# Euclidean Reconstruction from Uncalibrated Views *

*Richard I. Hartley*

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

**Abstract**

The possibility of calibrating a camera from image data alone, based on matched points identified in a series of images by a moving camera was suggested by Mayband and Faugeras. This result implies the possibility of Euclidean reconstruction from a series of images with a moving camera, or equivalently, Euclidean structure-from-motion from an uncalibrated camera. No tractable algorithm for implementing their methods for more than three images have been previously reported. This paper gives a practical algorithm for Euclidean reconstruction from several views with the same camera. The algorithm is demonstrated on synthetic and real data and is shown to behave very robustly in the presence of noise giving excellent calibration and reconstruction results.

## 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 ([13, 4]). 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. A method of solving these equations to obtain the camera calibration has been reported in [13, 4, 12] based on directly solving these quadratic equations using 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.

In this paper a method is given based partly on the well known Levenberg-Marquardt (LM) parameter estimation algorithm, partly on new non-iterative algorithms and partly on techniques of Projective Geometry for solving this self-calibration problem. This algorithm has the advantage of being applicable to large numbers of views, and in fact performs best when many views are given. As a consequence, the algorithm can be applied to the structure-from-motion problem to determine the structure of a scene from a sequence of views with the same uncalibrated camera. Indeed, since the calibration of the camera may be determined from the correspondence data, it is possible to compute a Euclidean reconstruction of the scene. That is, the scene is reconstructed, relative to the placement of one of the cameras used as reference, up to an unknown scaling.

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

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

## 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 $3 \times 4$ 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.

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. 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 internal camera parameters. 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.

## 3 The Euclidean Reconstruction Problem

The reconstruction problem to be solved in this paper will now be described. Consider a situation in which a set of 3D points $\mathbf{x}_j$ are viewed by a set of $N$ cameras with matrices $M_i$ numbered from 0 to $N-1$. Denote by $\mathbf{u}^i_j$ the coordinates of the $j$-th point as seen by the $i$-th camera. Given the set of coordinates $\mathbf{u}^i_j$ it is required to find the set of camera matrices, $M_i$ and the points $\mathbf{x}_j$. This is the reconstruction problem. A *reconstruction* based on a set of image correspondences $\{\mathbf{u}^i_j\}$ consists of a set of camera matrices $M_i$ and points $\mathbf{x}_j$ such that $M_i\mathbf{x}_j \approx \mathbf{u}^i_j$. (The notation $\approx$ denotes equality up to a non-zero scale factor.) For compactness we denote the reconstruction by the pair $(\{M_i\}, \{\mathbf{x}_j\})$. Without further restriction on the $M_i$ or $\mathbf{x}_j$, such a reconstruction is called a projective reconstruction, because the points $\mathbf{x}_j$ may differ by an arbitrary 3D projective transformation from the *true* reconstruction ([3, 5]). 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 paper to mean a similarity transform, namely the composition of a rotation, a translation and a uniform scaling.

According to the result of Maybank and Faugeras ([13]) 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 [13] but was clarified by Luong ([12])). Consequently, we assume in this paper that all cameras have the same calibration, so that $M_i = K(R_i \mid -R_i\mathbf{t}_i)$, 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 $M_i$ and $\mathbf{x}_j$ from a set of image correspondences $\mathbf{u}_j^i$. The points $\mathbf{x}_j$ and the camera matrices, $M_i$ 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 $M_i$ 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 paper, therefore 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 [2].

# 4 Levenberg Marquardt Minimization

The Levenberg-Marquardt (LM) algorithm is a well known algorithm for parameter estimation ([15]). However, since it is such an important ingredient of our reconstruction method, it is described here in detail.

## 4.1 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 $\hat{\mathbf{y}} = f(\hat{\mathbf{x}}) + \hat{\boldsymbol{\epsilon}}$ for which $||\hat{\boldsymbol{\epsilon}}||$ 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 $\hat{\mathbf{y}} = f(\mathbf{x}_0) + \boldsymbol{\epsilon}_0$. We assume that the function $f$ is approximated at $\mathbf{x}_0$ by $f(\mathbf{x}_0 + \boldsymbol{\Delta}) = f(\mathbf{x}_0) + J\boldsymbol{\Delta}$, where $J$ is the linear mapping represented by the Jacobian matrix $J = \partial\mathbf{y}/\partial\mathbf{x}$. Setting $\mathbf{x}_1 = \mathbf{x}_0 + \boldsymbol{\Delta}$ leads to $\hat{\mathbf{y}} - f(\mathbf{x}_1) = \hat{\mathbf{y}} - f(\mathbf{x}_0) - J\boldsymbol{\Delta} = \boldsymbol{\epsilon}_0 - J\boldsymbol{\Delta}$. It is required to minimize $||\boldsymbol{\epsilon}_0 - J\boldsymbol{\Delta}||$. Solving for $\boldsymbol{\Delta}$ is a linear minimization problem that can be solved by the method of normal equations. The minimium occurs when $J\boldsymbol{\Delta} - \boldsymbol{\epsilon}_0$ is perpendicular to the row space of $J$, which leads to the so-called *normal equations* $J^\top(J\boldsymbol{\Delta} - \boldsymbol{\epsilon}_0) = 0$ or $J^\top J\boldsymbol{\Delta} = J^\top\boldsymbol{\epsilon}_0$ . Thus, the solution is obtained by starting with an estimate $\mathbf{x}_0$ and computing successive approximations according to the formula

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

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

$$J^\top J\boldsymbol{\Delta}_i = J^\top\boldsymbol{\epsilon}_i \ .$$

Matrix $J$ is the Jacobian $\partial \mathbf{y}/\partial \mathbf{x}$ evaluated at $\mathbf{x}_i$ and $\boldsymbol{\epsilon}_i = \hat{\mathbf{y}} - f(\mathbf{x}_i)$ 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 $\mathbf{x}_0$.

## 4.2 Levenberg-Marquardt Iteration

The Levenberg-Marquardt (abbreviated LM) iteration method is a slight variation on the Newton iteration method. The normal equations $N\boldsymbol{\Delta} = J^\top J\boldsymbol{\Delta} = J^\top \boldsymbol{\epsilon}$ are replaced by the *augmented normal equations* $N'\boldsymbol{\Delta} = J^\top \boldsymbol{\epsilon}$, where $N'_{ii} = (1+\lambda)N_{ii}$ and $N'_{ij} = N_{ij}$ for $i \neq j$. The value $\lambda$ 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 $\lambda$ is divided by 10 before the next iteration. On the other hand if the value $\boldsymbol{\Delta}$ leads to an increased error, then $\lambda$ 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 $\lambda$ until an acceptable $\boldsymbol{\Delta}$ is found constitutes one iteration of the LM algorithm.

## 4.3 Implementation

Based on the implementation of the LM algorithm in [15] I 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 $\delta$ 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, I have seen almost no disadvantage in using numerical differentiation, though for simple functions $f$ I prefer to provide a routine to compute $J$, partly for aesthetic reasons, partly because of a possible slightly improved convergence and partly for speed.

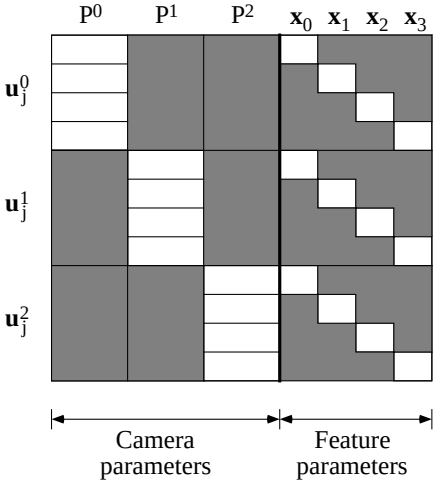
As an alternative to all the dependent variables being equally weighted, it is possible to provide a weight matrix specifying the weights of the dependent variables $\mathbf{y}$. This weight matrix may be diagonal specifying independent weights for each of the $y_i$, or else it may be symmetric, equal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

## 4.4 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

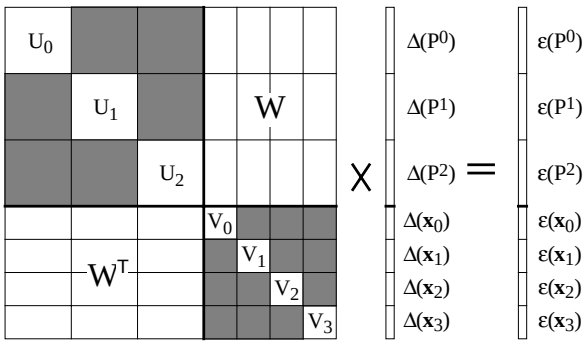
sparse block structure of the normal equations. It is possible to take advantage of this to achieve an enormous simplification in the solution of the normal equations. This method is described in [17] but is presented here for the convenience of the reader.

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 $M_i$ 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 $M_0$ 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 $J^\top J\boldsymbol{\Delta} = J^\top \boldsymbol{\epsilon}$ has a special block structure as follows:




(2)

It is possible to give specific formulae for each of the blocks in the normal equations. Specifically, let $D(\mathbf{u}_j^i, M_i)$ 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 $M_i$. 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

$$\begin{aligned} U_i &= \sum_j D(\mathbf{u}_j^i, M_i)^\top D(\mathbf{u}_j^i, M_i) \\ V_j &= \sum_i D(\mathbf{u}_j^i, \mathbf{x}_j)^\top D(\mathbf{u}_j^i, \mathbf{x}_j) \\ W_{ij} &= D(\mathbf{u}_j^i, M_i)^\top D(\mathbf{u}_j^i, \mathbf{x}_j) \\ \epsilon(M_i) &= \sum_j D(\mathbf{u}_j^i, M_i)^\top \epsilon(\mathbf{u}_j^i) \\ \epsilon(\mathbf{x}_j) &= \sum_i D(\mathbf{u}_j^i, \mathbf{x}_j)^\top \epsilon(\mathbf{u}_j^i) \end{aligned} \tag{3}$$

The normal equations (2) may be written in the form

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

where each of the matrices $U$, $V$ and the vectors $\Delta(M)$, $\Delta(X)$, $\epsilon(M)$ 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

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

The resulting set of equations

$$\begin{pmatrix} U - WV^{-1}W^\top & 0 \\ W^\top & V \end{pmatrix} \begin{pmatrix} \Delta(M) \\ \Delta(X) \end{pmatrix} = \begin{pmatrix} \epsilon(M) - WV^{-1}\epsilon(X) \\ \epsilon(X) \end{pmatrix} \tag{4}$$

may be divided into two sets of equations, to be solved separately. From the top half of (4), one obtains

$$(U - WV^{-1}W^\top)\Delta(M) = \epsilon(M) - WV^{-1}\epsilon(X) \tag{5}$$

which may be solved to get $\Delta(M)$. The resulting solution may then be substituted back into the bottom half of (4) providing a set of equations $V\Delta(X) = \epsilon(X) - W^\top\Delta(M)$, or

$$\Delta(X) = V^{-1}(\epsilon(X) - W^\top\Delta(M)) \ . \tag{6}$$

Because of the block-diagonal form of $V$, the equations (5) may be computed efficiently using the quantities computed in (3). Specifically, the matrix $A = U - WV^{-1}W^\top$ divides naturally into sub-blocks, where the $(i, j)$-th sub-block is the matrix

$$A_{ij} = \delta_{ij}U_i - \sum_k W_{ik}V_k^{-1}{W_{jk}}^\top \tag{7}$$

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

$$\mathbf{b}_i = \epsilon(M_i) - \sum_j W_{ij}V_j^{-1}\epsilon(\mathbf{x}_j) \tag{8}$$

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). 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 (9) may be done block-by-block as follows :

$$\Delta(\mathbf{x}_j) = V_j^{-1}(\epsilon(\mathbf{x}_j) - \sum_i W_{ij}{}^\top \Delta(M_i)) \tag{9}$$

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 Fig 2. 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, I 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 $15000 \times 15000$, and solving it using usual methods (for instance Gaussian elimination) would be out of the question.

# 5 Reconstruction by Direct Levenberg-Marquardt Iteration

A direct approach to the Euclidean reconstruction problem is to solve directly for the unknown camera matrices, $M_i = K(R_i \mid -R_i\mathbf{t}_i)$ and points $\mathbf{x}_j$. In particular, we search for $M_i$ of the required form, and $\mathbf{x}_j$ such that $\hat{\mathbf{u}}_j^i = M_i\mathbf{x}_j$ 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 relies on an 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 $\mathbf{u}_j^i$.

There are various methods of parametrizing the rotations. Horn ([8, 9]) uses quaternions to do this. 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 $\theta_i$, $\phi_i$ and $\kappa_i$. Initially, $X_i$ is set to the initial estimate of the rotation, and $\theta_i$, $\phi_i$ and $\kappa_i$ are all set to zero (and hence $\Delta$ 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 $\theta_i$, $\phi_i$ and $\kappa_i$ are reset to zero.

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 ([8, 9]) In fact the algorithm given in [8] amounts essentially to Newton iteration by solving the normal equations, using the method of back-substitution mentioned in Section 4.4, 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 Projective 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. Nevertheless, the estimation of the 3D transformation is itself carried out in several sub-steps.

For the present, we drop the assumption that all the cameras have the same calibration. The basic fact about projective reconstruction is the theorem ([3, 5]) that any two reconstructions of a scene from a set of (sufficiently many) image correspondences in images

taken with uncalibrated cameras differ by a 3D projective transformation. In particular a solution in which all the cameras have the same calibration must differ by a projective transformation from any other solution in which the cameras are possibly different.

Various methods of projective reconstruction from two or more views have been given previously ([3, 5, 14]). The method given in [5] is a straight-forward non-iterative construction method from two views. Where high precision is required, it should be followed by iterative refinement. Mohr et. al. ([14]) have reported a direct LM approach to projective reconstruction. However, with my recoding of their algorithm I have been unable to obtain reliable convergence in all cases. Therefore, I 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, $M_i$ and points $\mathbf{x}_j$ such that $\hat{\mathbf{u}}_j^i = M_i \mathbf{x}_j$ 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 $M_0 = (I \mid 0)$. This least-squares minimization problem is different from the one described in Section 5. 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 $M_i$ for $i > 0$. The dependent variables $\mathbf{y}$ are the image coordinates.

The main difference between my algorithm and that of Mohr et. al. is that whereas they fix the locations of five points in space, I fix the location and the orientation of one of the cameras. In particular, I set $M_0 = (I \mid 0)$. 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 my algorithm. Setting $M_0 = (I \mid 0)$ 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 $M_0$, which may be done easily be choosing three points that do not map to collinear points in the image corresponding to $M_0$.

This method will not converge, however, if a good initial estimate is not known. Fortunately, there exist linear methods for computing an initial reconstruction. The strategy is as follows. We use two of the views to compute a projective reconstruction of those points seen in the two images. The locations of these points and their images in the other views are used to solve one by one for the positions of the other cameras (in the arbitrary projective frame of the initial reconstruction). The 3D locations of additional points may be computed as soon as the camera parameters are known for two cameras in which these points are visible.

In particular, let $F$ be the fundamental matrix for a pair of cameras. If $F$ factors as a product $F = [\mathbf{p}']_\times M$, then the matrices $(I \mid 0)$ and $(M|\mathbf{p}')$ are one choice of a pair of camera matrices for the two cameras. Let $\mathbf{u} \leftrightarrow \mathbf{u}'$ be a pair of matched points in the two images, then it may be shown ([7]) that the point $M\mathbf{u}$ lies on the epipolar line $\mathbf{u}' \times \mathbf{p}'$ in the second image ($\mathbf{p}'$ is the epipole). If in particular $M\mathbf{u} = \beta\mathbf{u}' - \alpha\mathbf{p}'$ then the

corresponding object space point $\mathbf{x}$ is $\begin{pmatrix} \mathbf{u} \\ \alpha \end{pmatrix}$. It is easily verified that this point maps onto $\mathbf{u}$ and $\mathbf{u}'$ in the two images. Using this method we may reconstruct the points seen in these two images. This initial reconstruction from two views may be refined by LM iteration if required. In fact this is done in our implementation.

Given this initial reconstruction of a subset of the points visible in the first two images, it is now possible to compute directly the camera matrix for any other cameras in which at least six of these points are visible. This is done using the direct linear transformation (DLT) method as described by Sutherland ([19]). The order in which the camera matrices are computed is done so as to maximize the number of already reconstructed points seen by each camera in turn.

After all the camera matrices and 3D point locations have been computed in this way, the LM camera modelling program is run to refine the camera matrices and the point locations as already described.

# 7 Converting Projective to Euclidean Reconstruction

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 $(\{M_i\}, \{\mathbf{x}_j\})$ is a projective reconstruction, then any other reconstruction is of the form $(\{M_i H^{-1}\}, \{H\mathbf{x}_j\})$ where $H$ is a $4 \times 4$ non-singular matrix. We seek such a matrix $H$ such that the transformed camera matrices $M_i H^{-1}$ all have the same (yet to be determined) calibration matrix, $K$. In other words, we seek $H$ such that $M_i H^{-1} = K(R_i \mid -R_i \mathbf{t}_i)$ 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 $M_0 = (I \mid 0)$, it follows that $H^{-1}$ may be assumed to have the restricted form

$$H^{-1} = \begin{pmatrix} K & \mathbf{0} \\ \mathbf{v}^\top & \alpha \end{pmatrix} .$$

Since the constant $\alpha$ 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

$$H^{-1} = \begin{pmatrix} K & 0 \\ -\mathbf{v}^\top K & 1 \end{pmatrix} = \begin{pmatrix} I & \mathbf{0} \\ -\mathbf{v}^\top & 1 \end{pmatrix} \begin{pmatrix} K & 0 \\ 0 & 1 \end{pmatrix} \tag{10}$$

Now, writing each $M_i = (A_i \mid -A_i \mathbf{t}_i)$ and multiplying out leads to a requirement that

$$A_i(I + \mathbf{t}_i \mathbf{v}^\top)K \approx K R_i \tag{11}$$

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 we may 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 (11) 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

$$A_i(I + \mathbf{t}_i\mathbf{v}^\top)K = K_i'R_i \ . \tag{12}$$

Subsequently, we compute the matrices $X_i = K^{-1}K_i'$ for all $i$. Since we have assumed that $M_0 = (A_0 \mid -A_i\mathbf{t}_i) = (I \mid 0)$, it follows that $X_0 = I$. Furthermore, if $K$ and $\mathbf{v}$ satisfy the desired condition (11) 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 $\alpha_i$ 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

$$\sum_{i>0} ||\alpha_i X_i - I||^2 \tag{13}$$

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 fuction $f : R^8 \mapsto R^{6(N-1)}$ where $N$ is the total number of cameras. The function $f$ maps the eight [2] 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 3 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 $K$.

# 8 Euclidean From Affine Reconstruction

With $H^{-1}$ of the form (10) matrix $H$ may be written as

$$H = \begin{pmatrix} K^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & 0 \\ \mathbf{v}^\top & 1 \end{pmatrix} \ .$$

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.

[2] 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

We will now present a simple non-iterative algorithm for the determination of $K$, and hence of the Euclidean structure.

Equation (11) 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 $\alpha_i$ 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

$$(K K^\top) {B_i}^{-\top} = B_i (K K^\top) \tag{14}$$

Given sufficiently many views and corresponding matrices $B_i$ equation 14 may be used to solve for the entries of the matrix $K K^\top$. In particular, denoting $K K^\top$ by $C$ and writing

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

the equation (14) 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 (14). Consequently, $C$ can only be solved up to a constant factor. It turns out that because of redundancy, the nine equations derived from (14) 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 ([15]). 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 $C = K K^\top$ is found it is an easy matter to solve for $K$ using the Choleski factorization ([1, 15]). 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 $C = K K^\top$ 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.

It may be remarked that the matrix $C$ has a geometric interpretation. It is the dual of the image of the absolute conic. The condition that $C = B C B^\top$ is related to the fact that $C$ is invariant under translation and rotation of the camera.

### 8.1 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 $\infty$ 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 [2]. On the other hand, Sparr ([18]) gives a method of computing affine structure given a single view, and Koenderink and van Doorn [11] give a method for computing affine structure from pairs of orthographic views. Quan [16] gives a method of affine construction from two views given affine constraints.

## 9 Quasi-affine Reconstruction

We are interested, however, in finding the plane at infinity without any extra given information. The first step will be to get an approximation to the plane at infinity. This will be done by considering the *cheirality* of the images, in other words, by taking into account the fact that the points must lie in front of the cameras that view them.

The subject of cheirality of cameras was considered in detail in [6]. It was shown in that paper that if $(\{M_i\}, \{\mathbf{x}_j\})$ is a projective reconstruction of a set of image correspondences derived from a real scene, then there exist constants $\eta_j$ and $\epsilon_i$ equal to $\pm 1$, such that $\epsilon_i \eta_j M_i \mathbf{x}_j = (u^i_j, v^i_j, w^i_j)^\top$ where each $w^i_j > 0$. It should be noted that the equality sign here means exact equality, and not equality up to a constant factor. Given the reconstruction $(\{M_i\}, \{\mathbf{x}_j\})$ we may replace $M_i$ by $\epsilon_i M_i$ and $\mathbf{x}_j$ by $\eta_j \mathbf{x}_j$ to obtain a reconstruction such that $M_i \mathbf{x}_j = (u^i_j, v^i_j, w^i_j)^\top$ and each $w^i_j > 0$. Suppose that this has been done. Now ([6]) there exists a matrix $H = \begin{pmatrix} \beta I & 0 \\ \alpha \mathbf{v}^\top & \alpha \end{pmatrix}$ with $\alpha, \beta = \pm 1$ such that $H\mathbf{x}_j = (x'_j, y'_j, z'_j, s'_j)^\top$ with $s'_j > 0$ for all $j$, and such that $M_i H^{-1} = (A'_i \mid -A'_i \mathbf{t}_i)$ with $\det A'_i > 0$ for all $i$.

The conditions satisfied by the matrix $H$ transform into inequalities. In particular, $s'_j > 0$ means that

$$\alpha(\mathbf{v}^\top 1)\mathbf{x}_j > 0 \tag{15}$$

for each point $\mathbf{x}_j$. The condition $\det A'_i < 0$ also gives rise to a linear inequality as follows. Writing $M_i = (A_i \mid -A_i \mathbf{t}_i)$ then $M_i H^{-1} = (A'_i \mid -A'_i \mathbf{t}'_i)$ where $A'_i = \beta A_i (I + \mathbf{t}_i \mathbf{v}^\top)$. Then

$$\det A'_i = \beta \det A_i \det(I + \mathbf{t}_i \mathbf{v}^\top) = \beta(1 + \mathbf{t}_i{}^\top \mathbf{v}) \det A_i \ .$$

Since $A_i$ and $\mathbf{t}_i$ are known this gives a linear inequality

$$\beta(1 + \mathbf{t}_i{}^\top \mathbf{v}) \det A_i > 0 \tag{16}$$

in the entries of $\mathbf{v}$. These set of inequalities (15) and (16) constraining the placement of the plane at infinity are called the *cheiral inequalities*.

Naturally, we propose to solve the cheiral inequalities using linear programming (LP). The four cases corresponding to the choices of $\alpha$ and $\beta$ must be considered. In order to obtain a single solution it is necessary to define an appropriate goal function to optimize. We choose to maximize the margin by which the given inequalities are satisfied, since this should correspond informally to a placement of the plane at infinity at a maximum distance from the points and the cameras. For this to make sense, the homogeneous coordinate expression for $\mathbf{x}_j = (x_j, y_j, z_j, s_j)^\top$ should first be normalized so that $||\mathbf{x}_j|| = 1$. Now, we have a set of inequalities of the form $\mathbf{f}_i{}^\top \mathbf{v} \geq g_i$, where $\mathbf{f}_i$ is simply the vector of coefficients of the $i$-th equation. We add an extra variable $\delta$ to obtain equations of the form $\mathbf{f}_i{}^\top \mathbf{v} - \delta \geq g_i$. The LP problem is to maximize $\delta$ subject to the given inequalities. If $\delta > 0$ in the optimum solution, then the original inequalities have a solution, and this is the solution that we accept to obtain $\mathbf{v}$. Once $\mathbf{v}$ has been found by solving the LP problem, the projective reconstruction is transformed by the corresponding matrix $H$. The new reconstruction may be termed a *quasi-affine* reconstruction.

By solving this cheiral inequalities, we find a candidate value for $\mathbf{v}$. By the method of Section 8 we can now compute the corresponding value of $K$. This estimate may then be refined using the method described in Section 7. 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 I have done in practice.

# 10 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 6)
   (a) Compute the essential matrix $Q$ 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 9)

(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 $H = \begin{pmatrix} I & 0 \\ \mathbf{v}^\top & 1 \end{pmatrix}$ 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 9)

   (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 8 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 7 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 5.

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.

# 11 Experimental Evaluation

This algorithm has been evaluated on both real and synthetic data.

## 11.1 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 $700 \times 600$ pixels. The degradation becomes progressively worse for greater degrees of noise, however the ratio $k_u/k_v$ remains

| Noise | $p_u$ | $p_v$ | $k_v$ | $skew$ | $k_u/k_v$ | $\Delta$ |
|---|---|---|---|---|---|---|
| – | 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 1: Reconstruction from Three Views

| Noise | $p_u$ | $p_v$ | $k_v$ | $skew$ | $k_u/k_v$ | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ |
|---|---|---|---|---|---|---|---|---|
| – | 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 2: Reconstruction from 15 Views

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 $\Delta$ 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.

## 11.2 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 ([10]) 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.

$\Delta_1$ The error in reconstruction, namely the distance between the actual and the reconstructed point locations.

$\Delta_2$ The residual pixel error after step 4 of the algorithm in Section 10.

$\Delta_3$ The residual pixel error after step 5 of the algorithm. This shows only a 10% reduction compared with $\Delta_2$.

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 $600 \times 600$ (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.

### 11.3 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 [14]. The image coordinates were integer numbers ranging between 0 and 500. Figure 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 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.


Figure 1: On the left one of the views of a house. On the right a view of the reconstructed house.

# References

[1] K.E. Atkinson. *An Introduction to Numerical Analysis, 2nd Edition.* John Wiley and Sons, New York, 1989.

[2] B. Boufama, R. Mohr, and F. Veillon. Euclidean constraints for uncalibrated reconstruction. *Technical Report, LIFIA - IRIMAG*, 1993.

[3] 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.

[4] 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.

[5] 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.

[6] R. I. Hartley. Cheirality invariants. In *Proc. DARPA Image Understanding Workshop*, pages 745 – 753, 1993.

[7] Richard Hartley and Rajiv Gupta. Computing matched-epipolar projections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 549 – 555, 1993.

[8] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4:59 – 78, 1990.

[9] B. K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America, A*, Vol. 8, No. 10:1630 – 1638, 1991.

[10] 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.

[11] Jan J. Koenderink and Andrea J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America, A*, 1992.

[12] Q.-T Luong. *Matrice Fondamentale et Calibration Visuelle sur l'Environnement.* PhD thesis, Universite de Paris-Sud, Centre D'Orsay, 1992.

[13] 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.

[14] 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.

[15] 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.

[16] Long Quan. Affine stereo calibration for relative affine shape reconstruction. In *Proc. BMVC*, pages 659–668, 1993.

[17] C. C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, VA, fourth edition, 1980.

[18] Gunnar Sparr. Depth computations from polyhedral images. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 378–386, 1992.

[19] I.E. Sutherland. Three dimensional data input by tablet. *Proceedings of IEEE*, Vol. 62, No. 4:453–461, April 1974.