} \approx \mathrm{diag}(1, 1, k'^2)$ for the other. These are circles of imaginary radius centred at the origin. Their equations are $u^2 + v^2 + k^2 = 0$ and $u'^2 + v'^2 + k'^2 = 0$.

Consider now two images J and J'. Let π be a plane in space passing through the two camera centres and tangent to the absolute conic. This plane is mapped into the image J as a line λ through the epipole $\mathbf{e}$ tangent to the image of the absolute conic. Similarly it maps to a line λ' in the second image. The two lines λ and λ' in the two images are a matching pair of epipolar lines. There are in fact two tangent planes through the two camera centres tangent to the absolute conic, which results in two pairs of epipolar lines $\lambda_1 \leftrightarrow \lambda'_1$ and $\lambda_2 \leftrightarrow \lambda'_2$ all tangent to the image of the absolute conic.

Let the two epipoles be $\mathbf{e}$ and $\mathbf{e}'$. We assume that neither $\mathbf{e}$ nor $\mathbf{e}'$ lies at the origin, which means that neither camera lies on the principal ray of the other. Simply by rotating each of the images, it may be ensured that the two epipoles lie on the positive u-axis. Rotating the image about the origin is equivalent to rotating the camera about the principal ray. Note that this observation is true only because of the assumption of now skew s and square pixels, $k_u = k_v$. Otherwise the image will undergo distortion as the camera as rotated.

Suppose that the two epipoles are $\mathbf{e} = (e_1, 0, e_3)$ and $\mathbf{e}' = (e'_1, 0, e'_3)$. Under these conditions, the fundamental matrix has a particularly simple form. From the conditions $F\mathbf{e} = F(e_1, 0, e_3)^\top = 0$ and $\mathbf{e}'^\top F = (e'_1, 0, e'_3)F = 0$, we derive

$$F \approx \begin{bmatrix} e'_3 & & \\ & 1 & \\ & & -e'_1 \end{bmatrix} \begin{bmatrix} a & b & a \\ c & d & c \\ a & b & a \end{bmatrix} \begin{bmatrix} e_3 & & \\ & 1 & \\ & & -e_1 \end{bmatrix} \tag{6.10}$$

for real numbers a, b, c and d.

Now, we consider the tangent lines from the epipole $(e_1, 0, e_3)^\top$ to the conic with matrix $\mathrm{diag}(1, 1, k^2)$. The polar ([60]) of the point $(e_1, 0, e_3)^\top$ with respect to the conic $\mathrm{diag}(1, 1, k^2)$ is the line $(e_1, 0, k^2 e_3)^\top$, in other words the line $e_1 u + k^2 e_3 = 0$. The points of tangency from $(e_1, 0, e_3)^\top$ to the conic are therefore the intersections of the line $e_1 u + k^2 e_3 = 0$ with the conic $u^2 + v^2 + k^2 = 0$. These are the points $\hat{\mathbf{u}} = (-k^2 e_3, ik(e_1^2 + k^2 e_3^2)^{1/2}, e_1)^\top$ and the complex conjugate point. Transforming this point by F, we will obtain the corresponding epipolar line in the second image, namely the tangent line from $\mathbf{e}' = (e_1', 0, e_3')^\top$ to the conic $\mathrm{diag}(1, 1, k'^2)$. Writing $\Delta = (e_1^2 + k^2 e_3^2)^{1/2}$, we compute

$$\begin{aligned} F\hat{\mathbf{u}} &\approx \begin{pmatrix} (-a\Delta^2 + bik\Delta)e_3' \\ -c\Delta^2 + dik\Delta \\ (a\Delta^2 - bik\Delta)e_1' \end{pmatrix} \approx \begin{pmatrix} (-a\Delta + bik)e_3' \\ -c\Delta + dik \\ (a\Delta - bik)e_1' \end{pmatrix} \\ &\approx \begin{pmatrix} -e_3' \\ \frac{-c\Delta + dik}{a\Delta - bik} \\ e_1' \end{pmatrix} = \begin{pmatrix} -e_3' \\ \frac{(-c\Delta + dik)(a\Delta + bik)}{a^2\Delta^2 + b^2k^2} \\ e_1' \end{pmatrix} \\ &= \begin{pmatrix} -e_3' \\ \frac{-ac\Delta^2 - bdk^2 + ik(ad - bc)\Delta}{a^2\Delta^2 + b^2k^2} \\ e_1' \end{pmatrix} \stackrel{\mathrm{def}}{=} \begin{pmatrix} -e_3' \\ \mu + i\nu \\ e_1' \end{pmatrix} \end{aligned} \tag{6.11}$$

The other tangent line is obtained as the complex conjugate of this line, namely $(-e_3', \mu - i\nu, e_1')$. On the other hand, these two tangent lines are tangents from the epipole $\mathbf{e}'$ to the conic $\mathrm{diag}(1, 1, k'^2)$, which is symmetric about the u-axis. In other words, if $(-e_3', \mu + i\nu, e_1')$ is one tangent line, then the other is $(-e_3', -(\mu + i\nu), e_1')$. Consequently, $-(\mu + i\nu) = \mu - i\nu$, and so $\mu = 0$. This gives

$$\mu = \frac{-ac\Delta^2 - bdk^2}{a^2\Delta^2 + b^2k^2} = 0 \tag{6.12}$$

whence

$$-ac\Delta^2 - bdk^2 = -ac(e_1^2 + k^2 e_3^2) - bdk^2 = 0 \ .$$

Finally, this leads to

$$k^2 = \frac{-ace_1^2}{ace_3^2 + bd} \ . \tag{6.13}$$

A formula for k' may be computed by reversing the role of the two images. This corresponds to taking the transpose of the fundamental matrix. Consequently, we may write

$$k'^2 = \frac{-abe_1'^2}{abe_3'^2 + cd} \ . \tag{6.14}$$

Since the focal lengths must be positive, we select the positive values of k and k' satisfying (6.13) and (6.14).

6:5.3 Algorithm Outline

Now, we are able to describe the complete algorithm for computation of the magnifications of the two cameras. It is assumed that the cameras have camera matrices $\mathrm{diag}(k, k, 1)$ and $\mathrm{diag}(k', k', 1)$ and that the fundamental matrix F corresponding to the

(ordered) pair of cameras is known. Matrix F has rank 2. It is required to determine k and k'.

1. **Determination of the epipoles :** The two epipoles $\mathbf{e}$ and $\mathbf{e}'$ are determined by solving the equations $F\mathbf{e} = 0$ and $F^\top \mathbf{e}' = 0$.

2. **Normalizing the position of the epipoles :** The images are rotated so that the two epipoles both lie on the x-axis, namely at points $(e_1, 0, e_3)^\top$ and $(e'_1, 0, e'_3)^\top$ respectively. Subsequently, correct the fundamental matrix to reflect this change. More precisely, if T and T' are the two image rotations, then F is replaced by the new F equal to $T'^{-\top} F T^{-1}$.

3. Compute the decomposition of F in the form (6.10).

4. **Computation of k and k' :** Compute k and k' according to the formulae (6.13) and (6.14).

Once the scales k and k' are known, it is an easy matter to compute the relative placement of the two cameras. One simply computes the essential matrix $Q = \mathrm{diag}(k', k', 1)^\top F \,\mathrm{diag}(k, k, 1)$ for cameras with identity calibration matrices and then computes the camera placement using the algorithm given in [42] or [23].

6:5.4 Failure

No solution possible : There may be no solution possible, if the right sides of (6.13) or (6.14) are negative. This indicates that the data is faulty. The fundamental matrix F does not correspond to a pair of cameras with the assumed simple calibration matrices.

The Principal Rays Meet : There are other cases, however, in which the algorithm may fail. The most interesting situation is when the principal rays of the cameras intersect. Any point along the principal ray is projected to the principal point in the image, which is assumed to be the origin $(0, 0, 1)^\top$. Therefore, the point of intersection of the principal rays maps to the origin in both images. This is an image correspondence $(0, 0, 1)^\top \leftrightarrow (0, 0, 1)^\top$. It follows that $(0, 0, 1)F(0, 0, 1)^\top = 0$, and hence that the $(3, 3)$-entry of F is zero. With F as in (6.10), this means $a = 0$. Now, from (6.12), we have $\mu = -bdk^2/b^2k^2 = -d/b$. It follows that $\mu = 0$ if and only if $d = 0$, and this condition is independent of k. This means that k and k' can not be determined from the fundamental matrix. This condition is somewhat troublesome in practice, since if two images of the same object are taken, then there is a good chance that they may both be taken with the camera aimed at the same point in the scene.

In this case, however, the two magnification factors k and k' may not be varied independently. In fact, if $a = d = 0$, then from (6.11) we see that $F\hat{\mathbf{u}} = (-1, ic\Delta/bk, 1)^\top$. These are the two tangent lines to the image of the absolute conic in the second image. On the other hand, the points of tangency from point $\mathbf{e}' = (1, 0, 1)^\top$ to the conic $u'^2 + v'^2 + k'^2 = 0$ are the two points $(-k'^2, \pm ik'\Delta', 1)^\top$. However $F\hat{\mathbf{u}}$ is a tangent line, and hence passes through one of these points. Multiplying out, we get

$$\Delta'^2 \pm \frac{c\Delta k'\Delta'}{bk} = 0$$

from which it follows that $b\Delta'/k' = \pm c\Delta/k$. Squaring, substituting for Δ and Δ' and simplifying gives

$$b^2(1 + k'^{-2}) = c^2(1 + k^{-2}) \ .$$

The two magnification factors may vary freely as long as they satisfy this relationship.

Other Configurations : A similar situation occurs when $b = 0$. From (6.12) one deduces as before that $c = 0$, and the condition that $\mu = 0$ is independent of k.

Uniqueness It is evident that the geometric configurations that lead to failure of the algorithm through ambiguity are somewhat special cases. In fact, in the space of all possible configurations of cameras, the set of configurations that give rise to ambiguous solutions in the determination of the focal lengths constitute a critical set of lower dimension. This allows us to state the following uniqueness result.

Theorem 6.1. *For almost all configurations of a pair of pinhole cameras with known principal points, the focal lengths and relative orientation of the two cameras are determined uniquely by the correspondence of points seen in the two images.*

The words "almost all configurations" are to be interpreted as meaning all configurations, except those lying in a lower dimensional critical set. Similarly, the term pinhole camera is intended to denote a camera carrying out central projection from object space onto an image plane. As explained previously, such a camera has isotropic coordinates in the image plane, and no skew.

6:5.5 Experiments

Simulations were carried out to evaluate the algorithm in the presence of varying degrees of noise, and with varying proximity to a critical configuration. The set of experiments were carried out with a simulated 35mm camera with a focal length of 28mm. This focal length is equal to the lower limit of focal lengths for one commonly available zoom lens. Such a camera has a field of view approximately 64°. This is a fairly wide field of view, but also within typical ranges for aerial cameras. For this experiment, both simulated cameras had the same focal length. Images were assumed to be digitized with pixels of size 70μ, which means 500 pixels across the 35mm width of the image. The f parameter of the images was 400 (the focal length in pixels).

Two images were assumed taken with one camera rotated 30 degrees from the other. The two principal axes were skew lines in space chosen to be varying distances apart. A set of 30 points lying in a sphere of diameter 1.5 units were viewed by the two cameras. The cameras were placed so that the set of points approximately filled the field of view of each of the cameras. The images of the points were computed in each of the two cameras.

As seen in Section 6:5.4, if the principal axes meet, then the algorithm will fail and the focal lengths can not be determined. Suppose the principal axis of each camera could actually be seen in the image taken with the other camera. If the two principal axes meet in space, then the image of the principal axis of one camera, as seen by the other camera, will pass through the principal point of the image. The distance (in pixels) from the principal point in an image to the image of the principal axis of the other camera

is a measure of how far the configuration differs from the critical configuration. This parameter will be denoted by α. For the configurations used in the experiment, the value of α was the same for both images (though this need not be so).

To determine how closeness to the critical configuration affects the stability of the computation a set of tests were carried out with values of α equal to 20, 39, 58 and 75 pixels. For each α, varying degrees of gaussian noise was added to each image pixel coordinate. The standard deviation of the added noise varied from 0.25 pixels to 5 pixels in each axial direction, for each point. For each noise level and value of α the computation was repeated 100 times with different added noise.

To measure the performance of the algorithm, four criteria were used.

1. Successful completion of the algorithm. In some cases, because of noise, the values on the right of (6.13) or (6.14) will be negative, and no solution is possible.
2. Accuracy of the computation of the focal length.
3. Accuracy of the ratio of the focal lengths of the two cameras. It turns out that errors in the computed focal lengths of the two cameras are strongly correlated. The ratio of the focal lengths is more accurately computed than the individual focal lengths.
4. Reconstruction error. Once the focal lengths were computed, the scene was reconstructed and compared with the actual values of the object points. To do this the reconstructed scene was superimposed on the actual scene using the algorithm of Horn ([38]) and the RMS construction error was measured.

The results of these experiments are reported in Figs 1 – 4.

6:5.6 Conclusions

The algorithm described in this section provides a very simple method of computing the focal lengths and relative position and orientation of a pair of pinhole cameras given a set of image correspondences. As long as the principal axes of the two cameras are sufficiently distant from each other the algorithm performs quite well in the presence of reasonable amounts of noise. Realistic noise levels should be on the order of at most 2 pixels in each axial direction. The deviation of the estimated from the correct values of focal length are due to the inherent instability of the problem, rather than to an inadequacy of the algorithm. This has been demonstrated by using least-squares minimization methods to find the optimal solution iteratively. Images taken with a narrow field of view and long focal length cameras differ only slightly from orthographic views, and what difference there is can easily be swamped by noise.

6:6 Camera Rotating about a Fixed Point

In this section, another method is given for calibration of a camera in the case of a certain restricted motion of the camera. This differs from the other self-calibration methods discussed in the previous sections, in which unrestricted movements of the camera were allowed. In the method discussed in this section, at least three images are taken from the

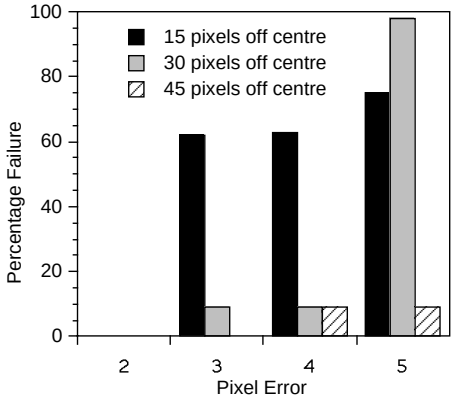


Figure 6.2: **Failure.** *The algorithm fails if the value of k^2 or k'^2 computed from (4) and (5) is negative. This plot shows the percentage failure of the algorithm. For RMS pixel errors of less than 3 pixels, the algorithm succeeded in all cases. Similarly for a value of α of 75 pixels, the algorithm always succeeded for all noise levels. The graphs shows the percentage failure for values of α of 20, 39 and 58 pixels at RMS noise levels of 3, 4 and 5 pixels. For a value of α of 20 pixels, the algorithm is not reliable for RMS pixel errors exceeding 2 pixels.*

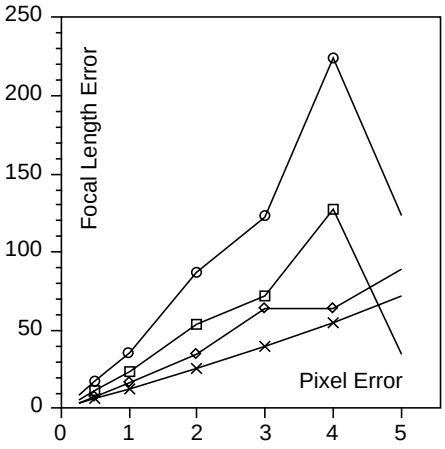


Figure 6.3: **Error in estimation of focal length.** *The plot shows the standard deviation in the estimate of focal length at different noise levels. In all cases, the mean estimated focal length was close to 400 (the correct value). The four plots show the standard deviation for values of α equal to (from the top) 20, 39, 58 and 75 pixels. Thus, in most cases, for noise levels less than 2 pixels, the estimated focal length lies in the range between 350 and 450. The apparent improvement in performance for pixel errors of 5 pixels in the top two plots is due to the fact that the algorithm failed for a substantial percentage of such tests, which were therefore not included in this statistic.*

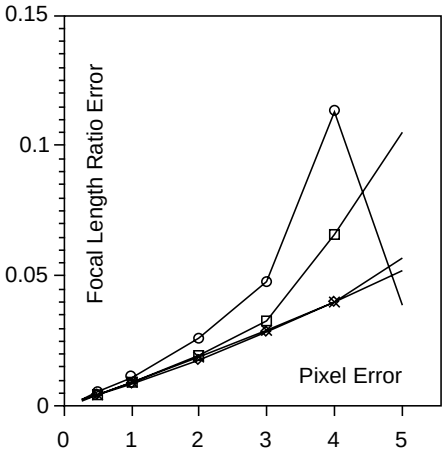


Figure 6.4: **Error in focal length ratio.** *This plot shows standard deviation in the estimated ratio of the focal lengths of the two cameras for differing noise levels. Each curve represents a different value of α – from the top 20, 39, 58 and 75 pixels. The graph shows that except for high noise levels, the estimated ratio of the two focal lengths lies between 0.95 and 1.05. Once more the apparent improvement in performance for $\alpha = 20$ pixels at a noise level of 5 pixels is due to the exclusion of those cases where the algorithm fails.*

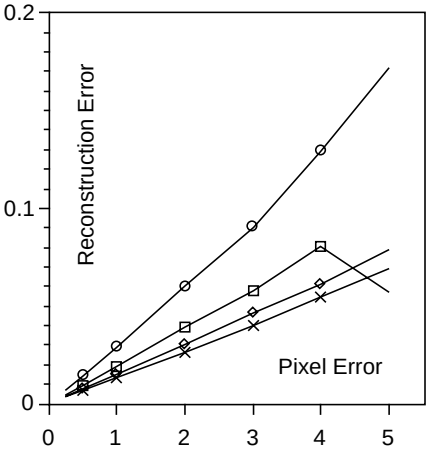


Figure 6.5: **Reconstruction Error.** *This graph shows the RMS reconstruction error for different noise levels and the same values of α as before. The vertical axis gives the absolute RMS reconstruction error in length units where the reconstructed set of points has radius about 1.5 units.*

same point in space with different orientations of the camera and calibration is computed from an analysis of point matches between the images. The method requires no knowledge of the orientations of the camera. Calibration is based on the image correspondences only. This method differs fundamentally from the previous methods of self-calibration using the epipolar structure of image pairs. In the method of this section, there is no epipolar structure since all images are taken from the same point in space, and so methods based on epipolar structure, and Kruppa's equations do not apply. Since the images are all taken from the same point in space, determination of point matches is considerably easier than for images taken with a moving camera, since problems of occlusion or change of aspect or illumination do not occur. A non-iterative calibration algorithm is given that works with any number of images. An iterative refinement method that may be used with noisy data is also described. The algorithm is implemented and validated on several sets of synthetic and real image data.

Recently several papers on self-calibration have appeared ([10, 3, 11]). relying on known motions of the cameras. In [10] the motion of the camera is assumed to be purely translational. In [3, 11] rotational motions of the camera are considered, but the rotation must be through known angles. This simplifies the calibration task enormously. For instance, in this case, the focal length of the camera can be estimated simply as a ratio of feature displacement to incremental angle of rotation ([11]). In addition, the methods of [3, 11] require tracing features in the image through many frames. In [3] an approximate guess at the location of the principal point is also necessary. In this section, on the other hand, calibration is carried out solely on the basis of image content, and without *a priori* assumptions of calibration values. Calibration can be carried out by finding point matches in as few as three images, though for best results, more images may be used. The method is based on analysis of the projective distortion that an image undergoes when the camera is rotated.

The calibration algorithm is demonstrated on real and synthetic data and is shown to perform robustly in the presence of noise.

The purpose of this section is to give a method for determining the matrix K of internal camera parameters of a perspective camera. In the method to be described, the camera will be held in the same location in space and rotated to different orientations. For convenience, the common location of all the cameras will chosen to be the origin of the coordinate system. We will speak of several cameras each with its own camera matrix, whereas in fact the cameras will be the same camera, with the same interior parameters, differing only in their orientation. Thus, we consider a set of cameras with camera matrices $M_j = K[R_j \mid 0]$. Often, we will identify a camera with its transformation matrix.

A point $\mathbf{x} = (x, y, z, 1)^\top$ is mapped by the camera M_j to the point $\mathbf{u} = K[R_j \mid 0](x, y, z, 1)^\top = KR_j(x, y, z)$. In other words, since the last column of M_j is always 0, the fourth coordinate of $\mathbf{x}$ is immaterial. Therefore, in this section, we will drop the fourth column of the camera matrix, and write instead

$$M_j = KR_j$$

where K is upper triangular, the same for all cameras, and R_j is a rotation matrix. This transformation sends points $\mathbf{x} = (x, y, z)^\top$ to $\mathbf{u} = KR_j\mathbf{x}$. Note that the points $k\mathbf{x}$, where k is a non-zero factor, are all mapped to the same point independent of the scale factor. Consequently, M_j represents a mapping between a two-dimensional projective object space with coordinates $(x, y, z)^\top$ and two-dimensional projective image space with

coordinates $(u, v, w)^\top$. This situation has a very convenient feature, not shared by the usual 3D to 2D projective mapping, namely that the mapping M_j from object to image space is invertible.

6:6.1 Rotating the Camera

Now, we will consider what happens to an image taken by a camera when the camera is rotated. Thus, let $M = KR$ and $M' = KR'$ be two cameras, and let $\mathbf{u}_i = KR\mathbf{x}_i$ and $\mathbf{u}'_i = KR'\mathbf{x}_i$. From this it follows that

$$\mathbf{u}'_i = KR'R^{-1}K^{-1}\mathbf{u}_i$$

This simple observation gives the following important result

Proposition 6.2. *Given a pair of images taken by cameras with the same interior parameters from the same location, then there is a projective transformation P taking one image to the other. Furthermore, P is of the form $P = KRK^{-1}$ where R is a rotation matrix and K is the calibration matrix.*

In standard terminology, the relation $P = KRK^{-1}$ may be described by saying that P is a conjugate of a rotation matrix, K being the conjugating element. Since P is meaningfully defined only up to a non-zero factor, Proposition 6.2 may be interpreted as meaning that $P = KRK^{-1}$ only up to a non-zero factor. However, the right hand side of this equation has unit determinant. Therefore, if P is chosen to have unit determinant (as may always be done by multiplying P by an appropriate factor if necessary), then exact equality will hold.

Now, suppose we have several cameras with matrices $M_j = KR_j$ for $j = 0, \ldots, N$. For convenience, we assume that the coordinate axes are chosen to be aligned with the 0-th camera, so that $R_0 = I$, the identity matrix, and hence $M_0 = K$. Write $P_j = M_j M_0^{-1} = KR_jK^{-1}$. This gives the following proposition.

Proposition 6.3. *Given a set of images $J_0, \ldots J_N$ taken from the same location by cameras with the same calibration (or with the same camera), then there exist 2D projective transforms, represented by matrices P_j, taking image J_0 to image J_j. The matrix P_j may be written in the form*

$$P_j = KR_jK^{-1}$$

where K is the common calibration matrix of the cameras, and R_j represents the rotation of the $j-th$ camera with respect to the 0-th. The camera matrix for the j-th camera is $M_j = KR_j = P_jK$.

6:6.2 Algorithm Idea

The idea of the calibration algorithm will now be described. Suppose we are given a set of overlapping images $J_0, J_1, \ldots, J_N$ where $N \geq 2$, all taken from the same location with cameras with the same calibration (or the same camera). It is required to determine the common calibration matrix of the cameras. The steps of the algorithm are as follows.

1. Establish point correspondences between the images. (Section 6:6.7)

2. For each $j = 1, \ldots, N$ compute the 2D projective transformation P_j matching J_0 to J_j. (Section 6:6.3)

3. Find an upper triangular matrix K such that $K^{-1}P_jK = R_j$ is a rotation matrix for all $j > 0$. The matrix K is the calibration matrix of the cameras, and R_j represents the orientation of the $j - th$ camera with respect to the 0-th camera. (Section 6:6.4)

4. Refine the estimated camera matrix using Levenberg-Marquardt iterative techniques. (Section 6:6.6)

The steps of this algorithm will be described in detail later in this section of the report, as indicated. The main subject of this section comprises the last three steps of this algorithm, which will be described first. The first step (establishing point correspondences) is of peripheral interest, and a description of the method used for point matching in validation of this algorithm will be postponed to a later section.

6:6.3 Determination of the Transformations

Consider a set of matched points $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$. It is required to find a two-dimensional projectivity, P mapping each $\mathbf{u}_i$ to $\mathbf{u}'_i$. In the presence of noise, the matches will not be exact. Therefore, a best approximation will be computed instead. First, of all, a quick linear, but non-optimal method for computing P will be described.

Linear determination of P

Writing $\mathbf{u}_i = (u_i, v_i, 1)^\top$ and $\mathbf{u}'_i = (u'_i, v'_i, 1)^\top$, the 2D transform is given by the equation $w'_i(u'_i, v'_i, 1)^\top = P(u_i, v_i, 1)^\top$, where w'_i is unknown. Denoting the rows of P by vectors $\mathbf{p}_1{}^\top$, $\mathbf{p}_2{}^\top$ and $\mathbf{p}_3{}^\top$, this set of equations can be written as

$$\begin{aligned} w'_i u'_i &= \mathbf{p}_1{}^\top \mathbf{u}_i \\ w'_i v'_i &= \mathbf{p}_2{}^\top \mathbf{u}_i \\ w'_i &= \mathbf{p}_3{}^\top \mathbf{u}_i \end{aligned}$$

Eliminating the unknown w'_i leads to two equations

$$\begin{aligned} \mathbf{p}_3{}^\top \mathbf{u}_i u'_i &= \mathbf{p}_1{}^\top \mathbf{u}_i \\ \mathbf{p}_3{}^\top \mathbf{u}_i v'_i &= \mathbf{p}_2{}^\top \mathbf{u}_i \end{aligned}$$

This is a set of two linear equations in the entries of P. Four such point matches provide a set of eight equations in the entries of P. Since P is determined only up to a scale, this is enough equations to solve linearly for the entries of P. If there are more than four matched points, then we have an overdetermined set of equations of the form $A\mathbf{p} = 0$, where $\mathbf{p}$ is a vector consisting of the entries of P. We seek to find $\mathbf{p}$ such that $||\mathbf{p}|| = 1$ and such that $||A\mathbf{p}||$ is minimized. The solution is the eigenvector corresponding to the smallest eigenvalue of $A^\top A$, and may be conveniently found using the Singular Value Decomposition of A or Jacobi's method to find the eigenvalues of the symmetric matrix $A^\top A$ ([55]).

Computing all the transforms

Now, we consider the case where point matches are known in several images. It is not assumed, however, that all points are visible in all images. As a first step the images are reordered. We start by choosing the image J_0 to be the one for which the greatest number of image matches are given. The next image J_1 is the image with the greatest number of matches with J_0, then J_2 is the image with the greatest number of matches with J_0 and J_1. Once J_i is chosen, then J_{i+1} is the image with the greatest number of matches with the images $J_0, \ldots, J_i$.

We choose $P_0 = I$, the identity transform. It is desired to find the other transformations P_j for $j > 1$. Suppose that transformations P_0 to P_{j-1} have been determined and we are to determine the transformation P_j. Consider a matched point between a point in image k and a point in image j, where $k < j$. We denote this as $\mathbf{u}^k \leftrightarrow \mathbf{u}^j$ where the superscripts identify the image involved. Since the transformation P_k is known, we may relate the point $\mathbf{u}^k$ back to a point $P_k^{-1}\mathbf{u}^k$ in image J_0. Thus, the match $\mathbf{u}^k \leftrightarrow \mathbf{u}^j$ between points in the k-th and j-th images is equivalent to a match $P_k^{-1}\mathbf{u}^k \leftrightarrow \mathbf{u}^j$ between points in images J_0 and J_j. If there are at least four of these image matches, we may solve for the transformation P_j such that $P_j P_k^{-1}\mathbf{u}^k = \mathbf{u}^j$ for all the matched points. If there are more than four matches, the equation is of course to be solved in the least-squares sense described previously. Proceeding in this way, we identify all the transformations P_j, as long as sufficient matches are given.

Refining the transforms

First, suppose that a set of matches $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ are known between a pair of images. Suppose that there are errors in the measurement of both $\mathbf{u}_i$ and $\mathbf{u}'_i$, and suppose further that errors are Gaussian and independent (the usual assumption). Then, the optimum (maximum likelihood) transform P is found by estimating the transform P and points $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{u}}'_i$ (the "correct" values of $\mathbf{u}_i$ and $\mathbf{u}'_i$) so as to minimize the squared error sum,

$$\sum d(\mathbf{u}_i, \hat{\mathbf{u}}_i)^2 + d(\mathbf{u}'_i, \hat{\mathbf{u}}'_i)^2$$

where $\hat{\mathbf{u}}'_i = P\hat{\mathbf{u}}_i$, and $d(*)$ represents Euclidean distance. This non-linear problem can be solved by iterative techniques starting from an initial guess with $\hat{\mathbf{u}}_i = \mathbf{u}_i$ and $\hat{\mathbf{u}}'_i = \mathbf{u}'_i$ and P provided by the linear solution.

This method generalizes to the case of several images with point matches. Denote by $\mathbf{u}_i^j$ the coordinates of some point $\mathbf{x}_j$ as seen in the i-th image. It is not assumed that all points are seen in all images. Once more, we assume that errors in the measured coordinates $\mathbf{u}_i^j$ are gaussian and independent. The optimal estimate of the transformations P_j is obtained by minimizing the error term

$$\sum d(\mathbf{u}_i^j, \hat{\mathbf{u}}_i^j)^2 \tag{6.15}$$

where the sum is over all pairs (i, j) for which $\mathbf{u}_i^j$ is defined. The values $\hat{\mathbf{u}}_i^j$ are estimates of the "correct" image point locations, which must satisfy the equation

$$\hat{\mathbf{u}}_i^j = P_j P_k^{-1} \hat{\mathbf{u}}_i^k \tag{6.16}$$

whenever both $\mathbf{u}_i^j$ and $\mathbf{u}_i^k$ are defined. Both the transformations P_j and the estimates $\hat{\mathbf{u}}_i^j$ are to be varied in minimizing the error expression (6.15), subject to the constraint (6.16).

One can turn this into an unconstrained minimization problem by introducing variables $\mathbf{x}_i$ defined by the equation $\hat{\mathbf{x}}_i = P_j^{-1}\hat{\mathbf{u}}_i^j$ if $\mathbf{u}_i^j$ is defined. According to (6.16), it does not matter which point $\hat{\mathbf{u}}_i^j$ is used in defining $\hat{\mathbf{x}}_i$, since $P_j^{-1}\hat{\mathbf{u}}_i^j = P_k^{-1}\hat{\mathbf{u}}_i^k$ for all applicable j and k. The problem now becomes to minimize the error expression (6.15), where $\hat{\mathbf{u}}_i^j = P_j\hat{\mathbf{x}}_i$. The transforms P_j and the points $\hat{\mathbf{x}}_i$ are to be varied in minimizing (6.15), but transform P_0 should be locked to the value $P_0 = I$ in to avoid over-parametrization. The problem is solved by a Levenberg-Marquardt iterative minimization method ([55]). The transformations P_j are initialized to the values found using the non-iterative method given above, and the value of $\hat{\mathbf{x}}_i$ is initially set to $P_j^{-1}\mathbf{u}_i^j$ for some j such that $\mathbf{u}_i^j$ is known.

This problem is essentially the same as a camera parameter estimation problem as described in [68]. In fact, what we are doing is effectively equivalent to computing a projective reconstruction of the scene. The vectors $\hat{\mathbf{x}}_i$ represent the directions to the reconstructed scene points. It is of course not possible to determine the depths of the points from the common camera centre. The minimization problem was solved with minimal extra coding using a general purpose camera-parameter estimation program called *Carmen*written by the author. The general method used is a Levenberg-Marquardt least-squares parameter estimation method ([55]). The minimization problem formulated in the previous paragraph has the advantage that measurements in each of the images are treated equally. If one assumes that the pinhole camera model is exact, and that errors in measurements take the form of independent gaussian variables, then this problem formulation leads to an estimate of the transformations, P_j, corresponding to the optimal (maximum likelihood) estimate of the true image point positions.

It may appear that there are a large number of parameters to be estimated, namely the entries of each of the transformation matrices P_j, as well as the values of $\mathbf{x}_i$ for all i. However, in solving this problem one may (in fact must) take advantage of the block structure of the Jacobian matrix (of the measurements with respect to the parameters). At each iteration, this allows one to compute the updated estimates of the transforms P_j first, and then to get the values of $\mathbf{x}_i$ by a sort of back-substitution. This is a standard technique in photogrammetry, and is well described in the Manual of Photogrammetry ([68]), and also in [29]. Using this technique, it was possible to handle cases with more than 30 transformations and over 4,500 point matches within a reasonable time (a few minutes on a Sun Sparcstation 2). Without this refinement, each iteration would take thousands of times longer, if it would be possible at all.

In estimating the transformations P_j for a large number of cameras using the direct non-iterative approach, it is advantageous to pause after every few transformations are computed to apply the iterative least-squares method. In this way, errors are prevented from accumulating. In our experiments the approach of allowing one step of iteration after the computation of each transformation, P_j, and then five steps of iteration at the end proved more than adequate, while not taking excessive time. If time were important, the iterative estimation steps could be applied less frequently. Alternatively, a Kalman filter approach could be used.

6:6.4 Determining the Calibration Matrix

We now suppose that transformations P_j are known for $j = 1, \ldots, N$. We wish to find the calibration matrix K, which will be an upper triangular matrix satisfying the condition that $K^{-1}P_jK = R_j$ is a rotation matrix for all j. The condition that P should be a conjugate of a rotation matrix means that P is somewhat special, as will be seen now. For any non-singular matrix A, let $A^{-\top}$ be the inverse transpose of A. For a rotation matrix R, we have $R = R^{-\top}$. From the relation $R_j = K^{-1}P_jK$ it follows that $R_j = K^\top {P_j}^{-\top} K^{-\top}$. Equating the two expressions for R_j gives $K^\top {P_j}^{-\top} K^{-\top} = K^{-1}P_jK$, from which it follows that

$$(KK^\top){P_j}^{-\top} = P_j(KK^\top) \tag{6.17}$$

Given sufficiently many views and corresponding matrices P_j equation 6.17 may be used to solve for the entries of the matrix $KK^\top$. In particular, denoting $KK^\top$ by C and writing

$$C = KK^\top = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

the equation (6.17) gives rise to a set of nine linear equations in the six independent entries of C. It may be seen that multiplying C by a constant factor does not have any effect on the equation (6.17). Consequently, C can only be determined up to a constant factor. It turns out that because of redundancy, the nine equations derived from (6.17) for a single known transformation P_j are not sufficient to solve for C (see Section 6:6.9). However, if two or more such P_j are known, then we may solve for C. In particular, for each view and corresponding P_j for $j = 1, \ldots, N$ we have nine equations in the entries of C. This overconstrained system of equations may be written in the form $X\mathbf{a} = 0$, where X is a matrix of dimension $9N \times 6$ and the vector $\mathbf{a}$ contains the independent entries of C. This is the same sort of minimization problem as in Section 6:6.3. The least-squares solution is the eigenvector corresponding to the least eigenvalue of $X^\top X$. Note that the views are numbered starting at 0, so we need three views to provide two independent transforms P_j, and hence to solve for C.

Once $C = KK^\top$ is found it is an easy matter to solve for K by Choleski factorization ([1]). A solution for K is only possible when C is positive-definite. This is guaranteed for noise-free data, since by construction, C possesses such a factorization. With noisy input data, it is possible that the matrix C turns out not to be positive-definite, and so the calibration matrix can not be found. In practice this was found to happen only in the case of gross errors in the point matching. In fact, this algorithm was found to work very well, as will be seen later.

6:6.5 Interpretation of Calibration using the Absolute Conic

This method of camera calibration may be interpreted in terms of the absolute conic. The connection between the absolute conic and camera calibration is well known. For instance, in [46] it is shown how Kruppa's equations ([41]) are related to the dual of the absolute conic.

The absolute conic is a conic on the plane at infinity consisting of points $(x, y, z, t)^\top$ such that $t = 0$ and $x^2 + y^2 + z^2 = 0$. Writing as usual $\mathbf{x} = (x, y, z)^\top$, this last condition is $\mathbf{x}^\top\mathbf{x} = 0$. The image point corresponding to such an object point is given

by $\mathbf{u}^j = KR_j\mathbf{x}$, from which we obtain $\mathbf{x} = R_j^{-1}K^{-1}\mathbf{u}^j$. Then from $\mathbf{x}^\top\mathbf{x} = 0$ follows $\mathbf{u}^{j\top}K^{-\top}R_jR_j^{-1}K^{-1}\mathbf{u}^j = \mathbf{u}^{j\top}(KK^\top)^{-1}\mathbf{u}^j = 0$. In other words, $\mathbf{u}^j$ is on the image of the absolute conic if and only if $\mathbf{u}^{j\top}(KK^\top)^{-1}\mathbf{u}^j = 0$. Thus, the image of the absolute conic is a plane conic represented by the matrix $(KK^\top)^{-1}$. In other words, $KK^\top$ is the dual of the image of the absolute conic. By finding the image of the absolute conic, one can retrieve K using the Choleski factorization, as already discussed.

The image of the absolute conic is unaffected by the location and orientation of the camera. Consequently, if P_j is a projective transformation from image J_0 to J_j taking a point in J_0 to its matching point in J_j, then in particular it must take a point on the image of the absolute conic in J_0 to a point on the image of the absolute conic in J_j. In short, P_j must preserve the image of the absolute conic. Since a 2D projective transform P acting on a conic C transforms it to the conic $P^{-\top}CP^{-1}$ it follows that ${P_j}^{-\top}CP_j^{-1} = C$ where C is the absolute conic $(KK^\top)^{-1}$. In other words,

$$ {P_j}^{-\top}(KK^\top)^{-1}P_j^{-1} = (KK^\top)^{-1} $$

from which it follows that

$$ P_j(KK^\top) = (KK^\top){P_j}^{-\top} \ , $$

which is the same equation as (6.17).

6:6.6 Iterative Estimation of the Calibration matrix

In section 6:6.3 a method was given for determining the transformations P_i. Similar least-square techniques are also available for an iterative determination of the calibration matrix K. In particular, we seek a set of points $\mathbf{x}_i$, a matrix K and a set of rotation matrices R_i such that

$$ \mathbf{u}_i^j = KR_j\mathbf{x}_i + \epsilon_i^j $$

for each pair (i, j) for which $\mathbf{u}_i^j$ is defined, and such that the squared error sum, $\sum \epsilon_i^{j2}$ is minimized. The difference between this and the iteration problem described in 6:6.3 is that matrix K is common to all the transforms KR_j, and that the matrix R_j must be constrained to be a rotation matrix. There is no particular technical problem with sharing the transform K between all the transforms. Adapting the sparse block techniques described in [68] to this added complication is straight-forward enough. Indeed it is built in to our camera-parameter estimation program.

The matrix K is parametrized by its five independent entries. This makes it easy to set any of the camera parameters to known values (for instance *skew* may be forced to zero, or the two magnifications k_u and k_v may be forced to be equal). This capability is built into Carmen.

Before carrying out this iterative estimate of K, it is necessary to provide an initial estimate. This initial estimate is provided by the methods of Sections 6:6.3 and 6:6.4. In particular, from Section 6:6.3 or Section 6:6.3 we obtain a set of transformations P_j and points $\mathbf{x}_i$ such that $P_0 = I$ and $P_j\mathbf{x}_i = \mathbf{u}_i^j$ whenever $\mathbf{u}_i^j$ is defined. From Section 6:6.4 we obtain rotation matrices R_j and a calibration matrix K such that $P_j = KR_jK^{-1}$ for all j. Now, writing $\mathbf{x}_i' = K^{-1}\mathbf{x}_i$, one verifies that

$$ KR_j\mathbf{x}_i' = P_jKK^{-1}\mathbf{x}_i = P_j\mathbf{x}_i = \mathbf{u}_i^j $$

as required, with $\mathbf{x}'_i$ being the initial point locations.

Using Carmen, therefore, an optimal estimate of the calibration matrix and the orientation of the parameters is possible. However, in the examples used for experimentation it turned out that this did not yield very great benefits. The solution for K given by the non-iterative method of section 6:6.4 was so good that the difference between the estimates found with and without this final estimation step did not differ very significantly.

6:6.7 Finding Matched Points

Finding matched points between images taken from the same point is easier than the general point-matching problem, because apart from the image transformation determined by the changing orientation of the camera, the images look essentially the same. There is no occlusion and no lighting changes. Points that are visible in one image are visible in the other (provided that they are inside the field of view). One method of finding matched points in sequences of video images would be to track them from from frame to frame. In the experiments carried out to test the calibration algorithm, individual images, rather than an image sequence were used, and a different approach to image matching was taken.

To find match points between images a correlation-based matching algorithm was used. The algorithm was based on parts of the STEREOSYS stereo algorithm ([19, 20]) adapted to the particular purposes of the current problem. The matching algorithm consisted of the following steps

1. Identify manually a small number (at least four) matching points between overlapping images. This identification need not be made very exactly.

2. Automatically find matches between pairs of overlapping images by resampling the second image of each pair to the same reference frame as the first, and then carrying out correlation based hierarchical matching.

3. Weed out outliers (false matches) among the matched points by a least median error algorithm.

Details of the second step are as follows. Given the small number of seed matches, a projective transformation mapping each selected point in the second image to its matching point in the first image is computed. The second image is then resampled according to this transformation. After resampling, the two images should correspond precisely, pixel for pixel. In reality, the accuracy of the initial matches is only approximate, so the match will not be exact. However, it will be sufficiently good for a correlation-based matching algorithm to work effectively. The accuracy of the point matching is ensured by doing the match in both directions. In a first step, a point $\mathbf{u}$ in the first image is matched with a point $\mathbf{u}'$ in the second image. In the second step, $\mathbf{u}'$ is matched with a point $\mathbf{u}''$ in the first image. Only if $\mathbf{u}$ and $\mathbf{u}''$ are close together (within one pixel) is the match $\mathbf{u} \leftrightarrow \mathbf{u}'$ accepted.

Using this method, about 100 or more matches between each pair of overlapping images were found without difficulty. Even with this two-way matching method, it is possible for there to be some erroneous matches, and it is important to detect and eliminate them. This was done using a least median error approach. If we assume a small percentage

of outliers (false matches), not exceeding 25%, then a set of four matches chosen at random will contain no outliers with about 32% probability. If sufficiently many sets of four matches are chosen, then one can be almost certain that one of the sets contains no outliers. For instance, if we test 100 sets of four matches chosen at random, then the probability of not selecting one set without an outlier is inconceivably small. The complete algorithm is as follows.

1. Select several sets of four matched points and carry out the following steps for each of these sets.
2. Given four matched points, compute the projective transformation P consistent with these four matches.
3. Compute the distance $\delta_i = d(P\mathbf{u}_i, \mathbf{u}'_i)$ for all matched point pairs $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$.
4. Sort the set of distances δ_i, and find the median distance (or alternatively the 75th, or any other percentile).
5. Find the set of four matched points that leads to the least median distance δ_i, and accept this as being close to the correct transform.
6. Discard all matched point pairs $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i$ for which the error δ_i exceeds some threshold (for instance, three times the median error).

This least median error approach is particularly suitable to apply to the present problem for two reasons. First, the small number (four) matches required to determine the transform P means that one can be very sure of selecting an outlier-free set with a small number of trials. Second, the computation of each trial is very fast, since a 2D projective transformation is very quick to compute. Because of this, the time to weed out the outliers is very small compared with the time to find the point matches by correlation-based search.

One slight refinement is used in the selection of sets of four matched points. The matched points are divided into four equal sized sets, denoted NW, NE, SW and SE (after the compass directions) corresponding to their position in the first image. Then, sets of four matched points are selected by taking one point at random from each of the four sets. This means that the four points will not be clustered together in one part of the image, and the projective transform that they determine will be more accurately defined for the whole image.

6:6.8 Experimental Verification of the Algorithm

Tests with Synthetic Data

First of all, the calibration algorithm was carried out on synthetic data to determine its performance in the presence of noise.

The synthetic data was created to simulate the images taken with a 35mm camera with a 50mm lens, and digitized with 20 pixels per mm. For such a camera, the image measures approximately 35mm by 23mm. When digitized with 20 pixels per mm, the image measures 700×460 pixels. The field of view is approximately $38^\circ \times 26^\circ$. This is approximately the resolution of the images used for the experiments with real images

Noise	k_u	k_v	p_u	p_v	skew
–	1000.0	1000.0	0.0	0.0	0.0
0.125	995.3	995.8	-0.5	1.3	0.1
0.25	990.6	991.5	-1.0	2.5	0.2
0.5	981.4	983.2	-2.0	4.9	0.3
1.0	963.4	967.0	-3.6	9.4	0.6
2.0	946.0	952.6	-7.3	20.9	1.2
4.0	898.2	910.0	-10.2	35.8	2.2
8.0	864.4	882.3	-10.8	40.1	5.4
16.0	715.9	744.6	54.5	20.4	-4.6

Table 6.3: *Calibration from three images in the presence of various degrees of noise, with one run at each noise level. The three views directions lie in a circle of radius* 10°. *The table shows the results of the Choleski (that is, non-iterative) algorithm. The first row shows the expected parameter values, whereas subsequent rows show the effects of different levels of noise (measured in pixels). Although the noise level differs for different runs, the displacements of each pixel due to noise are in the same direction for all noise levels.*

described later. For such images, the magnification factors, k_u and k_v in the two image-plane axial directions are equal to the focal length in pixels. In other words, $k_u = k_v = 1000$. The skew calibration parameter, s was taken to be zero, and image coordinates were taken to be centred at the principal point of the image, so that $p_u = p_v = 0.0$.

A set of N camera matrices were chosen with arbitrary orientations so that the principal ray of the camera lay within a prescribed angle θ of the positive z-axis. A set of 100 points were chosen, randomly placed on the unit sphere, subject to the restriction that each point is visible in at least two cameras. The image of each of the points was computed in each camera for which it lay inside the field of view. These image coordinate values were then used to compute the camera calibration using the algorithm of section 6:6.2. A Levenberg-Marquardt iteration algorithm was used to refine the estimate given by the non-iterative method. Ideally, the computed calibration parameters should be close to the true values given in the previous paragraph.

Two experiments are reported here. The first experiment was with $N = 3$ with principal rays lying within an angle $\theta = 10°$ of the positive z axis. The results are summarized in tables 6.3, 6.4 and 6.5. The second experiment was with $N = 10$ images with view directions lying within an angle $\theta = 30°$ of the positive z-axis. Results are summarised in tables 6.6, 6.7 and The results are very satisfactory. Experiments with real images to be described later indicate that images may be matched with an RMS error of about 0.5 pixels, which suggests that this is a realistic noise level. The results with synthetic data show that the algorithms are robust for noise levels well beyond this range. The noise levels indicated in the table are the standard deviation of the deltas applied to *each* of u and v. Hence the actual RMS pixel displacement is $\sqrt{2}$ times the indicated value.

Tests with Real Images

Calibration tests were carried out on two sets of real images. In the first set of images five images of the Capitol building in Washington (Fig 6.6) were taken with a 35mm camera

Noise	k_u	k_v	p_u	p_v	skew
–	1000.0	1000.0	0.0	0.0	0.0
0.125	999.2	999.5	-0.2	-0.3	0.0
0.25	998.4	999.0	-0.4	-0.5	0.1
0.5	996.8	998.0	-0.7	-0.9	0.1
1.0	993.5	996.0	-1.5	-1.8	0.2
2.0	956.1	960.7	-7.5	19.1	0.8
4.0	946.0	955.3	-12.4	26.4	1.5
8.0	938.7	956.6	-15.8	23.6	3.7
16.0	1077.9	1108.7	-0.2	-13.7	5.1

Table 6.4: *Calibration from three views. The table shows the results of Levenberg-Marquardt algorithm with one run at each of the noise levels. The results of the non-iterative calibration algorithm are used for initialization. Results show significant improvement over those of the non-iterative algorithm.*

Noise	Algorithm	statistic	k_u	k_v	p_u	p_v	skew
1.0	Choleski	Mean	997.6	997.8	0.9	-1.1	-0.1
		σ	24.5	24.3	7.5	8.7	1.0
	Marquardt	Mean	1016.2	1016.4	5.6	-13.0	-0.2
		σ	29.1	29.2	7.5	14.7	0.9
2.0	Choleski	Mean	1005.7	1006.3	-1.8	-0.5	-0.1
		σ	81.9	92.1	24.4	4.4	8.5
	Marquardt	Mean	979.4	976.1	18.5	-1.1	-4.2
		σ	44.0	45.2	15.2	2.8	7.5

Table 6.5: *Result of 100 runs with 3 views, with random noise of 1 and 2 pixels. The parameters k_u and k_v were highly correlated, whereas other parameters showed little correlation. The Levenberg-Marquardt algorithm does not show a clear advantage over the non-iterative algorithm.*

Noise	k_u	k_v	p_u	p_v	skew
–	1000.0	1000.0	0.0	-0.0	0.0
0.125	996.1	997.0	-1.7	2.0	-2.2
0.25	992.9	994.3	-3.4	4.4	-4.3
0.5	986.0	988.9	-6.8	8.5	-8.6
1.0	970.7	976.6	-14.2	16.2	-17.4
2.0	945.8	958.0	-29.0	28.7	-33.3
4.0	1224.9	1163.7	-30.8	-310.8	-44.6
8.0	739.1	815.4	-95.0	15.4	-83.0

Table 6.6: *Calibration from ten images in the presence of various degrees of noise. The three views directions lie in a circle of radius 30°. The table shows the results of the non-iterative algorithm. The first row shows the expected parameter values. For noise level of 16 pixels, the calibration failed due to failure of the Choleski factorization, the matrix $KK^\top$ not being positive-definite.*

Noise	k_u	k_v	p_u	p_v	skew
–	1000.0	1000.0	0.0	-0.0	0.0
0.125	1000.8	1000.6	0.1	-0.2	-0.2
0.25	1002.3	1001.8	-0.0	-0.6	-0.3
0.5	1004.5	1003.7	-0.1	-1.2	-0.6
1.0	1008.8	1007.0	-0.2	-2.7	-1.2
2.0	972.2	968.1	-10.8	17.6	-0.8
4.0	1489.0	1467.2	-27.7	-240.2	-16.3
8.0	984.5	971.9	-14.0	5.0	-3.2

Table 6.7: *Calibration from ten views in the presence of various degrees of noise. Results of iterative Levenberg-Marquardt algorithm. The results of the non-iterative calibration algorithm are used for initialization. Results are satisfactory, except for noise-level 4 pixels, where a local minimum has been found.*

Method	k_u	k_v	p_u	p_v	skew	residual
Choleski	964.4	966.4	392.8	282.0	-4.9	unknown
Marquardt	956.8	959.3	392.0	281.4	-6.4	0.33

Table 6.8: *Calibration results for five images of the Capitol with a 35mm camera. The results from the two methods of calibration are very similar. The calibration seems very plausible, since the measured skew is small, magnification is almost the same in both directions and the principal point is near the centre of the image. The last column gives the difference in pixels between predicted image coordinates (given the calibration and reconstruction) and the measured values. A value of k_u or k_v of 960 corresponds to a focal length of approximately $35 \times 960/776 = 43.3mm$.*

Figure 6.6: *Five images of the capitol, numbered 1 – 5 left-to-right and top-to-bottom.*

with a zoom lens. The focal length of the lens was approximately 40mm (though not known exactly, since it was a zoom lens). The camera was hand-held, and no particular care was taken to ensure that the camera centre remained stationary. The images were printed, enlarged and digitized. The images were then scanned at 150 pixels per inch, resulting in images of size 776×536 pixels. Corresponding points were found between the images according to the algorithm of section 6:6.7, and the calibration was carried out. A composite of the five images is shown in Fig 6.7. The calibration results are summarized in Table 6.8.

A second set of 29 images were taken covering a region of about 48×22 degrees with a 105mm lens. The images were of size 470×320 pixels. The lens has a fairly small field of view, which increases the difficulty of calibration using the methods of this section. The results of this experiment were as shown in Table 6.9. Two of the images used are shown in Fig 6.8. The Levenberg-Marquardt iteration was carried out using an extra parameter to estimate the radial distortion in the image proportional to the square of the radius. However, the effect was found to be minimal.

Figure 6.7: *A composite image constructed from five different views of the Capitol. The composite image shows very clearly the projective distortion necessary for matching the images. Analysis of this projective distortion provides the basis for the calibration algorithm.*

Method	k_u	k_v	p_u	p_v	skew	residual
Choleski	1226.1	1226.5	238.1	170.4	-5.1	unknown
Marquardt	1242.1	1242.7	245.5	169.4	-6.6	0.26

Table 6.9: *Results of camera calibration of a set of image of a parking lot. The results suggest that the focal length of the camera shows some instability, but that the ratio of the magnifications k_u and k_v is very stable.*

Figure 6.8: *Two images of a parking lot*

6:6.9 Calibration from only two views

The constraint that the transformation matrix P must be the conjugate of a rotation is not sufficient to determine the conjugating element K exactly. Nevertheless, with just one additional constraint on the calibration matrix it is possible to determine K uniquely. For instance, it will be shown that under the assumption that the skew parameter $s = 0$, calibration matrix K is uniquely determined, and it is possible to calibrate from only two views. Since s is usually very small, the assumption that $s = 0$ is a very reasonable one, commonly used by other authors ([3]). Alternatively, one may make other assumptions about the calibration, for instance that the camera has square pixels, $k_u = k_v$.

According to Proposition 6.2, given two views the transformation taking one image to the other is of the form $P = KRK^{-1}$ where K is the calibration matrix and R is a rotation representing the relative orientation of the two cameras. Matrix P may be normalized so that its determinant $\det P = 1$. Given such a P, it will next be shown how to find an upper-triangular matrix K such that $P = KRK^{-1}$. It will turn out that there exist many such K (in fact a one-parameter family), but for now, we will concentrate on how to find just one of them. Later it will be shown how to find such a K with given desired properties (such as zero skew).

The fact that P is a conjugate of a rotation matrix has the immediate consequence that P and R have the same eigenvalues. The eigenvalues of a rotation matrix are equal to 1, $\exp(i\theta)$ and $\exp(-i\theta)$, where θ is the angle of rotation. Therefore, by finding the eigenvalues of P, we are able to find the angle of rotation of R. Furthermore, it is possible to find a matrix K' such that $P = K'\mathrm{diag}(1, \exp(i\theta), \exp(-i\theta))K'^{-1}$. The columns of K' are the eigenvectors of P. Since the eigenvectors are defined only up to multiplication by a non-zero factor, so are the columns of K'. Multiplying the columns of K' by independent factors preserves the condition that $P = K'\mathrm{diag}(1, exp(i\theta), exp(-i\theta))K'^{-1}$. One could continue this line or reasoning to determine the required calibration matrix, but this involves computations using complex numbers. Instead, we proceed slightly differently.

Any rotation is conjugate to a rotation about the x axis. Since P is conjugate to a rotation, it is therefore conjugate to a rotation about the x axis. From the eigenvalues of P one may determine the angle of rotation, θ. Then one may write $P = HR_xH^{-1}$, and hence $PH = HR_x$. We write

$$R_x = \begin{bmatrix} 1 & & \\ & c & -s \\ & s & c \end{bmatrix}$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. Further, write $H = (\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3)$ where $\mathbf{h}_i$ is the i-th column of H. Then from $PH = HR_x$ we obtain equations

$$\begin{aligned} P\mathbf{h}_1 &= \mathbf{h}_1 \\ P\mathbf{h}_2 &= c\mathbf{h}_2 + s\mathbf{h}_3 \\ P\mathbf{h}_3 &= -s\mathbf{h}_2 + c\mathbf{h}_3 \end{aligned}$$

This gives rise to a pair of equations

$$(P - I)\mathbf{h}_1 = 0 \tag{6.18}$$

and

$$\begin{bmatrix} P - cI & -sI \\ sI & P - cI \end{bmatrix} \begin{pmatrix} \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \ . \tag{6.19}$$

Because of the choice of c and s, the matrices in (6.18) and (6.19) will be singular. Consequently, we can solve (6.18) to find $\mathbf{h}_1$ and (6.19) to find $\mathbf{h}_2$ and $\mathbf{h}_3$. In the presence of noise, P will not be exactly equal to a conjugate of a rotation. In this case, the equations (6.18) and (6.19) will not have an exact solution. The least-squares solution is to be used. From the $\mathbf{h}_i$ we may reassemble a matrix H. This matrix will satisfy $P = HR_xH^{-1}$. Now, using QR decomposition, we may obtain $H = KR$, where K is upper-triangular and R is a rotation. It follows that $P = KRR_xR^{-1}K^{-1} = K\hat{R}K^{-1}$ as required.

It was shown above how to find a matrix H such that $HR_xH^{-1} = P$. Such an H is not unique, and so we now inquire how other solutions may be found. Suppose that $HR_xH^{-1} = P = H'R_xH'^{-1}$. It follows that $(H^{-1}H')R_x = R_x(H^{-1}H')$, in other words, $H^{-1}H'$ commutes with R_x. It may be shown by direct symbolic manipulation that if R_x is not a rotation through 0 or π radians, then $H^{-1}H' = \mathrm{diag}(\alpha_1, \alpha_2, \alpha_2)R'_x$ where R'_x is some other rotation about the x axis. Hence, $H' = H\mathrm{diag}(\alpha_1, \alpha_2, \alpha_2)R'_x$. Since we are only concerned with finding H up to a non-zero scale factor, we may assume that $H' = H\mathrm{diag}(\alpha, 1, 1)R'_x$. Now, if $H = KR$, and $R\mathrm{diag}(\alpha, 1, 1)$ has QR decomposition $K''R''$, then

$$H' = H\mathrm{diag}(\alpha, 1, 1)R'_x = KR\mathrm{diag}(\alpha, 1, 1)R'_x = KK''R''R'_x \ .$$

The foregoing discussion may be summarized in the following proposition.

Proposition 6.4. *Let P be a 2D projective transformation matching two images taken from the same location with the same camera. Let $P = HR_xH^{-1}$ where R_x is a rotation about the x axis. Further, let $H = KR$ be the QR decomposition of H. Then K is a calibration matrix for the camera, consistent with the transformation P. Any other calibration matrix K' consistent with P is of the form $K' = KK''$ where $K''R''$ is the QR decomposition of $R\mathrm{diag}(\alpha, 1, 1)$ for some α.*

This shows that the set of calibration matrices K corresponding to a given transformation matrix P is a one-parameter family. To find a unique calibration matrix, one extra constraint is necessary.

We next turn to the problem of finding a calibration matrix K satisfying additional constraints. To do this, we investigate the QR decomposition of a matrix $R\mathrm{diag}(\alpha, 1, 1)$. Let (r_{ij}) be the entries of the matrix R. The QR decomposition may be computed explicitly. Indeed, it may be verified after some computation that $R\mathrm{diag}(\alpha, 1, 1) = K''R''$ with K'' defined by

$$K'' = \frac{1}{\sqrt{AB}} \begin{bmatrix} \alpha\sqrt{A} & (\alpha^2 - 1)r_{11}r_{21} & (\alpha^2 - 1)r_{11}r_{31}\sqrt{B} \\ 0 & B & (\alpha^2 - 1)r_{21}r_{31}\sqrt{B} \\ 0 & 0 & A\sqrt{B} \end{bmatrix} \tag{6.20}$$

where $A = (1 - r_{31}^2) + \alpha^2 r_{31}^2$ and $B = r_{11}^2 + \alpha^2(1 - r_{11}^2)$.

There seems to be no pretty way of demonstrating the truth of this formula, and so it must be done by algebraic manipulation. The best way is probably to verify that $K''K''^{\top} = I + (\alpha^2 - 1)\mathbf{r}_1\mathbf{r}_1{}^{\top} = R\mathrm{diag}(\alpha, 1, 1)(R\mathrm{diag}(\alpha, 1, 1))^{\top}$ where $\mathbf{r}_1$ is the first column of R. From this it follows that $R\mathrm{diag}(\alpha, 1, 1) = K''R''$ for some rotation R'' as required. This formula leads us to the following extension to Proposition 6.4.

Proposition 6.5. *Let $P = HR_xH^{-1}$ and $H = KR$. Any calibration matrix consistent with P may be written as KK'' where K'' is of the form given in (6.20) for some $\alpha > 0$.*

The condition that $\alpha > 0$ is required to ensure that the magnification factor k_u of KK'' remains positive. Now, it is an easy matter to choose α so that the calibration matrix KK'' has desired properties.

Zero skew. We consider the condition that the skew parameter is zero. Suppose $K = (k_{ij})$ and $R = (r_{ij})$. The $(1,2)$-entry (that is, the skew) in the product KK'' is zero exactly when $k_{11}(\alpha^2 - 1)r_{11}r_{21} + k_{12}B = 0$. Solving for α gives

$$\alpha^2 = \frac{k_{11}r_{11}r_{21} - k_{12}r_{11}^2}{k_{11}r_{11}r_{21} - k_{12}(r_{11}^2 - 1)} \quad ; \quad \alpha > 0 \tag{6.21}$$

This gives a simple algorithm for the calibration of a camera from two views, assuming that the skew is zero.

1. Compute the transformation matrix P that matches points in the two images, such that $\det P = 1$.
2. Compute the rotation angle θ which is the argument of one of the complex eigenvalues of the matrix P.
3. Find a matrix H such that $P = HR_xH^{-1}$ where R_x is a rotation through angle θ about the x-axis. This is done by solving the equations (6.18) and (6.19).
4. Take the QR-decomposition $H = KR$.
5. Find $\alpha > 0$ by solving (6.21).
6. Compute the QR decomposition $H\mathrm{diag}(\alpha, 1, 1) = K'R'$. The matrix K' is the calibration matrix.

Square pixels. An alternative to setting the skew to zero is to set the two magnifications k_u and k_v in the two axial directions to be equal. Multiplying out KK'' and equating the first two diagonal entries leads to an equation $k_{11}\alpha\sqrt{A} = k_{22}B$. Squaring both sides of this equation leads to a quadratic equation in α^2. In particular, we obtain

$$\alpha^4(k_{11}^2 r_{31}^2) + \alpha^2\left(k_{11}^2(1 - r_{31}^2) - k_{22}^2(1 - r_{11}^2)\right) - k_{22}^2 r_{11}^2 = 0$$

This equation is easily solved for α, but in this case there may be two solutions, since a quadratic equation is involved. We have chosen the strategy of selecting the solution that has the smaller skew. The algorithm for finding the calibration matrix is otherwise the same as the previous one.

6:6.10 Exceptional Cases

It was seen that the calibration algorithm fails if P represents a rotation through 0 or π radians. The first case means that the two images are identical, and the second means that two images are taken with the camera pointing in opposite directions. These special cases are of no interest. There are, however other exceptional cases.

Rotation about the x-axis. If the rotation is about the x axis, then the transformation matrix P is of the form $P = KR_xK^{-1}$ where R_x is a matrix of the form previously given. Any other conjugating element K' satisfying this relationship is of the form $K' = K\text{diag}(\alpha, 1, 1)$ for any α. However, the matrix K' so obtained is the same as K, except that the $(1,1)$ entry, representing the parameter k_u is multiplied by α. The skew is unchanged. It follows therefore, that constraining the skew to be zero may be an impossible constraint, and in any case puts no restriction on k_u. In other words, we can not determine k_u if the rotation is about the x axis.

Rotation about the y-axis. Similar considerations apply to rotations about the y-axis. In this case, if $P = KR_yK^{-1}$, then any other conjugating element K' must be of the form $K' = K\text{diag}(1, \alpha, 1)$. In this case, the the value of k_v can not be determined.

Rotation about the z-axis. Unless the camera is calibrated, we do not know precisely where the principal axis (that is the z-axis) is. However, if the rotation does happen to be about the z axis, so that K satisfies the condition $P = KR_zK^{-1}$, then any other matrix of the form $K' = K\text{diag}(\alpha, \alpha, 1)$ will do so as well. This means that the two magnification factors, k_u and k_v as well as the skew are multiplied by the factor α. Consequently, it is not possible to determine any of these parameters. Only the position of the principal point and the ratio k_u/k_v may be computed.

6:6.11 Experiments with Calibration from Two Images

Tests with Synthetic Data

First of all, the calibration algorithm was applied to synthetic data to determine its performance in the presence of noise.

The synthetic data was created to simulate the images taken with a 35mm camera with a 50mm lens, and digitized with 20 pixels per mm. For such a camera, the image measures approximately 35mm by 23mm. When digitized with 20 pixels per mm, the image measures 700×460 pixels. The field of view is approximately $38^\circ \times 26^\circ$. This is approximately the resolution of the images used for the experiments with real images described later. For such images, the magnification factors, k_u and k_v in the two image-plane axial directions are equal to the focal length in pixels. In other words, $k_u = k_v = 1000$. The skew calibration parameter, s was taken to be zero, and the principal point was taken to have coordinates $(p_u, p_v) = (20, 30)$.

The square-pixel constraint: A first set of experiments were conducted with two images overlapping by 50% side-by-side. Thus, the rotation was through an angle of 19.29° (that is, half the image width) about the y axis. A set of 100 matched points were generated, and varying degrees of noise were added. Noise was zero mean Gaussian noise, with the indicated standard deviation. The quoted noise levels are for the deviation applied to *each* of the u and v image coordinates, hence the root-mean-squared pixel displacement is $\sqrt{2}$ times as great. The calibration algorithm was run with the constraint that magnification factors were equal : $k_u = k_v$. First the non-iterative calibration algorithm was run. It was found that for large amounts of noise the skew parameter s became substantially different from zero. Therefore, starting from the calibration already

Non-iterative algorithm						Levenberg-Marquardt		
Noise	k_u	skew	p_u	p_v	angle	k_u	p_u	p_v
0.0	1000.0	0.0	20.0	30.0	19.29	1000.0	20.0	30.0
0.1	1002.3	0.7	19.8	31.0	19.25	1002.3	19.9	31.0
0.25	1005.7	1.9	19.6	32.6	19.18	1005.7	19.7	32.5
0.5	1011.7	3.8	19.2	35.2	19.07	1011.4	19.4	35.1
1.0	1023.5	9.6	18.2	40.7	18.86	1022.5	18.7	40.4
2.0	1050.7	21.2	16.3	52.4	18.38	1046.6	17.4	51.9
3.0	1082.3	35.5	14.4	65.5	17.85	1072.5	15.9	64.5
4.0	1119.0	53.4	12.4	80.2	17.27	1100.3	14.3	78.5
5.0	1162.2	76.0	10.4	97.0	16.62	1130.4	12.5	94.1

Table 6.10: *Calibration from two images with 50% overlap assuming the condition $k_u = k_v$. For the Levenberg-Marquardt iteration, the condition that skew $s = 0$ was also assumed. The 6-th column shows the computed rotation angle between the two views. The rotation was 19.29 degrees about the x axis.*

obtained, an iterative Levenberg-Marquardt optimization was run, clamping the skew to zero and maintaining the condition $k_u = k_v$. The results of these experiments are found in table 6.10. As may be seen, the calibration becomes progressively less exact as noise increases, but for noise levels of the order of 0.5 pixels, which may be obtained in practice, the magnification is accurate to about 1% and the principal point is displaced by about 5 pixels. The results obtained by the Levenberg Marquardt algorithm are not significantly better, except for the zero skew. Note that setting skew to zero does not affect the other parameters very much, which suggests that skew is somewhat hard to estimate exactly.

The zero-skew constraint: A second set of experiments were conducted with the second image panned sideways through 10° and then rotated 90° about the principal axis. In this case, calibration was carried out assuming zero skew. Because of the 90° rotation about the principal axis, the ratio of k_u/k_v was computed very exactly, and a complete Levenberg-Marquardt optimization makes little difference to the final result. These results are shown in table 6.11.

Using knowledge of the rotation: During the Levenberg-Marquardt parameter fitting it is easy to add a constraint fixing the camera rotation to the known value. This was done for comparison using the same data as in table 6.10 for noise level of 2.0 pixels. The results of the calibration were then :

$$k_u = k_v = 1000.35 \quad ; \quad p_u = 15.9 \quad ; \quad p_v = 47.7$$

This is (as expected) considerably more accurate that the results for with unknown camera motion. The magnification factors are determined almost exactly, though there is still some error in the estimated position of the principal point (about 20 pixels).

Experiments with real data: Finally, calibration was carried out on the Capitol data set (Fig 6.6). The calibration computed using all five views is provided as a good

Non-iterative algorithm						Levenberg-Marquardt		
Noise	k_u	k_v	p_u	p_v	angle	k_u	p_u	p_v
0.0	1000.0	1000.0	20.0	30.0	90.63	1000.0	20.0	30.0
0.1	1002.6	1002.5	19.7	30.5	90.60	1002.3	19.7	30.4
0.25	1006.6	1006.4	19.2	31.2	90.57	1005.9	19.2	30.9
0.5	1013.6	1013.0	18.4	32.3	90.51	1012.3	18.3	31.7
1.0	1028.2	1027.1	16.6	34.7	90.39	1027.0	16.0	33.0
2.0	1088.8	1086.5	0.1	34.2	90.18	1080.4	4.4	28.5
3.0	1160.8	1157.4	-17.4	36.4	89.96	1150.3	-10.6	25.0
4.0	1260.1	1255.8	-43.0	38.4	89.72	1253.3	-34.5	20.2
5.0	1409.2	1404.6	-85.4	40.4	89.48	1457.3	-85.3	15.4

Table 6.11: *Calibration from two images assuming no skew. For the Levenberg-Marquardt iteration, the condition that* $k_u = k_v$ *was also assumed. The rotation angle is* 10° *about the* x*-axis and* 90° *about the* z *axis, for a combined rotation of* 90.63°.

Image numbers	constraint	k_u	k_v	skew	p_u	p_v	angle
1,2,3,4,5	–	964.4	966.4	-4.9	392.8	282.0	–
2,3	k	1002.9	1002.9	-25.0	330.1	214.8	25.49
2,5	k	963.6	963.6	-11.3	396.5	286.6	31.43
3,5	k	882.2	882.2	38.0	386.1	277.2	23.40
4,5	k	943.7	943.7	-4.7	389.3	250.8	9.57
1,5	k	1197.3	1197.3	-43.7	531.4	416.7	54.15
1,5	s	812.7	819.4	-0.0	381.3	224.3	54.15

Table 6.12: **Calibration from real images.** *The second column shows the type of constraint used (k = square-pixels, s = zero-skew). The first line gives the result of a calibration using all five images, provided as (approximate) ground truth. The next four lines show results of calibration for pairs of images for which the main component of rotation is a panning rotation. For such a rotation, the constraint skew = 0 will not give good results. The sixth and seventh lines show the result for a pair of images that differ by a rotation with its major component about the principal axis. As demonstrated theoretically, rotations about the principal axis do not lead to good calibration results. Accordingly, the results in the last two lines are substantially inferior.*

approximation to truth, since it is derived from more images and is expected to be accurate. Pairs of images were then taken and calibration carried out. Between 100 and 200 matched points were found between image pairs. The results are given in table 6.12.

In general, magnification is accurate within 10%, usually much less, and the principal point is accurate within 30 pixels. These results verify the conclusion suggested by the results with synthetic data that best results are obtained using panning rotations and the square-pixel constraint.

6:6.12 Handling translations

It is a basic assumption of the method described in this section that the camera centre remains fixed for all the images. In practice, the camera centre is not easily determined. Furthermore, for cameras mounted on a robot, the task of rotating about the camera centre, even if it is accurately known, may require careful calibration of the robot. For this reason, we are led to consider what strategy to adopt to account for small translatory motions of the camera from view to view. The methods described here are given as suggestions only, and no results are given to validate their performance.

In the images used for the experiments reported in this section, no particular care was taken to fix the camera centre. For instance the parking lot images were taken with a hand-held camera, only a token effort being made to keep the camera approximately fixed. In this case, the possible displacements of the camera are very small compared with the distance to the scene, and the displacements of the image points caused by translatory motion of the camera will be small, and may be safely ignored. In fact if the points in the scene are at infinite distance, then translations of the camera centre have no effect whatever. In a general case, therefore, we wish to determine the image of the plane at infinity. The projective transformations of the image of the plane at infinity, caused by the camera rotation, may be used to determine the camera calibration.

Finding the plane at infinity. A strategy will be suggested now for handling outdoor scenes in which most of the scene is distant from the camera. We suppose, however that there are near-by objects that may be displaced appreciably by the translatory motion of the camera. Our task is to ignore these points and determine the transformation of the points at infinity only (or distant points). A way to do this is provided in [6], where a method is described for determining a 2D transformation between two images of a plane. In this method, determination of the 2D transformation is cast as a parameter optimization problem in which the variable parameters are the 8 parameters of a plane-to-plane transformation and the quantity to be minimized is the difference in image intensity at corresponding points, summed over the image. One proceeds from coarse to fine resolution using a multi-resolution pyramid, the 2D transformation found at one level being used as the initial estimate at the next level. As observed in [6], the 2D projective transformation ultimately determined by this method will "lock" on to the dominant plane in the images. If this plane is the plane at infinity, then we obtain the desired transformation. One minor difference between the method of [6] and what is proposed here is that the transformation model they use is a quadratic model, and not the projective motion model assumed here, but with this minor modification, the method should apply unchanged.

Even in cases where the plane at infinity is not the dominant plane in the image, this method may be valuable if the plane at infinity can be determined, at least approximately, by other means. For instance, points on the plane at infinity may be determined by vanishing points in the image, or by known ratios of distances along a line. If four corresponding points on the plane at infinity are determined, then these may be used to initialize the transformation between the images. This should cause the transformation found by iteration to lock onto the plane at infinity. Relying on such extraneous information, however, restricts the generality of the method.

Note that this method of determining the transformations P_j does not require the explicit identification of matched points in the two images. Even if there is no effect due to translational motion of the camera, this method provides an attractive alternative to the methods described in this report (Sections 6:6.3 and 6:6.7) for determining the transformations.

Determining the translations. A more generally applicable method is to allow for small translations of the camera centre and solve for the rotation, and the translations all together. According to [46], it is possible to find the camera calibration explicitly from three views or more taken from a camera undergoing arbitrary motions. In [29] an iterative algorithm was given to find this calibration, provided that a sufficiently good initial estimate was known. In the algorithm described in the present section, a final iterative refinement of the camera calibration is suggested (Section 6:6.6) as a method of improving the calibration results. It would be an easy matter to modify this final iterative step to allow for a full 3×4 matrix camera model, which includes camera translation. For details of the iterative solution method, refer to section 6:6.6, or [29]. As in section 6:6.6, initialization is important. In section 6:6.6, the projection matrices are initialized to KR_j, and the points to values $\mathbf{x}' = (x', y', z')^\top$. Instead of this, we initialize the cameras to $[KR_j \mid \mathbf{0}]$, and the points to values $(\alpha x', \alpha y', \alpha z', 1)^\top$, where α is chosen such that $\alpha^2(x'^2 + y'^2 + z'^2) = 1$. Thus, the point lies on a unit sphere centred at the origin. In addition, the sign of α should be chosen such that the point lies in front of the cameras in which it is visible. This will be possible for all the cameras simultaneously, unless some

gross error of calibration has occurred.

6:6.13 Conclusions

The self-calibration algorithm given here represents a practical approach to camera calibration, giving good accuracy, and showing graceful degradation in the presence of noise. The non-iterative algorithm based on Choleski factorization does not show markedly inferior results to the optimal Levenberg-Marquardt method, and should be preferred except where highest possible accuracy is needed.

The use of the iterative Levenberg-Marquardt method allows the calibration problem to be cast as a general parameter fitting problem and allows the imposition of additional constraints, such as the known aspect ratio k_u/k_v, zero skew, or even known rotation angles for the various images. In addition, it allows the possibility of small translations of the camera to be taken into account.

The two-view algorithm given in this section derives the camera calibration from the smallest possible number of views, without using calibration rigs with known geometry. Naturally, the results are inferior to those obtained with a greater number of views, but they suggest that for suitable rotations, particularly panning rotations, the results are quite good. Further work is required to determine the optimal rotation that should be applied to give best calibration.

The mathematical derivations in this section make clearer the theory behind self-calibration schemes such as those of [3, 11]). As was demonstrated in Section 6:6.11, knowledge about the actual motion of the camera (which was assumed in [3, 11]) may be incorporated into our algorithm to give high quality results.

Clearly, the algorithms in this section can not hope to give such accurate results as will be obtained by calibration methods involving calibration grids, or other metric data. Nevertheless, for many purposes they will be adequate. As a means of calibrating cameras in the field, the methods of this section seem much more practical than methods based on a moving camera ([46]), both because of the ease of point matching and the simplicity of the calibration algorithms (for instance compare with [43, 29]).

The greatest use of such algorithms is expected to be in the calibration of robot-mounted cameras. The calibration obtained by this method could be used to do euclidean scene reconstruction, for purposes of navigation, or grasping. This would be particularly useful for cameras for which the calibration is subject to change, such as a camera with a zoom lens. Beardsley et. al. ([4, 5]) discuss navigation in a "quasi-euclidean" frame obtained by making a rough guess at the camera calibration. They also use purely translational motions of the camera to obtain an affine estimate of the coordinate frame. The algorithm of this section provides an alternative method, which furnishes euclidean, rather than just affine information, and which eliminates the need to guess at the camera calibration.

Also in [4, 5] a method is described for projective scene reconstruction from image sequences, using a Kalman filter. That method could be adapted to carry out euclidean scene reconstruction. The methods of the present section would provide a good initial estimate for calibration which could be refined by the Kalman filter. In this way, Euclidean reconstruction would be possible from a sequence of images from a camera undergoing unrestrained motion, provided the sequence begins with a series of purely rotational camera motions.

Finally, it is important to realize the restrictions on the self-calibration algorithms of this section. The algorithm relies ultimately on detecting the curvature of the vision sphere. As such, it works best for wide angle images. For the parking-lot images the field of view was only 18.92° in the maximum dimension (a 105mm lens in a 35mm camera). Calibration was possible, but a mosaic of 29 images was used. For a 40mm lens, only 5 images were needed.

Bibliography

[1] K.E. Atkinson. *An Introduction to Numerical Analysis, 2nd Edition.* John Wiley and Sons, New York, 1989.

[2] Eamon. B. Barrett, Michael H. Brill, Nils N. Haag, and Paul M. Payton. Invariant linear methods in photogrammetry and model matching. In J.L. Mundy and A. Zisserman, editors, *Geometric Invariance in Computer Vision*, pages 277 – 292. MIT Press, Boston, MA, 1992.

[3] Anup Basu. Active calibration: Alternative strategy and analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 495–500, 1993.

[4] P. A. Beardsley, A. Zisserman, and D. W. Murray. Navigation using affine structure from motion. In *Computer Vision - ECCV '94, Volume II, LNCS-Series Vol. 801, Springer-Verlag*, pages 85–96, 1994.

[5] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential update of projective and affine structure from motion. Report OUEL 2012/94, Oxford University, 1994. To appear in IJCV.

[6] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh. Hingorani. Hierarchical model-based motion estimation. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 237–252, 1992.

[7] B. Boufama, R. Mohr, and F. Veillon. Euclidean constraints for uncalibrated reconstruction. *Technical Report, LIFIA - IRIMAG*, 1993.

[8] Stefan Carlsson. The double algebra: An effective tool for computing invariants in computer vision. In *Applications of Invariance in Computer Vision : Proc. of the Second Joint European - US Workshop, Ponta Delgada, Azores – LNCS-Series Vol. 825, Springer Verlag*, pages 145 – 164, October 1993.

[9] R. Deriche, Z. Zhang, Q.-T. Luong, and O. Faugeras. Robust recovery of the epipolar geometry for an uncalibrated stereo rig. In *Computer Vision - ECCV '94, Volume I, LNCS-Series Vol. 800, Springer-Verlag*, pages 567–576, 1994.

[10] Lisa Dron. Dynamic camera self-calibration from controlled motion sequences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 501–506, 1993.

[11] Fenglei Du and Michael Brady. Self-calibration of the intrinsic parameters of cameras for active vision systems. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 477–482, 1993.

[12] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 563 – 578, 1992.

[13] O. D. Faugeras, Q.-T Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 321 – 334, 1992.

[14] Olivier Faugeras and Bernard Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *Proc. International Conference on Computer Vision*, pages 951 – 956, 1995.

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Second edition.* The Johns Hopkins University Press, Baltimore, London, 1989.

[16] P. Gros and L. Quan. Projective Invariants for Vision. Technical Report RT 90 IMAG - 15 LIFIA, IRIMAG–LIFIA, Grenoble, France, December 1992.

[17] P. Gros and L. Quan. 3D Projective Invariants from Two Images. In *Geometric Methods in Computer Vision II, SPIE 1993 International Symposium on Optical Instrumentation and Applied Science*, pages 75–86, July 1993.

[18] Patrick Gros. How to use the cross ratio to compute projective invariants from two images. In *Applications of Invariance in Computer Vision : Proc. of the Second Joint European - US Workshop, Ponta Delgada, Azores – LNCS-Series Vol. 825, Springer Verlag*, pages 107–126, October 1993.

[19] M. J. Hannah. Bootstrap stereo. In *Proc. Image Understanding Workshop, College Park, MD*, pages 210–208, April 1980.

[20] M. J. Hannah. A description of SRI's baseline stereo system. Technical Report Tech. Note 365, SRI International Artificial Intelligence Center, Oct. 1985.

[21] R. Hartley. Invariants of points seen in multiple images. unpublished report, May 1992.

[22] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 761–764, 1992.

[23] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 579 – 587, 1992.

[24] R. I. Hartley. Camera calibration using line correspondences. In *Proc. DARPA Image Understanding Workshop*, pages 361–366, 1993.

[25] R. I. Hartley. Invariants of lines in space. In *Proc. DARPA Image Understanding Workshop*, pages 737–744, 1993.

[26] R. I. Hartley. A linear method for reconstruction from lies and points. In *Proc. International Conference on Computer Vision*, pages 882 – 887, 1995.

[27] R. I. Hartley and A. Kawauchi. Polynomials of amphicheiral knots. *Math. Ann*, 243:63 – 70, 1979.

[28] Richard Hartley and Rajiv Gupta. Computing matched-epipolar projections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 549 – 555, 1993.

[29] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision : Proc. of the Second Joint European - US Workshop, Ponta Delgada, Azores – LNCS-Series Vol. 825, Springer Verlag*, pages 237–256, October 1993.

[30] Richard I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16:1036–1041, October 1994.

[31] Richard I. Hartley. Projective reconstruction from line correspondences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 903–907, 1994.

[32] Richard I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, March 1997.

[33] Richard I. Hartley and Peter Sturm. Triangulation. In *Proc. ARPA Image Understanding Workshop*, pages 957–966, 1994.

[34] Anders Heyden. *Geometry and Algebra of Multiple Projective Transformations*. PhD thesis, Department of Mathematics, Lund University, Sweden, December 1995.

[35] Anders Heyden. Reconstruction from multiple images using kinetic depths. Technical Report ISRN LUFTD2/TFMA-95/7003-SE, Department of Mathematics, Lund University, 1995.

[36] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4:59 – 78, 1990.

[37] B. K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America, A*, Vol. 8, No. 10:1630 – 1638, 1991.

[38] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America, A*, Vol. 4:629 – 642, 1987.

[39] T. S. Huang and O. D. Faugeras. Some properties of the e-matrix in two-view motion estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11:1310 – 1312, 1989.

[40] Jan J. Koenderink and Andrea J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America, A*, 1992.

[41] E. Kruppa. Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung. *Sitz.-Ber. Akad. Wiss., Wien, math. naturw. Abt. IIa.*, 122:1939–1948, 1913.

[42] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, Sept 1981.

[43] Q.-T Luong. *Matrice Fondamentale et Calibration Visuelle sur l'Environnement*. PhD thesis, Universite de Paris-Sud, Centre D'Orsay, 1992.

[44] Quang-Tuan Luong, Rachid Deriche, Olivier D. Faugeras, and Theodore Papadopoulo. On determining the fundamental matrix: analysis of different methods and experimental results. Report RR-1894, INRIA, 1993.

[45] S. J. Maybank. The projective geometry of ambiguous surfaces. *Phil. Trans. R. Soc. Lond.*, A 332:1 – 47, 1990.

[46] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8:2:123 – 151, 1992.

[47] R. Mohr, L. Quan, F. Veillon, and B. Boufama. Relative 3D reconstruction using multiples uncalibrated images. Technical Report RT 84-I-IMAG LIFIA 12, IRIMAG–LIFIA, 1992.

[48] R. Mohr, F. Veillon, and L. Quan. Relative 3D reconstruction using multiple uncalibrated images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 543 – 548, 1993.

[49] Theo Moons, Luc Van Gool, Marc van Diest, and Eric Pauwels. Affine reconstruction from perspective image pairs obtained by a translating camera. In *Applications of Invariance in Computer Vision : Proc. of the Second Joint European - US Workshop, Ponta Delgada, Azores – LNCS-Series Vol. 825, Springer Verlag*, pages 297 – 316, October 1993.

[50] L. Morin. *Quelques Contributions des Invariants Projectifs à la Vision par Ordinateur.* PhD thesis, Institut National Polytechnique de Grenoble, January 1993.

[51] L. Morin, P. Brand, and R. Mohr. Indexing with projective invariants. In *Proceedings of the Syntactical and Structural Pattern Recognition workshop, Nahariya, Israel.* World Scientific Pub., 1995.

[52] J. L. Mundy and A. Zisserman. Introduction – towards a new framework for vision. In J.L. Mundy and A. Zisserman, editors, *Geometric Invariance in Computer Vision*, pages 1 – 39. MIT Press, Boston, MA, 1992.

[53] Jean Ponce, Todd Cass, and David Marimont. Relative stereo and motion reconstruction. Report UIUC-BI-AI-RCV-93-07, Beckman Institute, University of Illinois, 1993.

[54] Jean Ponce, David H. Marimont, and Todd A. Cass. Analytical methods for uncalibrated stereo and motion reconstruction. In *Computer Vision - ECCV '94, Volume I, LNCS-Series Vol. 800, Springer-Verlag*, pages 463–470, 1994.

[55] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

[56] L. Quan. Invariants of 6 Points from 3 Uncalibrated Images. Rapport Technique RT 101 IMAG 19 LIFIA, LIFIA-IMAG, Grenoble, October 1993. To appear in ECCV94.

[57] Long Quan. Affine stereo calibration for relative affine shape reconstruction. In *Proc. BMVC*, pages 659–668, 1993.

[58] L. Robert and O.D. Faugeras. Relative 3D positioning and 3D convex hull computation from a weakly calibrated stereo pair. In *Proc. International Conference on Computer Vision*, pages 540–544, 1993.

[59] Charles A. Rothwell, Andrew Zisserman, David A. Forsyth, and Joseph L. Mundy. Canonical frames for planar object recognition. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 757 – 772, 1992.

[60] J.G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, Oxford, 1952.

[61] Larry S. Shapiro, Andrew Zisserman, and Michael Brady. Motion from point matches using affine epipolar geometry. In *Computer Vision - ECCV '94, Volume II, LNCS-Series Vol. 801, Springer-Verlag*, pages 73–84, 1994.

[62] A. Shashua and P. Anandan. Trilinear constraints revisited: Generalized trilinear constraints and the tensor brightness constraint. In *Proc. ARPA Image Understanding Workshop*, pages 815 – 820, 1996.

[63] Amnon Shashua. On geometric and algebraic aspects of 3D affine and projective structures from perspective 2D views. In *Applications of Invariance in Computer Vision : Proc. of the Second Joint European - US Workshop, Ponta Delgada, Azores – LNCS-Series Vol. 825, Springer Verlag*, pages 127 – 144, October 1993.

[64] Amnon Shashua. Projective depth: A geometric invariant for 3D reconstruction from two perspective/orthographic views and for visual recognition. In *Proc. International Conference on Computer Vision*, pages 583–590, 1993.

[65] Amnon Shashua. Trilinearity in visual recognition by alignment. In *Computer Vision - ECCV '94, Volume I, LNCS-Series Vol. 800, Springer-Verlag*, pages 479–484, 1994.

[66] Amnon Shashua. Algebraic functions for recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):779–789, August 1995.

[67] Amnon Shashua and Michael Werman. Trilinearity of three perspective views and its associated tensor. In *Proc. International Conference on Computer Vision*, pages 920 – 925, 1995.

[68] C. C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, fourth edition, 1980.

[69] Gunnar Sparr. Depth computations from polyhedral images. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 378–386, 1992.

[70] Minas E. Spetsakis and John Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:3:171–183, 1990.

[71] I.E. Sutherland. Sketchpad: A man-machine graphical communications system. Technical Report 296, MIT Lincoln Laboratories, 1963. Also published by Garland Publishing Inc, New York, 1980.

[72] I.E. Sutherland. Three dimensional data input by tablet. *Proceedings of IEEE*, Vol. 62, No. 4:453–461, April 1974.

[73] Bill Triggs. The geometry of projective reconstruction I: Matching constraints and the joint image. unpublished report, 1995.

[74] Bill Triggs. Matching constraints and the joint image. In *Proc. International Conference on Computer Vision*, pages 338 – 343, 1995.

[75] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces. *IEEE Trans. Patt. Anal. Machine Intell.*, PAMI-6:13–27, 1984.

[76] T. Viéville and Q.T. Luong. Motion of points and lines in the uncalibrated case. Report RR-2054, INRIA, 1993.

[77] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from line correspondences: Closed-form solution, uniqueness and optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 3:318–336, March, 1992.

[78] S. Wolfram. *Mathematica : A System for Doing Mathematics by Computer.* Addison-Wesley, Redwood City, California, 1988.

[79] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Report RR-2273, INRIA, 1994.