Self-Calibration of Stationary Cameras ⋆

Richard I. Hartley

G.E. CRD, Schenectady, NY, 12301.
Email : hartley@crd.ge.com

Abstract

A new practical method is given for the self-calibration of a camera. In this method, at least three images are taken from the 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 previous results by Maybank and Faugeras on self-calibration using the epipolar structure of image pairs. In the method of this paper, there is no epipolar structure since all images are taken from the same point in space, and so Maybank and Faugeras's method does 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.

Keywords : self-calibration, projective transformation, camera matrix.

⋆The research described in this paper has been supported by DARPA Contract #MDA972-91-C-0053

1 Introduction

The possibility of calibrating a camera based on the identification of matching points in several views of a scene taken by the same camera has been shown by Maybank and Faugeras ([**?**, **?**]). Using techniques of Projective Geometry they showed that each pair of views of the scene can be used to provide two quadratic equations in the five unknown parameters of the camera. For this, it is necessary that the two views be taken from different viewpoints. Given three pairs of views, a method of solving these equations to obtain the camera calibration has been reported in [**?**, **?**] based on directly solving these quadratic equations using homotopy continuation. It has been reported however that this method requires extreme accuracy of computation, and seems not to be suitable for routine use. In addition with large numbers of cameras (more than three or four) this method threatens to be unworkable, because the number of potential solutions grows rapidly as a function of the number of views. An alternative algorithm for calibration of a moving camera has been given in [**?**], which works for any number of views. However, the algorithm is lengthy and tricky to implement. Other methods are discussed in [**?**]. Nevertheless, the task of self-calibration in the case of arbitrary camera motions is difficult, and none of the methods given so far is entirely satisfactory. The applicability of these methods is further complicated by the problem of finding matched points in images taken from different viewpoints. This task can be difficult, because of occlusion, aspect changes and lighting changes that inevitably occur when the camera moves.

Recently several papers on self-calibration have appeared ([**?**, **?**, **?**]). These papers all rely on known motions of the cameras. In [**?**] the motion of the camera is assumed to be purely translational. In [**?**, **?**] 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 ([**?**]). In addition, the methods of [**?**, **?**] require tracing features in the image through many frames. In [**?**] an approximate guess at the location of the principal point is also necessary. In this paper, 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. It has recently come to my attention that the possibility of calibrating a camera undergoing rotatory motions has also been suggested by work of Viévilled, Luong and Faugeras ([**?**]), but they did not pursue this line of research.

The calibration algorithm is demonstrated on real and synthetic data and is shown to perform robustly in the presence of noise.

2 The Camera Model

A commonly used model for perspective cameras is that of projective mapping from 3D projective space, $\mathcal{P}^3$, to 2D projective space, $\mathcal{P}^2$. This map may be represented by a 3×4 matrix, M of rank 3. The mapping from $\mathcal{P}^3$ to $\mathcal{P}^2$ takes the point $\mathbf{x} = (x, y, z, 1)^\top$ to $\mathbf{u} = M\mathbf{x}$ in homogeneous coordinates. (Note: the equality relation when applied to

homogeneous vectors really means equality up to a non-zero scale factor).

Provided the camera centre is not located on the plane at infinity, the matrix M may be decomposed as $M = K(R|-R\mathbf{t})$, where $\mathbf{t}$ represents the location of the camera, R is a rotation matrix representing the orientation of the camera with respect to an absolute coordinate frame, and K is an upper triangular matrix called the *calibration matrix* of the camera. The matrix $(R|-R\mathbf{t})$ represents a rigid transformation (rotation and translation) of R^3. Given a matrix M it is a very simple matter to obtain this decomposition, using the QR-decomposition of matrices.

The entries of the matrix K may be identified with certain physically meaningful quantities known as the *internal parameters* of the camera. Indeed, K may be written as

$$K = \begin{pmatrix} k_u & s & p_u \\ 0 & k_v & p_v \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

where

- k_u is the magnification in the u coordinate direction
- k_v is the magnification in the v coordinate direction
- p_u and p_v are the coordinates of the principal point
- s is a skew parameter corresponding to a skewing of the coordinate axes.

Note that K is non-singular. This follows from the requirement that M should have rank 3.

The parameters k_u and k_v are related to the focal length f of the camera. In particular, if we assume that the skew $s = 0$, and that the pixels are square ($k_u = k_v$), then k_u is equal to the focal length of the camera measured in pixels. For CCD cameras, the pixels in the sensor array will not be square. In the case of non-square pixels, k_u is equal to the ratio of f to the pixel dimension measured in the u direction, and k_v is analogously defined.

For CCD cameras, it is quite unlikely that the skew s differs significantly from 0, since CCD arrays can be manufactured with high precision. However, many images used in aerial photogrammetry applications are taken on film and are later digitized. A common means of digitization is to use a digitizing camera that takes a photograph of the original film image. This digitization process causes a further 2D projective transformation to be applied to the original image. Nevertheless, the composite imaging process may still be expressed as a 3D to 2D projective mapping $\mathbf{u} = M\mathbf{x}$. In this case, the calibration matrix for the digitized image depends on the characteristics of both the original camera and the digitizing camera. A non-zero value of s will result if the principal axis of the digitizing camera is not precisely perpendicular to the image being digitized. This non-perpendicularity will also result in non-equal values for k_u and k_v, even if the digitizing camera has square pixels.

The images used for the experiments reported in this paper were obtained as follows. A 35mm camera with ordinary black and white film was used to produce negatives. These negatives were enlarged to produce positive 3.5×5 inch prints, which were then digitized using a flat-bed scanner. The enlargement process can lead to a non-zero value

of s and non-equal values of k_u and k_v if the negative and print paper are not precisely parallel. For digitized enlarged images obtained in this manner, the values of k_u and k_v are dependent on the focal length of the original camera, the degree of enlargement and the pixel-size of scanner. For this reason, the term magnification, rather than focal length is preferred here.

The term *camera* is used in this paper as an abstraction of the complete imaging process that maps world points to image points in a digitized image by one of these processes (or some other equivalent process).

The purpose of this paper is to give a method for determining the matrix K of internal camera parameters. 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 paper, 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.

3 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 3.1. *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 3.1 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 3.2. *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 $M_j = KR_j = P_jK$.

4 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 9)
2. For each $j = 1, \ldots, N$ compute the 2D projective transformation P_j matching J_0 to J_j. (Section 5)
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)
4. Refine the estimated camera matrix using Levenberg-Marquardt iterative techniques. (Section 8)

The steps of this algorithm will be described in detail in subsequent sections of this paper, as indicated. The main subject of this paper 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.

5 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.

5.1 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 $A^\top A$ ([**?**]).

5.2 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 greates 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.

5.3 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

$$\sum d(\mathbf{u}_i^j, \hat{\mathbf{u}}_i^j)^2 \tag{2}$$

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

$$\hat{\mathbf{u}}_i^j = P_j P_k^{-1} \hat{\mathbf{u}}_i^k \tag{3}$$

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 (2), subject to the constraint (3).

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 (3), 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 (2), 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 (2), 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 ([**?**]). 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 [**?**]. 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 ([**?**]). 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 ([**?**]), and also in [**?**]. 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 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

$$(KK^\top){P_j}^{-\top} = P_j(KK^\top) \tag{4}$$

Given sufficiently many views and corresponding matrices P_j equation 4 may be used to solve for the entries of the matrix $KK^\top$. In particular, denoting $KK^\top$ by C and writing

$$C = KK^\top = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

the equation (4) 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 (4). Consequently, C can only be determined up to a constant factor. It turns out that because of redundancy, the nine equations derived from (4) for a single known transformation P_j are not sufficient to solve for C (see Section 11). 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 5. 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 using the Choleski factorization ([**?**]). 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. The Choleski factorization of C is easily computed as follows. Since C is symmetric, it may be written as $C = UDU^\top$ where D is a diagonal matrix containing the eigenvalues of C, all real and positive, and U is an orthogonal matrix the columns of which are the corresponding eigenvectors. This factorization is easily found using the Jacobi method for eigenvalue determination ([**?**]) or else the Singular Value Decomposition ([**?**, **?**]). Since D is diagonal, real and positive, we may take its square root, writing $D = EE^\top$ where E is diagonal. Then $C = VV^\top$ where $V = UE$. The matrix V is not upper-triangular yet. However, we may apply the QR decomposition ([**?**, **?**]) to write $V = KR$ where K is upper triangular and R is a rotation matrix. Then $C = VV^\top = KRR^\top K^\top = KK^\top$ as required. This is the Choleski factorization of C. It is easy to prove that the Choleski factorization is almost unique. Specifically, if K_1 and K_2 are two upper triangular matrices satisfying $K_1 {K_1}^\top = K_2 {K_2}^\top$ then $K_2^{-1} K_1 = {K_2}^\top {K_1}^{-\top}$. Since the left side of this equation is upper-triangular, and the right side is lower triangular, they must both in fact be diagonal. Hence, $K_1 = K_2 D$ where D is diagonal. Furthermore, $D = K_2^{-1} K_1 = {K_2}^\top {K_1}^{-\top}$ is equal to its own inverse transpose, and hence is a diagonal matrix with diagonal entries equal to ± 1. Hence, if we insist that the diagonal entries or K are positive, then the Choleski factorization $C = KK^\top$ is unique.

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.

7 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 [**?**] it is shown how Kruppa's equations ([**?**]) 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 4.

8 Iterative Estimation of the Calibration matrix

In section 5.3 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

$$ \mathbf{u}_i^j = KR_j\mathbf{x}_i + \boldsymbol{\epsilon}_i^j $$

for each pair (i, j) for which $\mathbf{u}_i^j$ is defined, and such that the squared error sum, $\sum \boldsymbol{\epsilon}_i^{j2}$ is minimized. The difference between this and the iteration problem described in 5.3 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 [**?**] 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. There are various methods of parametrizing rotations. This estimation problem is similar to the Relative Orientation problem solved by Horn ([**?**, **?**]) using quaternions. In the method of quaternions, a rotation is represented by a unit quaternion. This method of parametrization has the disadvantage of using four parameters per rotation (a quaternion having four coordinates) and requiring the quaternions to be renormalized at each step. I 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 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 represents a substantial speed increase when large numbers of rotations are being estimated.

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 5 and 6. In particular, from Section 5.1 or Section 5.3 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 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 was so good that the difference between the estimates found with and without this final estimation step did not differ very significantly.

9 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 ([**?**, **?**]) 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.

10 Experimental Verification of the Algorithm

10.1 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 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 4. 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 $\theta = 10^\circ$ of the positive z axis. The results are summarized in tables 1, 2 and 3. The second experiment was with $N = 10$ images with view directions lying within an angle $\theta = 30^\circ$ of the positive z-axis. Results are summarised in tables 4, 5 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.

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 1: *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.*

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 2: *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 3: *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 4: *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 5: *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: *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 1: *Five images of the capitol, numbered 1 – 5 left-to-right and top-to-bottom.*

10.2 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 1) were taken with a 35mm camera 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 9, and the calibration was carried out. A composite of the five images is shown in Fig 2. The calibration results are summarized in Table 6.

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 difficully of calibration using the methods of this paper. The results of this experiment were as shown in Table 7. Two of the images used are shown

Figure 2: *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 7: *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.*

in Fig 3. 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 3: *Two images of a parking lot*

11 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 ([**?**]). Alternatively, one may make other assumptions about the calibration, for instance that the camera has square pixels, $k_u = k_v$.

According to Proposition 3.1, 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

$$R_x = \begin{pmatrix} 1 & & \\ & c & -s \\ & s & c \end{pmatrix}$$

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

$$(P - I)\mathbf{h}_1 = 0 \tag{5}$$

and

$$\begin{pmatrix} P - cI & -sI \\ sI & P - cI \end{pmatrix} \begin{pmatrix} \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \ . \tag{6}$$

Because of the choice of c and s, the matrices in (5) and (6) will be singular. Consequently, we can solve (5) to find $\mathbf{h}_1$ and (6) 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 (5) and (6) will not have an exact solution. The least-squares solution is to be used. From the $\mathbf{h}_i$ we may resassemble 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 11.3. *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

$$K'' = \frac{1}{\sqrt{AB}} \begin{pmatrix} \alpha\sqrt{A} & (\alpha^2 - 1)r_{11}r_{21} & (\alpha^2 - 1)r_{11}r_{31}\sqrt{B} \\ 0 & B & (\alpha^2 - 1)r_{21}r_{31}\sqrt{B} \\ 0 & 0 & A\sqrt{B} \end{pmatrix} \tag{7}$$

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 11.3.

Proposition 11.4. *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 (7) 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

$$\alpha^2 = \frac{k_{11}r_{11}r_{21} - k_{12}r_{11}^2}{k_{11}r_{11}r_{21} - k_{12}(r_{11}^2 - 1)} \quad ; \quad \alpha > 0 \tag{8}$$

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 (5) and (6).
4. Take the QR-decomposition $H = KR$.
5. Find $\alpha > 0$ by solving (8).
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.

12 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.

13 Experiments with Calibration from Two Images

13.1 Tests with Synthetic Data

First of all, the calibration algorithm was applied to synthetic data to determine its performance in the presence of noise.

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 8: *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.*

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 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 8. 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.

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 9: *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°.

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 9.

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 8 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 1). The calibration computed using all five views is provided as a good 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 10.

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.

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 10: **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.*

14 Handling translations

It is a basic assumption of the method described in this paper 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 paper, 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 [**?**], 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 [**?**], 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 [**?**] 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 paper (Sections 5 and 9) 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 [**?**], it is possible to find the camera calibration explicitly from three views or more taken from a camera undergoing arbitrary motions. In [**?**] 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 paper, a final iterative refinement of the camera calibration is suggested (Section 8) 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 8, or [**?**]. As in section 8, initialization is important. In section 8, 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.

15 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 paper 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 paper make clearer the theory behind self-calibration schemes such as those of [**?**, **?**]). As was demonstrated in Section 13, knowledge about the actual motion of the camera (which was assumed in [**?**, **?**]) may be incorporated into our algorithm to give high quality results.

Clearly, the algorithms in this paper 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 paper seem much more practical than methods based on a moving camera ([**?**]), both because of the ease of point matching and the simplicity of the calibration algorithms (for instance compare with [**?**, **?**]).

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. ([**?**, **?**]) 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 paper 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 [**?**, **?**] 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 paper 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 paper. 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.