# Projective Reconstruction from Line Correspondences

Richard I. Hartley
GE - Corporate Research and Development,
P.O. Box 8, Schenectady, NY, 12301.

## Abstract

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

## 1 Introduction

This paper gives an effective algorithm for the projective reconstruction of a scene consisting of lines in space as seen in at least three views with uncalibrated cameras. The placement of the cameras with respect to the scene is also determined. At least three views are necessary, since as discussed in [8], no information whatever about camera placements may be derived from any number of line-to-line correspondences in fewer than three views. With three arbitrary cameras with unknown possibly different calibrations it is not possible to specify the scene more precisely than up to an arbitrary projective transformation of space. This contrasts with the situation for calibrated cameras in which a set of sufficiently many lines may be determined up to a scaled Euclidean transformation from three views ([7, 8]).

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 ([4]) for which a robust algorithm has been given in [3]. 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 [7, 8] 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.

It will be assumed that three different views are taken of a set of fixed lines in space. That is, it is assumed that the cameras are moving and the lines are fixed, which is opposite to the assumption made in [8]. In general, it will not be assumed that the images are taken with the same camera. Thus the three cameras are uncalibrated and possibly different.

## 2 Notation and Basics

The three-dimensional space containing the scene will be considered to be the 3-dimensional projective space $\mathcal{P}^3$ and points in space will be represented by homogeneous 4-vectors $\mathbf{x}$. Similarly, image space will be regarded as the 2-dimensional projective space $\mathcal{P}^2$ and points in an image will be represented by homogeneous 3-vectors $\mathbf{u}$. The space-image mapping induced by a pinhole camera may be represented by a $3 \times 4$ 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 $3 \times 3$ matrix $A$ and a column vector $\mathbf{c}$, as follows : $M = (A \mid \mathbf{c})$. 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 $\mathbf{ab}^\top$ is a matrix.

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 $\boldsymbol{\lambda}^\top\mathbf{u} = 0$.

**Projective Reconstruction** Consider a set of lines in space viewed by several cameras, and let $\boldsymbol{\lambda}_j^i$ be the image of the $i$-th line in the $j$-th image. The task of projective reconstruction is to find a set of camera matrices $M_j$ and 3D-lines $\boldsymbol{\chi}_i$ so that line $\boldsymbol{\chi}_i$ is indeed mapped to the line $\boldsymbol{\lambda}^i_j$ by the mapping $M_j$. For the present we pass over the questions of how to represent lines in space and how they are acted on by camera matrices $M_j$. If the camera matrices are allowed to be arbitrary, then it is well known ([1, 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 $M_0$, $M_1$ and $M_2$. We make the assumption that no two of the cameras are located at the same point in space. Let $H$ be formed by adding one extra row to $M_0$ to make a non-singular $4 \times 4$ matrix. Then since $HH^{-1} = I_{4\times4}$, it follows that $M_0H^{-1} = (I|0)$. Since $M_0$ may be transformed to $(I \mid 0)$, by applying transformation $H$ to the reconstruction we may assume without loss of generality that $M_0 = (I \mid 0)$. Next we turn to the form of $M_1 = (A_1 \mid \mathbf{c}_1)$. Since cameras $M_0$ and $M_1$ are not located at the same points, $\mathbf{c}_1 \neq 0$ and $M_1$ may therefore be scaled so that $\mathbf{c}_1{}^\top\mathbf{c}_1 = 1$. It may be observed that further multiplication on the right by a matrix $H^{-1} = \begin{pmatrix} I & 0 \\ -\mathbf{c}_1{}^\top A_1 & 1 \end{pmatrix}$ transforms $(I \mid 0)$ to itself, while mapping $(A_1 \mid \mathbf{c}_1)$ to $(A_1 - \mathbf{c}_1\mathbf{c}_1{}^\top A_1 \mid \mathbf{c}_1)$. This matrix has the interesting property that the columns of $A_1 - \mathbf{c}_1\mathbf{c}_1{}^\top A_1$ are perpendicular to $\mathbf{c}_1$, since $\mathbf{c}_1{}^\top(A_1 - \mathbf{c}_1\mathbf{c}_1{}^\top A_1) = 0$. It follows of course that $A_1$ has rank 2, and so the camera is located on the plane at infinity, but this remark is not used.

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

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

where for simplicity in future computations we have changed notation to avoid proliferation of subscripts.

## 3 The Transfer Equations

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

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

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

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

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

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

$$\boldsymbol{\lambda}_0 = (R^\top\boldsymbol{\lambda}_1)(\mathbf{s}_4{}^\top\boldsymbol{\lambda}_2) - (S^\top\boldsymbol{\lambda}_2)(\mathbf{r}_4{}^\top\boldsymbol{\lambda}_1) \qquad (1)$$

This important formula allows us to transfer lines from a pair of images to a third image directly. In general, we will represent a line in space simply by giving its images $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ with respect to the two cameras with matrices $M_1$ and $M_2$.

Now, writing $R = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ and $S = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ where the $\mathbf{r}_k$ and $\mathbf{s}_k$ are the columns of $R$ and $S$, we see that the $i$-th entry (or row) of $\boldsymbol{\lambda}_0$ given in (1) is $(\mathbf{r}_k{}^\top\boldsymbol{\lambda}_1)(\mathbf{s}_4{}^\top\boldsymbol{\lambda}_2) - (\mathbf{s}_k{}^\top\boldsymbol{\lambda}_2)(\mathbf{r}_4{}^\top\boldsymbol{\lambda}_1)$, which may be rearranged as $\boldsymbol{\lambda}_1{}^\top(\mathbf{r}_k\mathbf{s}_4{}^\top - \mathbf{r}_4\mathbf{s}_k{}^\top)\boldsymbol{\lambda}_2$. This leads to a second form of (1).

$$\boldsymbol{\lambda}_0 = \begin{pmatrix} \boldsymbol{\lambda}_1{}^\top(\mathbf{r}_1\mathbf{s}_4{}^\top - \mathbf{r}_4\mathbf{s}_1{}^\top)\boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_1{}^\top(\mathbf{r}_2\mathbf{s}_4{}^\top - \mathbf{r}_4\mathbf{s}_2{}^\top)\boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_1{}^\top(\mathbf{r}_3\mathbf{s}_4{}^\top - \mathbf{r}_4\mathbf{s}_3{}^\top)\boldsymbol{\lambda}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\lambda}_1{}^\top T_1\boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_1{}^\top T_2\boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_1{}^\top T_3\boldsymbol{\lambda}_2 \end{pmatrix} \qquad (2)$$

where the $3 \times 3$ matrices $T_k$ are defined by this equation. Formula (2) is much the same as a formula given for calibrated cameras in [8], but proven here for uncalibrated cameras. In [8], the letters $E$, $F$ and $G$ are used instead of $T_k$. However, since $F$ is the standard notation for the fundamental matrix, we prefer to use $T_k$. Equations (1) and (2) may be termed the *transfer equations* in two alternative forms.

If sufficiently many line matches are known, it is possible to solve for the three matrices $T_k$. In fact, since each $\boldsymbol{\lambda}^i_0$ has two degrees of freedom, each set of matched lines $\boldsymbol{\lambda}^i_0 \leftrightarrow \boldsymbol{\lambda}^i_1 \leftrightarrow \boldsymbol{\lambda}^i_2$ gives rise to two linear equations in the entries of $T_1$, $T_2$ and $T_3$. Exactly how these equations may best be formulated will be discussed later. Since the $T_1$, $T_2$ and $T_3$ have a total of 27 entries, but are defined only up to a common scale factor, 13 line matches are sufficient to solve for the three matrices. With more than 13 line matches, a least-squares solution may be computed.

## 4 Retrieving the Camera Matrices

Formula (2) gives a formula for the transfer matrices $T_k$ 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 $T_k$. It will be assumed in this discussion that the rank of each of the matrices $T_k$ is at least 2, which will be the case except in certain special camera configurations. See [8] for a discussion of methods applying to calibrated cameras in the case where the rank of $T_k$ is less than 2.

Now, note that $(\mathbf{r}_4 \times \mathbf{r}_k)^\top T_k = 0$ since $(\mathbf{r}_4 \times \mathbf{r}_k)^\top \mathbf{r}_k = (\mathbf{r}_4 \times \mathbf{r}_k)^\top \mathbf{r}_4 = 0$. It follows that we can compute $\mathbf{r}_4 \times \mathbf{r}_k$ up to an unknown multiplicative factor by finding the null-space of $T_k$ for each $k = 1, \ldots, 3$. Since the camera matrix $M_1 = (R \mid \mathbf{r}_4) = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4)$ has rank 3, the set of vectors $\{\mathbf{r}_4 \times \mathbf{r}_k\}$ has rank 2. Consequently, $\mathbf{r}_4$ (or $-\mathbf{r}_4$) may be computed as the unit vector normal to all of $\mathbf{r}_4 \times \mathbf{r}_k$ for $k = 1, \ldots, 3$. The vector $\mathbf{s}_4$ may be computed in the same way.

To derive formulae for the camera matrices $M_1$ and $M_2$. we make use of the assumption (here for the first time) that ${\mathbf{r}_4}^\top \mathbf{r}_k = 0$ for each $i < 4$. Then one verifies that ${\mathbf{r}_4}^\top T_k = -{\mathbf{s}_k}^\top$. This means that

$$M_2 = (S \mid \mathbf{s}_4) = (-{T_1}^\top \mathbf{r}_4, -{T_2}^\top \mathbf{r}_4, -{T_3}^\top \mathbf{r}_4, \mathbf{s}_4) \ . \tag{3}$$

Furthermore, substituting ${\mathbf{r}_4}^\top T_k = -{\mathbf{s}_k}^\top$ into the formula $T_k = \mathbf{r}_k {\mathbf{s}_4}^\top - \mathbf{r}_4 {\mathbf{s}_k}^\top$ and multiplying by $\mathbf{s}_4$ gives $T_k \mathbf{s}_4 = \mathbf{r}_k + \mathbf{r}_4 {\mathbf{r}_4}^\top T_k \mathbf{s}_4$, from which one obtains $\mathbf{r}_k = (I - \mathbf{r}_4 {\mathbf{r}_4}^\top) T_k \mathbf{s}_4$. Thus,

$$M_1 = (R \mid \mathbf{r}_4) = (AT_1\mathbf{s}_4, AT_2\mathbf{s}_4, AT_3\mathbf{s}_4, \mathbf{r}_4) \tag{4}$$

where $A = I - \mathbf{r}_4 {\mathbf{r}_4}^\top$.

The correctness of these formulae is dependent on the fact that $T_k$ is of the form $T_k = \mathbf{r}_k {\mathbf{s}_4}^\top - \mathbf{r}_4 {\mathbf{s}_k}^\top$. In other words, if one computes $M_1$ and $M_2$ from the $T_k$ using (4) and (3) and then recomputes $T_k$ using (2) then one does not retrieve the same values of $T_k$ unless $T_k$ is of the correct form.

## 5 Finding a Linear Solution.

We suppose for now that lines in an image are defined by specifying a pair of end points. Let $\boldsymbol{\lambda}_0 \leftrightarrow \boldsymbol{\lambda}_1 \leftrightarrow \boldsymbol{\lambda}_2$ be three corresponding lines and suppose that the two end points of $\boldsymbol{\lambda}_0$ are $\mathbf{u}_0 = (u_0, v_0, 1)^\top$ and $\mathbf{u}'_0 = (u'_0, v'_0, 1)^\top$. Since $\boldsymbol{\lambda}_0$ passes through $\mathbf{u}_0$ and $\mathbf{u}'_0$ we get equations ${\mathbf{u}_0}^\top \boldsymbol{\lambda}_0 = {\mathbf{u}'_0}^\top \boldsymbol{\lambda}_0 = 0$. Substituting (2) into these two equations we obtain two linear equations in the entries of matrices $T_k$. Therefore 13 line correspondences are sufficient to solve for the matrices $T_k$ up to a common scale factor. With more than 13 correspondences a least-squares solution is found. The least-squares solution does not minimize the distance of the transferred line from the end points $\mathbf{u}_0$ and $\mathbf{u}'_0$, since the left hand sides of these two equations do not represent precisely the distance from the transferred line to the endpoints of the measured line $\boldsymbol{\lambda}_0$. As we shall see, a normalizing factor is missing. This is the price we pay, however, to have a linear algorithm. To understand this algorithm better, we now consider the effect of noise, at the same time generalizing to lines defined by several points, rather than just two end points.

We will suppose that lines in an image are defined by specifying a number of points and that the best line is the one that minimizes the sum of squares of distances from the points to the line. If $\mathbf{u} = (u, v, 1)^\top$ and $\boldsymbol{\lambda} = (\lambda, \mu, \nu)^\top$, then the perpendicular distance from the point $\mathbf{u}$ to the line $\boldsymbol{\lambda}$ is given by

$$d(\boldsymbol{\lambda}, \mathbf{u}) = \frac{\mathbf{u}^\top \boldsymbol{\lambda}}{(\lambda^2 + \mu^2)^{1/2}} \ . \tag{5}$$

If a line $\boldsymbol{\lambda}$ is defined by a set of points $\mathbf{u}^j$, and $\hat{\boldsymbol{\lambda}} = (\hat{\lambda}, \hat{\mu}, \hat{\nu})^\top$ is another line, then we define

$$\begin{aligned} d^2(\hat{\boldsymbol{\lambda}}, \boldsymbol{\lambda}) &= \sum_j d^2(\hat{\boldsymbol{\lambda}}, \mathbf{u}^j) \\ &= \hat{\boldsymbol{\lambda}}^\top \left( \sum_j \mathbf{u}^j {\mathbf{u}^j}^\top \right) \hat{\boldsymbol{\lambda}} / (\hat{\lambda}^2 + \hat{\mu}^2) \\ &= \hat{\omega} \hat{\boldsymbol{\lambda}}^\top \Lambda \hat{\boldsymbol{\lambda}} \end{aligned} \tag{6}$$

where $\Lambda = \sum_j \mathbf{u}^j {\mathbf{u}^j}^\top$ and $\hat{\omega} = 1/(\hat{\lambda}^2 + \hat{\mu}^2)$.

If lines $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are known, then the transfer equation (2) states that a transferred line $\hat{\boldsymbol{\lambda}}_0$ may be expressed as a linear combination of the entries of the matrices $T_k$. We may write this as $\hat{\boldsymbol{\lambda}}_0 = B\mathbf{t}$ where $\mathbf{t}$ is a 27-dimensional vector of the entries of the matrices $T_k$. Then we may write

$$d^2(\hat{\boldsymbol{\lambda}}_0, \boldsymbol{\lambda}_0) = \hat{\omega} \mathbf{t}^\top \left( B^\top \Lambda_0 B \right) \mathbf{t} \ . \tag{7}$$

Summing over a set of matches lines indexed by a superscript $i$, one sees that the quantity to be minimized is

$$\sum_i d^2(\hat{\boldsymbol{\lambda}}^i_0, \boldsymbol{\lambda}^i_0) = \mathbf{t}^\top \left( \sum_i \hat{\omega}^i {B^i}^\top \Lambda^i_0 B^i \right) \mathbf{t} \tag{8}$$

Unfortunately, the constants $\hat{\omega}^i$ are not known a-priori, but depend on the values of the matrices $T_k$. A linear solution may be found by assuming that all the values of $\hat{\omega}^i$ are equal to 1, or equivalently by seeking to minimize the following expression.

$$\sum_i d^2(\hat{\boldsymbol{\lambda}}^i_0, \boldsymbol{\lambda}^i_0)/\hat{\omega}^i = \mathbf{t}^\top \left( \sum_i {B^i}^\top \Lambda^i_0 B^i \right) \mathbf{t} \tag{9}$$

The value of $\mathbf{t}$ that minimizes this expression is the eigenvector corresponding to the minimum eigenvalue of $\sum_i B^{i\top} \Lambda_0^i B^i$, which is easily found using Jacobi's method ([5]). With perfect data, the smallest eigenvalue will be zero. Thus, one sees that using the linear method to solve for the $T_k$ corresponds to the somewhat arbitrary decision to set all the $\hat{\omega}^i$ to 1 instead of their proper values. This has the effect of weighting all the lines differently, and leads to sub-optimal results.

## 6 Getting the best solution

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

### 6.1 Scaling the 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 $\nu >> \lambda, \mu$. In this case, the matrix in (9) is poorly conditioned, having more than one eigenvalue close to zero. By experiment, it has been found that scaling all pixel coordinates so that the pixel values in the data range between about $-1.0$ and $1.0$ works well, giving a far closer match of the transferred line to the actual data than with the unscaled coordinates. An alternative approach is to try a range of scale factors, selecting the one that gives the best sum-of-squares residual error. The residual error is obtained by minimizing (9) to find the $T_k$, then computing line $\hat{\boldsymbol{\lambda}}_0$ using (2) and finally computing the error using (8) to estimate the true quality of the fit. Since this computation is quite fast, one can afford repeated trials of this nature.

### 6.2 Converging on the optimal solution

In the presence of noise, the matrices $T_k$ obtained by minimizing (9), or even (8) will not have the correct form as given in (2). Hence the camera matrices computed using (4) and (3) will not correspond precisely to the computed $T_k$. Furthermore, minimizing (9) does not correspond exactly to minimizing the distance of the constructed line $\hat{\boldsymbol{\lambda}}_0$ to the endpoints of $\boldsymbol{\lambda}_0$. Thus, the computed camera matrices can only be considered as an approximation to the optimal solution – and not a very good approximation either. Nevertheless, it is good enough to initialize an iterative algorithm to converge to the optimal solution.

Starting from the initial solution found by the linear methods already described, we proceed by varying the entries of the camera matrices $M_1$ and $M_2$ to converge to the optimal solution. This is a straightforward parameter minimization problem, solved using the Levenberg-Marquardt algorithm ([5]). (See also the paper [3] in these proceedings for applications of the Levenberg-Marquardt algorithm to other reconstruction problems.) The varying parameters are the 24 entries of the matrices $M_1$ and $M_2$ and the quantity to be minimized is the true error expression (8). The lines $\boldsymbol{\lambda}_1^i$ and $\boldsymbol{\lambda}_2^i$ are not varied, but are set to the best fit lines in the two images. The first form of the transfer equation (1) is used to compute the lines $\boldsymbol{\lambda}_0^i$. Typically convergence occurs within 10 iterations. Furthermore, each iteration is very fast, since construction of the normal equations ([5]) requires time linear in the number of lines, and the normal equations are only of size $24 \times 24$. 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.

### 6.3 The Optimal Solution

The solution given in the previous example is not quite optimal, since all the error is confined to measurement in the zero-th image, instead of being shared among all three images. The true optimal solution can be found by approximating lines $\boldsymbol{\lambda}_1^i$ and $\boldsymbol{\lambda}_2^i$ by lines $\hat{\boldsymbol{\lambda}}_1^i$ and $\hat{\boldsymbol{\lambda}}_2^i$. The variable parameters are the two camera matrices $M_1$ and $M_2$ as well as the coordinates of the lines $\hat{\boldsymbol{\lambda}}_1^i$ and $\hat{\boldsymbol{\lambda}}_2^i$. The goal is to minimize the sum of squares error given by a sum of terms of the form (6) for all lines $\hat{\boldsymbol{\lambda}}_j^i$ for $j = 1, 2, 3$. As before, the lines $\hat{\boldsymbol{\lambda}}_0^i$ are computed using (1).

The disadvantage of this 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 [6]. Carrying out this final iteration to obtain a true optimal solution gives minimal gain over the method described in the previous section. Moreover, it is definitely not a good idea to skip the previous iterative refinement step and attempt to find the optimal solution right away. Convergence problems can arise if this is done.

## 7 Reconstruction

Once the camera matrices are computed, it is a simple task to compute the positions of the lines in space. In particular, the line in space corresponding to a set of matched lines $\boldsymbol{\lambda}_0 \leftrightarrow \boldsymbol{\lambda}_1 \leftrightarrow \boldsymbol{\lambda}_2$ must be the

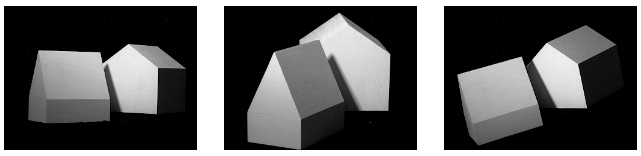

Figure 1: *Three photos of houses*

intersection of the three planes $M_i^\top \boldsymbol{\lambda}_i$. If the three lines $\boldsymbol{\lambda}_0$, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ satisfy the transfer equation (1), then the three planes will meet exactly. This will be the case with the lines estimated using either of the two iterative methods described above. Consequently, the line may be expressed as the intersection of any two of the three planes.

## 8 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 [2]. To carefully control noise insertion, image coordinates were adjusted (by an average of about 0.5 pixels) so as to make the projective reconstruction agree exactly with the pixel coordinates.

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

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

In order to judge the quality of the reconstruction, and present it in a simple form, the errors in the positions of the epipoles were chosen. The epipolar positions are related to the relative positions and orientations of the three cameras. If the computed camera positions are correct, then so will be the reconstruction. The epipoles in images $M_1$ and $M_2$ corresponding to the centre of projection of camera $M_0$ are simply the last columns of $M_1$ and $M_2$ respectively. To measure whether two epipoles are close, the following method was used. Let $\mathbf{p}$ and $\hat{\mathbf{p}}$ be actual and computed positions of the epipole, each vector being normalized to have unit length. We define $d(\mathbf{p}, \hat{\mathbf{p}'}) = 180.0 * min(||\mathbf{p} - \hat{\mathbf{p}}||, ||\mathbf{p} + \hat{\mathbf{p}}||)/\pi$. If the epipoles are close to the centre of the image, then this quantity gives a measure of their distance. If they are far from the image centre (which they are in this case – the epipoles are at locations (8249, 2006) and (-17876, 23000) in Euclidean coordinates), this is an approximate measure of the angular difference between the radial directions to the epipoles. The factor $180/\pi$ is included to give this angle in degrees.

| 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* $640 \times 484$ *pixels. The last line represents the reconstruction from 17 real data lines.*

The results of these experiments are given in Table 1. The columns of this table have the following meanings.

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

As can be seen from this table, the algorithm performs quite well with noise levels up to about 2.0 pixels (the image size being $640 \times 484$ 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.

## 9 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 $512 \times 512$ pixels, the problem of projective reconstruction becomes badly behaved. There exist multiple near-optimal solutions. For high resolution images where the relative errors may be expected to be smaller, the algorithm will show enhanced performance.

It is to be expected that (as with reconstruction from points [3]) 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 [3] shows that a projective reconstruction may be converted to a Euclidean reconstruction if all the cameras have the same calibration, or alternatively Euclidean constraints are imposed on the scene.

## References

[1] 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.

[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] Richard I. Hartley. An algorithm for self calibration from several views. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 908–912, 1994.

[4] 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.

[5] 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.

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

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

[8] 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.