International Journal of Computer Vision, 4, 171–183 (1990)
© 1990 Kluwer Academic Publishers, Manufactured in The Netherlands.

Structure from Motion Using Line Correspondences

MINAS E. SPETSAKIS* AND JOHN (YIANNIS) ALOIMONOS
Computer Vision Laboratory, Center for Automation Research and Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742-3411

Abstract
A theory is presented for the computation of three-dimensional motion and structure from dynamic imagery, using only line correspondences. The traditional approach of corresponding microfeatures (interesting points—highlights, corners, high-curvature points, etc.) is reviewed and its shortcomings are discussed. Then, a theory is presented that describes a closed form solution to the motion and structure determination problem from line correspondences in three views. The theory is compared with previous ones that are based on nonlinear equations and iterative methods.

1 Introduction

The importance of the estimation of the three-dimensional motion of a moving object (or of the sensor) from a sequence of images in robotics (visual input to a manipulator, proprioceptive abilities, navigation, structure computation for recognition, etc.) can hardly be overemphasized.

Up to now there have been three approaches toward the solution of the problem of computation of three-dimensional motion from a sequence of images:

1. The first method assumes the dynamic image to be a three-dimensional function of two spatial arguments and a temporal argument. Then, if this function is locally well behaved and its spatiotemporal gradients are computable, the image velocity or optical flow may be computed [1–3].
2. The second method considers cases where the motion is "large" and the previous technique is not applicable. In these instances the measurement technique relies upon isolating and tracking features in the image through time. These features can be microfeatures (highlights, corners, points of high curvature, interest points) or macrofeatures (contours, areas, lines, etc.). In other words, operators are applied to both images which output a set of features in each image, and then the correspondence problem between these two sets of features has to be solved (i.e., finding which features in both dynamic images are projections of the same world feature) [4, 5].

In both of the above approaches, after the optic flow field, the discrete displacement field (which can be sparse) or the correspondence between macrofeatures is computed; algorithms are constructed for the determination of the three-dimensional motion, based on the image flow or on the correspondence [6–13].

3. In the third method, the three-dimensional motion parameters are directly computed from the spatial and temporal derivatives of the image intensity function. In other words, if f is the intensity function and (u, v) the optic flow at a point, then the equation $f_x u + f_y v + f_t = 0$ holds approximately. All methods in this category are based on substitution of the optic flow values in terms of the three-dimensional motion parameters in the above equation, and there is promising work in this direction [14–16]. Also, there is work on "correspondenceless" motion detection in the discrete case, where a set of points is put into correspondence with another set of points (the sets correspond, not the individual points [14].

As the problem has been formulated over the years, one camera is used, and so the number of three-dimensional motion parameters that have to be and can be computed is five: two for the direction of translation and three for the rotation.

In this paper we present a theory for the determination of three-dimensional motion and structure from

*Current address: Dept. of Computer Science, York University, 4700 Keele Street, North York, Ontario, CANADA, M3J-IP3.

line correspondences in three views, since it can be easily shown that from two views there are infinitely many solutions if one corresponds only straight lines. A line is represented by its slope and intercept (although we use an equivalent representation that has more uniform behavior), and not by its end points, even if such points exist.

2 Motivation and Previous Work

The basic motivation for this research is the fact that optical flow (or discrete displacement) fields produced from real images by existing techniques are corrupted by noise and are partially incorrect [17]. Most of the algorithms in the literature that use the retinal motion field to recover three-dimensional motion, or are based on the correspondence of microfeatures, fail when the input (retinal motion) is noisy. Some algorithms work reasonably well, but only for images in a specific domain.

A possible solution to this difficulty is as follows: Instead of using correspondences between microfeatures such as points, why not try to use correspondences of macrofeatures? In this case, on the one hand the retinal correspondence process will be much easier, greatly reducing false matches, and on the other hand the constraint that relates three-dimensional motion to retinal motion will be different and perhaps not as sensitive to small perturbations resulting from discretization effects. As macrofeatures, we can use lines or contours, since they appear in a rich variety of natural images. The contour-based approach has been examined in [14]. Research on the problem of motion interpretation based on line correspondences has been carried out by T.S. Huang and his colleagues [18, 19]. There, the problem of three-dimensional motion computation has been successfully addressed in the restricted cases of only rotational or only translational motion. In the case of unrestricted rigid motion some good results have been obtained in the above references, but the solution is obtained iteratively from a system of nonlinear equations, and convergence of the solution to a unique value is not guaranteed if the initial value that is fed to the iterative procedure is not close to the actual solution.

3 Statement of the Problem

The problem we are addressing is to compute the 3D motion and structure of a rigid object from its successive perspective projections, using only line correspondences. So, from three views of a scene such as the one in figure 1 and using only line correspondences (where lines are represented with slope and intercept) we will be able, applying this theory, to recover the 3D motion between the views and the structure of the scene. Since the structure can easily be computed when the motion is known, we will first derive the equation of a 3D line given the motion parameters and the images of the line in two successive frames. Then, we will show how to recover 3D motion from line correspondences.

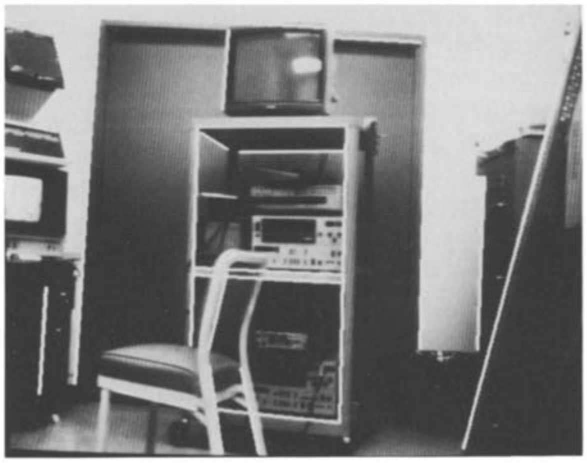
Fig. 1.

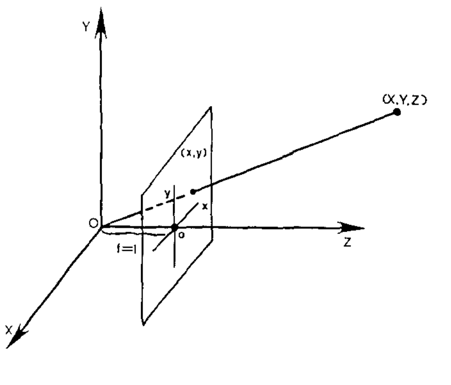

Fig. 2. Imaging geometry.

The imaging geometry is the usual one (figure 2): The system *OXYZ* is the object space coordinate system with the image plane perpendicular to the optical axis (z axis) at the point $o = [0, 0, 1]^T$, the focal length being 1. Let *ox*, *oy* be the axes of the naturally induced coordinate system on the image plane (*ox//OX*, *oy//OY*). The focal point (nodal point of the eye) is O and so an object point $[X, Y, Z]^T$ is projected onto the point $[x, y]^T$ on the image plane, where

$$x = \frac{X}{Z} \quad y = \frac{Y}{Z} \tag{1}$$

Finding structure is equivalent to finding the equations of all the 3D lines of interest. These equations have the following form:

E_i: $[X = A_{xi}Z + B_{xi},\ Y = A_{yi}Z + B_{yi}]$ $i = 1, 2, \ldots$

We use as motion parameters the rotation matrix R, representing a rotation around an axis that passes through the origin, and the translation vector T, where

$$R \equiv \begin{bmatrix} r_1 & r_4 & r_7 \\ r_2 & r_5 & r_8 \\ r_3 & r_6 & r_9 \end{bmatrix} \quad T \equiv \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$r_1, r_2, \ldots, r_9$ are the elements of an orthnormal matrix. A point $[X, Y, Z]^T$ before the motion is related to itself $[X', Y', Z']^T$ after the motion by

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T$$

The above is enough to describe any rigid motion. The images are known 2D lines of the form

ϵ_{il}: $y = a_{il}x + b_{il}$ $i = 1, 2, 3, \ldots$ l: a, b, c

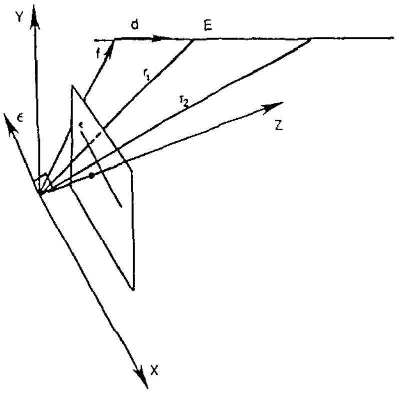

Fig. 3. A line in the image is represented either as its equation or as a 3D vector orthogonal to the line and to a ray from the origin to the line.

Frames are denoted by letters. Also the lines ϵ_{il} and $\epsilon_{il'}$ correspond to the same line E_i in space.

We are also going to use another representation of the lines in vector form which, although dual to the equation form, can make the rotation computations look more natural (figure 3). We represent an image line by the vector normal to the plane defined by the origin and the object line (which also contains the image line)

ϵ_{il}: $y = a_{il}x + b_{il}$ $i = 1, 2, 3, \ldots$ l: a, b, c

or in vector form

$$\epsilon_{il}: \begin{bmatrix} a_{il} \\ -1 \\ b_{il} \end{bmatrix}$$

We use a displacement and a direction vector to represent the object line:

$$E_i: \begin{cases} X = A_{xi}Z + B_{xi} \\ Y = A_{yi}Z + B_{yi} \end{cases}$$

or

$$d_i = \begin{bmatrix} A_{xi} \\ A_{yi} \\ 1 \end{bmatrix} \quad f_i = \begin{bmatrix} B_{xi} \\ B_{yi} \\ 0 \end{bmatrix}$$

and E_i: $f_i + Zd_i$.

The first problem for which we propose a solution is that of finding structure from *motion and line correspondence*, i.e., finding the equation, before the motion, of a line in 3D given the equations of two successive images of it, as well as its motion parameters R, T. We first consider the case of no rotation and then we introduce rotation. The second problem that we will solve is that of finding the motion and structure knowing only the *line correspondences* over three frames, by solving the first problem twice, first for frames 1 and 2 and then for frames 1 and 3. Clearly we should obtain the same line representation. This is the only constraint on the motion parameters of the problem, and is enough to solve it if we have an adequate number of line correspondences. (We need a minimum of 6, as pointed out in [18, 19]; and in order to have linear equations, we need 13 according to our theory.)

4 Structure from Motion and Correspondence in the Pure Translation Case

A line E:

$$X = A_xZ + B_x$$
$$Y = A_yZ + B_y$$

when translated by T: $[t_x, t_y, t_z]^T$ becomes

$$X = A_xZ + B_x + t_x - A_xt_z$$
$$Y = A_yZ + B_y + t_y - A_yt_z \tag{2}$$

The images are known to exist and are

$$e_a\colon y = a_ax + b_a \tag{3}$$
$$e_b\colon y = a_bx + b_b \tag{4}$$

for the two frames respectively. From the imaging geometry we find from (2) and the relations of perspective projection (1), by eliminating X, Y, Z, that

$$y = x\frac{B_y}{B_x} - \frac{A_xB_y - A_yB_x}{B_x} \tag{5}$$

$$y = x\frac{B_y + t_y - A_yt_z}{B_x + t_x - A_xt_z} - \frac{A_x(B_y + t_y - A_yt_z) - A_y(B_x + t_x - A_xt_z)}{(B_x + t_x - A_xt_z)} \tag{6}$$

By equating the x, y coefficients of (3)-(5) and (4)-(6) we get four equations in four unknowns (the parameters of the 3D lines). Solving them we get only two solutions (one of the solutions is spurious). These solutions are

$$A_x = \frac{t_x}{t_z},\quad A_y = \frac{t_y}{t_z},\quad B_x = 0,\quad B_y = 0$$

and

$$A_x = -\frac{b_b - b_a}{a_b - a_a}$$

$$A_y = -\frac{a_ab_b - a_bb_a}{a_b - a_a}$$

$$B_x = -\frac{b_bt_z - t_y + a_bt_x}{a_b - a_a}$$

$$B_y = -\frac{a_a(b_bt_z - t_y + a_bt_x)}{a_b - a_a}$$

The first is the spurious one because it represents a 3D line that passes through the origin and is sliding along itself. Such a line does not give a line image in both frames, which contradicts the natural assumption that the camera sees a line as a line. The other solution is the one we want, and the only one we keep as valid in the rest of the presentation. This also proves the uniqueness of the solution. An alternative way to write the same result is the vector form we described above:

$$d = \frac{\epsilon_a \times \epsilon_b}{\hat{Z}\cdot(\epsilon_a \times \epsilon_b)} \tag{7}$$

$$f = \frac{(T\cdot\epsilon_b)(\epsilon_a \times \hat{Z})}{\hat{Z}\cdot(\epsilon_a \times \epsilon_b)} \tag{8}$$

where $\hat{Z}$ is the unit vector along the z axis.

5 Introducing Rotation

The general case with both rotation and translation can be derived directly from the pure translation case quite easily. We first establish the following result which is also used in [18, 19]:

An image line ϵ_a (in vector form) of a line in space that is rotating with rotation R around the origin is transformed into an image line $R \cdot \epsilon_a$.

Proof. The above result has appeared in the past and its rigorous proof is rather trivial. So instead of just repeating, we provide a more descriptive and self-explanatory version of the proof. The vector ϵ_a is the normal to the plane defined by origin and the object line, which intersects the image plane along ϵ_a. When the line rotates then the plane also rotates with the same rotation, and so does the normal of the plane, which then becomes $R \cdot \epsilon_a$.

The importance of the above result is that the rotated image can be found without any knowledge about the object line, which implies that no constraint can be derived from the pure rotation case to lead to a solution similar to that in the pure translational case. So we consider now the general case of both rotation and translation.

The movement of the line consists of a rotation followed by a translation. So if we rotate the first image ϵ_a to $R \cdot \epsilon_a$ then we can solve the pure translation case with the image of the first frame being $R \cdot \epsilon_a$ and the image of the second being ϵ_b, and what we get is the object line rotated by R. All we need then is to rotate back by R^T. This way equations (7) and (8) become

$$R \cdot d = \frac{(R \cdot \epsilon_a) \times \epsilon_b}{\hat{Z} \cdot [(R \cdot \epsilon_a) \times \epsilon_b]}$$

and

$$R \cdot f = \frac{(T \cdot \epsilon_b)\,[(R \cdot \epsilon_a) \times \hat{Z}]}{\hat{Z} \cdot [(R \cdot \epsilon_a) \times \epsilon_b]}$$

from which we get, after some manipulations,

$$d = \frac{\epsilon_a \times (R^T \cdot \epsilon_b)}{\hat{Z} \cdot [\epsilon_a \times (R^T \cdot \epsilon_b)]} \tag{9}$$

$$f = \frac{(T \cdot \epsilon_b)(\epsilon_a \times \hat{Z})}{\hat{Z} \cdot [\epsilon_a \times (R^T \cdot \epsilon_b)]} \tag{10}$$

In the above expressions the z components of the vectors are 1 and 0 respectively. This not only makes the duality of the vector and equation forms obvious but is is also a sufficient property to guarantee that two equal lines are always represented by the same pair of vectors, a fact we use in the next section.

6 Motion and Structure from Line Correspondences

In the previous section we showed how to compute the structure given the line correspondences and the motion. Here we are concerned with finding motion from line correspondences alone.

Given the images of one line in three successive frames (a, b, c), the solution (as a function of the R and T parameters) must be the same for both pairs of frames $a-b$ and $a-c$. So

$$d = \frac{\epsilon_a \times (R_a^T \cdot \epsilon_b)}{\hat{Z} \cdot [\epsilon_a \times (R_a^T \cdot \epsilon_b)]} = \frac{\epsilon_a \times (R_b^T \cdot \epsilon_c)}{\hat{Z} \cdot [\epsilon_a \times (R_b^T \cdot \epsilon_c)]} \tag{11}$$

and

$$f = \frac{(T_a \cdot \epsilon_b)(\epsilon_a \times \hat{Z})}{\hat{Z} \cdot [\epsilon_a \times (R_a^T \cdot \epsilon_b)]} = \frac{(T_b \cdot \epsilon_c)(\epsilon_a \times \hat{Z})}{\hat{Z} \cdot [\epsilon_a \times (r_b^T \cdot \epsilon_c)]} \tag{12}$$

where ϵ_c is the image of the line in the third frame and T_a, T_b, R_a, R_b represent the translation and rotation for frames $a-b$ and frames $a-c$ respectively. We now simplify these vector equations since they represent four equations, only two of which are independent. The reason is the following: the vector f represents the point where the line cuts the plane $Z = 0$. This point belongs to this plane and to the plane defined by the origin and the image line, which of course contains the object line, so it belongs to their intersection which we can find from the image alone. Thus given the x (y) component of the f vector, the y (x) component can be found. This implies that another equation in f is superfluous. For the d vector we know that it has $x = 1$ and is orthogonal to the image line vector. The only additional information we need to specify is one of the other two components. The rest can be found then. So we have only one independent equation in the d vector.

Equations (11) and (12) can be expanded to

$$\begin{bmatrix} \dfrac{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{X})}{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{Z})} \\[2ex] \dfrac{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{Y})}{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{Z})} \\[2ex] 1 \end{bmatrix} = \begin{bmatrix} \dfrac{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{X})}{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{Z})} \\[2ex] \dfrac{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{Y})}{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{Z})} \\[2ex] 1 \end{bmatrix}$$

$$\begin{bmatrix} \dfrac{(T_a \cdot \epsilon_b)(\epsilon_a \cdot \hat{Y})}{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{Z})} \\[2ex] \dfrac{-(T_a \cdot \epsilon_b)(\epsilon_a \cdot \hat{X})}{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{Z})} \\[2ex] 0 \end{bmatrix} = \begin{bmatrix} \dfrac{(T_b \cdot \epsilon_c)(\epsilon_a \cdot \hat{Y})}{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{Z})} \\[2ex] \dfrac{-(T_b \cdot \epsilon_c)(\epsilon_a \cdot \hat{X})}{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{Z})} \\[2ex] 0 \end{bmatrix}$$

where $(\cdot, \cdot, \cdot)$ is the scalar triple product of vectors. From the above we choose the ones that come from equating the x components of the vectors. There is no reason for this, other than the fact that they lead to simpler equations and are independent. We can write them also as

$$\frac{(T_a \cdot \epsilon_b)}{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{Z})} = \frac{(T_b \cdot \epsilon_c)}{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{Z})} \tag{13}$$

$$\frac{(T_a \cdot \epsilon_b)}{(\epsilon_a, R_a^T \cdot \epsilon_b, \hat{X})} = \frac{(T_b \cdot \epsilon_c)}{(\epsilon_a, R_b^T \cdot \epsilon_c, \hat{X})} \tag{14}$$

By simplifying the triple products and cross multiplying and then substituting

$$K = (T_b \cdot R_{a1}^T)^T - T_a \cdot R_{b1}^T$$

$$L = (T_b \cdot R_{a2}^T)^T - T_a \cdot R_{b2}^T$$

$$M = (T_b \cdot R_{a3}^T)^T - T_a \cdot R_{b3}^T$$

where R_{a1} is the first column of the matrix R_a, etc., we get finally

$$a_a(\epsilon_b^T \cdot L \cdot \epsilon_c) + (\epsilon_b^T \cdot K \cdot \epsilon_c) = 0 \tag{15}$$

$$b_a(\epsilon_b^T \cdot L \cdot \epsilon_c) + (\epsilon_b^T \cdot M \cdot \epsilon_c) = 0 \tag{16}$$

from equations (13) and (14) respectively. The above equations are nonlinear in terms of the motion parameters but linear in terms of the elements of the matrices K, L, M, and they come from considering just one line. By using 13 lines we can get 26 linear equations, set any of the 27 elements of the matrices to 1, and solve the 26×26 system; then we can find the elements of the K, L, M matrices (hereafter essential parameters), which in terms of the motion parameters are

$$K = \begin{bmatrix} r_{a1}t_{bx} - r_{b1}t_{ax} & r_{a1}t_{by} - r_{b4}t_{ax} & r_{a1}t_{bz} - r_{b7}t_{ax} \\ r_{a4}t_{bx} - r_{b1}t_{ay} & r_{a4}t_{by} - r_{b4}t_{ay} & r_{a4}t_{bz} - r_{b7}t_{ay} \\ r_{a7}t_{bx} - r_{b1}t_{az} & r_{a7}t_{by} - r_{b4}t_{az} & r_{a7}t_{bz} - r_{b7}t_{az} \end{bmatrix}$$

$$L = \begin{bmatrix} r_{a2}t_{bx} - r_{b2}t_{ax} & r_{a2}t_{by} - r_{b5}t_{ax} & r_{a2}t_{bz} - r_{b8}t_{ax} \\ r_{a5}t_{bx} - r_{b2}t_{ay} & r_{a5}t_{by} - r_{b5}t_{ay} & r_{a5}t_{bz} - r_{b8}t_{ay} \\ r_{a8}t_{bx} - r_{b2}t_{az} & r_{a8}t_{by} - r_{b5}t_{az} & r_{a8}t_{bz} - r_{b8}t_{az} \end{bmatrix}$$

$$M = \begin{bmatrix} r_{a3}t_{bx} - r_{b3}t_{ax} & r_{a3}t_{by} - r_{b6}t_{ax} & r_{a3}t_{bz} - r_{b9}t_{ax} \\ r_{a6}t_{bx} - r_{b3}t_{ay} & r_{a6}t_{by} - r_{b6}t_{ay} & r_{a6}t_{bz} - r_{b9}t_{ay} \\ r_{a9}t_{bx} - r_{b3}t_{az} & r_{a9}t_{by} - r_{b6}t_{az} & r_{a9}t_{bz} - r_{b9}t_{az} \end{bmatrix}$$

In this way it is easy to find the numerical values of the three matrices. By equating their values with the functions of the motion parameters that they represent we get 27 nonlinear equations involving the motion parameters only. By setting one of the values to 1 we actually set the scale factor of the solution to some value.

If we want to increase the robustness of the computation of the elements of the matrices K, L, M, there is a heuristic that is widely used and proven good experimentally [13]. The heuristic is to use the least-squares technique that is proven optimal under some conditions (one of which is the independence of the unknowns and is clearly violated here). So by just using least squares we are not guaranteed optimality, but there is considerable improvement that can be justified intuitively and experimentally.

Another way to solve the system is by using Singular Value Decomposition (SVD). The SVD is defined by the following theorem [20, 25]:

Any $m \times n$ matrix A (where $m \geq n$) can be factored into the product of three matrices $U_{m\times n}$, the diagonal matrix $\Sigma_{n\times n}$ and $V_{n\times n}$ (the U, V being orthonormal) such that

$$\begin{bmatrix} \\ A \\ \\ \end{bmatrix} = \begin{bmatrix} \\ U \\ \\ \end{bmatrix} \cdot \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \cdots & \\ & & & \cdots \\ & & & w_n \end{bmatrix} \cdot \begin{bmatrix} \\ V^T \\ \\ \end{bmatrix}$$

The elements of the diagonal matrix are called singular values.

We can take equations (15) and (16) for a set of at least 13 lines and form the matrix A. The $\eta \times 27$ matrix, where η is at least 26 can be supplemented by rows of zeros to a minimum of $\eta = 27$. The solution for the 27 essential parameters then are the elements of the column of V that correspond to the smallest singular value. In the absence of noise this singular value is exactly zero. But when noise is present it is not zero. Moreover if the configuration of lines we used for constructing matrix A is a pathological one[1] then there are going to be at least two very small and almost equal singular values. By means of some thresholds we can detect when such singular values appear, in which case we know that we are dealing with an ill-conditioned problem.

This solution is equivalent to a least-squares [25] one and has one more advantage. In order to take care of the indeterminate scale factor, instead of arbitrarily setting one element of the vector of the unknowns to 1, we set the length of the vector to 1 (since it is a column of an orthonormal matrix). This way we avoid the ill-conditioning that occurs if we choose to set to 1 an element that is meant to be zero. These are the main reasons why the SVD method behaves better.

7 Solving for the Motion Parameters

The three matrices contain all the information the motion parameters themselves contain but they are not that useful a representation. So we present here a method to compute the motion parameters up to a scale factor for the translations. The three matrices are themselves known up to a scale factor since we either set one of their elements to one or they are computed as an eigenvector of length one.

The three matrices can be written as

$$\begin{aligned} K &= \begin{bmatrix} t_{ax} & r_{a1} \\ t_{ay} & r_{a4} \\ t_{az} & r_{a7} \end{bmatrix} \cdot \begin{bmatrix} -r_{b1} & -r_{b4} & -r_{b7} \\ t_{bx} & t_{by} & t_{bz} \end{bmatrix} \\ L &= \begin{bmatrix} t_{ax} & r_{a2} \\ t_{ay} & r_{a5} \\ t_{az} & r_{a8} \end{bmatrix} \cdot \begin{bmatrix} -r_{b2} & -r_{b5} & -r_{b8} \\ t_{bx} & t_{by} & t_{bz} \end{bmatrix} \\ M &= \begin{bmatrix} t_{ax} & r_{a3} \\ t_{ay} & r_{a6} \\ t_{az} & r_{a9} \end{bmatrix} \cdot \begin{bmatrix} -r_{b3} & -r_{b6} & -r_{b9} \\ t_{bx} & t_{by} & t_{bz} \end{bmatrix} \end{aligned} \tag{17}$$

These matrices are all singular so at least one of their singular values must be zero. The singular vector f_1 of K that corresponds to the zero singular value is orthogonal to the R_{b1} and T_b (assume singular values distinct for the time being). The other two matrices yield similar relations between the corresponding singular vectors f_2 and f_3 and the translation vector. So it is easy to compute the direction of the line the translation lies on [21]. The cases where this does not work can in principle be identified and treated separately. But it is not at all easy to decide that two values are equal in the presence of noise. Here it is even worse because we have to check nine pairs in order to decide how to treat the problem. Programming and verifying such a thing is very difficult. The following observations are helpful in developing a simple algorithm to find the direction and the magnitude of the translation (but not its polarity, which is computed last).

- There is degeneracy in computing the singular vectors only when the two smallest singular values (remember all are non-negative) are almost equal to each other (and almost equal to zero), and not all three of them can effectively be zero.
- In order for K to have two zero singular values either R_{a1} and T_a must be collinear or R_{b1} and T_b must be collinear. The same for the other two matrices and columns of R_a and R_b respectively.
- There can be at most two matrices out of three that are degenerate: one due to a collinearity of the a motion parameters and one due to a collinearity of the b parameters. From the third matrix we can compute the singular vector we are interested in.
- The sum of the squares of the projections of a unit vector on the three columns of the rotation matrix equals one.

We can define the vectors f_1, etc., through the singular value decomposition of the three matrices, which looks like this:

$$\begin{aligned} K &= \begin{bmatrix} f_1' & g_1' & h_1' \end{bmatrix} \cdot \begin{bmatrix} 0 & & \\ & k_1 & \\ & & k_2 \end{bmatrix} \cdot \begin{bmatrix} f_1 & g_1 & h_1 \end{bmatrix}^T \\ L &= \begin{bmatrix} f_2' & g_2' & h_2' \end{bmatrix} \cdot \begin{bmatrix} 0 & & \\ & l_1 & \\ & & l_2 \end{bmatrix} \cdot \begin{bmatrix} f_2 & g_2 & h_2 \end{bmatrix}^T \\ M &= \begin{bmatrix} f_3' & g_3' & h_3' \end{bmatrix} \cdot \begin{bmatrix} 0 & & \\ & m_1 & \\ & & m_2 \end{bmatrix} \cdot \begin{bmatrix} f_3 & g_3 & h_3 \end{bmatrix}^T \end{aligned} \tag{18}$$

We now compute the products $K \cdot f_1$, $L \cdot f_1$ and $M \cdot f_1$.

$$K \cdot f_1 = 0 = -T_a(R_{b1} \cdot f_1) \tag{19}$$

[1]One for which no unique solution exists.

because R_{b1} is normal to f_1,

$$\begin{aligned} L \cdot f_1 &= \begin{bmatrix} t_{ax} & r_{a2} \\ t_{ay} & r_{a5} \\ t_{az} & r_{a8} \end{bmatrix} \cdot \begin{bmatrix} -r_{b2} & -r_{b5} & -r_{b8} \\ t_{bx} & t_{by} & t_{bz} \end{bmatrix} \cdot f_1 \\ &= \begin{bmatrix} t_{ax} & r_{a2} \\ t_{ay} & r_{a5} \\ t_{az} & r_{a8} \end{bmatrix} \cdot \begin{bmatrix} -R_{b2} \cdot f_2 \\ 0 \end{bmatrix} \qquad (20) \\ &= -T_a(R_{b2} \cdot f_1) \end{aligned}$$

and similarly

$$M \cdot f_1 = -T_a(R_{b3} \cdot f_1) \qquad (21)$$

The sum of the squares of the above is

$$\begin{aligned} &(K \cdot f_1)^2 + (L \cdot f_1)^2 + (M \cdot f_1)^2 \\ &= T_a^2[(R_{b1} \cdot f_1)^2 + (R_{b2} \cdot f_1)^2 + (R_{b3} \cdot f_1)^2] \qquad (22) \\ &= T_a^2 \neq 0 \end{aligned}$$

where T_a^2 is the inner product of T_a with itself. Equations (19)-(22) have two consequences. One is that we can compute the magnitude of the translation. The other is that from even a single singular vector f_i we can compute the direction of the translation vector; it is parallel to any nonzero product $K \cdot f_1$, $L \cdot f_1$, $M \cdot f_1$. Since the sum of their squares is nonzero (T_a, T_b cannot be zero because in this case the problem itself is ill-posed and can have infinitely many solutions as described below) at least one of them is nonzero.

The procedure of finding T_a up to a sign is the following: Find one matrix for which the computation of the smallest singular value is furthest away from making the matrix degenerate and find the corresponding singular vector. Multiply the vector by the three matrices, and the vectors that result are parallel to T_a and the sum of their squares is the square of the length of T_a. Do the same with the transpose to find T_b. If noise is present we get the direction of T_a by averaging the three vectors.

It is worth noticing that the only property of the f_i vectors we need is their orthogonality to the T_b vector. In order to proceed to the computation of the columns of the rotation matrices we need a set of at least two noncollinear unit vectors orthogonal to T_b (and another such set for T_a), which is always trivial to find. We show the derivation using the f_i vectors which are orthogonal to T_b and not collinear (otherwise the columns of the rotation matrix would be coplanar). In case of degeneracy the f_i's cannot be computed; instead we construct a set of three noncoplanar vectors that are orthogonal to T_b but not necessarily orthogonal to the columns of the rotation matrices.

As we already saw

$$\begin{aligned} K \cdot f_1 &= -T_a(R_{b1} \cdot f_1) \\ L \cdot f_1 &= -T_a(R_{b2} \cdot f_1) \\ M \cdot f_1 &= -T_a(R_{b3} \cdot f_1) \end{aligned}$$

and in exactly the same way

$$\begin{aligned} K \cdot f_2 &= -T_a(R_{b1} \cdot f_2) \\ L \cdot f_2 &= -T_a(R_{b2} \cdot f_2) \\ M \cdot f_2 &= -T_a(R_{b3} \cdot f_2) \\ K \cdot f_3 &= -T_a(R_{b1} \cdot f_3) \\ L \cdot f_3 &= -T_a(R_{b2} \cdot f_3) \\ M \cdot f_3 &= -T_a(R_{b3} \cdot f_3) \end{aligned}$$

Define

$$F_b = [f_1 \ \ f_2 \ \ f_3]$$

We can now compute up to a sign a matrix A_b such that

$$A_b = s_b R_b^T \cdot F_b$$

The s_b represents the sign ambiguity inherited from the computation of the translation vector. Matrix F_b is composed from vectors f_1, f_2, and f_3 whose length is the unity. In fact, any choice of length or sign of these vectors will do.

Leaving the s_b aside for the time being,

$$\begin{aligned} A_b \cdot A_b^T &= R_b^T \cdot F_b \cdot F_b^T \cdot R_b = U_b \cdot \Sigma^2 \cdot U_b^T \\ A_b^T \cdot A_b &= F_b^T \cdot F_b = V_b \cdot \Sigma^2 \cdot F_b^T \end{aligned}$$

This implies that the SVDs of the matrices A_b and F_b can have the same V_b.

$$A_b = U_b \cdot \Sigma \cdot V_b^T \qquad F_b = X_b \cdot \Sigma \cdot V_b^T$$

Since the SVD is not unique the V_b's might be different. So we compute the SVD of the F_b matrix from the SVD of the A_b.

$$X_b \cdot \Sigma = F_b \cdot V_b \Rightarrow X_b = [\pm X_{b_1} \ \ X_{b_2} \ \ X_{b_3}]$$

The ambiguity in the sign comes from the zero singular value. There is only one zero singular value because the f_i's are not collinear.

So

$$A_b = R_b^T \cdot F_b \Rightarrow U_b \cdot \Sigma \cdot V_b^T = R_b^T \cdot X_b \cdot \Sigma \cdot V_b^T$$

Premultiplying and postmultiplying by U_b^T and V_b,

$$\Sigma = U_b^T \cdot R_b^T \cdot X_b \cdot \Sigma$$

Now it is obvious that

$$U_b^T \cdot R_b^T \cdot X_b = \begin{bmatrix} \sigma_b & & \\ & 1 & \\ & & 1 \end{bmatrix} \tag{23}$$

where

$$\sigma_b = \det(U_b) \cdot \det(X_b) \tag{24}$$

The σ_b takes on the values +1 or −1 and equation (23) gives the one that results in a right-handed rotation matrix. So (23) and (24) yield

$$R_b = X_b \cdot \begin{bmatrix} \sigma_b & & \\ & 1 & \\ & & 1 \end{bmatrix} \cdot U_b^T$$

The matrix A_b is known up to a sign. So the two solutions for the rotation matrix are

$$R_b = X_b \cdot \begin{bmatrix} \sigma_b & & \\ & s_b & \\ & & s_b \end{bmatrix} \cdot U_b^T \tag{25}$$

where s_b is ±1. Which of the two is to be determined. The same with the other matrix

$$R_a = X_a \cdot \begin{bmatrix} \sigma_a & & \\ & s_a & \\ & & s_a \end{bmatrix} \cdot U_a^T \tag{26}$$

There are two possible solutions for each rotation pair and each translation pair. By constructing the matrices K, L, M from the candidate solutions, only one yields the three initial matrices. In the next paragraph we prove that there is only one out of the four that yields the initial ones except when there is not a unique solution at all.

7.1 Uniqueness of the solution

We observe that if R_b is the actual solution and R_b' is the solution with the opposite sign option then

$$R_b^T \cdot T_b = U_b \cdot \Sigma_b \cdot X_b^T \cdot T_b$$

$$= U_b \cdot \Sigma_b \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = R_b'^T \cdot T_b \tag{27}$$

The information we used to find the solution for the rotation and translation is the products of the f vectors by the three matrices. The ambiguity of the signs is due to the lack of information along the direction of T. The ambiguity can be proved resolvable if we make use of this fact.

$$T_a^T \cdot K \cdot T_b = \begin{bmatrix} t_{ax} & t_{ay} & t_{az} \end{bmatrix} \cdot \begin{bmatrix} s_b t_{ax} & r_{a1} \\ s_b t_{ay} & r_{a4} \\ s_b t_{az} & r_{a7} \end{bmatrix}$$

$$\cdot \begin{bmatrix} -r_{b1} & -r_{b4} & -r_{b7} \\ s_a t_{bx} & s_a t_{by} & s_a t_{bz} \end{bmatrix} \cdot \begin{bmatrix} t_{bx} \\ t_{by} \\ t_{bz} \end{bmatrix}$$

$$= \begin{bmatrix} s_b T_a^2 & R_{a1} \cdot T_a \end{bmatrix} \cdot \begin{bmatrix} -R_{b1} \cdot T_b \\ s_a \cdot T_b^2 \end{bmatrix}$$

$$= s_a T_b^2 \cdot (R_{a1} \cdot T_a) - s_b T_a^2 \cdot (R_{b1} \cdot T_b)$$

For simplicity of the above expression we did not specify which of the two candidate solutions for the rotation matrices we used because when multiplied by the translation vectors they yield the same result. There are four solutions that correspond to the four different combinations of +1 or −1 for s_a and s_b. In the following three equations

$$\begin{aligned} T_a^T \cdot K \cdot T_b &= s_a T_b^2 \cdot (R_{a1} \cdot T_a) - s_b T_a^2 \cdot (R_{b1} \cdot T_b) \\ T_a^T \cdot L \cdot T_b &= s_a T_b^2 \cdot (R_{a2} \cdot T_a) - s_b T_a^2 \cdot (R_{b2} \cdot T_b) \\ T_a^T \cdot M \cdot T_b &= s_a T_b^2 \cdot (R_{a3} \cdot T_a) - s_b T_a^2 \cdot (R_{b3} \cdot T_b) \end{aligned} \tag{28}$$

we can compute $T_a^T \cdot K \cdot T_b$, etc. Also due to (27) we can compute $(R_{a1} \cdot T_a)$, etc. In the 3×3 matrix that is formed from (28),

$$\begin{bmatrix} -T_a^T \cdot K \cdot T_b & T_b^2 \cdot (R_{a1} \cdot T_a) & -T_a^2 \cdot (R_{b1} \cdot T_b) \\ -T_a^T \cdot L \cdot T_b & T_b^2 \cdot (R_{a2} \cdot T_a) & -T_a^2 \cdot (R_{b2} \cdot T_b) \\ -T_a^T \cdot M \cdot T_b & T_b^2 \cdot (R_{a3} \cdot T_a) & -T_a^2 \cdot (R_{b3} \cdot T_b) \end{bmatrix} \tag{29}$$

we know all the elements. At least one of the three singular values must be zero.

Case 1. The matrix has only one zero singular value. The corresponding singular vector then gives the solution for s_a and s_b.

Case 2. The matrix has two zero singular values. Then almost always the problem has a solution: Let two singular vectors be v_1 and v_2. And let $\hat{s}_i = [1 \ \pm 1 \ \pm 1]^T$, $i = 1, \ldots, 4$ be the four possible sign combinations. The solution must be parallel to one of them and it lies in the plane defined by the vectors v_1, v_2. This suggests that the vector $\hat{s}_i$ that satisfies $(\hat{s}_i, v_1, v_2) = 0$, where $(\cdot, \cdot, \cdot)$ is the triple scalar product, is the solution if there is only one such vector. When two of the $\hat{s}_i$ satisfy the condition then the rows of the matrix (29) are multiples of either $[1, \pm 1, 0]$ or $[1, 0, \pm 1]$ or $[0, 1, \pm 1]$. The first four possibilities are impossible because they suggest that either the second column is zero and the third is not, or the reverse, which cannot happen because if the second column is zero then one of the translation vectors is zero and so is the third column. The last two possibilities are possible and in this case if we multiply all equations in (28) by s_a, s_b then

$$\begin{aligned} 0 &= (s_bT_a)^T \cdot K \cdot (s_aT_b) \\ &= (s_aT_b)^2 \cdot [R_{a1} \cdot (s_bT_a)] \\ &\quad - (s_bT_a)^2 \cdot [R_{b1} \cdot (s_aT_b)] \\ 0 &= (s_bT_a)^T \cdot L \cdot (s_aT_b) \\ &= (s_aT_b)^2 \cdot [R_{a2} \cdot (s_bT_a)] \qquad (30) \\ &\quad - (s_bT_a)^2 \cdot [R_{b2} \cdot (s_aT_b)] \\ 0 &= (s_bT_a)^T \cdot M \cdot (s_aT_b) \\ &= (s_aT_b)^2 \cdot [R_{a3} \cdot (s_bT_a)] \\ &\quad - (s_bT_a)^2 \cdot [R_{b3} \cdot (s_aT_b)] \end{aligned}$$

Equation (30) implies that either

$$\begin{aligned} &T_a = 0 \quad \text{or} \quad T_b = 0 \quad \text{or} \\ &R_a^T \cdot (s_bT_a) = R_b^T \cdot (s_aT_b) \qquad (31) \end{aligned}$$

This is the condition for the problem not to have a unique solution when the matrix (29) has two zero singular values (figure 4).

Case 3. If there are three zero singular values then the matrix is all zero. This implies that either T_a or T_b are zero. Of course then the problem has no unique solution as above.

For all three cases condition (31) is necessary and sufficient for the problem not to have a unique solution. In case (31) holds then there is not only ambiguity in the signs but the problem cannot be solved at all. The reason for this is that we need three views a, b, c to solve the problem. If the problem could be solved even when (31) holds then we could solve it from two views as follows: construct the c view by rotating either view a by some rotation (in this case T_b is zero) or by rotating view b by some rotation (in this case $s_bR_a^TT_a = s_aR_b^TT_b$ holds) and then solve the problem.

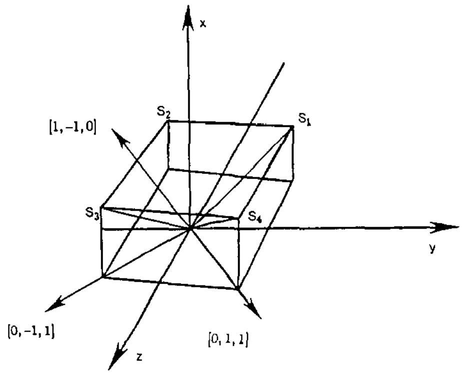

Fig. 4. The solution of the system (28) can be one of $S_1 \ldots S_4$. In case we have a degeneracy in the two smallest singular values the solution exists if the plane of the corresponding singular values contains only one of the S_i's. If it contains two of them then the rows of the matrix (29) must be parallel to the diagonal bisectors of the *yz* plane, which implies equation (31).

There is also another issue regarding uniqueness. We proved that there is one solution for the motion parameters given the three matrices. The solution is correct up to a scale factor which cannot be determined. Its sign, though, can be determined from the fact that a line image is visible only on one side of its vanishing point. An opposite sign in the scale factor makes the line visible from the other side of the vanishing point. Rewriting (9) and (10) in a more convenient form,

$$d = \epsilon_a \times (R^T \cdot \epsilon_b)$$

$$f = \frac{(T \cdot \epsilon_b)(d \times \epsilon_a)}{d^2}$$

we see that the direction vector d of the line does not depend on the scale factor. The vanishing point depends only on d so it does not depend on the scale factor either. The procedure to find if the sign of the scale factor is correct then is: first find a visible point p_1 on the actual line image (this is a point that the edge detector located in the first place), then a visible point p_2 of the image of the computed line (the projection of a point of the line that is in front of the image plane), and check the sign of

$$\left(\frac{d}{d\cdot\hat{z}} - p1\right)\cdot\left(\frac{d}{d\cdot\hat{z}} - p2\right)$$

If it is positive the scale factor is correct. If it is negative the scale factor is wrong and if it is zero then pick another line or other points for the test.

8 Experimental Results

We performed several experiments with artificial data in order to study the sensitivity of the algorithm to noise. The input was lines in a random configuration and the rotation and translation were also random. After the images of the lines (three) were computed, noise was added; the algorithm was executed; and the results were compared with the actual. Table 1 shows the results. In table 1, the first column shows the error in the input as a fraction of the focal length. When we say that a line is in error we mean that we haven't found exactly the position of the line, i.e., the operators that will extract the images are not perfect [22, 23]. Since we represent the image lines throughout the paper with the vector ϵ (see figure 3, where ϵ is perpendicular to the plane defined by the world line E, the image line ϵ and the nodal point), when we say that we add noise to the image lines we mean that we add noise to their corresponding vectors. When ϵ has noise ϵ', then the new (noisified) ϵ will be $\epsilon + \epsilon'$. However note that these vectors are three dimensional, i.e., we can add ϵ' perpendicular to ϵ in any direction (see figure 5 which illustrates a pictorial description of the error).

When the error in input is $1.0e - 03$, for example, this means that the length of the error vector is $0.1f/100$, where f is the focal length (in a random orientation). This means that in the image the line of interest would be in a zone of width 1 pixel (if $f = 500$ pixels for example) (see figure 6).

To give another example, if the error in input is $1.0e - 02$, this means that the image line can be anywhere in a zone of width $2f/100 = 10$ pixels (if $f = 500$ pixels). Since the lines are macrofeatures we assume that we don't have any error in corresponding lines. The only noise in the input is the uncertainty in the position of the lines.

Table 1.

Error in Input as Fraction of Unity	Number of Lines in Least Squares	Angle Between Computed and Actual Rotation Axis for a' Motion	Error in Rotation Angle for a' Motion	Angle Between Computed and Actual Rotation Axis for b' Motion	Error in Rotation Angle for b' Motion	Percent Error in a' Translation	Percent Error in b' Translation
1. 0e − 03	1. 4e + 01	3. 1d − 01	5. 9e − 01	2. 9e − 01	5. 0e − 01	1. 7e + 00	1. 7e + 00
3. 2e − 03	1. 4e + 01	1. 0e + 00	2. 3e + 00	7. 4e − 01	9. 0e − 01	3. 9e + 00	4. 4e + 00
1. 0e − 02	1. 4e + 01	9. 5e + 00	3. 4e + 01	4. 9e + 00	2. 5e + 01	1. 5e + 01	1. 4e + 01
3. 2e − 01	1. 4e + 01	1. 7e + 01	3. 4e + 01	1. 8e + 01	5. 1e + 01	3. 6e + 01	2. 5e + 01
1. 0e − 01	1. 4e + 01	1. 6e + 01	8. 6e + 01	2. 4e + 01	8. 8e + 01	3. 9e + 01	2. 2e + 01
1. 0e − 03	2. 2e + 01	6. 8e − 02	1. 3e − 01	6. 6e − 02	9. 8e − 02	3. 0e − 01	2. 8e − 01
3. 2e − 03	2. 2e + 01	2. 2e − 01	4. 1e − 01	2. 1e − 01	3. 1e − 01	9. 6e − 01	9. 0e − 01
1. 0e − 02	2. 2e + 01	7. 2e − 01	1. 3e + 00	7. 6e − 01	1. 0e + 00	3. 1e + 00	2. 8e + 00
3. 2e − 02	2. 2e + 01	4. 9e + 00	2. 0e + 01	4. 8e + 00	1. 6e + 01	1. 2e + 01	1. 3e + 01
1. 0e + 01	2. 2e + 01	1. 8e + 01	8. 4e + 01	1. 4e + 01	8. 7e + 01	8. 4e + 01	2. 2e + 01
1. 0e − 03	3. 0e + 01	5. 1e − 02	7. 0e − 02	4. 1e − 02	7. 3e − 02	1. 9e − 01	1. 8e − 01
3. 2e − 03	3. 0e + 01	1. 6e − 01	2. 2e − 01	1. 3e − 01	2. 2e − 01	6. 0e − 01	5. 7e − 01
1. 0e − 02	3. 0e + 01	5. 1e − 01	6. 8e − 01	4. 0e − 01	7. 4e − 01	1. 8e + 00	1. 7e + 00
3. 2e − 02	3. 0e + 01	2. 8e + 00	9. 3e + 00	1. 2e + 00	3. 0e + 00	5. 5e + 00	6. 3e + 00
1. 0e − 01	3. 0e + 01	1. 5e + 01	1. 7e + 01	1. 7e + 01	6. 9e + 01	3. 5e + 01	4. 3e + 01
1. 0e − 03	3. 8e + 01	3. 1e − 02	5. 7e − 02	3. 6e − 02	7. 8e − 02	2. 0e − 01	1. 4e − 01
3. 2e − 03	3. 8e + 01	1. 7e − 01	2. 4e − 01	2. 2e − 01	2. 9e − 01	4. 5e − 01	3. 6e − 01
1. 0e − 02	3. 8e + 01	3. 2e − 01	5. 4e − 01	3. 5e − 01	5. 7e − 01	1. 9e + 00	1. 3e + 00
3. 2e − 02	3. 8e + 01	1. 2e + 00	1. 9e + 00	1. 1e + 00	2. 0e + 00	5. 6e + 00	4. 5e + 00
1. 0e − 01	3. 8e + 01	1. 1e + 01	5. 5e + 01	1. 0e + 01	6. 6e + 01	2. 6e + 01	2. 3e + 01
1. 0e − 01	4. 8e + 01	1. 6e + 01	6. 8e + 01	1. 4e + 01	6. 0e + 01	2. 6e + 01	2. 5e + 01

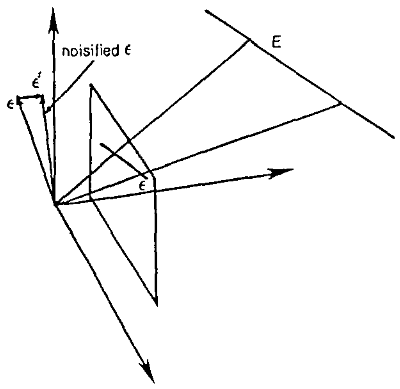

Fig. 5. The noise vector ϵ' can be in any direction perpendicular to ϵ.

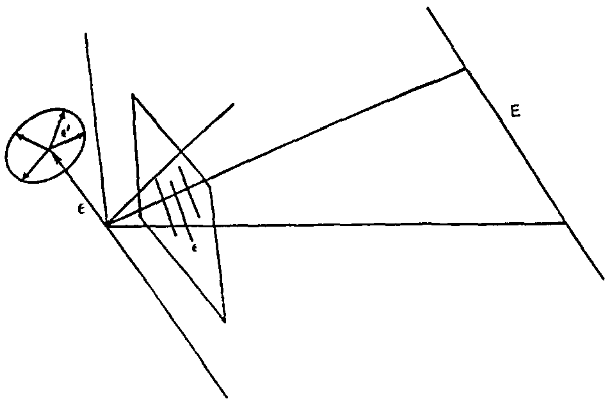

Fig. 6. The line would be in a zone of uncertainty of width at most $2(0.1f/100) = 1$ pixel, for $f = 500$ and $\|\epsilon'\| = 0.1f/100$.

Every row of table 1 has been calculated by considering the error vector ϵ' in several directions and averaging the results. The second column shows the number of lines used in the input. The third column shows the angle (in degrees) between the computed and actual rotation axes for the first motion and the fifth column shows the same thing for the second motion. The fourth and sixth columns show the error in the rotation angle (in degrees) for the first and second motion. Finally, the last two columns show the percent error in translation for both motions.

As a summary of the experimental results we can say that:

- The algorithm is unstable for a small number of lines in the input but becomes stable as the number of lines increases. Increasing the number of lines above 30 gives a little improvement. For example, in a case where there is uncertainty in the image lines such that a line can be anywhere in a zone of width 10 pixels—which is something that current line finders and edge detectors can guarantee [22, 23, 24]—when 30 lines are used, the results are 0.5°, 0.6°, 0.4°, 0.7° and 1.8% and 1.7%. These results are considered very satisfactory and our extensive experimentation shows that the algorithm is robust is a practical sense, if several lines are used.
- The algorithm has two parts: the computation of the elements of the three matrices and from these the motion parameters. The first part exhibits instability, while the second is quite stable and partly makes up for the instability of the first. The reader might wish to compare our results (table 1) with results from the algorithm by Tsai and Huang [1984] (tables 2 and 3). This algorithm computes 3D motion from point correspondences in two views, in a two-step process. It computes the entries of a matrix (essential parameters) and then from these the motion parameters. The error in the input is the error in the correspondences. Clearly the line-based algorithm is more stable than the point-based one, as is shown from the experiments.
- The computing time is about 5 seconds on a VAX/785.

Table 2. Error of motion parameters vs. error of point correspondences; 8 point correspondences are used. This table is from Tsai and Huang [1984].

Error of point correspondences	0%	0.1%	0.5%	1%	2%
Error of E (essential parameters)	0%	7.30%	29.36%	47.13%	67.54%
Error of rotation parameters	0%	1.36%	6.96%	14.32%	30.28%
Error of translations	0%	0.51%	20.66%	53.97%	94.63%

Table 3. Error of motion parameters vs. number of point correspondences for 2.5% error on point correspondences. This table is from Tsai and Huang [1984].

Number of point correspondences	8	20
Error of E (essential parameters)	73.91%	19.49%
Error of rotation parameters	38.70%	2.40%
Error of translations	103.6%	29.66%

9 Conclusions and Future Directions

We have presented a theory for the computation of structure and 3D motion from line correspondences in three views. The solution comes in closed form and not from iterative methods. The uniqueness of the essential parameters, i.e., classification of pathological cases for which the essential parameters cannot be computed, is an open research issue. We also consider as an interesting future research issue the theoretical stability analysis of the present theory. Also, the application of the theory to the problems of camera calibration and pose determination for object recognition.

References

1. B.K.P. Horn and B.G. Schunck, "Determining optical flow," *Artificial Intelligence* 17: 185-204, 1981.
2. S. Ullman and E. Hildreth, "The measurement of visual motion," *Physical and Biological Processing of Images* (Proc. Intern. Symp., Rank Prize Funds. London), O.J. Braddick and A.C. Sleigh (eds.), Springer-Verlag, pp. 154-176, September 1982.
3. H.H. Nagel, "Displacement vectors derived from second order intensity variations in image sequences," *Comput. Vision, Graphics, and Image Process.* 21: 85-117, 1983.
4. S. Ullman, "The interpretation of visual motion." Ph.D. thesis, 1977.
5. A. Bandopadhay. Ph.D. thesis, Department of Computer Science, University of Rochester, 1986.
6. G. Adiv, "Determining 3-D motion and structure from optical flow generated from several moving objects," COINS Tech. Rept. 84-07, July 1984.
7. A. Bruss and B.K.P. Horn, "Passive navigation," *Comput. Vision, Graphics, and Image Process.* 21: 3-20, 1983.
8. H.C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proc. Roy. Soc. London B* 208: 385-397, 1980.
9. H.C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature* 293: 133-135, 1981.
10. K. Prazdny, "Egomotion and relative depth map from optical flow," *Biol. Cybernetics* 36: 87-102, 1980.
11. K. Prazdny, "Determining the instantaneous direction of motion from optical flow generated by a curvilinearly moving observer," *Comput. Vision, Graphics, and Image Process.* 17: 94-97, 1981.
12. R.Y. Tsai and T.S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects." In *Image Understanding 1984*, S. Ullman and W. Richards (eds.). Ablex Publishing: Norwood, NJ, 1984.
13. R.Y. Tsai and T.S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces," *IEEE Trans.* PAMI 6: 13-27, 1984.
14. J. Aloimonos, "Low level visual computations." Ph.D. thesis, Department of Computer Science, University of Rochester, August 1986.
15. E. Ito and J. Aloimonos, "Computing transformation parameters from images," *Proc. IEEE Conf. on Robotics and Automation*, 1987.
16. S. Negadahripour. Ph.D. thesis, AI Lab, MIT, 1986.
17. S. Ullman, "Analysis of visual motion by biological and computer systems," *IEEE Comput.* 14: 57-69, 1981.
18. Y. Liu and T.S. Huang, "Estimation of rigid body motion using straight line correspondences," *Proc. Intern. Conf. Pattern Recog.*, Paris, October 1986.
19. Y. Liu and T.S. Huang, "Estimation of rigid body motion using straight line correspondences," *IEEE Workshop on Motion: Representation and Analysis*, Kiawah Island, SC, May 1986.
20. G.W. Stewart, *Introduction to Matrix Computations*. Academic Press, San Diego, CA, 1980.
21. M.E. Spetsakis and J.Y. Aloimonos, "Closed form solution to the structure from motion problem from line correspondences," TR-1798, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, March 1987.
22. B.K.P. Horn, "The Binford-Horn edge finder," MIT AI Memo 285, December 1973.
23. J. Canny, M.S. thesis, Artificial Intelligence Laboratory, MIT, 1984.
24. J.B. Burns, A.R. Hanson, and E.M. Riseman, "Extracting straight lines," *IEEE Trans. PAMI* 8: 425-455, July 1986.
25. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press: New York, 1988.