A linear method for reconstruction from lines and points

Richard I. Hartley,
GE-Corporate Research and Development,
Schenectady, NY, 12309,
email : hartley@crd.ge.com

Abstract

This paper discusses the basic role of the trifocal tensor in scene reconstruction. 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 maybe computed by a linear algorithm from a set of 13 line correspondences in three views. It is further shown in this paper to be essentially identical to a set of coefficients introduced by Shashua to effect point transfer in the three view case. This observation means that the 13-line algorithm may be extended to allow for the computation of the trifocal tensor given any mixture of sufficiently many line and point correspondences. From the trifocal tensor the camera matrices of image may be computed, and the scene may be reconstructed. For unrelated uncalibrated cameras, this reconstruction will be unique up to projectivity. Thus, projective reconstruction of a set of lines and points may be reconstructed linearly from three views.

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. Most previous reconstruction algorithms have been specific to points ([11, 9, 10]) or lines ([16, 19]), 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 constraint 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.

For the case of calibrated cameras, an algorithm that handles both lines and points has been given by Spetsakis and Aloimonos ([17]). They explain the application of a set of three matrices, previously used for reconstruction from lines ([16]) to the point reconstruction problem. That paper contains the elements of ideas contained in the present paper, which extends the results to uncalibrated cameras. While being very relevant to the subject of the present paper, their paper also foreshadows Shashua's trilinearity relationships ([14]).

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 ([11]), for lines at least three views are necessary ([19]). On the other hand, lines have several advantages. Firstly, they can normally be determined more accurately than points, often with an accuracy of better than a tenth of a pixel. Secondly, line matches may be used in cases where occlusions occur. Often end points of lines are not visible in the images. On the other hand, if we are to use line matches, then at least three views must be used, since no information whatever about camera placements may be derived from any number of line-to-line correspondences in fewer than three views.

Outline. The key observation of this paper is the connection (see equation (6)) between Shashua's 3-view trilinearity relationships ([14]) and 7-point algorithm and an algorithm ([4, 7]) for projective reconstruction from lines. It immediately follows from that observation that one can amalgamate the two algorithms into one. This means that one can noniteratively carry out projective reconstruction from three views of 7 points, or 13 lines or anything in between (so to speak). Projective reconstruction from 7 lines in three views was lurking behind Shashua's work, but never explicit derived previously.

In order to derive this key observation, I rederive Shashua's trilinearity relationships and place them in the standard context of projection using camera matrices. My hope is that this rederivation has the merit of throwing further light on the meaning of those relationships.

I take the opportunity in section 8 to provide a better method of determining the camera matrices than the one previously published in [7]).

A somewhat longer version of this paper, giving

0-8186-7042-8/95 $4.00

details of experimental evaluation of the algorithm is given in [5].

2 The Trifocal Tensor

A basic tool in the analysis of this paper is an entity called the *trifocal tensor*. Since this entity has appeared previously in the literature in different guises, and it is appropriate to discuss its history. With hindsight, we may attribute the discovery of the trifocal tensor to Spetsakis and Aloimonis ([15, 16] who use it for scene reconstruction from lines in the case of calibrated cameras. It also appears in similar work on line-reconstruction by Weng, Huang and Ahuja ([19]). It was later shown by the present author in [4, 7] 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 ([18]) who continued the work on scene reconstruction from lines.

These three matrices were later shown to be applicable to reconstruction from points in the calibrated case by Spetsakis and Aloimonos ([17]). 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 ([14]). Subsequently ([13]) 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 ([14]) are exactly the same as the entries of the three 3×3 matrices of ([16, 19, 4]). The importance of this result is that it allows an amalgamation of the linear algorithms for points (Shashua [13]) and for lines ([7]). This results in an algorithm of significantly greater applicability and power than either of the line or point algorithms alone.

Whereas the line papers [19, 4, 7] 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. Using the usual tensor convention of summation over repeated indices results in a compact notation.

3 Notation and Basics

Vectors, sometimes denoted by bold lower-case letters such as $\mathbf{u}$ are singly-indexed ensembles of real numbers. The vector $\mathbf{u}$ (and in the same way other vectors) will more often be denoted as u_i, where i is an index running over the appropriate range (usually $1, \ldots, 3$ or $1, \ldots, 4$). In a similar manner, a notation such as a_{ij} denotes a doubly indexed quantity, namely a matrix. We will also be concerned with triply indexed quantities, such as T_{ijk}. We adopt the usual summation convention for tensors :

> Any repeated index in a product of vectors, matrices and tensors implies a summation over the range of index values. If the index range is not the same in both instances of the repeated index (as happens occasionally), then summation over the intersection of the two ranges is implied. Any formula involving indices is intended to hold for any choice of values of the free indices (which means those indices that are not repeated).

The three-dimensional space containing the scene will be considered to be the 3-dimensional projective space $\mathcal{P}^3$ and points in space will be represented by homogeneous 4-vectors $\mathbf{x}$. Similarly, image space will be regarded as the 2-dimensional projective space $\mathcal{P}^2$ and points in an image will be represented by homogeneous 3-vectors $\mathbf{u}$. Homogeneous quantities (vectors, matrices or tensors) that differ by a non-zero scale factor are considered to be equal. We use the symbol $\approx$ to indicate equality up to a constant non-zero scale.

The space-image mapping induced by a pinhole camera may be represented by a 3×4 matrix $M = (m_{ij})$ of rank 3, such that if $\mathbf{x}$ and $\mathbf{u}$ are corresponding object and image points then $\mathbf{u} \approx M\mathbf{x}$, or in tensor notation, $u_i \approx m_{ij} x_j$. Such a matrix will be called a camera matrix. One special camera matrix is denoted by $(I \mid 0)$, made up of a 3×3 identity matrix I and a final zero column.

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 $\lambda_i u_i = 0$.

Normal Form for Camera Matrices. Consider a set of lines and points in space viewed by several cameras. We use the word *feature* to represent either a line or a point. We suppose that the image coordinates of each feature as seen in each image are given, but the actual positions of the features in space are unknown. The task of projective reconstruction is to find a set of camera matrices M_j and 3D-lines and points so that each such 3D feature is indeed mapped to the given image feature in each of the images. If the camera matrices are allowed to be arbitrary, then it is well known ([2, 3]) 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 $HH^{-1} = I_{4\times4}$, it follows that $MH^{-1} = (I|0)$. Since M may be transformed to $(I \mid 0)$, by applying transformation H to the

reconstruction we may assume without loss of generality that $M = (I \mid 0)$.

To save the reader the trouble of having to count primes, we denote the entries of the camera matrices M' and M'' by a_{ij} and b_{ij} respectively, instead of by m'_{ij} and m''_{ij}. Thus, the three camera matrices M, M' and M'' may be written in the form $M = (I \mid 0)$, $M' = (a_{ij})$ and $M'' = (b_{ij})$

4 Transferring lines

Some of the main results of [4, 7] will now be summarized using the tensor notation of this paper. Given three cameras with matrices $M = (I \mid 0)$, $M' = (a_{ij})$ and $M'' = (b_{ij})$, one defines a $3 \times 3 \times 3$ tensor, T_{ijk} by the expression

$$T_{ijk} = a_{ji}b_{k4} - a_{j4}b_{ki} \ . \tag{1}$$

If $\boldsymbol{\lambda} \leftrightarrow \boldsymbol{\lambda}' \leftrightarrow \boldsymbol{\lambda}''$ are three corresponding lines in the three images, then

$$\lambda_i \approx \lambda'_j \lambda''_k T_{ijk} \tag{2}$$

Given T_{ijk} and the coordinates of matching lines $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$, this expression may be used to compute the line in the other image. On the other hand, if at least 13 line matches $\boldsymbol{\lambda} \leftrightarrow \boldsymbol{\lambda}' \leftrightarrow \boldsymbol{\lambda}''$ are known, it is possible to solve for the entries of the tensor T_{ijk}, since each line match provides two linear equations in the 27 unknown tensor entries. In particular, if the line $\boldsymbol{\lambda}$ is presented by specifying the two endpoints, then each endpoint $\mathbf{u} = (u_1, u_2, u_3)$ gives rise to an equation

$$u_i \lambda'_j \lambda''_k T_{ijk} = 0 \tag{3}$$

To normalize these equations, the line $\boldsymbol{\lambda}'$ (and similarly $\boldsymbol{\lambda}''$) should be scaled so that $\lambda'^2_1 + \lambda'^2_2 = 1$, and each end point $\mathbf{u}$ should be scaled so that $u_3 = 1$.

5 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 $M = (I \mid 0)$, $M' = (a_{ij})$ and $M'' = (b_{ij})$.

We suppose that the point $\mathbf{x}$ is seen at positions $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$ in the three images, where $\mathbf{u}$ (and similarly $\mathbf{u}'$ and $\mathbf{u}''$) is a 3-vector $\mathbf{u} = (u_1, u_2, u_3)$, the representation of the point in homogeneous coordinates. The coordinates $(u_1/u_3, u_2/u_3)$ are the coordinates actually seen in the image. We wish to find a relationship between the coordinates of the points $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$. At any point in the following derivation, we may set u_3, u'_3 or u''_3 to 1 to obtain equations relating to measured image coordinates.

Because of the form of the matrix $M = (I \mid 0)$, it is extremely simple to give a formula for the position of the point in space. In particular, since $(I \mid 0)\mathbf{x} \approx \mathbf{u}$, we may write $\mathbf{x} = \begin{pmatrix} \mathbf{u} \\ t \end{pmatrix}$ for some t, yet to be determined. It may be verified that t is the same as the "relative affine invariant", k, considered by Shashua ([14]). Now, projecting this point into the second image by the usual formula $u'_i \approx a_{ij}x_j$ gives

$$u'_i \approx a_{ik}u_k + a_{i4}t$$

The notation $\approx$ denotes equality up to an unknown scale factor. We may eliminate this scale factor to obtain equations

$$u'_i(a_{jk}u_k + a_{j4}t) = u'_j(a_{ik}u_k + a_{i4}t)$$

where each choice of the free indices i and j gives a separate equation. Of the three resulting equations, only two are independent. From each of these equations independently, one may compute the value of t. We obtain three separate estimates for t.

$$t = u_k(u'_i a_{jk} - u'_j a_{ik})/(u'_j a_{i4} - u'_i a_{j4}) \tag{4}$$

Substituting the value of t from (4) we see that the point $\mathbf{x}$ may be written as

$$\begin{aligned} \mathbf{x} &= \begin{pmatrix} \mathbf{u} \\ u_k(u'_i a_{jk} - u'_j a_{ik})/(u'_j a_{i4} - u'_i a_{j4}) \end{pmatrix} \\ &\approx \begin{pmatrix} (u'_j a_{i4} - u'_i a_{j4})\mathbf{u} \\ u_k(u'_i a_{jk} - u'_j a_{ik}) \end{pmatrix} \end{aligned}$$

Now, projecting this point via the third camera, $u''_l \approx b_{lk}x_k$ we find that

$$\begin{aligned} u''_l &\approx b_{lk}u_k(u'_j a_{i4} - u'_i a_{j4}) \\ &+ b_{l4}u_k(u'_i a_{jk} - u'_j a_{ik}) \\ &\approx u_k u'_i(a_{jk}b_{l4} - a_{j4}b_{lk}) \\ &- u_k u'_j(a_{ik}b_{l4} - a_{i4}b_{lk}) \end{aligned} \tag{5}$$

Now, referring to (1), we recognize the tensor coefficients T_{ijk} in this expression :

$$u''_l \approx u_k(u'_i T_{kjl} - u'_j T_{kil}) \tag{6}$$

As before we may eliminate the unknown scale factor implied by the $\approx$ sign to get (after some slight rearranging) the equations

$$\begin{aligned} &u_k(u'_i u''_m T_{kjl} - u'_j u''_m T_{kil}) = \\ &u_k(u'_i u''_l T_{kjm} - u'_j u''_l T_{kim}) \ . \end{aligned} \tag{7}$$

These are the trilinearity relationships of Shashua ([14]). In these equations, the indices i, j, l and m are free variables, and there is one equation for each choice of indices with $i \neq j$ and $l \neq m$. Since we get the same relation by interchanging i and j, or l and m, we may assume that $i < j$ and $l < m$. There are therefore 9 different equations defined by this expression. However, only two of the three choices of pair (i, j) given

independent equations, and the same is true for pairs (l, m). Hence, there are 4 independent equations.

One choice of the four independent equations is obtained by setting $j = m = 3$, and letting i and l range freely. As stated previously, we may set u_3, u'_3 and u''_3 to 1 to obtain a relationship between observed image coordinates. The equations then become

$$u_k(u'_i u''_l T_{k33} - u''_l T_{ki3} - u'_i T_{k3l} + T_{kil}) = 0 \ . \quad (8)$$

The four different choices of $i, l = 1, 2$ give four different equations in terms of the observed image coordinates.

Given 7 point correspondences, we have 28 equations, which is enough to solve for the tensor values T_{ijk}. Shashua states that better results are obtained by including 6 equations for each point match. The experiments reported in [5] use all 9 equations, but it is not clear how much advantage (if any) this gives.

6 Solving using lines and points.

We have seen that the entries of the trifocal tensor, T occur in equations involving both points (8) and lines (3). This 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. Therefore, provided that $2\#lines + 4\#points \geq 26$ we have sufficiently many matches to solve for the T_{ijk}. The tensor entries T_{ijk} are found up to a common scale, and we find a solution such that sum of squares of the entries is one. In the case where there are more than 26 equations, we find a least-squares solution satisfying this constraint.

The set of equations we construct are of the form, $A\mathbf{t} = \mathbf{0}$, where A is the equation matrix and $\mathbf{t}$ is a vector containing the elements of T_{ijk} to be found. We are not interested in the solution $\mathbf{t} = 0$, and to avoid ambiguity, we impose the constraint $||t|| = 1$. Since we do not expect an exact solution, our task is to minimize the quantity $||A\mathbf{t}||$ subject to this constraint. The solution to this problem is easily seen (using Langrange multipliers, for instance) to be the unit eigenvector corresponding to the least eigenvalue of $A^\top A$. Being symmetric and positive definite, $A^\top A$ has only real positive eigenvalues. A good algorithm for finding this eigenvector is the method of Jacobi ([12]).

Before setting out to write and solve the equations, it is a very good idea to normalize the data by scaling and translating the points. The algorithm does not do well if all points are of the form $(u_1, u_2, 1)^\top$ with u_1 and u_2 very much larger than 1. A heuristic that works well is to translate the points in each image so that the centroid of all measured points is at the origin of the image coordinates, and then scaling so that the average distance of a point from the origin is $\sqrt{2}$ units. Reasons for carrying out this normalization are given in [6].

7 Retrieving the Camera Matrices

Formula (1) gives a formula for the trifocal tensor T_{ijk} in terms of the camera matrices. It is possible to go the other way and retrieve the camera matrices, M' and M'' from the tensor T_{ijk}.

This is done in two stages. In the first stage, one finds the vectors a_{i4} and b_{i4} (that is, the last columns of M' and M''). As shown in [4, 7], vectors a_{i4} and b_{i4} may be found as the common perpendiculars to the left (respectively, right) null-spaces of the three matrices, T_{1ij}, T_{2ij} and T_{3ij}. In the second stage, one finds the remaining entries.

The direct method In [7] closed form formulas were given for the camera matrices in terms of T_{ijk}. However, this method is unstable in the presence of noise. An alternative method is to observer that if a_{i4} and b_{i4} and T_{ijk} are all known, the equations (1) form a redundant set of linear equations in the remaining entries of the matrices M' and M''. To be specific, one may write equations (1) in the form $\mathbf{t} = H\mathbf{y}$ where $\mathbf{y}$ is the vector of the entries a_{ij} and b_{ij} that we are seeking, and H is the linear relationship expressed by (1). We may solve these equations using linear least-squares methods ([12, 1]) to find $\mathbf{y}$ and hence $M' = (a_{ij})$ and $M'' = (b_{ij})$.

Unfortunately, this method does not seem to give significantly better results than the closed form solution, and is also **not** recommended. The reason that this method gives poor results is that the trifocal tensor, T_{ijk} contains small as well as large entries, differing by several orders of magnitude. The small entries are however important, and changes in the large entries are less significant than changes of equal magnitude in the small entries. Thus, a method that weights changes to all the entries of T_{ijk} equally (as the direct method does) gives bad results.

The recomputation method. The recommended method is to go right back to the equations (3) and (8) and substitute the formula (1) to get equations in terms of a_{ij} and b_{ij}. Now that a_{j4} and b_{j4} are known these equations are linear in the remaining a_{ij} and b_{ij}. In particular, the tensor T_{ijk} was found by solving a system of equations $A\mathbf{t} = 0$ where $\mathbf{t}$ was a vector comprising the desired elements of T_{ijk}. Substituting the relationship $\mathbf{t} = H\mathbf{y}$ derived from (1) we obtain equations $AH\mathbf{y} = 0$. Our task therefore is to minimize $||AH\mathbf{y}||$ subject to the constraint $||\mathbf{y}|| = 1$. The solution is the eigenvector corresponding to the least eigenvalue of $H^\top A^\top AH$. Note that the matrix $A^\top A$ was previously computed in order to find the tensor T_{ijm} by solving the equations $A\mathbf{t} = 0$, and so it need not be recomputed. This observation can save a lot of recomputation.

Unfortunately, there is one small problem which causes an instability in this method as just described. In particular, it may be verified that the values of a_{ij}

and b_{ij} are not uniquely determined by T_{ijk}. In particular, if a_{ij} is replaced by $a_{ij} + \alpha_j a_{i4}$ for any vector α_j, and similarly b_{ij} is replaced by $b_{ij} + \alpha_j b_{i4}$ for the same α_j, then the value of T_{ijk} defined by equation (1) is unchanged. This means that the matrix H described above is not of full rank, and consequently the matrix $H^\top A^\top A H$ will have a multidimensional eigenspace corresponding to the eigenvalue 0. This means that the solution found will be unstable. This may not be a significant problem, since any solution found by this method will be a valid solution, giving one solution for the camera matrices. Nevertheless, I prefer to constrain the solution by adding constraints on the entries a_{ij} and b_{ij} so as to ensure a stable solution.

One method of constraining a_{ij} is by specifying that $a_{ij}a_{i4} = 0$. This gives three constraints, one for each value of $j = 1, \ldots, 3$, which may be interpreted as meaning that 4-th column of $M' = (a_{ij})$ is orthogonal to all the other columns. One may verify that this condition is achieved by a suitable choice of the vector α_j in the last paragraph (in particular, setting $\alpha_j = -a_{ij}a_{i4}$). The condition $a_{ij}a_{i4} = 0$ gives a set of three linear constraints, which may be written in matrix form as a set of equations $C\mathbf{y} = 0$. The algorithm is now summarized:

Algorithm for computing camera-matrices from T_{ijk}.

1. Retain the matrix $A^\top A$ used to solve the set of equations $A\mathbf{t} = \mathbf{0}$ where $\mathbf{t}$ is the vector containing the entries of T_{ijk}
2. Compute the set of equations $\mathbf{t} = H\mathbf{y}$ from (1), where $\mathbf{y}$ is the vector of still unknown entries of a_{ij} and b_{ij}.
3. Compute the matrix C such that $C\mathbf{y} = 0$ expresses the three constraints $a_{ij}a_{i4} = 0$.
4. Find vector $\mathbf{y}$ that minimizes $\mathbf{y}^\top H^\top (A^\top A) H\mathbf{y}$ subject to constraints $\mathbf{y}^\top\mathbf{y} = 1$ and $C\mathbf{y} = 0$.

This minimization problem is a standard quadratic minimization problem with linear constraints. A detailed method of solution is given in [5].

Since this algorithm is more complicated than either of the two previous algorithms (closed-form solution, and direct linear solution) it is comforting to verify that it performs at least 10 times better (in terms of measured errors) than they do. Therefore, the two earlier algorithms may be consigned to the scrap heap.

8 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 [8]. For lines, a method was described in [4, 7]

9 Algorithm Outline

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

1. **Coordinate scaling and translation.** For each image separately, translate and scale the coordinates of the points such that the centroid of all given coordinates is at the origin, and the average distance of a point from the origin is $\sqrt{2}$.
2. **Computing and normalizing the lines.** Each line $\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}''$ is computed from its endpoints, and normalized.
3. **Construct the equations.** For each line correspondence, construct a pair of equations of the form (2) in the entries of the tensor T_{ijk}. Similarly, for each point correspondence, construct four equations of the form (8) also in the entries of tensor T_{ijk}.
4. **Finding the trifocal tensor.** Solve the set of equations to find an estimate of T_{ijk}.
5. **Computation of the epipoles.** Find a_{i4} and b_{i4} as the common normal to the left (respectively, right) null spaces of the three matrices T_{1ij}, T_{2ij} and T_{3ij}.
6. **Computation of the Camera Matrices.** Solve for the remaining entries a_{ij} and b_{ij} using the method given in section 7
7. **Reconstruction.** Given the camera matrices, the points may be reconstructed using the triangulation methods of [8]. The lines may be reconstructed as described in section 8.
8. **Unscaling** The effect of the initial scaling and translation of the images may be undone by replacing the image coordinates by their original values, and making a corresponding adjustment to the computed camera matrices.

10 Conclusions

Results of experimental verification and evaluation of this algorithm are given in [5]. The results obtained indicate that this algorithm behaves very well in the presence of moderate amounts of noise. The results seem to be more stable than those obtained for reconstruction from points seen in two views. This improvement may be attributed to the stabilizing influence of a third image. Shashua has made a similar observation ([13] and conversations). The advantage that this

algorithm has over two view algorithms is that it applies to lines as well. Lines in images may often be computed with great precision. On the other hand, the algorithm benefits from the presence of points in the input data. The results obtained using this algorithm are much better than those previously obtained with just lines. This is due to the use of mixed point and line data, and also to the improved algorithm for extracting the camera matrices, reported in section 7.

The trifocal tensor seems to be a basic object in the analysis of the three-view situation. Apart from the use here described in projective scene reconstruction, it has also been used for point transfer and object recognition ([14]), and is suitable for line transfer as well, as shown in this paper. Just as the fundamental matrix neatly encapsulates the geometry of the two-view case, the trifocal tensor serves a similar purpose for three views.

Acknowledgements

This paper draws from many sources and relies on the work of many other people. Section 5 is largely a reworking of the work of Shashua ([14]), except of course for the critical connection with the same tensor T_{ijk} defined for line transfer. I first heard of the idea of using tensor terminology in connection with the matrices from Vieville ([18]) and subsequently convinced myself of its merit. The idea that there may be a connection between my work on line transfer in three views and Shashua's work with points in three views first arose during conversations with Andrew Zisserman, who also pointed out the significance of Shashua's 7-point algorithm. Finally, the key ideas of the paper came to me after several wide ranging discussions with Long Quan during a stay as visitor at Lifia, Grenoble. In some way, all these people have contributed to this paper, and I wish to thank them.

References

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

[2] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Computer Vision - ECCV '92, LNCS-Series Vol. 588, Springer-Verlag*, pages 563 – 578, 1992.

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

[5] R. I. Hartley. Lines and points in three views - an integrated approach. In *Proc. ARPA Image Understanding Workshop*, pages 1009–1016, 1994.

[6] R. I. Hartley. In defence of the 8-point algorithm. In *Proc. International Conference on Computer Vision*, 1995.

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

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

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

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

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

[12] 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.

[13] Amnon Shashua. Algebraic functions for recognition. PAMI–in press.

[14] 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.

[15] Minas E. Spetsakis and John Aloimonos. Closed form solution to the structure from motion problem. In *Proc. AAAI*, 1987.

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

[17] Minas E. Spetsakis and John Aloimonos. A unified theory of structure from motion. In *DARPA IU Proceedings*, 1990.

[18] T. Vieville and Q.T. Luong. Motion of points and lines in the uncalibrated case. Report RR-2054, INRIA, 1993.

[19] 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, March, 1992.