Lines and Points in Three Views and the Trifocal Tensor

Richard I. Hartley

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

Abstract

This paper discusses the basic role of the trifocal tensor[1] in scene reconstruction from three views. This $3 \times 3 \times 3$ tensor plays a role in the analysis of scenes from three views analogous to the role played by the fundamental matrix in the two-view case. In particular, the trifocal tensor may be computed by a linear algorithm from a set of 13 line correspondences in three views. It is further shown in this paper, that the trifocal tensor is essentially identical to a set of coefficients introduced by Shashua to effect point transfer in the three view case. This observation means that the 13-line algorithm may be extended to allow for the computation of the trifocal tensor given any mixture of sufficiently many line and point correspondences. From the trifocal tensor the fundamental matrices of each pair of images may be computed, and the scene may be reconstructed. For unrelated uncalibrated cameras, this reconstruction will be unique up to projectivity. Thus, projective reconstruction of a set of lines and points may be reconstructed linearly from three views. If the tree cameras have the same calibration, then reconstruction up to a similarity transform may be achieved (though not linearly).

1 Introduction

This paper gives an effective algorithm for the projective reconstruction of a scene consisting of lines and points in space as seen in three views with uncalibrated cameras. The placement of the cameras with respect to the scene is also determined, as is the epipolar geometry (as expressed by the fundamental matrix) for each pair of images. This algorithm is unique in the literature in that it gives a unified linear approach that can deal with a mixture of points and lines. For instance, previous algorithms have been specific to points ([9, **?**, 8]) or lines ([15, 16]), but could not handle both. True, one could always use pairs of matching points to determine line matches, which can then be used in an algorithm for reconstruction from lines. This strategem, however, achieves a unified approach for lines and points at considerable expense, since a pair of point matches contains much more information than a single line match (as will be made clear quantitively in this paper). The restraint of using only points or lines forces one to ignore important information available in most images, particularly of man-made objects, since typically, one can find both distinguished points and lines in matching images.

Points are important in that they give much more information than lines. For instance although one can do relative reconstruction from only two views of a set of points ([9]),

[1] Suggestions for possible alternative names are solicited. One possibility is the fundamental tensor ...

for lines at least three views are necessary ([16]). On the other hand, lines have several advantages. Firstly, they can normally be determined more accurately than points, typically with an accuracy of better than a tenth of a pixel. Secondly, line matches may be used in cases where occlusions occur. Often end points of lines are not visible in the images. For instance, in Fig 1, the left hand slanting roof edge of the rear house may be matched in the three images, although its end point is occluded behind the front house. On the other hand, if we are to use line matches, then at least three views must be used, since no information whatever about camera placements may be derived from any number of line-to-line correspondences in fewer than three views.

Euclidean Reconstruction. 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 ([15, 16]). 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 ([10]) for which a robust algorithm has been given in [4]. 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 [15, 16] 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.

2 The Trifocal Tensor

This paper deals with the properties of an entity called, here for the first timer, the *trifocal tensor*. Since this entity has appeared previously in the literature in different guises, and it is therefore appropriate to discuss its history. With hindsight, we may atribute the discovery of the trifocal tensor to Spetsakis and Aloimonis ([15] and Weng, Huang and Ahuja ([16]), where it was used for scene reconstruction from lines in the case of calibrated cameras. It was later shown by the present author in [1, 3, 6] to be equally applicable to projective scene reconstruction from 13 lines in the uncalibrated case. Those papers form the basis for part of this article. In all of the above papers, the entity referred to here as the trifocal tensor was not considered as a tensor, but rather as a set of three 3×3 matrices. Perhaps the first author to refer to it as a tensor was Vieville ([**?**]) who continued the work on scene reconstruction from lines.

Meanwhile in independent work, Shashua introduced a set of 27 coefficients for a set of four independent tri-linearity conditions relating the coordinates of corresponding points in three views with uncalibrated cameras ([12]). Subsequently ([**?**]) Shashua gave a linear method for computation of the coefficients from only 7 point matches in three views.

A key result of this paper is that the set of coefficients introduced by Shashua ([12]) are exactly the same as the entries of the three 3×3 matrices of ([16, 3]), except for a change of sign[2] and rearrangement of the indices. The importance of this result is that

[2]In order to avoid the sign discrepancy, Shashua's coefficients will be defined with opposite sign in

it allows an amalgamation of the linear algorithms for points (Shashua [**?**]) and for lines ([6]). This results in an algorithm of significantly greater applicability and power than either of the line or point algorithms alone.

Whereas the line papers [16, 3, 6] consider three 3×3 matrices, Shashua's paper defines his coefficients as the entries of nine 3-vectors. In fact, essentially, we are dealing with a triply indexed $3 \times 3 \times 3$ array of values, which it is natural to treat as a tensor, as suggested by Vieville. Therefore, in this paper, we refer to this three-dimensional array as a tensor, though without making significant use of tensor notation or machinery. In recent unpublished work, Shashua has also considered his set of coefficients as a tensor. As for the name, I suggest the words *trifocal tensor* in an attempt to establish a standard terminology. Despite the potentially fundamental role played by this tensor in three-view stereo, I believe that the word *fundamental* is too often used for us to adopt the term *fundamental tensor*.

3 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. 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

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

A basic tool used in this paper is the solution of redundant sets of linear equations. The sets of equations that we generally need to solve are of the form $A\mathbf{x} = \mathbf{0}$. We

this paper

are not interested in the solution $\mathbf{x} = \mathbf{0}$. Generally, in the presence of noise, no exact solution is possible, since A will have full rank, hence we wish to find the nearest solution possible. The solution of such a system is defined up to an indeterminate (and usually uninteresting) scale factor. This system may be solved by minimizing $||A\mathbf{x}||$ subject to the restriction $||\mathbf{x}|| = 1$. The required $\mathbf{x}$ is the unit eigenvector corresponding to the smallest eigenvalue of $A^\top A$. It may be found using either Singular Value Decomposition, or Jacobi's method for finding eigenvalues of symmetric matrices ([11]).

Projective Reconstruction Consider a set of lines and points in space viewed by several cameras. We use the word *feature* to represent either a line or a point. We suppose that the image coordinates of each feature as seen in each image are given, but the actual positions of the features in space are unknown. The task of projective reconstruction is to find a set of camera matrices M_j and 3D-lines and points so that each such 3D feature is indeed mapped to the given image feature in each of the images. 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 well known ([**?**, 2]) that the scene can not be reconstructed more precisely than up to an arbitrary 3D projective transformation.

Consider now a reconstruction from three views, and let the three camera matrices be M, M' and M''. We make the assumption that no two of the cameras are located at the same point in space. Let H be formed by adding one extra row to M to make a non-singular 4×4 matrix. Then since $HH^{-1} = I_{4\times 4}$, it follows that $MH^{-1} = (I|0)$. Since M may be transformed to $(I \mid 0)$, by applying transformation H to the reconstruction we may assume without loss of generality that $M = (I \mid 0)$.

In this case, the three camera matrices M, M' and M'' may be written in the form $M = (I \mid 0)$, $M' = (A \mid \mathbf{a}_4)$ and $M'' = (B \mid \mathbf{b}_4)$, where $\mathbf{a}_4$ and $\mathbf{b}_4$ denote the final columns of M' and M''.

Assuming (as we always shall) that neither of the latter two cameras is located at the same location as the first (that is the origin), we see that $M'(0, 0, 0, 1)^\top = \mathbf{a}_4$. That is $\mathbf{a}_4$ is the projection of the centre of the first camera as seen from the second – it is the epipole. We may normalize $\mathbf{a}_4$ such that $||\mathbf{a}_4|| = 1$. Similar considerations apply to $\mathbf{b}_4$.

4 Transferring lines

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 $M = (I \mid 0)$, $M' = (A \mid \mathbf{a}_4)$ and $M'' = (B \mid \mathbf{b}_4)$. Let $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$ 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

$$\begin{pmatrix} \boldsymbol{\lambda} & A^\top \boldsymbol{\lambda}' & B^\top \boldsymbol{\lambda}'' \\ 0 & {\mathbf{a}_4}^\top \boldsymbol{\lambda}' & {\mathbf{b}_4}^\top \boldsymbol{\lambda}'' \end{pmatrix} .$$

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,

$$\boldsymbol{\lambda} = (A^\top \boldsymbol{\lambda}')({\mathbf{b}_4}^\top \boldsymbol{\lambda}'') - (B^\top \boldsymbol{\lambda}'')({\mathbf{a}_4}^\top \boldsymbol{\lambda}') \tag{2}$$

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 $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$ with respect to the two cameras with matrices M' and M''.

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

$$\boldsymbol{\lambda} = \begin{pmatrix} {\boldsymbol{\lambda}'}^\top(\mathbf{a}_1 {\mathbf{b}_4}^\top - \mathbf{a}_4 {\mathbf{b}_1}^\top)\boldsymbol{\lambda}'' \\ {\boldsymbol{\lambda}'}^\top(\mathbf{a}_2 {\mathbf{b}_4}^\top - \mathbf{a}_4 {\mathbf{b}_2}^\top)\boldsymbol{\lambda}'' \\ {\boldsymbol{\lambda}'}^\top(\mathbf{a}_3 {\mathbf{b}_4}^\top - \mathbf{a}_4 {\mathbf{b}_3}^\top)\boldsymbol{\lambda}'' \end{pmatrix} \tag{3}$$

Now, defining 3×3 matrices T_i by the equation

$$T_i = \mathbf{a}_1 {\mathbf{b}_4}^\top - \mathbf{a}_4 {\mathbf{b}_1}^\top \tag{4}$$

we obtain a simple set of equations

$$\boldsymbol{\lambda} = \begin{pmatrix} {\boldsymbol{\lambda}'}^\top T_1 \boldsymbol{\lambda}'' \\ {\boldsymbol{\lambda}'}^\top T_2 \boldsymbol{\lambda}'' \\ {\boldsymbol{\lambda}'}^\top T_3 \boldsymbol{\lambda}'' \end{pmatrix} \tag{5}$$

Formula (5) is much the same as a formula given for calibrated cameras in [16], but proven here for uncalibrated cameras. In [16], 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 (5) may be termed the *transfer equation* for lines.

If sufficiently many line matches are known, it is possible to solve for the three matrices T_i. In fact, since each $\boldsymbol{\lambda}$ has two degrees of freedom, each set of matched lines $\boldsymbol{\lambda} \leftrightarrow \boldsymbol{\lambda}' \leftrightarrow \boldsymbol{\lambda}''$ 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.

5 Formulating the Line Equations.

The previous description of the form of the transfer equations (5) was a little vague. In this section, the exact manner of forming these equations will be described.

Scaling the coordinates. The first step is the normalization of the image coordinates. 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 may be poorly conditioned. By experiment, it has been found that scaling all pixel coordinates so that the pixel values lie in the data range between about -1.0 and 1.0 works well, giving a closer match of the transferred line to the actual data than with the unscaled coordinates. A further possible transformation is to translate all the pixels so that the centroid of the measured pixel values is at the origin. This can be done independently to each image, without affecting the results of projective reconstruction. Later, we will be considering point matches as well as line matches. This scaling and translation operation should be applied to the coordinates of matched points, as well as lines.

Presentation of lines. So far, we have assumed perfect data. Of course, this will never be the case, and the equations (5) used to solve for the transfer matrices T_i will only be satisfied approximately. 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.

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 a number of points on the line. In the following discussion it will be assumed that a line is defined by specifying two end points. The method generalizes easily to the case where a line is defined as the best fit to a set of measured points, as is shown in the appendix to this paper. We denote the line defined by two points $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$ as $\boldsymbol{\lambda}(\mathbf{u}^{(1)}, \mathbf{u}^{(2)})$. Let $\boldsymbol{\lambda}'$ be another line. We define the distance $d(\boldsymbol{\lambda}', \boldsymbol{\lambda}_{\mathbf{u}^{(1)}\mathbf{u}^{(2)}})$ 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 $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$ respectively, as given by (1).

Normalization of lines. In the equations 5), the weight given to equations represented by different line correspondences depends on the specific representation of the lines $\boldsymbol{\lambda}_i$ and $\boldsymbol{\lambda}_i''$. To standardize this, we define a line $\mathbf{l}ambda' = (\lambda', \mu', \nu')^\top$ (and similarly for $\boldsymbol{\lambda}''$) to be normalized if $\lambda'^2 + \mu^2 = 1$. Thus, the lines $\boldsymbol{\lambda}_i'$ and $\mathbf{l}ambda_i''$ are computed by taking the cross-product of the two end points, and then normalizing.

The line equations. Now, we return to equations (5). Let lines $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$ be computed and normalized. Let $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{u}^{(1)}, \mathbf{u}')$. From the equation $\boldsymbol{\lambda}(\mathbf{u}^{(1)}, \mathbf{u}^{(2)}) = (\boldsymbol{\lambda}'^\top T_1 \boldsymbol{\lambda}'', \boldsymbol{\lambda}'^\top T_2 \boldsymbol{\lambda}'', \boldsymbol{\lambda}'^\top T_3 \boldsymbol{\lambda}'')^\top$ we obtain two equations

$$\begin{aligned} u^{(1)} \boldsymbol{\lambda}'^\top T_1 \boldsymbol{\lambda}'' + v^{(1)} \boldsymbol{\lambda}'^\top T_2 \boldsymbol{\lambda}'' + \boldsymbol{\lambda}'^\top T_3 \boldsymbol{\lambda}'' &= 0 \\ u^{(2)} \boldsymbol{\lambda}'^\top T_1 \boldsymbol{\lambda}'' + v^{(2)} \boldsymbol{\lambda}'^\top T_2 \boldsymbol{\lambda}'' + \boldsymbol{\lambda}'^\top T_3 \boldsymbol{\lambda}'' &= 0 \end{aligned} \tag{6}$$

where $\mathbf{u}^{(1)} = (u^{(1)}, v^{(1)}, 1)^\top$ and $\mathbf{u}^{(2)} = (u^{(2)}, v^{(2)}, 1)^\top$. These equations specify that the two end points of the line $\boldsymbol{\lambda}$ lie on the transferred line.

The left-hand sides of these equations are of the form $(u, v, 1)^\top \mathbf{l}\hat{ambda}$ where $\mathbf{l}\hat{ambda}$ is the transferred line $\lambda'^\top T_1 \boldsymbol{\lambda}'', \boldsymbol{\lambda}'^\top T_2 \boldsymbol{\lambda}'', \boldsymbol{\lambda}'^\top T_3 \boldsymbol{\lambda}'')^\top$. This quantity does not measure

precisely the distance from the transferred line to the line endpoint $(u, v, 1)^\top$, since $\hat{\boldsymbol{\lambda}}$ will not in general be normalized, and the normalizing factor in the denominator of (1) is missing. This is the price we pay to have a linear algorithm. However, we will come back to discuss this point in section 8.

6 Transferring Points.

Suppose that a point $\mathbf{x}$ in space is seen in three images, and that the three cameras are given in the normalized form $M = (I \mid 0)$, $M' = (A \mid \mathbf{a}_4)$ and $M'' = (B \mid \mathbf{b}_4)$.

We suppose that the point $\mathbf{x}$ is seen at positions $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$ in the three images, where $\mathbf{u}$ (and similarly $\mathbf{u}'$ and $\mathbf{u}''$) is a 3-vector $\mathbf{u} = (u, v, w)$, the representation of the point in homogeneous coordinates. The coordinates $(u/w, v/w)$ are the coordinates actually seen in the image. We wish to find a relationship between the coordinates of the points $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$. At any point in the following derivation, we may set w, w' or w'' to 1 to obtain equations relating to measured image coordinates.

Because of the form of the matrix $M = (I \mid 0)$, it is extremely simple to give a formula for the position of the point in space. In particular, since $(I \mid 0)\mathbf{x} \approx \mathbf{u}$, we may write $\mathbf{x} = \begin{pmatrix} \mathbf{u} \\ t \end{pmatrix}$ for some t, yet to be determined. It may be verified that t is the same as the "relative affine invariant", k, considered by Shashua ([13]). Now, projecting this point into the second image, we see that

$$\mathbf{u}' \approx M'\mathbf{x} = (A \mid \mathbf{a}_4)\begin{pmatrix} \mathbf{u} \\ t \end{pmatrix}$$

Denoting the i-th row of A by $\hat{\mathbf{a}}_i{}^\top$, we may write this equation as three separate equations

$$\begin{array}{rcl} u' & = & k\hat{\mathbf{a}}_1{}^\top\mathbf{u} + a_{14}t \\ v' & = & k\hat{\mathbf{a}}_2{}^\top\mathbf{u} + a_{24}t \\ w' & = & k\hat{\mathbf{a}}_3{}^\top\mathbf{u} + a_{34}t \end{array} \tag{7}$$

where k is an unknown scale factor. Eliminating k in different ways, we may obtain three equations

$$\begin{array}{rcl} u'(\hat{\mathbf{a}}_3{}^\top\mathbf{u} + a_{34}t) & = & w'\hat{\mathbf{a}}_1{}^\top\mathbf{u} + a_{14}t \\ v'(\hat{\mathbf{a}}_1{}^\top\mathbf{u} + a_{14}t) & = & u'\hat{\mathbf{a}}_2{}^\top\mathbf{u} + a_{24}t \\ w'(\hat{\mathbf{a}}_2{}^\top\mathbf{u} + a_{24}t) & = & v'\hat{\mathbf{a}}_3{}^\top\mathbf{u} + a_{34}t \end{array} \tag{8}$$

Of these three equations, only two are independent. From each of these equations independently, one may compute the value of t. We obtain

$$\begin{array}{rcl} t & = & \dfrac{w'\hat{\mathbf{a}}_1{}^\top\mathbf{u} - u'\hat{\mathbf{a}}_3{}^\top\mathbf{u}}{u'a_{34} - w'a_{14}} \\ & = & \dfrac{u'\hat{\mathbf{a}}_2{}^\top\mathbf{u} - v'\hat{\mathbf{a}}_1{}^\top\mathbf{u}}{v'a_{14} - u'a_{24}} \end{array}$$

$$= \frac{v'\hat{\mathbf{a}}_3{}^\top\mathbf{u} - w'\hat{\mathbf{a}}_2{}^\top\mathbf{u}}{w'a_{24} - v'a_{34}} \tag{9}$$

Considering only the first of these expressions for t, we see that the point $\mathbf{x}$ may be written as

$$\begin{aligned}\mathbf{x} &= \begin{pmatrix} \mathbf{u} \\ (w'\hat{\mathbf{a}}_1{}^\top\mathbf{u} - u'\hat{\mathbf{a}}_3{}^\top\mathbf{u})/(u'a_{34} - w'a_{14}) \end{pmatrix} \\ &\approx \begin{pmatrix} (u'a_{34} - w'a_{14})\mathbf{u} \\ w'\hat{\mathbf{a}}_1{}^\top\mathbf{u} - u'\hat{\mathbf{a}}_3{}^\top\mathbf{u} \end{pmatrix}\end{aligned}$$

Now, projecting this point via the third camera, we find that

$$\begin{aligned}\mathbf{u}'' &\approx (B \mid \mathbf{b}_4)\mathbf{x} \\ &\approx (u'a_{34} - w'a_{14})B\mathbf{u} + \mathbf{b}_4(w'\hat{\mathbf{a}}_1{}^\top\mathbf{u} - u'\hat{\mathbf{a}}_3{}^\top\mathbf{u}) \\ &\approx \left((u'a_{34} - w'a_{14})B + \mathbf{b}_4(w'\hat{\mathbf{a}}_1{}^\top - u'\hat{\mathbf{a}}_3{}^\top)\right)\mathbf{u}\end{aligned}$$

Writing this in terms of the rows of B (denoted by $\hat{\mathbf{b}}_i{}^\top$), we have an expression

$$\mathbf{u}'' \approx \begin{pmatrix} (u'a_{34} - w'a_{14})\hat{\mathbf{b}}_1{}^\top + b_{14}(w'\hat{\mathbf{a}}_1{}^\top - u'\hat{\mathbf{a}}_3{}^\top) \\ (u'a_{34} - w'a_{14})\hat{\mathbf{b}}_2{}^\top + b_{24}(w'\hat{\mathbf{a}}_1{}^\top - u'\hat{\mathbf{a}}_3{}^\top) \\ (u'a_{34} - w'a_{14})\hat{\mathbf{b}}_3{}^\top + b_{34}(w'\hat{\mathbf{a}}_1{}^\top - u'\hat{\mathbf{a}}_3{}^\top) \end{pmatrix}\mathbf{u} \ .$$

We may rearrange terms to obtain

$$\mathbf{u}'' \approx \begin{pmatrix} u'(a_{34}\hat{\mathbf{b}}_1{}^\top - b_{14}\hat{\mathbf{a}}_3{}^\top) - w'(a_{14}\hat{\mathbf{b}}_1{}^\top - b_{14}\hat{\mathbf{a}}_1{}^\top) \\ u'(a_{34}\hat{\mathbf{b}}_2{}^\top - b_{24}\hat{\mathbf{a}}_3{}^\top) - w'(a_{14}\hat{\mathbf{b}}_2{}^\top - b_{24}\hat{\mathbf{a}}_1{}^\top) \\ u'(a_{34}\hat{\mathbf{b}}_3{}^\top - b_{34}\hat{\mathbf{a}}_3{}^\top) - w'(a_{14}\hat{\mathbf{b}}_3{}^\top - b_{34}\hat{\mathbf{a}}_1{}^\top) \end{pmatrix}\mathbf{u} \tag{10}$$

Following Shashua ([13]) we now write

$$\boldsymbol{\alpha}_{ij}{}^\top = a_{i4}\hat{\mathbf{b}}_j{}^\top - b_{j4}\hat{\mathbf{a}}_i{}^\top \tag{11}$$

Then, the equations (10) may be written as

$$\mathbf{u}'' \approx \begin{pmatrix} u'\boldsymbol{\alpha}_{31}{}^\top - w'\boldsymbol{\alpha}_{11}{}^\top \\ u'\boldsymbol{\alpha}_{32}{}^\top - w'\boldsymbol{\alpha}_{12}{}^\top \\ u'\boldsymbol{\alpha}_{33}{}^\top - w'\boldsymbol{\alpha}_{13}{}^\top \end{pmatrix}\mathbf{u} \tag{12}$$

Starting with the other two formulae for t given in (9) we obtain two other similar expressions for $\mathbf{u}''$. These can all be put together to give the set of equations :

$$\mathbf{u}'' \approx \begin{pmatrix} u'\boldsymbol{\alpha}_{31}{}^\top - w'\boldsymbol{\alpha}_{11}{}^\top \\ u'\boldsymbol{\alpha}_{32}{}^\top - w'\boldsymbol{\alpha}_{12}{}^\top \\ u'\boldsymbol{\alpha}_{33}{}^\top - w'\boldsymbol{\alpha}_{13}{}^\top \end{pmatrix}\mathbf{u} \approx \begin{pmatrix} v'\boldsymbol{\alpha}_{11}{}^\top - u'\boldsymbol{\alpha}_{21}{}^\top \\ v'\boldsymbol{\alpha}_{12}{}^\top - u'\boldsymbol{\alpha}_{22}{}^\top \\ v'\boldsymbol{\alpha}_{13}{}^\top - u'\boldsymbol{\alpha}_{23}{}^\top \end{pmatrix}\mathbf{u} \approx \begin{pmatrix} w'\boldsymbol{\alpha}_{21}{}^\top - v'\boldsymbol{\alpha}_{31}{}^\top \\ w'\boldsymbol{\alpha}_{22}{}^\top - v'\boldsymbol{\alpha}_{32}{}^\top \\ w'\boldsymbol{\alpha}_{23}{}^\top - v'\boldsymbol{\alpha}_{33}{}^\top \end{pmatrix}\mathbf{u} \ . \tag{13}$$

Once more, only two of these equations are independent. Just as before we may eliminate the unknown scale factor implied by the $\approx$ sign. In this way, from each expression for $\mathbf{u}''$ an we obtain three equations, of which only two are independent. This gives for each correspondence $\mathbf{u} \leftrightarrow \mathbf{u}' \leftrightarrow \mathbf{u}''$ a set of 9 equations in the unknown entries of the vectors α_{ij}. These are the trilinearity relationships of Shashu. Of these equations, only 4 are independent. Those equations involving both w' and w'' form an independent set, as follows :

$$\begin{array}{rcl} u''(u'\boldsymbol{\alpha}_{33}{}^{\top} - w'\boldsymbol{\alpha}_{13}{}^{\top})\mathbf{u} & = & w''(u'\boldsymbol{\alpha}_{31}{}^{\top} - w'\boldsymbol{\alpha}_{11}{}^{\top})\mathbf{u} \\ v''(u'\boldsymbol{\alpha}_{33}{}^{\top} - w'\boldsymbol{\alpha}_{13}{}^{\top})\mathbf{u} & = & w''(u'\boldsymbol{\alpha}_{32}{}^{\top} - w'\boldsymbol{\alpha}_{12}{}^{\top})\mathbf{u} \\ u''(v'\boldsymbol{\alpha}_{33}{}^{\top} - w'\boldsymbol{\alpha}_{23}{}^{\top})\mathbf{u} & = & w''(v'\boldsymbol{\alpha}_{31}{}^{\top} - w'\boldsymbol{\alpha}_{21}{}^{\top} \\ v''(v'\boldsymbol{\alpha}_{33}{}^{\top} - w'\boldsymbol{\alpha}_{23}{}^{\top})\mathbf{u} & = & w''(v'\boldsymbol{\alpha}_{32}{}^{\top} - w'\boldsymbol{\alpha}_{22}{}^{\top})\mathbf{u} \ . \end{array} \tag{14}$$

As stated previously, we may set w, w' and w' to 1 to obtain a relationship between observed image coordinates.

To be able to write the general form of the trilinearity equation, we write (u_1, u_2, u_3) instead of (u, v, w), and make the same notational change for the primed and doubly primed quantities. Furthermore, we denote the k-th entry of $\boldsymbol{\alpha}_{ij}$ by α_{ijk}. Then the general trilinearity relationship obtained from (13) may be written in a simple form :

$$\sum_{h=1}^{3}(u'_j u''_i \alpha_{klh} + u'_k u''_l \alpha_{jih})u_h = \sum_{h=1}^{3}(u'_k u''_i \alpha_{jlh} + u'_j u''_l \alpha_{kih})u_h \tag{15}$$

where i, j, k and l range over all indices such that $i \neq l$ and $j \neq k$. Since we get the same relation by interchanging i and l, or j and k, we may assume that $i < l$ and $j < k$. There are therefore 9 different equations defined by this expression.

Given 7 point correspondences, we have 28 equations, which is enough to solve for the vectors α_{ij}. Thus, we may obtain the transfer equations (13) linearly from a set of 7 point matches in all three images. Shashua states that better results are obtained by including 6 equations for each point match. Possibly the best results are obtained by including all 9 equations, but this has not been tested.

7 Connection with Line Equations

The fact that we have 27 unknown values in the three matrices T_i as well as in the 9 vectors $\boldsymbol{\alpha}_{ij}$ suggests that there could be a connection between these techniques. This is indeed the case as will now be demonstrated. We denote the k-th entry of the vector $\boldsymbol{\alpha}_{ij}$ by α_{ijk}. Similarly, we denote the (jk)-th entry of the matrix T_i by T_{ijk}. The entries of matrices A and B will be denoted by a_{ij} and b_{ij}.

Now, from (11) we have

$$\alpha_{ijk} = a_{i4}b_{jk} - b_{j4}a_{ik} \ . \tag{16}$$

Further, from (4) we have

$$T_{ijk} = a_{ji}b_{k4} - a_{j4}b_{ki} \tag{17}$$

One immediately sees that

$$\alpha_{ijk} = -T_{kij} \tag{18}$$

This equation has the significant implication that we may amalgamate the line and point algorithms into one algorithm. In particular, each line correspondence $\boldsymbol{\lambda} \leftrightarrow \boldsymbol{\lambda}' \leftrightarrow \boldsymbol{\lambda}''$ gives two linear equations in the entries T_{ijk}, whereas each point correspondence gives four linear equations in the entries α_{ijk} which are the same as the T_{ijk} except for the order of the indices. Therefore, provided that $2\#lines+4\#points \geq 26$ we have sufficiently many matches to solve for the T_{ijk}, and hence to carry out a projective scene reconstruction, as seen in Sections 9 and 10.

8 Iterative Linear Method

In finding a least-squares solution to a redundant set of equations of the form (eq:line-equations) and (eq:4-point-equations) linearly it is clear that we are not minimizing precisely what we would want to minimize, namely the distances between the transferred points, or lines and the measured ones.

Consider first of all the line transfer equation (6). The expression $(u,v,1)(\boldsymbol{\lambda}'^{\top}T_1\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_2\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_3\boldsymbol{\lambda}'')^{\top}$ is not equal to the distance between the point $(u,v,1)^{\top}$ and the transferred line $(\lambda,\mu,\nu) = (\boldsymbol{\lambda}'^{\top}T_1\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_2\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_3\boldsymbol{\lambda}'')^{\top}$. To get the correct distance, we need to divide by $sqrt\lambda^2+\mu^2$ as indicated by equation (1). This shows that to express the true distance between the point $(u,v,1)^{\top}$ and the transferred line $(\lambda,\mu,\nu) = (\boldsymbol{\lambda}'^{\top}T_1\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_2\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_3\boldsymbol{\lambda}'')$ we need to write an equation

$$(u,v,1)^{\top}(\lambda,\mu,\nu)/\sqrt{\lambda^2+\mu^2} \ .$$

Unfortunately, the resulting equation is not linear in the entries of the T_i. To overcome this problem, we adopt a strategy of adaptive weights. Let $W_0 = 1$. We construct and solve (see section sec:notation the set of linear equations (6), and any point equations of the form (refeq:4-point-equations) as well to get a solution for T_i. From this, we may compute the transferred line $(\lambda,\mu,\nu)^{\top} = (\boldsymbol{\lambda}'^{\top}T_1\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_2\boldsymbol{\lambda}'', \boldsymbol{\lambda}'^{\top}T_3\boldsymbol{\lambda}'')^{\top}$. From this we compute a weight

$$W_1 = 1/\sqrt{\lambda^2+\mu^2} = 1/sqrt(\boldsymbol{\lambda}'^{\top}T_1\boldsymbol{\lambda}'')^2 + (\boldsymbol{\lambda}'^{\top}T_2\boldsymbol{\lambda}'')^2 \ . \tag{19}$$

Now, we multiply each of the equations (6) by the weight W_1 and solve again. This process may be repeated several times, at each iteration computing the new weight W_i computed from the previous solution. Finally the values of the weights will converge (one hopes), and the residual error will indeed equal the distance from the point $(u,v,1)^{\top}$ to the transferred line. The least squares error will be equal to the sum of the squares of the errors as desired.

Of course, one should simultaneously do a similar thing with the point equations (14). In particular the residual error in the first of these equations is $u''(u'\boldsymbol{\alpha}_{33}{}^{\top} - \boldsymbol{\alpha}_{13}{}^{\top})\mathbf{u} - (u'\boldsymbol{\alpha}_{31}{}^{\top} - \boldsymbol{\alpha}_{11}{}^{\top})\mathbf{u}$. On the other hand, the difference between the measured image coordinate u'' and the coordinate of the transferred point is $u'' - (u'\boldsymbol{\alpha}_{31}{}^{\top} - \boldsymbol{\alpha}_{11}{}^{\top})\mathbf{u}/(u'\boldsymbol{\alpha}_{33}{}^{\top} - \boldsymbol{\alpha}_{13}{}^{\top})\mathbf{u}$. To get the correctly weighted error, we need to multiply the equation $u''(u'\boldsymbol{\alpha}_{33}{}^{\top} - \boldsymbol{\alpha}_{13}{}^{\top})\mathbf{u} - (u'\boldsymbol{\alpha}_{31}{}^{\top} - \boldsymbol{\alpha}_{11}{}^{\top})\mathbf{u}$ by a weight

$$W = 1/(u'\boldsymbol{\alpha}_{33}{}^{\top} - \boldsymbol{\alpha}_{13}{}^{\top})\mathbf{u} \ . \tag{20}$$

The other equations are to be multiplied by weights of a similar form. As before, this would result in non-linear equations. Consequently we again use the strategy of adaptive weights.

The adaptive weighting of the line and point equations should be carried out simultaneously. One particular result of doing this is that the line and point equations are assigned their correct weights relative to each other. Otherwise, there is no *a priori* method of correctly weighting the point and line equations relative to each other.

This iterative proceedure has several advantages compared with a full least-squares minimization scheme such as Levenberg-Marquardt. For instance, it is very simple to program once one has the linear method coded. Further, one does not have to worry about an initial estimate, since this method is self-initializing. Finally, there is no difficulty with a stopping criterion – one simply continues until the change in weights from iteration to iteration is small. The solution of the set of equations is quite insensitive to small variations in the weights. The stopping criterion I have used is to continue as long as $\sum_j (|W_j^i / W_j^{i-1}| - 1)^2 < 10^{-8}$, where W_j^i is the i-th measurement of the j-th weight. Convergence is usually rapid and reliable. An alternative is to iterate for a fixed small number of iterations (such as 3).

9 Retrieving the Camera Matrices

Formula (4) 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 for now 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.

Now, note that $(\mathbf{a}_4 \times \mathbf{a}_i)^\top T_i = 0$ since $(\mathbf{a}_4 \times \mathbf{a}_i)^\top \mathbf{a}_i = (\mathbf{a}_4 \times \mathbf{a}_i)^\top \mathbf{a}_4 = 0$. It follows that we can compute $\mathbf{a}_4 \times \mathbf{a}_i$ up to an unknown multiplicative factor by finding the null-space of T_i for each $i = 1, \ldots, 3$. Since the camera matrix $M' = (A \mid \mathbf{a}_4) = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4)$ has rank 3, the set of vectors $\{\mathbf{a}_4 \times \mathbf{a}_i : i = 1 \ldots 3\}$ has rank 2. Consequently, $\mathbf{a}_4$ (or $-\mathbf{a}_4$) may be computed as the unique unit vector normal to all of $\mathbf{a}_4 \times \mathbf{a}_i$ for $i = 1, \ldots, 3$. The vector $\mathbf{b}_4$ may be computed in the same way.

The vectors $\mathbf{a}_4$ and $\mathbf{b}_4$ were found as the common perpendicular to the null-spaces of the three transfer matrices. In some cases, one or more of the T_i will have rank 1, in which case the null-space will be 2-dimensional. This can occur when (for instance) $\mathbf{a}_4 \approx \mathbf{a}_i$. To find $\mathbf{a}_4$, it is not really necessary to assume that all the T_i have rank 2. We can make do easily if two of the matrices have rank 2, for we need only two vectors to find a common perpendicular. Even if all three matrices T_i have rank one (which is probably not possible), one can still find $\mathbf{a}_4$ as the common perpendicular to the three 2-dimensional null-spaces. See [16] for the a further discussion of methods applying to calibrated cameras for the case where the rank of T_i is less than 2.

As noted previously, $\mathbf{a}_4$ and $\mathbf{b}_4$ are the homogeneous coordinates of the epipoles in the second and third images corresponding to the centre of the first camera. Once $\mathbf{a}_4$ and $\mathbf{b}_4$ are known there are several ways to compute the matrices A and B to reconstruct the complete camera matrices. We continue by considering three such methods.

The linear method If $\mathbf{a}_4$ and $\mathbf{b}_4$ are known, the equations (4) form a redundant set of linear equations in the entries of A and B. We may solve these equations using linear least-squares methods ([11, **?**]) to find A and B, and hence $M' = (A \mid \mathbf{a}_4)$ and $M'' = (B \mid \mathbf{b}_4)$.

The recomputation method. An alternative method is to solve for the entries of A and B by going back to equations (2) and (13) and solving them again now that $\mathbf{a}_4$ and $\mathbf{b}_4$ are known. These equations can be written in terms of a_{ij} and b_{ij} by substituting using equations (16) and (17). The resulting equations will be linear in terms of the entries of A and B, and may be solved by the eigenvalue method (see section 3). In solving these equations, one may again use the method of adaptive weights. The initial weights chosen should ideally be the final weigths used for the initial solution for the T_i.

The Closed Form Method. In [6] closed form formulas were given for M' and M'', namely

$$\begin{array}{ll} M' & = (A \mid \mathbf{a}_4) \\ & = (XT_1\mathbf{b}_4, XT_2\mathbf{b}_4, XT_3\mathbf{b}_4, \mathbf{a}_4) \\ M'' & = (B \mid \mathbf{b}_4) \\ & = (-{T_1}^\top\mathbf{a}_4, -{T_2}^\top\mathbf{a}_4, -{T_3}^\top\mathbf{a}_4, \mathbf{b}_4) \end{array} \tag{21}$$

where $X = I - \mathbf{a}_4{\mathbf{a}_4}^\top$.

This method of determining M' and M'' was coded and evaluated in [6]. By carefully examining the results of that method, it has subsequently been found that using these formulae to determine M' and M'' is very unstable in the presence of noise, and hence is **not** recommended. Solving directly for A and B is a far more stable method, and the results of this paper have been found by using this method.

10 Reconstruction

Once the camera matrices are computed, it is a simple task to compute the positions of the points and lines in space. Different ways in which this may be done for points are described in [**?**].

For lines, the line in space corresponding to a set of matched lines $\boldsymbol{\lambda} \leftrightarrow \boldsymbol{\lambda}' \leftrightarrow \boldsymbol{\lambda}''$ 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 $X = ({M}^\top\boldsymbol{\lambda}, {M'}^\top\boldsymbol{\lambda}', {M''}^\top\boldsymbol{\lambda}'')$. 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 ([11]) 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.

11 Iterative Methods.

In the presence of noise, the matrices T_i obtained by solving equations (6) will not have the correct form as given in (4). This means that the epipoles $\mathbf{a}_4$ and $\mathbf{b}_4$ may not be extremely accurate. Furthermore, solving equations (6) and (14) even using the method of iterative weighting does not correspond exactly to minimizing the cost function defined by the sum of squares of distances between predicted and measured image coordinates.

Therefore, to obtain an optimal solution, a least-squares minimization technique may be used, in which the initial solution is provided by the linear methods described so far. I have implemented such a method using the Levenberg-Marquardt (LM) method for carrying out the least-squares optimization.

11.1 Converging on the optimal solution - Lines

A method that gains in simplicity at the cost of a not-quite optimal result is described first. The varying parameters are the 24 entries of the matrices M' and M'' and the quantity to be minimized is the sum of squares of Euclidean distances of the transferred lines or points to the measured lines or points. We assume that the lines $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$ occuring in the equations (6) are accurate, as are the points $\mathbf{u}$ and $\mathbf{u}'$ occuring in (14). As the entries of the matrices M' and M'' are made to vary, the matrices T_i are computed using (4) and the transferred lines $\boldsymbol{\lambda}$ and points $\mathbf{u}''$ are computed using (3) and (13. Finally, the differences between these transferred features and the measured lines $\boldsymbol{\lambda}$ and points $\mathbf{u}''$ are measured. It is the sum of squares of these errors that is to be minimized. Of course, in the case of lines, the measured error is the distance of the transferred line from the defining end points of the measured line.

Typically convergence occurs within 10 iterations. Furthermore, each iteration is very fast, since construction of the normal equations ([11]) 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 total time required for reconstruction of 20 lines in three views is not more than 5 seconds on a Sparc 2.

11.2 The Optimal Least Squares 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. To find the true optimal solution, we proceed differently. In this case, apart from the 24 parameters of the two camera matrices, M' and M'', we need parameters representing each of the 3D lines and points in the scene. This is a fairly standard least-squares minimization problem in which the parameters of the cameras, as well as the parameters used to represent the features (lines and points) are varied. The features are then projected into each view, and the errors compared with the measured image feature locations are found. The goal function to be minimized is the sum of squares of the errors.

The only novel point about our implementation of this is the manner of representing lines in space. Each line is parametrized by its image coordinates $\hat{\boldsymbol{\lambda}}'$ and $\hat{\boldsymbol{\lambda}}''$ in two images. In this way, we do not have to worry about how to project the line into these two images – the projections are $\hat{\boldsymbol{\lambda}}'$ and $\hat{\boldsymbol{\lambda}}''$ respectively. The projection $\hat{\boldsymbol{\lambda}}$ of the line in the other image is computed using (2). The error to be measured is the distance between the projected lines $\hat{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{\lambda}}'$ and $\hat{\boldsymbol{\lambda}}''$ and the measured lines $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$. More exactly we measure the distance from the projected lines to the endpoints of the measured lines.

The parametric representation for each point is as a homogeneous 4-vector representing the position of the image in space. In this way, the standard pinhole-camera mapping is

used to compute the projection of the point in each image.

The disadvantage of this complete minimization approach 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 [14, 5]. Carrying out this final iteration to obtain a true optimal solution gives minimal gain over the method described in the previous section.

12 Algorithm Outline

To tie together all the threads of the reconstruction algorithm, an outline will now be given. As input to this algorithm, we assume as set of point-to-point image correspondences $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i$, and a set of line correspondences $\boldsymbol{\lambda}_i \leftrightarrow \boldsymbol{\lambda}'_i \leftrightarrow \boldsymbol{\lambda}''_i$, where $\#$ lines $+ 2 * \#$ points ≥ 13. The lines are assumed to be specified by giving the two end points of each line. The steps of the algorithm are as follows.

1. **Coordinate scaling and translation.** For each image separately, translate and scale the coordinates of the points such that the centroid of all given coordinates is at the origin, and the coordinates lie in a range [-1.0, 1.0].

2. **Computing and normalizing the lines.** The lines $\boldsymbol{\lambda}'_i$ and $\boldsymbol{\lambda}''_i$ are computed from their enpoints, and normalized.

3. **Construct the equations.** For each line correspondence, construct a pair of equations of the form (5) in the the entries of the matrices T_i. Similarly, for each point correspondence, construct four equations of the form (14) also in the entries of matrices T_i. The relationship between the T_i and the α_{ij} is given by (18).

4. **Initial solution.** Solve the set of equations to find an estimate of the matrices T_i.

5. **Computing the weights.** Compute a set of weights to apply to each equation. The weights are of the form (19) or (20).

6. **Iterative weight adjustment.** Weight the set of equations, and solve again. Iterate on these last two steps until convergence, or for a small number of iterations (at most 5).

7. **Computation of the epipoles.** Find $\mathbf{a}_4$ and $\mathbf{b}_4$ as the common normal to the left (respectively, right) null spaces of the three matrices T_i.

8. **Computation of the Camera Matrices.** Now that $\mathbf{b}_4$ and $\mathbf{a}_4$ are known, the equations (5) and (14) become linear in the remaining entries of the camera matrices, taking into account the formula for T_i given by (4). These linear equations may be solved to find the camera matrices (the recomputation method). Alternatively, we may simply solve for the camera matrix entries using (4) directly (linear method).

9. **Reconstruction.** Given the camera matrices, the points may be reconstructed using the triangulation methods of [7]. The lines may be reconstructed as described in section 10.

Figure 1: *Three photos of houses*

10. **Iterative Refinement.** A full iterative least-squares refinement of the computed solution may be carried out.

11. **Unscaling** The effect of the initial scaling and translation of the images may be undone by replacing the image coordinates by their original values, and making a corresponding adjustment to the computed camera matrices.

13 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 [4]. 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' and M'' corresponding to the centre of projection of camera M are simply the last columns of M' and M'' 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 1. The columns of this table have the following meanings.

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.*

- 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.

14 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.

This equation has a number of significant implications.

1. Given a set of 7 point matches in 3 images, one has a linear method of computing the projective structure of the scene, including the three camera matrices and the fundamental matrices. One uses the 7-point linear method to compute the values α_{ijk} and from this one reassembles the matrices T_i. Finally, the methods of Section 9 may be used to compute the camera matrices.

2. We may compute (linearly) the projective structure of a set of lines and points seen in three views, provided that $2\#lines + 4\#points \geq 26$. In particular, for each point, equation (13) gives 4 equations in the values α_{ijk}, and for each line, equation (5) gives two equations in the entries T_{ijk}. However, the unknowns T_{ijk} and α_{ijk} are the same (modulo a change of sign and permutation of the indices). Thus, provided we have at least 26 equations, we may compute the values of T_{ijk} linearly and then reconstruct the scene as in Sections 9 and 10.

It is to be expected that (as with reconstruction from points [4]) the robustness of the reconstruction will increase substantially with more than the minimum number of views. This situation arises when an object is tracked through several frames by a video camera.

The work of [4] 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.

Appendix : Lines specified by several points

In this paper, we have considered the case where lines are specified by their two end-points. Another common way that lines may be specified in an image is as the best line fit to several points. In this appendix it will be shown how that case may easily be reduced to the case of a line defined by two end points. Consider a set of points $\mathbf{u}_i$ in an image, and let $\boldsymbol{\lambda} = (\lambda, \mu, \nu)^\top$ be a line, which we suppose is normalized such that $\lambda^2 + \mu^2 = 1$. In this case, the distance from a point $\mathbf{u}_i$ to the line $\boldsymbol{\lambda}$ is equal to ${\mathbf{u}_i}^\top \boldsymbol{\lambda}$. The squared distance may be written as $d^2 = \lambda^\top \mathbf{u}_i {\mathbf{u}_i}^\top \boldsymbol{\lambda}$, and the sum-of-squares of all distances is

$$\sum_i \lambda^\top \mathbf{u}_i {\mathbf{u}_i}^\top \boldsymbol{\lambda} = \lambda^\top (\sum_i \mathbf{u}_i {\mathbf{u}_i}^\top) \boldsymbol{\lambda} \ .$$

The matrix $E = (\sum_i \mathbf{u}_i {\mathbf{u}_i}^\top)$ is positive-definite and symmetric.

As in the standard method for line-fitting, we may compute the line that minimizes the sum-of-squares distance $\boldsymbol{\lambda}^\top E \boldsymbol{\lambda}$ as follows. The line in question will be the one that minimizes $\boldsymbol{\lambda}^\top E \boldsymbol{\lambda}$ subject to the condition $\lambda^2 + \mu^2 = 1$. Using the method of Lagrange multipliers, this comes down to minimizing $\boldsymbol{\lambda}^\top E \boldsymbol{\lambda} - d(\lambda^2 + \mu^2)$, where (since the conventional symbol λ is already in use) we denote the Lagrange coefficient by d. Now, taking the derivative with respect to $\boldsymbol{\lambda}$ and setting it to zero, we find that

$$2E\boldsymbol{\lambda} - d \begin{pmatrix} 2\lambda \\ 2\mu \\ 0 \end{pmatrix} = 0 \ .$$

This may be written as $(E - dJ)\boldsymbol{\lambda} = 0$. The minimizing solution is the line $\boldsymbol{\lambda}$ corresponding to the minimum root of the equation $\det(E - dJ) = 0$. Let this minimum root be d_0.

Now, $E - d_0 J$ being symmetric and positive semi-definite may be written in the form $E - d_0 J = V \mathrm{diag}(r, s, 0) V^\top$ where V is an orthogonal matrix and a and b are positive. It follows that

$$\begin{aligned} E - d_0 J &= V \mathrm{diag}(r, 0, 0) V^\top + V \mathrm{diag}(s, 0, 0) V^\top \\ &= r \mathbf{v}_1 {\mathbf{v}_1}^\top + s \mathbf{v}_2 {\mathbf{v}_2}^\top \end{aligned}$$

where $\mathbf{v}_i$ is the i-th column of V. Therefore $E = d_0 J + r \mathbf{v}_1 {\mathbf{v}_1}^\top + s \mathbf{v}_2 {\mathbf{v}_2}^\top$. If (as we have assumed) $\lambda^2 + \mu^2 = 1$, then

$$\begin{aligned} \sum_i (\mathbf{u}_i \boldsymbol{\lambda})^2 &= \boldsymbol{\lambda}^\top E \boldsymbol{\lambda} \\ &= d_0 + (\sqrt{r} {\mathbf{v}_1}^\top \boldsymbol{\lambda})^2 + (\sqrt{s} {\mathbf{v}_2}^\top \boldsymbol{\lambda})^2 \ . \end{aligned}$$

Thus, we have replaced the sum-of-squares of several points by a constant value d_0, which is not capable of being minimized, plus the weighted sum-of-squares of the distances to two points $\mathbf{v}_1$ and $\mathbf{v}_2$.

In constructing the equations (6) for a line defined by a set of point, we use the pair of equations

$$\begin{aligned} \sqrt{r} {\mathbf{v}_1}^\top \boldsymbol{\lambda} &= 0 \\ \sqrt{s} {\mathbf{v}_2}^\top \boldsymbol{\lambda} &= 0 \end{aligned} \tag{22}$$

where λ defined by (5) is the transferred line. Similarly, in a full LM minimization, we may consider the line to be defined by the two weighted end points, $\sqrt{r}\mathbf{v}_1$ and $\sqrt{s}\mathbf{v}_2$.

References

[1] R. Hartley. Invariants of points seen in multiple images. unpublished report, May 1992.

[2] 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.

[3] R. I. Hartley. Camera calibration using line correspondences. In *Proc. DARPA Image Understanding Workshop*, pages 361–366, 1993.

[4] 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.

[5] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision : Proc. of the Second Joint European - US Workshop, Ponta Delgada, Azores – LNCS-Series Vol. 825, Springer Verlag*, pages 237–256, October 1993.

[6] Richard I. Hartley. Projective reconstruction from line correspondences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 903–907, 1994.

[7] Richard I. Hartley and Peter Sturm. Triangulation. In *Proc. ARPA Image Understanding Workshop*, pages 957–966, 1994.

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

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

[10] 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.

[11] 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.

[12] Amnon Shashua. Trilinearity in visual recognition by alignment. In *Computer Vision - ECCV '94, Volume I, LNCS-Series Vol. 800, Springer-Verlag*, pages 479–484, 1994.

[13] Amnon Shashua. Algebraic functions for recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):779–789, August 1995.

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

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

[16] 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.