Determination of Camera Location from 2-D to 3-D Line and Point Correspondences

YUNCAI LIU, THOMAS S. HUANG, FELLOW, IEEE, AND OLIVIER D. FAUGERAS, MEMBER, IEEE

Abstract—**A new method for the determination of camera location from 2-D to 3-D straight line or point correspondences is presented in this paper. With this method, the computation of the rotation matrix and the translation vector of the camera is separable. First, the rotation matrix is found by a linear algorithm using eight or more line correspondences, or by a nonlinear algorithm using three or more line correspondences, where the line correspondences are either given or derived from point correspondences. Then, the translation vector can be obtained by solving a set of linear equations based on three or more line correspondences, or two or more point correspondences. Eight 2-D to 3-D line correspondences or six 2-D to 3-D point correspondences are needed for the linear approach; three 2-D to 3-D line or point correspondences for the nonlinear approach. Good results can be obtained in the presence of noise if more than the minimum required number of correspondences are used.**

Index Terms—**Camera calibration, cartography, photogrammetry.**

I. Introduction

DETERMINING camera location from image to space point correspondences (i.e., 2-D and 3-D correspondences) is a very basic problem in image analysis and cartography. It can be applied to aircraft location, robot calibration and so on. Roughly, this problem can be considered as estimating the three-dimensional location from which an image was taken by a set of recognized landmarks appearing in the image. The camera location determination problem was formally defined by Fischler and Bolles [1] as follows: "Given a set of m control points, whose three-dimensional coordinates are known in some coordinate frame, and given an image in which some subset of the m control points is visible, determine the location (relative to the coordinate system of the control points) from which the image was obtained." In solving this problem, we shall assume that the focal length of the camera is known (to be unity in this paper for simplicity) and the 2-D to 3-D line or point correspondences are given.

On this problem, early work was done by many researchers, some of the main references are: Keller and Tewinkel [2], Wolf [3], Fischler and Bolles [1], Ganapathy [4], and Lenz and Tsai [5]. Keller and Tewinkel, and Church (as described in Wolf) solved this problem routinely by nonlinear least-squares techniques. Fischler and Bolles found the solution by first computing the "legs" (the lengths of rays from the focus point of the camera to the control points in the image plane). They also established results on the number of solutions for various number of control points. According to their analysis, there are up to four solutions if three control points are used. Usually six control points are required to get a unique solution. Ganapathy showed similar results and presented some simplified algorithms. Dealing with a real time robot system, Lenz and Tsai proposed an algorithm which seemed more accurate and easier than the existing calibration techniques. The key of their method is that there is only one single rotary joint of a robot arm moving for each movement. In this way the multidimensional problem is decomposed into a series of one-dimensional problems, and the computation becomes easier. Note that all existing techniques are based on point correspondences. We are not aware of any working using line correspondences.

In this paper, a new method for determining camera location from straight line correspondences is presented. This method is based on our earlier work [6], [7]. Since lines can be created from given points, the method can also used for point correspondences. The camera coordinate system is related to the space coordinate system (located on the ground) by a rotation followed by a translation. The rotation matrix and the translation vector can be solved separately. Both nonlinear and linear algorithms are available for estimating the rotation. The linear method needs eight line correspondences or six point correspondences, while the nonlinear algorithm needs three line or point correspondences. In determining the translation vector, three line correspondences or two point correspondences are required and the algorithm is linear. Since the nonlinear methods for rotation need fewer correspondences and have reasonably wide convergence ranges (about 25–30 degrees for the three Euler angles), they may be preferable in many practical problems to linear methods. In this paper these algorithms will be described and computer simulation results presented.

II. Notation

The geometry of the problem is illustrated in Fig. 1. In the figure, let o–xyz be the space coordinate system fixed on the ground, and o'–$x'y'z'$ be the camera coordinate

Manuscript received July 30, 1987; revised August 31, 1989. Recommended for acceptance by J. L. Mundy. This work was supported by the National Science Foundation under Grant IRI-8605400.

Y. Liu and T. S. Huang are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1101 W. Springfield Ave., Urbana, IL 61801.

O. D. Faugeras is with INRIA-Roquencourt, B.P. 105, F-78153 Le Chesnay, France.

IEEE Log Number 8931115.

0162-8828/90/0100-0028$01.00 © 1990 IEEE

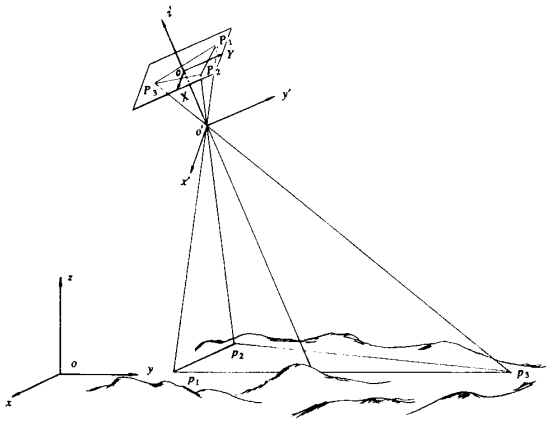


Fig. 1. Basic imaging geometry.

system with the origin at the focus point of the camera and the z'-axis coinciding with the optical axis and pointing to the back of the camera. Let us imagine that the coordinate system o'-$x'y'z'$ is obtained by first rotating the coordinate system o-xyz with rotation matrix R then translating it with vector $\vec{T}$. Let $p_i = (x_i, y_i, z_i)$ be a point in the space coordinate system, and its camera coordinate representation be $p_i' = (x_i', y_i', z_i')$. Then the relationship of the two coordinate representations is

$$\vec{p}_i = R\vec{p}_i' + \vec{T} \tag{1}$$

where R is a 3×3 rotation matrix and $\vec{T}$ is a three-element vector.

Let us use O-XY to denote the coordinate system on the image, then a 3-D point $p' = (x', y', z')$ represented in the camera coordinate system and its image $P = (X, Y)$ are related by central projection:

$$X = \frac{x'}{z'} \tag{2}$$

$$Y = \frac{y'}{z'}. \tag{3}$$

III. Camera Location Determination Using 2-D to 3-D Straight Line Correspondences

A. Algorithm

Let us denote a 3-D line l in the space coordinate system o-xyz in the parametric form

$$l: \quad \vec{p} = \vec{n}t + \vec{p}_0 \tag{4}$$

where $\vec{n} = (a, b, c)^T$ is the direction of the line and $\vec{p}_0 = (x_0, y_0, z_0)^T$ is a point in the line. The image of this 3-D line in the image coordinate system can be expressed

$$L: \quad AX + BY + C = 0 \tag{5}$$

where A, B, C are known constants subject to $A^2 + B^2 + C^2 = 1$.

Substituting (2) and (3) into (5), we have

$$M: \quad Ax' + By' + Cz' = 0 \tag{6}$$

Equation (6) is the equation of the projecting plane of the 3-D line l (i.e., the plane containing l and the camera focal point o') expressed in the coordinate system o'-$x'y'z'$ through which the image line L is obtained. Note that the vector

$$\vec{N} = (A, B, C)^T$$

is the normal of the projecting plane M. Therefore, we will use the normal vector $\vec{N}$ instead equation (5) to represent the 2-D image of a 3-D line. The 3-D line we represent by two triplets $\vec{n} = (a, b, c)^T$ and $\vec{p}_0 = (x_0, y_0, z_0)^T$.

In the camera coordinate system o'-$x'y'z'$, the direction of 3-D line l expressed in (4) is

$$\vec{n}' = R^{-1}\vec{n}.$$

Since a 3-D line is always perpendicular to the normal of its projecting plane, we have

$$\vec{N} \cdot \vec{n}' = 0$$

or

$$\vec{n}^T R\vec{N} = 0. \tag{7}$$

If we consider the nine elements of R as independent unknowns, then (7) is a homogeneous linear equation in these nine unknowns. Therefore, with eight or more 2-D to 3-D line correspondences, these nine unknowns can be obtained to within a scale factor by solving linear equations. This scale factor can be determined by imposing the constraint that the sum of the squares of these unknowns is equal to 3.

A rotation matrix can also be expressed in three Euler angles

$$R = \begin{bmatrix} c_2c_3 - c_1s_2s_3 & -c_2s_3 - c_1s_2c_3 & s_1s_2 \\ s_2c_3 + c_1c_2s_3 & -s_2s_3 + c_1c_2c_3 & -s_1c_2 \\ s_1s_3 & s_1c_3 & c_1 \end{bmatrix} \tag{8}$$

where

$$c_1 = \cos\theta \quad c_2 = \cos\psi \quad c_3 = \cos\phi$$

$$s_1 = \sin\theta \quad s_2 = \sin\psi \quad s_3 = \sin\phi$$

and θ, ψ, ϕ are Euler angles called the tilt, swing, and spin, respectively.

If we use the representation of (8) for rotation, then (7) becomes nonlinear but in the three unknowns θ, ψ, and ϕ. When three or more line correspondences are givens, their nonlinear equations (7) can be solved using the method of linearization described below.

If we consider θ, ψ, and ϕ as obtained from some angles θ_0, ψ_0, and ϕ_0 by small perturbations

$$\theta = \theta_0 + \Delta\theta$$

$$\psi = \psi_0 + \Delta\psi$$

$$\phi = \phi + \Delta\phi$$

where $\Delta\theta$, $\Delta\psi$, and $\Delta\phi$ are some small values, then (7) can be linearized at the angle triplet $(\theta_0, \psi_0, \phi_0)$, and becomes linear in the variables $\Delta\theta$, $\Delta\psi$, $\Delta\phi$. We denote the linearized equation by

$$\boldsymbol{L}(\theta_0, \psi_0, \phi_0; \Delta\theta, \Delta\psi, \Delta\phi) = 0. \tag{9}$$

We solve for the rotation using (9) in the following way.

First, let $\theta_0 = \psi_0 = \phi_0 = 0$, and solve

$$\boldsymbol{L}(0, 0, 0; \Delta\theta_0, \Delta\psi_0, \Delta\phi_0) = 0$$

for $\Delta\theta_0$, $\Delta\psi_0$, and $\Delta\phi_0$.

Then, let $\theta_1 = \Delta\theta_0$, $\psi_1 = \Delta\psi_0$, $\phi_1 = \Delta\phi_0$, and solve

$$\boldsymbol{L}(\theta_1, \psi_1, \phi_1; \Delta\theta_1, \Delta\psi_1, \Delta\phi_1) = 0$$

for $\Delta\theta_1$, $\Delta\psi_1$, and $\Delta\phi_1$.

Let

$$\theta_{k+1} = \sum_{m=0}^{k} \Delta\theta_m, \quad \psi_{k+1} = \sum_{m=0}^{k} \Delta\psi_m, \quad \phi_{k+1} = \sum_{m=0}^{k} \Delta\phi_m.$$

Solve

$$\boldsymbol{L}(\theta_{k+1}, \psi_{k+1}, \phi_{k+1}; \Delta\theta_{k+1}, \Delta\psi_{k+1}, \Delta\phi_{k+1}) = 0$$

for $\Delta\theta_{k+1}$, $\Delta\psi_{k+1}$, $\Delta\phi_{K+1}$.

This procedure is stopped when $|\Delta\theta_{k+1}| + |\Delta\psi_{k+1}| + |\Delta\phi_{K+1}|$ is less than some present threshold. Then, the three Euler angle of the rotation are

$$\theta = \sum_{m=0}^{k} \Delta\theta_m,$$

$$\psi = \sum_{m=0}^{k} \Delta\psi_m,$$

$$\phi = \sum_{m=0}^{k} \Delta\phi_m.$$

Our simulations indicate that this approximation works well with three or more 2-D to 3-D line correspondences if the three Euler angles are less than 30 degrees.

To determine the translation vector of the camera coordinate system, we imagine that there is an intermediate coordinate system $\bar{o}$–$\bar{x}\bar{y}\bar{z}$ which is formed by rotating the space coordinate system o–xyz by the rotation matrix R (but without any translation). It is evident that the relationship between the intermediate coordinate system $\bar{o}$–$\bar{x}\bar{y}\bar{z}$ and the camera coordinate system o'–$x'y'z'$ is a pure translation.

Let us construct a plane Π passing through the origin $\bar{o}$ of the intermediate coordinate system and perpendicular to the 3-D line l, and another plane $\overline{M}$ passing through the origin $\bar{o}$ and parallel to the projecting plane M. See Fig. 2. Then the projection of the translation vector of the camera from $\bar{o}$ to o' in the direction $\vec{N}$ is just the distance between the planes M and $\overline{M}$. Since a point lying in the 3-D line l is always in its projecting plane M, the distance

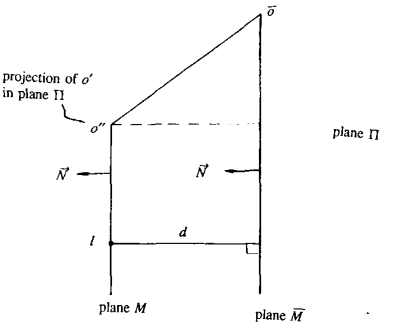


Fig. 2. Pertaining to the determination of translation.

from point p_0 (lying in the 3-D line l) to plane $\overline{M}$ is equal to the projection of the translation vector in the direction $\vec{N}$. Thus, we have

$$\vec{N} \cdot \vec{T} = \vec{N} \cdot (R^{-1}\vec{p}_0) \tag{10}$$

where we have the constraint $\|\vec{N}\| = 1$.

Equation (10) is a linear equation with three unknowns, and it is solvable if its coefficient matrix has rank 3. This is equivalent to saying that the translation vector of the camera can be uniquely determined if there are three or more 2-D to 3-D line correspondences and at least three of the image lines do not intersect in one point. Usually more than three line correspondences are available, and (10) becomes overdetermined and is solved by least-squares.

We note in passing that Chen [11] has shown that with three line correspondences, the maximum number of solutions for rotation is eight, which can in fact be attained.

B. Simulation Results

In our simulation experiments, we choose the focal length of the camera to be one unit and the image size to be a one by one square unit with the optical axis of the camera passing through the center of the image. This means that the view angle of the camera is about 53 degrees (the view angle of a 50 mm lens being 47 degrees). The rotation of the camera coordinates is so chosen that each of the three Euler angles is less than 30 degrees. The height of the camera from ground (z-component of the translation of the camera coordinate system) is from 20 to 30 units. The 3-D landmarks (lines or points) on the ground are chosen such that their images are within the view of the camera.

We performed the experiments on SUN 3/110. The CPU time of computation is typically 0.1 seconds per run.

1) Nonlinear Algorithm: Appendix 1 gives some simulation results the use of the nonlinear algorithm for solving the camera location determination problem from 2-D to 3-D straight line correspondences. For convenience, in our simulation experiments, we use a model in which the camera coordinate system is taken as the reference.

Simulation 1 is a noise-free case. Three line correspondences are used. The computation results are very accurate.

Simulations 2 and 3 are the experimental results for the case where the image line data contain noise. The noise is added as follows:

$$\vec{N}^* = \vec{N}(1 + \mu\vec{\xi})$$

where $\vec{N}$ and $\vec{N}^*$ are the image line representations before and after adding noise, μ is the noise level, and $\vec{\xi}$ is the noise vector whose elements have uniform distribution in the range $[-1, +1]$.

Simulation 2 has the same 2-D to 3-D line correspondences, rotation R and translation $\vec{T}$ as simulation 1, but 1 percent noise (i.e., $\mu = 0.01$) is added to the data of the image line representation ($\vec{N} = (A, B, C)^T$), and an additional 2-D to 3-D line correspondence is used (a total of four line correspondences) to combat the noise. The computation result shows that the relative error of the rotation matrix is 0.00173, and the relative error of the translation vector is 0.00684, which are reasonably accurate.

Simulation 3 has the same setup as simulation 2, but the noise level is 5 percent. The relative errors of rotation and translation are 0.00863 and 0.03467, respectively, which are still tolerable.

In order to examine the relationship between the errors in the computation results and the number of line correspondences used, we did some experiments for the case of noise level $\mu = 0.05$. The results shown in Table I are based on 100 random trials. From the table we can see that additional line correspondences can indeed improve the accuracy of the solutions. However, the improvement is most dramatic from three to four lines.

2) Linear Algorithm: Appendix 2 contains the simulation results of the linear algorithm for determining camera location using straight line correspondences. Simulation 4 is a noise-free case with the motion parameters the same as those of simulation 1 in Appendix 1. Eight 2-D to 3-D line correspondences are used in the simulation. With noise appearing in the data of the line correspondences, simulations 5 and 6 use nine straight line correspondences to combat noise. Simulations 4, 5, and 6 have the same motion parameters. When we compare results in Appendix 2 with those in Appendix 1, we see that the linear algorithm is much more sensitive to noise than the nonlinear algorithm.

IV. Camera Location Determination Using 2-D to 3-D Point Correspondences

A. Algorithm

In this section, we shall describe a method of determining camera location using 2-D to 3-D point correspondences. We first use line correspondences derived from point correspondences to estimate the rotation matrix by (7). The translation vector of the camera can then be easily computed using the point correspondences directly.

TABLE I

number of LC	3	4	5	6	7
relative error of rotation	0.08424	0.00884	0.00787	0.00578	0.00496
relative error of translation	0.20832	0.07417	0.03414	0.02762	0.01915

Suppose a set of 2-D to 3-D point correspondences is given:

$$P_i(X_i, Y_i) \leftrightarrow p_i(x_i, y_i, z_i) \qquad i = 1, 2, \cdots n$$

where the capital P_i is an image point with coordinates (X_i, Y_i) in the image coordinate system O–XY (or, with coordinates $(X_i, Y_i, 1)$ in the camera coordinate system o'–$x'y'z'$), p_i is a 3-D point with coordinates (x_i, y_i, z_i) in the space coordinate system o–xyz. Taking any two image points P_i, P_j and the corresponding space points p_i, p_j from the point correspondence set, we can generate an image line by connecting the points P_i and P_j:

$$AX + BY + C = 0$$

where

$$A = (Y_j - Y_i)/k$$

$$B = (X_i - X_j)/k$$

$$C = (X_iY_j - X_jY_j)/k$$

$$k = \left[(X_i - X_j)^2 + (Y_i - Y_j)^2\right]^{1/2}$$

and

$$\vec{N} = (A, B, C)^T, \tag{11}$$

and the 3-D line by connecting points p_i and p_j, which has the direction $\vec{n}$:

$$\vec{n} = (\vec{p}_i - \vec{p}_j)/\|\vec{p}_i - \vec{p}_j\|. \tag{12}$$

After finding the 2-D to 3-D line correspondences, we can use (7) and the method discussed in Section III to solve for the rotation matrix.

If (7) is solved linearly, eight or more line correspondences are needed. This means that five or more 2-D to 3-D point correspondences are needed for determining the rotation. However, according to our simulation experiments, with five point correspondences the rank of the coefficient matrix of the linear homogeneous equations is always less than eight, but with six point correspondences, the rank is eight. Since the rank of the coefficient matrix must be eight to make a set of linear homogeneous equations with nine unknowns have a unique solution (up to a scale factor), at least six point correspondences are necessary to yield a solution. This is consistent with Fischler and Bolles' conclusion [1].

Equation (7) can also be solved nonlinearly. When dealing with small rotations (all the three Euler angles being less than 30 degrees), we found that three 2-D to 3-D point correspondences (yielding three line correspondences) almost always give correct results if the iterative linearization method we discussed in the Section III is used.

To determine the translation vector of the camera coordinate system, we can use the point correspondences directly. From (1), we have

$$\vec{p}_i' = R^{-1}\vec{p}_i - R^{-1}\vec{T} \triangleq \vec{p}_{0i} - \vec{T}' \tag{13}$$

where

$$p_{0i} = R^{-1}p_i \triangleq \begin{pmatrix} x_{0i} \\ y_{0i} \\ z_{0i} \end{pmatrix}$$

$$\vec{T}' = R^{-1}\vec{T} \triangleq \begin{pmatrix} \Delta x' \\ \Delta y' \\ \Delta z' \end{pmatrix}.$$

From (2) and (3), the coordinates of the image point P_i can be written as

$$X_i = \frac{x_{0i} - \Delta x'}{z_{0i} - \Delta z'}$$

$$Y_i = \frac{y_{0i} - \Delta y'}{z_{0i} - \Delta z'}$$

or, expressed in matrix form

$$\begin{bmatrix} 1 & 0 & -X_i \\ 0 & 1 & -Y_i \end{bmatrix} \begin{bmatrix} \Delta x' \\ \Delta y' \\ \Delta z' \end{bmatrix} = \begin{bmatrix} x_{0i} - X_i z_{0i} \\ y_{0i} - Y_i z_{0i} \end{bmatrix}. \tag{14}$$

Equation (14) is a linear equation with three unknowns: $\Delta x'$, $\Delta y'$, and $\Delta z'$. Since every 2-D to 3-D point correspondence gives two equations, two point correspondences are enough to solve for $\vec{T}'$.

To inspect the condition under which (14) is solvable, let (X_i, Y_i) and (X_j, Y_j) be two image points; the coefficient matrix of (14) is then

$$\begin{bmatrix} 1 & 0 & -X_i \\ 0 & 1 & -Y_i \\ 1 & 0 & -X_j \\ 0 & 1 & -Y_j \end{bmatrix} \triangleq F.$$

Thus,

$$F^TF = \begin{bmatrix} 2 & 0 & -(X_i + X_j) \\ 0 & 2 & -(Y_i + Y_j) \\ -(X_i + X_j) & -(Y_i + Y_j) & X_i^2 + X_j^2 + Y_i^2 + Y_j^2 \end{bmatrix}$$

whose determinant is

$$\det(F^TF) = 2(x_i - x_j^2) + 2(y_i - y_j^2). \tag{15}$$

If (14) is solvable, the rank of its coefficient matrix must be 3, this means that $\det(F^TF)$ cannot be zero. From (15) we can see that this simply implies that the two image points (X_i, Y_i) and (X_j, Y_j) must be distinguishable.

Since at least three 2-D to 3-D point correspondences are needed to solve for the rotation, (14) is always overdetermined in the computation of translation $\vec{T}'$. We can use least-squares to solve for it. After the translation $\vec{T}'$ is found, the translation vector $\vec{T}$ of the camera coordinate system can be easily computed by

$$\vec{T} = R\vec{T}'.$$

B. Simulation Results

1) Nonlinear Algorithm: In Appendix 3, some simulation results of the use of nonlinear algorithm of camera location determination from 2-D to 3-D point correspondences are given. The arrangement is the same as in the cases of 2-D to 3-D straight line correspondences.

Simulation (7) is a noise-free case with three 2-D to 3-D point correspondences. The computation results are very accurate.

Simulations (8) and (9) are the experimental results with noise appearing in the data of the image points. The way to add noise to the image data is as follows.

$$X^* = X + m\xi/256$$

$$Y^* = Y + m\xi/256$$

where (X, Y) and (X^*, Y^*) are the image points before and after adding noise, m is the noise level in the number of pixels based on 256×256 pixel image, and ξ is a zero mean uniform distribution noise in the range of $[-0.5, +0.5]$.

Simulation (8) has the same 3-D points, rotation R and translation T as simulation (7), but we added half pixel noise ($m = 0.5$) to the image points and one additional point correspondence to the input data (a total of four point correspondences) to combat the noise. The computation results show that the relative error of the rotation matrix is 0.01490, and the relative error of the translation vector is 0.00392, which are reasonably accurate.

Simulation (9) has the same rotation R, translation T, and point correspondences as simulation (8), but two-pixel

TABLE II

number of PC	3	4	5	6	7
relative error of rotation	0.12292	0.05793	0.05288	0.04777	0.04489
relative error of translation	0.06661	0.03410	0.02257	0.02115	0.01717

noises are added to the image points. The relative errors of rotation matrix and translation vector are 0.06295 and 0.01827, respectively, which are still tolerable.

We have investigated the effect of the number of point correspondences on the error in the computation results. According to the statistical result of 100 trails (2-pixel noise case), we found that additional point correspondences can indeed combat the noise in the input data. But the improvement is most dramatic from three to four points.

2) Linear Algorithm: The simulation results of the linear algorithm using point correspondences are presented in Appendix 4. Simulations 10, 11, and 12 have the same motion parameters and camera arrangement as simulation 7 in Appendix 3. Simulation 10 is a noise-free case using six 2-D to 3-D point correspondences, while simulations 11 and 12 are noisy cases using seven 2-D to 3-D point correspondences. It appears that the linear algorithm is more sensitive to noise than the nonlinear algorithm. Comparison of computation results can be made between Appendix 3 and Appendix 4.

V. Conclusion

We have presented a new method for computing camera location using 2-D to 3-D straight line or point correspondences. Specifically a linear algorithm is described which required at least eight line correspondences or six point correspondences, and a nonlinear algorithm is described which required at least three line correspondences or three point correspondences. A main advantage of this new method is that it decouples rotation and translation, and then reduces computation.

With respect to error in the solution due to noise in the input image data, we have empirically observed the following: 1) Adding one more feature correspondence to required minimum number improves the solution accuracy drastically. However, further increase in the number of feature correspondence improve the solution accuracy only slowly. 2) The linear algorithm is more sensitive to noise than the nonlinear algorithm.

Appendix 1
Nonlinear Algorithm Using Straight Line Correspondences

Simulation 1:

Ground Truth: The rotation angle is (in degree):

30.00000000 0.00000000 0.00000000

The translation vector is:

2.00000000 2.00000000 20.00000000

The rotation matrix R is:

$$\begin{matrix} 0.1000000d+01 & 0.0000000d+00 & 0.0000000d+00 \\ 0.0000000d+00 & 0.8660254d+00 & -0.5000000d+00 \\ 0.0000000d+00 & 0.5000000d+00 & 0.8660254d+00 \end{matrix}$$

The corresponding $(n1, n2, n3)$, theta form is: the directional is:

$n1 = 1.000000$ $n2 = 0.000000$ $n3 = 0.000000$

the rotation angle is:

$$\beta = 30.000000$$

Input Data (without noise):

LC 1:

$\vec{n} = (a, b, c)^T$	0.1221445027	−0.9343472750	−0.3347773741
$p_0 = (x_0, y_0, z_0)^T$	0.2786720000	−1.3542210000	0.6162190000
$\vec{N} = (A, B, C)^T$	0.9414706748	0.3163931603	−0.1163113777

LC 2:

$\vec{n} = (a, b, c)^T$	−0.1041114075	−0.8264568029	−0.5532901298
$p_0 = (x_0, y_0, z_0)^T$	1.2797140000	−1.8449260000	−1.3235320000
$\vec{N} = (A, B, C)^T$	0.9711512684	0.1486670424	−0.1864492541

LC 3:

$\vec{n} = (a, b, c)^T$	−0.8692699079	−0.2742805633	−0.4112663369
$p_0 = (x_0, y_0, z_0)^T$	−2.2873670000	1.2172720000	1.6876750000
$\vec{N} = (A, B, C)^T$	−0.0198862882	−0.9948511869	0.0993763124

Computation Result (0.0 percent noise):

The number of iterations: 5.
Three rotation angles are:

30.00000 0.00000 0.00000

The rotation matrix is:

1.00000	0.00000	0.00000
0.00000	0.86603	−0.50000
0.00000	0.50000	0.86603

The translation vector of the camera is:

2.00000 2.00000 20.00000

The relative error of rotation matrix is 0.00000.
The relative error of translation vector is 0.00000.

Simulation 2 (1.0 percent noise added):
Computation Result:

The number of iterations: 4.
Three rotation angles are:

30.03544 0.06029 −0.08016

Rotation matrix is:

1.00000	−0.00196	0.00053
0.00196	0.86571	−0.50054
0.00053	0.50054	0.86572

The translation vector of the camera is:

1.98706 1.97956 19.86395

The relative error of rotation matrix is 0.00173.
The relative error of translation vector is 0.00684.

Simulation 3 (5.0 percent noise added):
Computation Result:

The number of iterations: 5.
Three rotation angles are:

30.17695 0.29993 −0.39909

The rotation matrix is:

0.99994	−0.00976	0.00263
0.00976	0.86442	−0.50267
0.00263	0.50266	0.86448

The translation vector of the camera is:

1.93245 1.89875 19.31030

The relative error of rotation matix is 0.00863.
The relative error of translation vector is 0.03467.

Appendix 2
Linear Algorithm Using Straight Line Correspondences

Simulation 4:
Input Data (without noise):

LC 1:

$\vec{n} = (a, b, c)^T$	−0.0550362335	0.3849718116	0.9212859042
$p_0 = (x_0, y_0, z_0)^T$	−0.3118320000	−0.9656830000	−0.2249210000
$\vec{N} = (A, B, C)^T$	0.8054505381	−0.5918368088	−0.0312829423

LC 2:

$\vec{n} = (a, b, c)^T$	−0.5305415036	0.4802977015	0.6984553177
$p_0 = (x_0, y_0, z_0)^T$	−0.9865060000	−0.2093540000	1.3899750000
$\vec{N} = (A, B, C)^T$	−0.0379401229	−0.9977673642	0.0549621150

LC 3:

$\vec{n} = (a, b, c)^T$	0.9162940895	−0.3522829597	−0.1905304645
$p_0 = (x_0, y_0, z_0)^T$	2.1417310000	1.0744330000	−2.0015540000
$\vec{N} = (A, B, C)^T$	0.1346062707	0.9635735099	−0.2311000711

LC 4:

$\vec{n} = (a, b, c)^T$	−0.8341662963	−0.5324650959	−0.1436924207
$p_0 = (x_0, y_0, z_0)^T$	−2.2898520000	−0.5475220000	−2.1957280000
$\vec{N} = (A, B, C)^T$	0.3634684405	−0.9208057050	0.1414480340

LC 5:

$\vec{n} = (a, b, c)^T$	0.7768523247	−0.5650866120	−0.2778085431
$p_0 = (x_0, y_0, z_0)^T$	2.1886130000	−0.9838890000	−1.2167450000
$\vec{N} = (A, B, C)^T$	0.3141647430	0.9357271220	−0.1603598061

LC 6:

$\vec{n} = (a, b, c)^T$	−0.8167041642	−0.5016822288	0.2851477677
$p_0 = (x_0, y_0, z_0)^T$	−2.0257540000	−1.8499800000	2.3905210000
$\vec{N} = (A, B, C)^T$	0.5768824189	−0.8162731898	−0.0300791353

LC 7:

$\vec{n} = (a, b, c)^T$	0.1221445027	−0.9343472750	−0.3347773741
$p_0 = (x_0, y_0, z_0)^T$	0.2786720000	−1.3542210000	0.6162190000
$\vec{N} = (A, B, C)^T$	0.9414706748	0.3163931603	−0.1163113777

LC 8:

$\vec{n} = (a, b, c)^T$	−0.1041114075	−0.8264568029	−0.5532901298
$p_0 = (x_0, y_0, z_0)^T$	1.2797140000	−1.8449260000	−1.3235320000
$\vec{N} = (A, B, C)^T$	0.9711512684	0.1486670424	−0.1864492541

Computation Result (0.0 percent noise):

The rotation matrix of the camera is:

1.00000	0.00000	0.00000
0.00000	0.86603	−0.50000
0.00000	0.50000	0.86603

The translation vector of the camera is:

2.00000	2.00000	20.00000

Simulation 5 (1.0 percent noise added):

Computation Result: The rotation matrix of the camera is:

0.98066	−0.00183	0.00771
0.00159	0.84403	−0.48257
0.01176	0.44201	0.94734

The translation vector of the camera is:

1.98179	1.96278	19.70503

The relative error of rotation matrix is 0.06148.
The relative error of translation vector is 0.01475.

Simulation 6 (5.0 percent noise added):

Computation Result: The rotation matrix of the camera is:

0.88007	−0.00861	0.03530
0.00725	0.73846	−0.40513
0.05358	0.21971	1.20976

The translation vector of the camera is:

1.85237	1.76551	18.01496

The relative error of rotation matrix is 0.28321.
The relative error of translation vector is 0.09923.

Appendix 3
Nonlinear Algorithm Using Point Correspondences

Simulation 7:

Ground Truth: The rotation angle is (in degrees):

30.00000000	0.00000000	0.00000000

The translation vector is:

0.00000000	5.00000000	20.00000000

The rotation matrix R is:

$0.1000000d + 01$	$0.0000000d + 00$	$0.0000000d + 00$
$0.0000000d + 00$	$0.8660254d + 00$	$-0.5000000d + 00$
$0.0000000d + 00$	$0.5000000d + 00$	$0.8660254d + 00$

The corresponding ($n1$, $n2$, $n3$), theta form is:
the directional is:

$n1 = 1.000000$ $n2 = 0.000000$ $n3 = 0.000000$

the rotation angle is:

beta = 29.958677

Input Data (without noise):

	x	y	z
PC 1:			
3-D-point	0.0000000000	5.0000000000	0.0000000000
2-D-point	0.0000000000	0.4146723120	1.0000000000
PC 2:			
3-D-point	6.0000000000	−13.0000000000	−1.0000000000
2-D-point	0.4749099307	−0.4557813699	1.0000000000
PC 3:			
3-D-point	8.0000000000	10.0000000000	1.5000000000
2-D-point	0.3041936351	0.4909021382	1.0000000000

Computation Result (0.0 pixel noise added):

The number of iterations: 6.
Three rotation angle are:

30.00000 0.00000 0.00000

The rotation matrix is:

1.00000	0.00000	0.00000
0.00000	0.86603	−0.50000
0.00000	0.50000	0.86603

The 2-D–3-D translation vector is:

0.00000 5.00000 20.00000

The relative error of the rotation matrix is 0.00000.
The relative error of the translation vector is 0.00000.

Simulation 8 (0.5 pixel noise added):
Computation result:

The number of iterations: 6.
From 3-D to 2-D, three rotation angle are:

29.66472 −0.51237 0.64961

The rotation matrix is:

0.99983	0.01671	−0.00443
−0.01671	0.86877	−0.49490
−0.00443	0.49489	0.86894

The 2-D to 3-D translation vector is:

−0.00530 5.06940 19.95893

The relative error of rotation matrix is 0.01490.
The relative error of translation vector is 0.00392.

Simulation 9 (2.0 pixel noise added):
Computation result:

The number of iterations: 6.
From 3-D to 2-D, three rotation angles are:

28.67375 −2.17834 2.73301

The rotation matrix is:

0.99687	0.07129	−0.01824
−0.07129	0.87429	−0.47947
−0.01824	0.47928	0.87737

The 2-D to 3-D translation vector is:

−0.02586 5.28413 19.75410

The relative error of rotation matrix is 0.06295.
The relative error of translation vector is 0.01827.

Appendix 4
Linear Algorithm Using Point Correspondences

Simulation 10:
Input Data (without noise):

	x	*y*	*z*
PC 1:			
3-D-point	0.0000000000	5.0000000000	0.0000000000
2-D-point	0.0000000000	0.4146723120	1.0000000000
PC 2:			
3-D-point	6.0000000000	−13.0000000000	−1.0000000000
2-D-point	0.4749099307	−0.4557813699	1.0000000000
PC 3:			
3-D-point	8.0000000000	10.0000000000	1.5000000000
2-D-point	0.3041936351	0.4909021382	1.0000000000
PC 4:			
3-D-point	4.0000000000	−6.0000000000	0.5000000000
2-D-point	0.2294497267	−0.0255923879	1.0000000000
PC 5:			
3-D-point	−5.0000000000	−5.0000000000	0.2000000000
2-D-point	−0.2829141617	0.0322450273	1.0000000000
PC 6:			
3-D-point	−10.0000000000	6.0000000000	−0.2000000000
2-D-point	−0.4380816508	0.4510555450	1.0000000000

Computation Result (0.0 pixel noise added):

The rotation matrix of the camera is:

1.00000	0.00000	0.00000
0.00000	0.86603	−0.50000
0.00000	0.50000	0.86603

The translation vector of the camera is:

0.00000 5.00000 20.00000

The relative error of rotation matrix is 0.00000.
The relative error of translation vector is 0.00000.

Simulation 11 (0.5 pixel noise added):
Computation Result: The rotation matrix of the camera is:

0.98726	0.00100	−0.00634
0.00410	0.85793	−0.48864
−0.00028	0.49713	0.89627

The translation vector of the camera is:

−0.01031	4.96164	19.76145

The relative error of rotation matrix is 0.02112.
The relative error of translation vector is 0.01173.

Simulation 12 (2.0 pixel noise added):
Computation Result: The rotation matrix of the camera is:

0.94463	0.00411	−0.02427
0.01632	0.82960	−0.45353
−0.00076	0.48560	0.98846

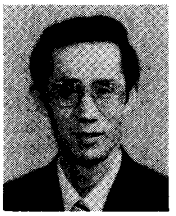
The translation vector of the camera is:

−0.03898	4.82216	18.95542

The relative error of rotation matrix is 0.08684.
The relative error of translation vector is 0.05143.

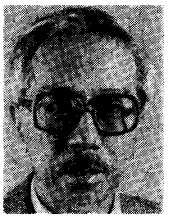
REFERENCES

[1] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.
[2] M. Keller and G. C. Tewinkel, "Space resection in photogrammetry," U. S. Coast and Geodetic Survey, ESSA Tech. Rep. 32, 1966.
[3] P. R. Wolf, *Elements of Photogrammetry*. New York: McGraw-Hill, 1974.
[4] S. Ganapathy, "Decomposition of transformation matrices for robot vision," in *Proc. Int. Conf. Robotics and Automation*, 1984, pp. 130–139.
[5] R. K. Lenz and R. Y. Tsai, "Calibrating a Cartesian robot with eye-hand configuration independent of eye-to-hand relationship," in *Proc. Computer Vision and Pattern Recognition*, Ann Arbor, MI, June 5–9, 1988, pp. 67–75.
[6] Y. C. Liu and T. S. Huang, "Estimation of rigid body motion using straight line correspondences," in *Proc. IEEE Workshop Motion*, Kiawah Island, SC, May 7–9, 1986, pp. 47–52.
[7] ——, "A linear algorithm for motion estimation using straight line correspondences," *Comput. Vision Graphics Image Processing*, vol. 44, no. 1, pp. 35–57, Oct. 1988.
[8] R. Y. Tsai and R. K. Lenz, "Real time versatile robotics hand/eye calibration using 3D machine vision," in *Proc. Int. Conf. Robotics and Automation*, Philadelphia, PA, Apr. 24–29, 1988.
[9] D. E. Whitney, C. A. Lozinski, and J. M. Rourke, "Industrial robot forward calibration method and results," *J. Dynamic Syst., Meas., Contr.*, 1986.
[10] P. R. Judd and A. B. Knasinski, "A technique to calibrate industrial robots with experimental verification," in *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, 1987.
[11] H. H. Chen, "Locating polyhedral objects with dextrons hand by straight finger motion," submitted to *IEEE Robotics and Automation Conf.*, Scotsdale, AZ, May 1989.



Yuncai Liu was born in China in 1948. He received the B.S. and M.S. degrees in electrical engineering from Shandong University, China, in 1974 and 1981, respectively.

He was an Electrical Engineer at the Shandong TV Bureau, China, from 1974 to 1978 and an Assistant Professor at Shandong University, China, from 1981 to 1984. In 1985, he joined the computer vision group of the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. Currently, he is a Research Assistant at that Laboratory and is working toward the Ph.D. degree in electrical engineering. His research interests are computer vision, image processing, and artificial intelligence.



Thomas S. Huang (S'61–M'63–SM'76–F'79) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, China, and the M.S. and Sc.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge.

He was on the Faculty of the Department of Electrical Engineering at M.I.T. from 1963 to 1973, and on the Faculty of the School of Electrical Engineering and Director of its Laboratory for Information and Signal Processing at Purdue University from 1973 to 1980. In 1980, he joined the University of Illinois at Urbana-Champaign, where he is now Professor of Electrical and Computer Engineering and Research Professor at the Coordinated Science Laboratory. During his sabattical leaves, he has worked at M.I.T. Lincoln Laboratory, IBM Thomas J. Watson Research Center, and the Rheinishes Landes Museum in Bonn, West Germany, and held visiting Professor positions at the Swiss Institutes of Technology in Zürich and Lausanne, the University of Hannover in West Germany, and INRS-Telecommunications of the University of Quebec in Montreal, Canada. He has served as a consultant to numerous industrial firms and government agencies both in the U.S. and abroad. His professional interests lie in the broad area of information technology, especially the transmission and processing of multidimensional signals. He has published 10 books, and over 200 papers on network theory, digital filtering, image processing, and computer vsion.

Dr. Huang is a Fellow of the Optical Society of America. He received a Guggenhem Fellowship (1971–1972), an A. V. Humboldt Foundation Senior U.S. Scientist Award (1976–1977), and a Fellowship from the Japan Society for the Promotion of Science (1986). He is an Editor of the international journal *Computer Vision, Graphics, and Image Processing*, Editor of the *Springer Series in Information Sciences*, published by Springer-Verlag, and Editor of the *Research Annual Series on Advances in Computer Vision and Image Processing* published by JAI Press.

Olivier Faugeras (S'76–M'76) is Director of Research at INRIA (National Institute for Research in Computer Science and Control Theory) where he leads the Robotics and Computer Vision Project. His research interests include computer vision, robotics, shape representation, computational geometry, and architecture for vision. In addition, he is a Lecturer in Applied Mathematics at the Ecole Polytechnique in Palaiseau where he teaches computer vision and computational geometry. He also teaches computer vision and robotics in the "Magistère de Mathématique et Informatique" at the Ecole Normale Supérieure de la rue d'Ulm and at l'Ecole Polytechnique Fédérale of Lausanne in Switzerland.

Dr. Faugeras is Associate Editor of several international scientific journals, including: IEEE TPAMI, *International Journal of Computer Vision, Internation Journal of Robotics Research, Pattern Recognition Letters, Signal Processing*, and *Robotics and Autonomous Systems*. In April 1989, he received the "Institut de France—Fondation Fiat" prize for his work in Vision and Robotics from the French Science Academy.