Projective Reconstruction from Line Correspondences

Richard I. Hartley

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

Abstract

The paper gives a practical rapid algorithm for doing projective reconstruction of a scene consisting of a set of lines seen in three or more images with uncalibrated cameras. The algorithm is evaluated on real and ideal data to determine its performance in the presence of varying degrees of noise. By carefully consideration of sources of error, it is possible to get accurate reconstruction with realistic levels of noise. The algorithm can be applied to images from different cameras or the same camera. For images with the same camera with unknown calibration, it is possible to do a complete Euclidean reconstruction of the image. This extends to the case of uncalibrated cameras previous results of Spetsakis and Aloimonos on scene reconstruction from lines.

Keywords : Projective reconstruction, Structure from motion, Epipolar geometry, Fundamental Matrix

1 Introduction

This paper gives an effective algorithm for the projective reconstruction of a scene consisting of lines in space as seen in at least three views with uncalibrated cameras. The placement of the cameras with respect to the scene is also determined. At least three views are necessary, since as discussed in [13], no information whatever about camera placements may be derived from any number of line-to-line correspondences in fewer than three views. With three arbitrary cameras with unknown possibly different calibrations it is not possible to specify the scene more precisely than up to an arbitrary projective transformation of space. This contrasts with the situation for calibrated cameras in which a set of sufficiently many lines may be determined up to a scaled Euclidean transformation from three views ([12, 13]).

In the case where all of the three cameras are the same, however, or at least have the same calibration, it is possible to reconstruct the scene up to a scaled Euclidean transformation. This result relies on the theory of self-calibration

expounded by Maybank and Faugeras ([9]) for which a robust algorithm has been given in [6]. In particular for the case of a stationary camera and a moving object the camera calibration remains fixed. This motion and structure problem for lines was solved in [12, 13] for calibrated cameras. The assumption of calibration means that a pixel in each image corresponds to a uniquely specified ray in space relative to the location and placement of the camera. The result of this paper is that this assumption is not necessary.

The *fundamental matrix* defined by Longuet-Higgins ([7]) (originally for calibrated cameras) contains all the information available about relative camera placements that can be derived from image point correspondences. The methods of this paper give a way of computing the fundamental matrix from line correspondences.

It will be assumed that three different views are taken of a set of fixed lines in space. That is, it is assumed that the cameras are moving and the lines are fixed, which is opposite to the assumption made in [13]. In general, it will not be assumed that the images are taken with the same camera. Thus the three cameras are uncalibrated and possibly different.

2 Notation and Basics

The three-dimensional space containing the scene will be considered to be the 3-dimensional projective space $\mathcal{P}^3$ and points in space will be represented by homogeneous 4-vectors $\mathbf{x}$. Similarly, image space will be regarded as the 2-dimensional projective space $\mathcal{P}^2$ and points in an image will be represented by homogeneous 3-vectors $\mathbf{u}$. The space-image mapping induced by a pinhole camera may be represented by a 3×4 matrix M of rank 3, such that if $\mathbf{x}$ and $\mathbf{u}$ are corresponding object and image points then $\mathbf{u} = M\mathbf{x}$. Such a matrix will be called a camera matrix. The pose and internal calibration of the camera are easily deduced from the camera matrix (for instance see [6]). It will often be desirable to decompose a camera matrix into a 3×3 matrix A and a column vector $\mathbf{t}$, as follows : $M = (A \mid \mathbf{t})$. If the camera centre is at a finite point, then A is non-singular, but we will not make this restriction.

All vectors are assumed to be column vectors. The transpose $\mathbf{u}^\top$ of $\mathbf{u}$ is a row vector. Notationally, vectors will be treated as $n \times 1$ matrices. In particular $\mathbf{a}^\top\mathbf{b}$ is the scalar product of vectors $\mathbf{a}$ and $\mathbf{b}$, whereas $\mathbf{a}\mathbf{b}^\top$ is a matrix.

If $\mathbf{t}$ is a vector and X is a matrix, then $\mathbf{t} \times X$ denotes the matrix $(\mathbf{t} \times \mathbf{x}_1, \mathbf{t} \times \mathbf{x}_2, \mathbf{t} \times \mathbf{x}_3)$ constructed by taking the cross product $\mathbf{t} \times \mathbf{x}_i$ with each of the columns $\mathbf{x}_i$ of X individually. Similarly, $X \times \mathbf{t}$ is a matrix obtained by taking the cross product of $\mathbf{t}$ with the rows of X individually.

Just as points in image space $\mathcal{P}^2$ are represented by homogeneous vectors so are lines in $\mathcal{P}^2$. Bold greek letters such as $\boldsymbol{\lambda}$ represent lines. The point $\mathbf{u}$ lies on the line $\boldsymbol{\lambda}$ if and only if $\boldsymbol{\lambda}^\top\mathbf{u} = 0$. The line through two points $\mathbf{u}$ and $\mathbf{u}'$ is given by the cross product $\mathbf{u} \times \mathbf{u}'$. Similarly, the intersection of lines $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ is equal to $\boldsymbol{\lambda} \times \boldsymbol{\lambda}'$. We will sometimes wish to consider the Euclidean space R^2 as a subset of $\mathcal{P}^2$ and determine the perpendicular Euclidean distance from a point to a line. If $\mathbf{u} = (u, v, w)^\top$ and $\boldsymbol{\lambda} = (\lambda, \mu, \nu)^\top$, then the perpendicular

distance is given by

$$d(\boldsymbol{\lambda}, \mathbf{u}) = \frac{\boldsymbol{\lambda}^\top \mathbf{u}}{w(\lambda^2 + \mu^2)^{1/2}} \tag{1}$$

The fundamental matrix : The fundamental matrix corresponding to a pair of cameras is of basic importance in analyzing pairs of images of a common scene. It is at the heart of algorithms for camera calibration [9, 4, 2], image rectification [5], scene reconstruction [6, 3, 7] and transfer [1].

The fundamental matrix may be computed from a pair of camera matrices as shown in the following proposition.

Proposition 2.1. *If M_0 and M_1 are two camera matrices and H is any non-singular 4×4 matrix, then the fundamental matrices corresponding to the pairs (M_0, M_1) and $(M_0 H, M_1 H)$ are equal.*
The fundamental matrix corresponding to a pair of cameras with matrices $(I \mid 0)$ and $(A \mid \mathbf{t})$ is equal to $\mathbf{t} \times A$.

The first statement is that the fundamental matrix is invariant under projective transformation of the two cameras. The formula for the fundamental matrix is a special form of a more general formula given in [4].

Methods have been given for the computation of the fundamental matrix from point correspondences [7, 8, 6]. It will be shown in this paper how the fundamental matrix may be computed from line correspondences.

Projective Reconstruction Consider a set of lines in space viewed by several cameras, and let $\boldsymbol{\lambda}_j^i$ be the image of the i-th line in the j-th image. The task of projective reconstruction is to find a set of camera matrices M_j and 3D-lines $\boldsymbol{\chi}_i$ so that line $\boldsymbol{\chi}_i$ is indeed mapped to the line $\boldsymbol{\lambda}_j^i$ by the mapping M_j. For the present we pass over the questions of how to represent lines in space and how they are acted on by camera matrices M_j.

If the camera matrices are allowed to be arbitrary, then it is easily seen that the scene can not be reconstructed more precisely than up to an arbitrary 3D projective transformation. Indeed, let H be a non-singular 4×4 matrix, and let every point $\mathbf{x}$ be transformed by the projective transformation $\mathbf{x} \mapsto H\mathbf{x}$. If the camera matrices M_j are simultaneously transformed to $M_j' = M_j H^{-1}$, then $M_j' H\mathbf{x} = M\mathbf{x}$. Thus, the correspondence between object and image points is preserved. It follows that the correspondence between 3D-lines and image lines is also preserved, since lines are made up of individual points.

Consider now a reconstruction from three views, and let the three camera matrices be M_0, M_1 and M_2. We make the assumption that no two of the cameras are located at the same point in space. Let H be formed by adding one extra row to M_0 to make a non-singular 4×4 matrix. Then since $HH^{-1} = I_{4\times 4}$, it follows that $M_0 H^{-1} = (I|0)$. Since M_0 may be transformed to $(I \mid 0)$, by applying transformation H to the reconstruction we may assume without loss of generality that $M_0 = (I \mid 0)$. Next we turn to the form of $M_1 = (A_1 \mid \mathbf{t}_1)$. Since cameras M_0 and M_1 are not located at the same points, $\mathbf{t}_1 \neq 0$ and M_1 may therefore be scaled so that ${\mathbf{t}_1}^\top \mathbf{t}_1 = 1$. It may be observed that further

multiplication on the right by a matrix $H^{-1} = \begin{pmatrix} I & 0 \\ -\mathbf{t}_1{}^\top A_1 & 1 \end{pmatrix}$ transforms $(I \mid 0)$ to itself, while mapping $(A_1 \mid \mathbf{t}_1)$ to $(A_1 - \mathbf{t}_1\mathbf{t}_1{}^\top A_1 \mid \mathbf{t}_1)$. This matrix has the interesting property that the columns of $A_1 - \mathbf{t}_1\mathbf{t}_1{}^\top A_1$ are perpendicular to $\mathbf{t}_1$, since $\mathbf{t}_1{}^\top(A_1 - \mathbf{t}_1\mathbf{t}_1{}^\top A_1) = 0$. It follows of course that A_1 has rank 2, and so the camera is located on the plane at infinity, but this remark is not used.

The result of this discussion is that in seeking a projective reconstruction of a scene from three views, we may assume without loss of generality that

1. $M_0 = (I \mid 0)$.
2. $M_1 = (R \mid \mathbf{r}_4)$, where $\mathbf{r}_4{}^\top R = 0$, and $\mathbf{r}_4{}^\top \mathbf{r}_4 = 1$.
3. $M_2 = (S \mid \mathbf{s}_4)$, where $\mathbf{s}_4{}^\top \mathbf{s}_4 = 1$

where for simplicity in future computations we have changed notation to avoid proliferation of subscripts. The observation that these conditions determine the reconstruction uniquely will not be needed.

3 The Transfer Equations

We now address the question of how lines are mapped by camera matrices. Instead of considering the forward mapping, however, we will consider the backward mapping – given a line in an image, determine the plane in space that maps onto it. This will be a plane passing through the camera centre, consisting of points that map to the given image line. This plane has a simple formula as follows.

> The plane in space mapped to the line $\boldsymbol{\lambda}$ by the camera with matrix M is equal to $M^\top \boldsymbol{\lambda}$.

To justify this remark, note that a point $\mathbf{x}$ lies on the plane with coordinates $M^\top \boldsymbol{\lambda}$ if and only if $\boldsymbol{\lambda}^\top M\mathbf{x} = 0$. This is also the condition for the point $M\mathbf{x}$ to lie on the line $\boldsymbol{\lambda}$.

Now, consider three cameras with matrices $M_0 = (I \mid 0)$, $M_1 = (R \mid \mathbf{r}_4)$ and $M_2 = (S \mid \mathbf{s}_4)$. Let $\boldsymbol{\lambda}_0$, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ be corresponding lines in the three images, each one the image of a common line in space. The planes corresponding to these three lines are the columns of the matrix

$$\begin{pmatrix} \boldsymbol{\lambda}_0 & R^\top \boldsymbol{\lambda}_1 & S^\top \boldsymbol{\lambda}_2 \\ 0 & \mathbf{r}_4{}^\top \boldsymbol{\lambda}_1 & \mathbf{s}_4{}^\top \boldsymbol{\lambda}_2 \end{pmatrix} .$$

Since these three planes must meet in a single line in space, the above matrix must have rank 2. Therefore, up to an insignificant scale factor,

$$\boldsymbol{\lambda}_0 = (R^\top \boldsymbol{\lambda}_1)(\mathbf{s}_4{}^\top \boldsymbol{\lambda}_2) - (S^\top \boldsymbol{\lambda}_2)(\mathbf{r}_4{}^\top \boldsymbol{\lambda}_1) \tag{2}$$

This important formula allows us to transfer lines from a pair of images to a third image directly. In general, we will represent a line in space simply by

giving its images $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ with respect to the two cameras with matrices M_1 and M_2.

Now, writing $R = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ and $S = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ where the $\mathbf{r}_i$ and $\mathbf{s}_i$ are the columns of R and S, we see that the i-th entry (or row) of $\boldsymbol{\lambda}_0$ given in (2) is $({\mathbf{r}_i}^\top \boldsymbol{\lambda}_1)({\mathbf{s}_4}^\top \boldsymbol{\lambda}_2) - ({\mathbf{s}_i}^\top \boldsymbol{\lambda}_2)({\mathbf{r}_4}^\top \boldsymbol{\lambda}_1)$, which may be rearranged as $\boldsymbol{\lambda}_1{}^\top(\mathbf{r}_i{\mathbf{s}_4}^\top - \mathbf{r}_4{\mathbf{s}_i}^\top)\boldsymbol{\lambda}_2$. This leads to a second form of (2).

$$\boldsymbol{\lambda}_0 = \begin{pmatrix} {\lambda_1}^\top(\mathbf{r}_1{\mathbf{s}_4}^\top - \mathbf{r}_4{\mathbf{s}_1}^\top)\boldsymbol{\lambda}_2 \\ {\lambda_1}^\top(\mathbf{r}_2{\mathbf{s}_4}^\top - \mathbf{r}_4{\mathbf{s}_2}^\top)\boldsymbol{\lambda}_2 \\ {\lambda_1}^\top(\mathbf{r}_3{\mathbf{s}_4}^\top - \mathbf{r}_4{\mathbf{s}_3}^\top)\boldsymbol{\lambda}_2 \end{pmatrix} = \begin{pmatrix} {\lambda_1}^\top T_1\boldsymbol{\lambda}_2 \\ {\lambda_1}^\top T_2\boldsymbol{\lambda}_2 \\ {\lambda_1}^\top T_3\boldsymbol{\lambda}_2 \end{pmatrix} \tag{3}$$

where the 3×3 matrices T_i are defined by this equation. Formula (3) is much the same as a formula given for calibrated cameras in [13], but proven here for uncalibrated cameras. In [13], the letters E, F and G are used instead of T_i. However, since F is the standard notation for the fundamental matrix, we prefer to use T_i. Equations (2) and (3) may be termed the *transfer equations* in two alternative forms.

If sufficiently many line matches are known, it is possible to solve for the three matrices T_i. In fact, since each $\boldsymbol{\lambda}_0^i$ has two degrees of freedom, each set of matched lines $\boldsymbol{\lambda}_0^i \leftrightarrow \boldsymbol{\lambda}_1^i \leftrightarrow \boldsymbol{\lambda}_2^i$ gives rise to two linear equations in the entries of T_1, T_2 and T_3. Exactly how these equations may best be formulated will be discussed later. Since the T_1, T_2 and T_3 have a total of 27 entries, but are defined only up to a common scale factor, 13 line matches are sufficient to solve for the three matrices. With more than 13 line matches, a least-squares solution may be computed.

4 Retrieving the Camera Matrices

Formula (3) gives a formula for the transfer matrices T_i in terms of the camera matrices. We now show that it is possible to go the other way and retrieve the camera matrices, M_i from transfer matrices T_i. It will be assumed in this discussion that the rank of each of the matrices T_i is at least 2, which will be the case except in certain special camera configurations. There are methods of proceeding in case one or more of the T_i has rank one, but we omit any further consideration of these cases, for lack of space. See [13] for the a discussion of methods applying to calibrated cameras. For general camera configurations all T_i have rank 2.

Now, note that $(\mathbf{r}_4 \times \mathbf{r}_i)^\top T_i = 0$ since $(\mathbf{r}_4 \times \mathbf{r}_i)^\top \mathbf{r}_i = (\mathbf{r}_4 \times \mathbf{r}_i)^\top \mathbf{r}_4 = 0$. It follows that we can compute $\mathbf{r}_4 \times \mathbf{r}_i$ up to an unknown multiplicative factor by finding the null-space of T_i for each $i = 1, \ldots, 3$. However, by (2.1) $\mathbf{r}_4 \times R = (\mathbf{r}_4 \times \mathbf{r}_1, \mathbf{r}_4 \times \mathbf{r}_2, \mathbf{r}_4 \times \mathbf{r}_3)$ is the fundamental matrix for cameras 0 and 1, and hence has rank 2. It follows that $\mathbf{r}_4$ (or $-\mathbf{r}_4$) may be computed as the unique unit vector normal to all of $\mathbf{r}_4 \times \mathbf{r}_i$ for $i = 1, \ldots, 3$. The vector $\mathbf{s}_4$ may be computed in the same way.

Once we have computed $\mathbf{r}_4$ and $\mathbf{s}_4$, the computation of the fundamental matrices

is easy, according to the following formulae :

$$\begin{aligned} F_{01} &= (\mathbf{r}_4 \times \mathbf{r}_1, \mathbf{r}_4 \times \mathbf{r}_2, \mathbf{r}_4 \times \mathbf{r}_3) = (\mathbf{r}_4 \times T_1\mathbf{s}_4, \mathbf{r}_4 \times T_2\mathbf{s}_4, \mathbf{r}_4 \times T_3\mathbf{s}_4) \\ F_{02} &= (\mathbf{s}_4 \times \mathbf{s}_1, \mathbf{s}_4 \times \mathbf{s}_2, \mathbf{s}_4 \times \mathbf{s}_3) = -(\mathbf{s}_4 \times {T_1}^\top\mathbf{r}_4, \mathbf{s}_4 \times {T_2}^\top\mathbf{r}_4, \mathbf{s}_4 \times {T_3}^\top\mathbf{r}_4) \end{aligned} \tag{4}$$

where F_{01} and F_{02} are the fundamental matrices corresponding to the camera pairs (M_0, M_1) and (M_0, M_2) respectively.

Next, we derive formulae for the camera matrices M_1 and M_2. To do this, we make use of the assumption that ${\mathbf{r}_4}^\top\mathbf{r}_i = 0$ for each i. Then one verifies that ${\mathbf{r}_4}^\top T_i = -{\mathbf{s}_i}^\top$. This means that $M_2 = (S \mid \mathbf{s}_4) = (-{T_1}^\top\mathbf{r}_4, -{T_2}^\top\mathbf{r}_4, -{T_3}^\top\mathbf{r}_4, \mathbf{s}_4)$. Furthermore, substituting ${\mathbf{r}_4}^\top T_i = -{\mathbf{s}_i}^\top$ into the formula $T_i = \mathbf{r}_i{\mathbf{s}_4}^\top - \mathbf{r}_4{\mathbf{s}_i}^\top$ and multiplying by $\mathbf{s}_4$ gives $T_i\mathbf{s}_4 = \mathbf{r}_i + \mathbf{r}_4{\mathbf{r}_4}^\top T_i\mathbf{s}_4$, from which one may solve for $\mathbf{r}_i$. This gives the following formulae for the camera matrices.

$$\begin{aligned} M_1 &= (R \mid \mathbf{r}_4) &= ((I - \mathbf{r}_4{\mathbf{r}_4}^\top)T_1\mathbf{s}_4, (I - \mathbf{r}_4{\mathbf{r}_4}^\top)T_2\mathbf{s}_4, (I - \mathbf{r}_4{\mathbf{r}_4}^\top)T_3\mathbf{s}_4, \mathbf{r}_4) \\ M_2 &= (S \mid \mathbf{s}_4) &= (-{T_1}^\top\mathbf{r}_4, -{T_2}^\top\mathbf{r}_4, -{T_3}^\top\mathbf{r}_4, \mathbf{s}_4) \end{aligned} \tag{5}$$

The correctness of this formula relies on the fact that T_i is of the form $T_i = \mathbf{r}_i{\mathbf{s}_4}^\top - \mathbf{r}_4{\mathbf{s}_i}^\top$. In other words, if one computes M_1 and M_2 from the T_i using (5) and then recomputes T_i using (3) then one does not retrieve the same values of T_i unless T_i is of the correct form.

5 How close are two lines?

If the algorithm outlined so far is used to compute the camera matrices from a set of line correspondences in three images, then the results are good as long as the data is noise-free. Of course, this will never be the case, and disappointingly, the results are extraordinarily bad in the presence of even small amounts of noise. Why is this the case, and how do we fix it ? These questions will be answered in the following sections.

The effect of noise will be to perturb lines to lines that lie close to the correct lines. The goal of the reconstruction algorithm must then be to find 3D lines in space and camera matrices that project the 3D lines to lines in the images "close" to the measured lines. But what does it mean for two lines to be close to each other, and how is this to be quantified. The obvious answer is to represent lines as homogeneous vectors of unit norm and to consider the two lines to be close when the vectors are close to each other. Formally, if $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ are two lines represented by unit homogenous vectors, then we can define

$$d(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = min(||\boldsymbol{\lambda} - \boldsymbol{\lambda}'||, ||\boldsymbol{\lambda} + \boldsymbol{\lambda}'||)$$

This definition turns out to be entirely unsuitable, however. As an example, consider two lines with equations $u = 100$ and $v = 100$. These two lines are perpendicular to each other, and no one would consider them to be "close". However, as normalized homogeneous vectors, they have representations approximately equal to $(0.01, 0, -1)^\top$ and $(0, 0.01, -1)^\top$ respectively. These two vectors are close to each other.

Intuitively, two lines can be considered close only if they have nearly the same slope. However, two lines with slightly differing slope may be considered close

near their point of intersection, but not at points far from their intersection. For instance, lines $u+v=1000$ and $u+0.999v=-1000$ should not normally be considered close, except near their intersection point $(-199000, 200000)$. This example makes the point, however, that closeness of lines depends on which region of the lines are being considered.

In general, in computer vision, we are interested in line segments, and not infinite lines. However, often different segments of the same line are seen in two different views, so the distance between endpoints is not a suitable metric. We will take the position in this paper that line segments are usually defined by specifying two end points. Lines can arise in other ways, such as by taking a best fit to a set of edgels on a purported straight line in the image, but most representations of lines may readily be reduced to one in terms of two end points. We denote the line defined by points $\mathbf{u}$ and $\mathbf{u}'$ as $\boldsymbol{\lambda}(\mathbf{u}, \mathbf{u}')$. Let $\boldsymbol{\lambda}'$ be another line. We define the distance $d(\boldsymbol{\lambda}', \boldsymbol{\lambda}_{\mathbf{uu}'})$ to be $(d_1^2 + d_2^2)^{1/2}$, where d_1 and d_2 are the perpendicular distances from the line $\boldsymbol{\lambda}'$ to the points $\mathbf{u}$ and $\mathbf{u}'$ respectively, as given by (1). The fact that this function $d(*,*)$ is not symmetric, and hence not a true metric, concerns us not at all.

Now, we return to equations (3). Let lines $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ be defined precisely by their end points, and let $\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}_0(\mathbf{u}_0, \mathbf{u}_0')$. From the equation $\boldsymbol{\lambda}_0(\mathbf{u}_0, \mathbf{u}_0') = (\boldsymbol{\lambda}_1{}^\top T_1 \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_1{}^\top T_2 \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_1{}^\top T_3 \boldsymbol{\lambda}_2)^\top$ we obtain two equations

$$\begin{aligned} u_0 \boldsymbol{\lambda}_1{}^\top T_1 \boldsymbol{\lambda}_2 + v_0 \boldsymbol{\lambda}_1{}^\top T_2 \boldsymbol{\lambda}_2 + \boldsymbol{\lambda}_1{}^\top T_3 \boldsymbol{\lambda}_2 &= 0 \\ u_0' \boldsymbol{\lambda}_1{}^\top T_1 \boldsymbol{\lambda}_2 + v_0' \boldsymbol{\lambda}_1{}^\top T_2 \boldsymbol{\lambda}_2 + \boldsymbol{\lambda}_1{}^\top T_3 \boldsymbol{\lambda}_2 &= 0 \end{aligned} \tag{6}$$

where $\mathbf{u}_0 = (u_0, v_0, 1)^\top$ and $\mathbf{u}_0' = (u_0', v_0', 1)^\top$. These two equations do not represent precisely the distance from the transferred line to the endpoints of the measured line $\boldsymbol{\lambda}_0$, since the normalizing factor in the denominator of (1) is missing. This is the price we pay, however, to have a linear algorithm. The equations (6) are linear in the entries of the T_i and 13 point correspondences are sufficient to solve for the matrices T_i up to a common scale factor.

6 Getting the best solution

As pointed out, the linear methods described here for computing the camera matrices are not stable in the presence of noise. It is necessary to take extra precautions in order to get a good solution that is relatively immune to noise. The methods described below give the best results among several different approaches that were tested to avoid problems with noise. In fact, using these techniques result in quite accurate and stable reconstruction.

6.1 Condition number

If the units in the image plane are pixel numbers, then a typical line will have an equation of the form $\lambda u + \mu v + \nu = 0$, where $\nu >> \lambda, \mu$. If the equations (6) are constructed using these unadjusted coordinates, then the resulting set of equations is poorly conditioned. By experiment, it has been found that scaling all pixel coordinates so that the pixel values in the data range between about

-1.0 and 1.0 works well, giving a far closer match of the transferred line to the actual data than with the unscaled coordinates. An alternative approach is to try a range of scale factors, selecting the one that gives the best sum-of-squares residual error. The residual error is obtained by solving the equations (6) to find the T_i, then computing line $\hat{\boldsymbol{\lambda}}_0$ using (3) and finally computing the Euclidean distance of $\hat{\boldsymbol{\lambda}}_0$ to the end points of $\boldsymbol{\lambda}_0$. Since this computation is quite fast, one can afford repeated trials of this nature.

6.2 Converging on the optimal solution

In the presence of noise, the matrices T_i obtained by solving equations (6) will not have the correct form as given in (3). Hence the camera matrices computed using (5) will not correspond precisely to the computed T_i. Furthermore, solving equations (6) does not correspond exactly to minimizing the distance of the constructed line $\hat{\boldsymbol{\lambda}}_0$ to the endpoints of $\boldsymbol{\lambda}_0$. Thus, the computed camera matrices can only be considered as an approximation to the optimal solution – and not a very good approximation either. Nevertheless, it is good enough to initialize an iterative algorithm to converge to the optimal solution.

Starting from the initial solution found by the linear methods already described, we proceed by varying the entries of the camera matrices M_1 and M_2 so as to minimize the sum of squares of distances of the transferred lines $\hat{\boldsymbol{\lambda}}_0^i$ to the measured defining endpoints of the lines $\boldsymbol{\lambda}_0^i$. We use the first form of the transfer equation (2). This is a straight-forward parameter minimization problem, solved simply using the Levenberg-Marquardt algorithm ([10]). The varying parameters are the 24 entries of the matrices M_1 and M_2 and the quantity to be minimized is the sum of squares of Euclidean distances. The lines $\boldsymbol{\lambda}_1^i$ and $\boldsymbol{\lambda}_2^i$ are not varied, but are set to the exact measured values. Typically convergence occurs within 10 iterations. Furthermore, each iteration is very fast, since construction of the normal equations ([10]) requires time linear in the number of points, and the normal equations are only of size 24×24. For construction of the normal equations, numerical (rather than symbolic) differentiation is adequate, and simplifies implementation. The fact that this problem is over-parametrized (since M_1 and M_2 are not uniquely determined) causes no problems whatever. The total time required for reconstruction of 20 lines in three views is not more than 5 seconds on a Sparc 2.

6.3 The Optimal Solution

The solution given in the previous example is not quite optimal, since all the error is confined to measurement in the zero-th image, instead of being shared among all three images. The true optimal solution can be found by approximating lines $\boldsymbol{\lambda}_1^i$ and $\boldsymbol{\lambda}_2^i$ by lines $\hat{\boldsymbol{\lambda}}_1^i$ and $\hat{\boldsymbol{\lambda}}_2^i$ which are allowed to vary, as well as the matrix entries. The goal is to minimize the sum-of-squares of distances of lines $\hat{\boldsymbol{\lambda}}_j^i$ for all $j = 1, 2, 3$. The disadvantage of doing this is that there may be a large number of varying parameters. This disadvantage is mitigated however by an implementation based on the sparseness of the normal equations as described in [11]. Carrying out this final iteration to obtain a true optimal solution gives

minimal gain over the method described in the previous section. Moreover, it is definitely not a good idea to skip the previous iterative refinement step and attempt to find the optimal solution right away. Convergence problems can arise if this is done.

7 Reconstruction

Once the camera matrices are computed, it is a simple task to compute the positions of the lines in space. In particular, the line in space corresponding to a set of matched lines $\boldsymbol{\lambda}_0 \leftrightarrow \boldsymbol{\lambda}_1 \leftrightarrow \boldsymbol{\lambda}_2$ must be the intersection of the three planes ${M_i}^\top \boldsymbol{\lambda}_i$. A good way to compute this line is as follow. One forms the matrix $X = ({M_0}^\top \boldsymbol{\lambda}_0, {M_1}^\top \boldsymbol{\lambda}_1, {M_2}^\top \boldsymbol{\lambda}_2)$. Then a point $\mathbf{x}$ will lie on the intersection of the three planes if and only if $\mathbf{x}^\top X = 0$. We need to find two such points to define the line in space. Let the singular value decomposition ([10]) be $X = UDV^\top$, where D is a diagonal matrix $\mathrm{diag}(\alpha, \beta, 0, 0)$. Since $(0,0,0,1)DV = 0$, it follows that $(0,0,0,1)U^\top(UDV^\top) = (0,0,0,1)U^\top X = 0$. The vector $(0,0,0,1)U^\top$ is simply the last column of U. The same argument shows that $(0,0,1,0)U^\top X = 0$. In summary, the third and fourth columns of the matrix U represent a pair of points on the intersection of three planes, and hence define the required line in space.

8 Algorithm Outline

Here is a brief outline of the algorithm for reconstruction. We start with a set of at least thirteen sets of matched lines in three views : $\boldsymbol{\lambda}_0^i \leftrightarrow \boldsymbol{\lambda}_1^i \leftrightarrow \boldsymbol{\lambda}_2^i$, each line defined by specifying two end points $\mathbf{u}_j^i$ and ${\mathbf{u}'}_j^i$ in the image.

1. Scale the coordinates of the line end points so that u and v coordinates lie in a range -1 to 1.

2. Set up and solve the set of equations (6). The solution is the singular vector corresponding to the smallest singular value of the matrix of equations. This gives three 3×3 matrices T_i

3. Compute the vectors $\mathbf{r}_4$ and $\mathbf{s}_4$, and then compute the camera matrices M_1 and M_2 using (5).

4. Adjust the initial computed values of M_1 and M_2 by Levenberg-Marquardt iteration so that the transferred lines $\boldsymbol{\lambda}_0^i$ given by (2) are optimally close to the measured end points of $\boldsymbol{\lambda}_0^i$. (See section 6.2.)

5. If desired do a full least-squares fit to the data as described in section 6.3.

6. Compute the projective reconstruction of the lines geometry using the by intersection three planes corresponding to the three matching lines (section 7).

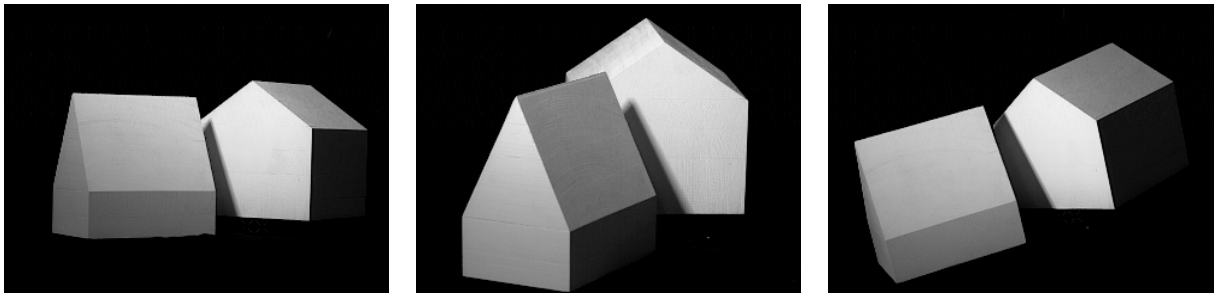

Figure 1: Three photos of houses

At this point we have a projective reconstruction of the scene. If we now assume that the three cameras were the same for all three views, then it is possible to adjust this reconstruction by the appropriate projective transformation to obtain a scaled Euclidean reconstruction of the scene. This possibility is ensured theoretically by [9]. A practical algorithm for converting a projective to a Euclidean reconstruction is given in [6] and may be applied with minor modifications to the case of lines.

9 Experimental Evaluation of the Algorithm

This algorithm was tested as follows. Three images of a scene consisting of two houses were acquired as shown in Fig 1. Edges and vertices were obtained automatically and matched by hand. In order to obtain some ground truth for the scene, a projective reconstruction was done based on point matches using the algorithm described in [6]. To carefully control noise insertion, image coordinates were adjusted (by an average of about 0.5 pixels) so as to make the projective reconstruction agree exactly with the pixel coordinates.

Lines were selected joining vertices in the image, only lines that actually appeared in the image being chosen (and not lines that join two arbitrary vertices), for a total of 15 lines. Next, varying degrees of noise were added to the endpoints defining the lines and the algorithm was run to compute the projective reconstruction.

Finally for comparison, the algorithm was run on the real image data. For this run, two extra lines were added, corresponding to the half obscured roof and ground line in the right hand house. Note that in the three images the endpoints of these lines are actually different points, since the lines are obscured to differing degrees by the left hand house. One of the advantages of working with lines rather than points is that such lines can be used.

In order to judge the quality of the reconstruction, and present it in a simple form, the errors in the positions of the epipoles were chosen. The epipolar positions are related to the relative positions and orientations of the three cameras. If the computed camera positions are correct, then so will be the reconstruction. The epipoles in images M_1 and M_2 corresponding to the centre of projection of camera M_0 are simply the last columns of M_1 and M_2 respectively. To measure

whether two epipoles are close, the following method was used. Let $\mathbf{p}$ and $\hat{\mathbf{p}}$ be actual and computed positions of the epipole, each vector being normalized to have unit length. We define $d(\mathbf{p}, \hat{\mathbf{p}'}) = 180.0 * min(||\mathbf{p} - \hat{\mathbf{p}}||, ||\mathbf{p} + \hat{\mathbf{p}}||)/\pi$. If the epipoles are close to the centre of the image, then this quantity gives a measure of their distance. If they are far from the image centre (which they are in this case – the epipoles are at locations (8249, 2006) and (-17876, 23000) in Euclidean coordinates), this is an approximate measure of the angular difference between the radial directions to the epipoles. The factor $180/\pi$ is included to give this angle in degrees.

The results of these experiments are given in table 9. The columns of this table have the following meanings.

- Column 1 gives the standard deviation of zero-mean gaussian noise added to both the u and v coordinates of the end-points of the lines
- Column 2 gives the residual error, which is the RMS distance of the images of the reconstructed lines from the measured noisy end points of the lines.
- Columns 3 and 4 (epipolar error) give the epipolar error (described above) for the epipoles in images 1 and 2 corresponding to the camera centre of image 0.

As can be seen from this table, the algorithm performs quite well with noise levels up to about 2.0 pixels (the image size being 640×484 pixels). For 3.0 and 4.0 pixels error the residual error is still small, but the epipolar error is large, meaning that the algorithm has found a solution other than the correct one. Since residual error should be of the order of the injected noise, the solution found is apparently just as good as the correct solution. Thus, the algorithm has worked effectively, but the problem is inherently unstable with this amount of noise. Note that 3–4 pixels' error is more than should occur with careful measurement.

The last line of the table gives the results for the real image data, and shows very good accuracy.

10 Conclusions

The algorithm described here provides an effective means of doing projective reconstruction from line correspondences in a number of images. The algorithm is rapid and quite reliable, provided the degree of error in the image-to-image correspondences is not excessive. It does, however require careful implementation to avoid convergence problems. For more than about 2 pixels of error in an image of size about 512×512 pixels, the problem of projective reconstruction becomes badly behaved. There exist multiple near-optimal solutions. For high resolution images where the relative errors may be expected to be smaller, the algorithm will show enhanced performance.

It is to be expected that (as with reconstruction from points [6]) the robustness of the reconstruction will increase substantially with more than the minimum

Noise	residual error	epipolar error 1	epipolar error 2
0.1	1.82e-02	4.55e-01	4.27e-01
0.25	4.50e-02	1.15e+00	1.07e+00
0.5	8.89e-02	2.31e+00	2.14e+00
1.0	1.74e-01	4.50e+00	4.26e+00
2.0	3.38e-01	7.29e+00	7.44e+00
3.0	9.96e-01	8.10e+01	2.56e+01
4.0	1.36e+00	2.15e+01	2.84e+01
–	3.10e-01	2.55e-01	7.27e-01

Table 1: Results of reconstruction for 15 lines from three views. Dimension of the image is 640×484 pixels. The last line represents the reconstruction from 17 real data lines.

number of views. This situation arises when an object is tracked through several frames by a video camera.

The work of [6] shows that a projective reconstruction may be converted to a Euclidean reconstruction if all the cameras have the same calibration, or alternatively Euclidean constraints are imposed on the scene.

References

[1] Eamon. B. Barrett, Michael H. Brill, Nils N. Haag, and Paul M. Payton. Invariant linear methods in photogrammetry and model matching. In J.L. Mundy and A. Zisserman, editors, *Geometric Invariance in Computer Vision*, pages 277 – 292. MIT Press, Boston, MA, 1992.

[2] 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.

[3] 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.

[4] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 579 – 587, 1992.

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

[6] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. In *Proc. of the Second Europe-US Workshop on Invariance, Ponta Delgada, Azores*, pages 187–202, October 1993.

[7] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, Sept 1981.

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

[9] 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.

[10] 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.

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

[12] Minas E. Spetsakis and John Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:3:171–183, 1990.

[13] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from line correspondences: Closed-form solution, uniqueness and optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 3:318–336, March, 1992.