

# Iterative Pose Computation from Line Correspondences\*

Stéphane Christy and Radu Horaud

GRAVIR-IMAG &amp; INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 Montbonnot, Saint-Martin, France

Received October 6, 1997; accepted July 7, 1998

**This paper presents a method for estimating the position and orientation of a camera with respect to a known 3-D object from line correspondences. The main idea of the method is to estimate a pose with either a weak perspective or a paraperspective camera model and to improve this pose iteratively. At convergence the result is compatible with a perspective camera model. This iterative improvement of a linear (affine) camera model has already been used for points but has never been extended to lines. Known methods which compute pose from line correspondences deal with a set of nonlinear equations which are solved either in closed-form or using minimization techniques. These methods have to deal with multiple solutions. In contrast our method starts with a solution which is very close to the true solution and converges in very few iterations (typically three to five iterations). The rank analysis of the linear system to be solved at each iteration allows us to characterize geometric configurations which defeat the algorithm.** © 1999 Academic Press

## 1. INTRODUCTION AND BACKGROUND

The problem of object pose from 2-D to 3-D feature correspondences has received a lot of attention in the past two decades. A large majority of the proposed methods use point correspondences [2, 5, 8, 10, 13]. However, from a practical point of view it is often advantageous to use lines. Indeed, line detection is more accurate and more reliable than points and line matching is more robust than point matching with respect to partial occlusions. Moreover, pose can be estimated from lines without finding their endpoints.

Nevertheless, the perspective camera model is nonlinear and hence pose estimation from point and/or line correspondences is nonlinear as well. With three line correspondences Dhome *et al.* [3] and Chen [1] showed that the solutions are given by an eight-degree equation in one unknown. If the three lines meet at a common point, the solution is given by a fourth-degree equation in one unknown [6] which degenerates to a second-degree equation if the 3-D lines are mutually orthogonal. For more than three lines closed form solutions become unpractical. Both [9] and [12] showed that two classes of methods are possible with lines: rotation then translation and rotation and translation. Both

these two classes lead to nonlinear minimization techniques. These techniques work well provided that an initial solution is provided.

Recently, Dementhon and Davis [2] proposed a method for determining the pose of a 3-D object with respect to a camera from 3-D to 2-D point correspondences. The method consists of iteratively improving the pose computed with a weak perspective camera model to converge, at the limit, to a pose estimation computed with a perspective camera model. Oberkampff *et al.* [11] extended this method to coplanar sets of 3-D points and Horaud *et al.* [8] established the link between perspective and paraperspective for point correspondences. They showed that the *iterative paraperspective method* has better convergence properties than the *iterative weak perspective method* both in terms of rate of convergence and of number of iterations.

In this paper we extend these methods, i.e., [2, 8, 11], to line correspondences. We establish the basic equations linking perspective to weak perspective and to paraperspective and we embed these equations into an algorithm. Moreover, we study line configurations which defeat this algorithm. Finally, we analyze the convergence of the algorithm, we study the accuracy of pose computation in the presence of image noise, and we compare point-based linear and nonlinear methods with line-based linear and nonlinear methods.

## 2. CAMERA MODELS

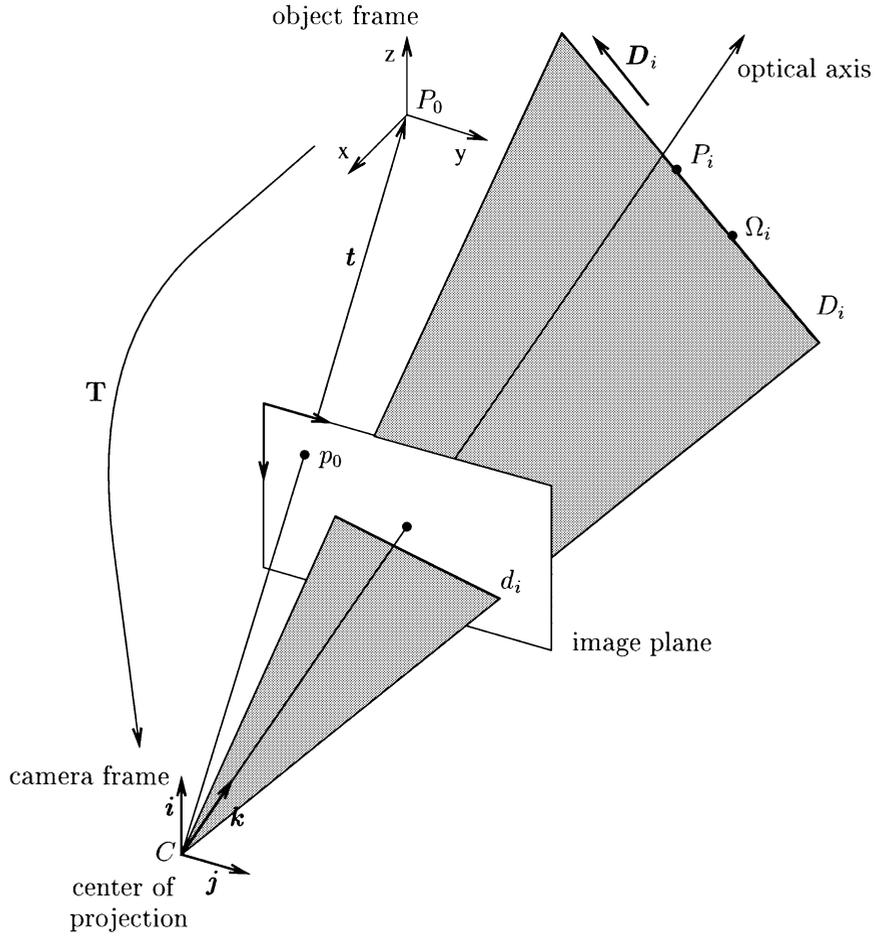
Figure 1 shows the general setup of the problem. The origin of the 3-D frame is an object point  $P_0$ . A 3-D (or object) line—denoted by  $D_i$ —is represented by a reference point  $\Omega_i$  and a direction  $D_i$ . Let us denote by  $(i, j, k)$  the rows of matrix  $\mathbf{R}$  and by  $\mathbf{t} = (t_x, t_y, t_z)$  the translation vector, both defining the rigid transformation between the object frame and the camera frame:

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}.$$

### 2.1. Perspective Camera Model

We assume that the camera is calibrated and hence image coordinates can be replaced by unitless camera coordinates. The  $3 \times 4$  matrix describing the projection of the 3-D Euclidean

\* This work has been supported by Société Aérospatiale and by DGA/DRET.



**FIG. 1.** This figure shows the general setup. The point  $P_0$  is the reference point of the object frame and its projection is  $p_0(x_0, y_0)$ . A 3-D line is represented by a reference point  $\Omega_i$  and a direction vector  $D_i$  expressed in the object frame ( $\Omega_i \perp D_i$ ). The projection of the 3-D line  $D_i$  is  $d_i$ .

space onto the 2-D image is in this case

$$\mathbf{M}_p = (\mathbf{R} \quad \mathbf{t}).$$

With the notations

$$\mathbf{I} = \frac{\mathbf{i}}{t_z} \quad \mathbf{J} = \frac{\mathbf{j}}{t_z} \quad \mathbf{K} = \frac{\mathbf{k}}{t_z} \quad x_0 = \frac{t_x}{t_z} \quad \text{and} \quad y_0 = \frac{t_y}{t_z}$$

we obtain

$$\mathbf{M}_p = \begin{pmatrix} \mathbf{I}^T & x_0 \\ \mathbf{J}^T & y_0 \\ \mathbf{K}^T & 1 \end{pmatrix}.$$

## 2.2. Weak Perspective Camera Model

Weak perspective may well be viewed as a linear approximation of full perspective. It is well known that in this case the

projection matrix becomes

$$\mathbf{M}_{wp} = \begin{pmatrix} \mathbf{I}^T & x_0 \\ \mathbf{J}^T & y_0 \\ \mathbf{0}^T & 1 \end{pmatrix}. \quad (1)$$

## 2.3. Paraperspective Camera Model

Paraperspective is a first-order Taylor expansion of perspective projection [8] and the projection matrix writes

$$\mathbf{M}_{pp} = \begin{pmatrix} \mathbf{I}_p^T & x_0 \\ \mathbf{J}_p^T & y_0 \\ \mathbf{0}^T & 1 \end{pmatrix} \quad (2)$$

with

$$\mathbf{I}_p = \mathbf{I} - x_0 \mathbf{K} \quad (3)$$

$$\mathbf{J}_p = \mathbf{J} - y_0 \mathbf{K}. \quad (4)$$

### 3. POSE EQUATIONS FROM LINE CORRESPONDENCES

The goal of pose estimation is to compute the orientation ( $\mathbf{i}$ ,  $\mathbf{j}$ ,  $\mathbf{k}$ ) and the translation ( $t_x$ ,  $t_y$ ,  $t_z$ ) of the camera frame with respect to the object frame. Equivalently one may compute  $\mathbf{I}$ ,  $\mathbf{J}$ ,  $x_0$ , and  $y_0$  in the case of weak perspective [2] or  $\mathbf{I}_p$ ,  $\mathbf{J}_p$ ,  $x_0$ , and  $y_0$  in the case of paraperspective [8].

#### 3.1. Perspective Model

Let us denote by  $P_i$  a 3-D point belonging to the 3-D line  $D_i$ . Thus, we may write

$$P_i = \Omega_i + \lambda D_i.$$

The projection of this point onto the image verifies

$$\begin{pmatrix} s p_i \\ s \end{pmatrix} = \mathbf{M}_p \begin{pmatrix} P_i \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} \cdot \Omega_i + x_0 \\ \mathbf{J} \cdot \Omega_i + y_0 \\ 1 + \eta_i \end{pmatrix} + \lambda \begin{pmatrix} \mathbf{I} \cdot D_i \\ \mathbf{J} \cdot D_i \\ \mu_i \end{pmatrix} \quad (5)$$

with

$$\eta_i = \mathbf{K} \cdot \Omega_i \quad \text{and} \quad \mu_i = \mathbf{K} \cdot D_i.$$

Moreover, point  $p_i$  is constrained to lie onto an image line and its coordinates verify  $a_i x + b_i y + c_i = 0$ . By substitution in the previous equation and after grouping terms we obtain

$$\begin{aligned} a_i \mathbf{I} \cdot \Omega_i + b_i \mathbf{J} \cdot \Omega_i + a_i x_0 + b_i y_0 + c_i (1 + \eta_i) \\ + \lambda (a_i \mathbf{I} \cdot D_i + b_i \mathbf{J} \cdot D_i + c_i \mu_i) = 0. \end{aligned}$$

This equation is verified for all points  $P_i$  of the 3-D line, i.e., for all values of  $\lambda$ . Therefore we obtain two equations:

$$a_i \mathbf{I} \cdot \Omega_i + b_i \mathbf{J} \cdot \Omega_i + a_i x_0 + b_i y_0 + c_i (1 + \eta_i) = 0 \quad (6)$$

$$a_i \mathbf{I} \cdot D_i + b_i \mathbf{J} \cdot D_i + c_i \mu_i = 0. \quad (7)$$

The unknowns are the pose parameters  $\mathbf{I}$ ,  $\mathbf{J}$ ,  $x_0$ , and  $y_0$  as well as  $\eta_i$  and  $\mu_i$  which encapsulate the perspective effect. For  $n$  correspondences, we have  $2n$  equations.

Equivalently, we can write these equations with  $\mathbf{I}_p$  and  $\mathbf{J}_p$ . From Eqs. (3) and (4) we have  $\mathbf{I} = \mathbf{I}_p + x_0 \mathbf{K}$  and  $\mathbf{J} = \mathbf{J}_p + x_0 \mathbf{K}$  and by substitution in Eqs. (6) and (7), we obtain

$$a_i \mathbf{I}_p \cdot \Omega_i + b_i \mathbf{J}_p \cdot \Omega_i + (a_i x_0 + b_i y_0 + c_i)(1 + \eta_i) = 0 \quad (8)$$

$$a_i \mathbf{I}_p \cdot D_i + b_i \mathbf{J}_p \cdot D_i + (a_i x_0 + b_i y_0 + c_i) \mu_i = 0. \quad (9)$$

#### 3.2. Weak Perspective

The equations that link a 3-D line to its projection for weak perspective can be easily derived from above. Indeed, from the

expression of the weak perspective projection matrix given by Eq. (1) we obtain the following equations:

$$a_i \mathbf{I} \cdot \Omega_i + b_i \mathbf{J} \cdot \Omega_i + a_i x_0 + b_i y_0 + c_i = 0 \quad (10)$$

$$a_i \mathbf{I} \cdot D_i + b_i \mathbf{J} \cdot D_i = 0. \quad (11)$$

Therefore, by setting  $\eta_i = \mu_i = 0$  in Eqs. (6) and (7), we obtain the weak perspective equations of 2-D to 3-D line matching.

#### 3.3. Paraperspective

Similarly, using the paraperspective projection matrix given by Eq. (2) we obtain the following equations:

$$a_i \mathbf{I}_p \cdot \Omega_i + b_i \mathbf{J}_p \cdot \Omega_i + a_i x_0 + b_i y_0 + c_i = 0 \quad (12)$$

$$a_i \mathbf{I}_p \cdot D_i + b_i \mathbf{J}_p \cdot D_i = 0. \quad (13)$$

Therefore, by setting  $\eta_i = \mu_i = 0$  in Eqs. (8) and (9), we obtain the paraperspective equations of 2-D to 3-D line matching.

## 4. POSE COMPUTATION

The sets of equations that we just established allow us to compute the pose parameters under various projection models:

- *Weak perspective*: Equations (10) and (11) are linear in  $\mathbf{I}$ ,  $\mathbf{J}$ ,  $x_0$ , and  $y_0$ . Since there are eight unknowns and each line correspondence provides two equations, a minimum of four lines are necessary to estimate these unknowns from which the pose parameters (rotation matrix and translation vector) can be easily estimated [2].

- *Paraperspective*: Equations (12) and (13) are linear in  $\mathbf{I}_p$ ,  $\mathbf{J}_p$ ,  $x_0$ , and  $y_0$  from which (under the same conditions as above) the pose parameters can be easily estimated as well [8].

- *Full perspective*: In this case the pose parameters can be estimated *iteratively* by solving either Eqs. (6) and (7) (weak perspective iterations) or Eqs. (8) and (9) (paraperspective iterations). In both cases the iterative algorithm is the following:

1. For all  $i$ ,  $i \in \{1 \dots n\}$ ,  $\eta_i = \mu_i = 0$ ;
2. Solve the overconstrained set of linear Eqs. (6) and (7) or (8) and (9);
3. Estimate the translation vector ( $t_x$ ,  $t_y$ , and  $t_z$ ) and the matrix with  $\mathbf{i}$ ,  $\mathbf{j}$ , and  $\mathbf{k}$  as row vectors; Orthogonalize this matrix to estimate the rotation  $\mathbf{R}$  [8];
4. For all  $i$ , compute

$$\eta_i = \frac{\mathbf{k} \cdot \Omega_i}{t_z} \quad \text{and} \quad \mu_i = \frac{\mathbf{k} \cdot D_i}{t_z}.$$

If the  $\eta_i$  and  $\mu_i$  computed at this iteration are equal to the  $\eta_i$  and  $\mu_i$  computed at the previous iteration then stop, otherwise go to step 2.

## 5. SOLVING THE LINEAR EQUATIONS

Both the weak perspective and paraperspective iterative algorithms need to solve an overconstrained system of linear equations, namely Eqs. (6) and (7) (weak perspective iterations) or Eqs. (8) and (9) (paraperspective iterations). In matrix form these equations can be written as

$$\underbrace{\mathbf{A}}_{2n \times 8} \underbrace{\mathbf{x}}_{8 \times 1} = \underbrace{\mathbf{b}}_{2n \times 1}. \quad (14)$$

More precisely, this matrix equation is

$$\begin{pmatrix} \dots & \dots & \dots & \dots \\ a_i \Omega_i^T & b_i \Omega_i^T & a_i & b_i \\ a_i \mathbf{D}_i^T & b_i \mathbf{D}_i^T & 0 & 0 \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \mathbf{I}^T \\ \mathbf{J}^T \\ x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} \dots \\ -c_i(1 + \eta_i) \\ -c_i \mu_i \\ \dots \end{pmatrix}.$$

Or, equivalently,

$$\begin{pmatrix} \dots & \dots & \dots & \dots \\ a_i \Omega_i^T & b_i \Omega_i^T & a_i(1 + \eta_i) & b_i(1 + \eta_i) \\ a_i \mathbf{D}_i^T & b_i \mathbf{D}_i^T & a_i \mu_i & b_i \mu_i \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \mathbf{I}_p^T \\ \mathbf{J}_p^T \\ x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} \dots \\ -c_i(1 + \eta_i) \\ -c_i \mu_i \\ \dots \end{pmatrix}.$$

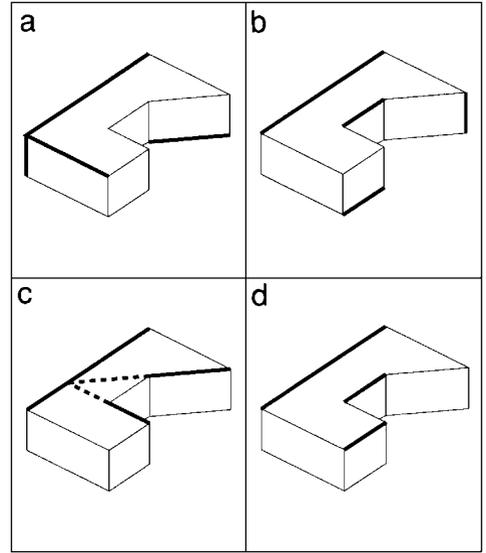
We recall that a 3-D line  $D_i$  is described by a 3-D point arbitrarily chosen along this line,  $\Omega_i$ , and a direction vector  $\mathbf{D}_i$ . The 2-D line matching this 3-D line is described by three scalars  $a_i$ ,  $b_i$ , and  $c_i$ .

The unknown vector  $\mathbf{x}$  has eight components. Therefore the  $2n \times 8$  measurement matrix  $\mathbf{A}$  must have a rank equal to 8.

### 5.1. Rank Analysis

We analyze under which geometric configurations matrix  $\mathbf{A}$  satisfies the rank constraint stated above and we identify the line configurations for which pose cannot be computed. Due to the fact that this rank analysis is based on the 3-D geometric configurations associated with known object lines, one can easily avoid configurations which defeat the method.

First, notice that the two equations associated with a line-match, (6) and (7)<sup>1</sup>, are independent. The only configuration for which these two equations are not independent is when the 3-D line passes through the center of projection. In such a case we must have  $\Omega_i = \alpha \mathbf{D}_i$  and the 2-D line is reduced to a point.



**FIG. 2.** This figure shows all the geometric configurations which defeat the pose computation method described in this paper. In the case of a noncoplanar configuration the only forbidden configuration is a pencil of three lines or more, whether they intersect in a common point (a) or they are parallel (b). In the case of a coplanar configuration three lines are sufficient but these three lines must not intersect in a common point (c) or be parallel (d).

Second, we consider two 3-D lines. These lines project onto two distinct image lines and hence there are two distinct projection planes associated with them; e.g., Fig. 1. Therefore two line correspondences have four independent equations associated with them.

Third, we consider three 3-D lines, Figs. 2a and 2b. If these lines intersect in a common point, the three associated projection planes form a pencil of planes and any plane in this pencil is a linear combination of the two other planes. Therefore, a pencil of lines, i.e., three or more parallel lines or three or more lines intersecting in a common point, will contribute as two lines.

Fourth, we consider four 3-D lines. A set of four lines is said to be in “general position” if the lines are not coplanar and if no three lines among them form a pencil of lines.

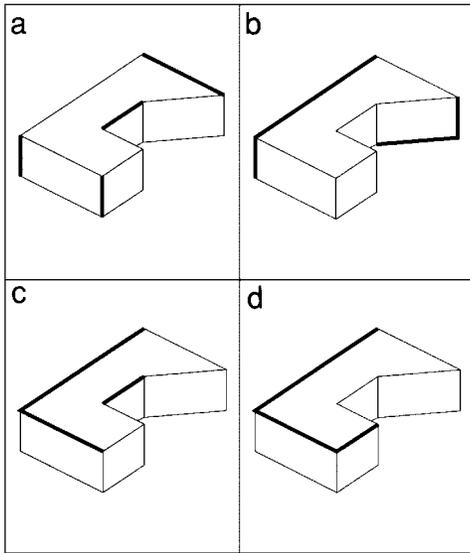
To conclude, the minimal configuration which allows pose computation is a set of four lines in general position. Figures 3a and 3b show a few examples of line configurations which can be used in conjunction with Eq. (14).

We are left with the case of coplanar object lines. It will be shown below that the coplanarity constraint can be explicitly taken into account such that three coplanar lines are sufficient to compute pose without ambiguity.

### 5.2. Coplanar Object Lines

Whenever the object lines lie in the same plane, the coplanarity constraint can be explicitly used to compute pose. As it will be shown below, the practical advantage is that a minimum of three lines are sufficient to estimate the pose parameters.

<sup>1</sup> Equations (8) and (9) are strictly equivalent to (6) and (7).



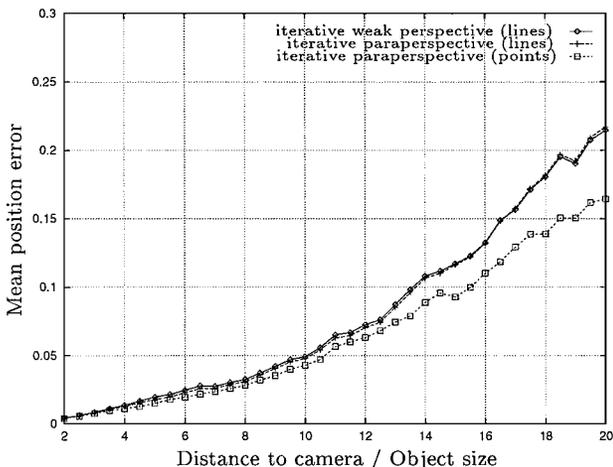
**FIG. 3.** This figure shows a few examples of noncoplanar, (a) and (b), and coplanar, (c) and (d), configurations which can be used to compute pose with either camera model described herein.

We consider the plane containing the object lines and let  $\mathbf{u}$  be the unit vector normal to this plane. Vectors  $\mathbf{I}$  and  $\mathbf{J}$  can be written as a linear combination of a vector belonging to this plane and of vector  $\mathbf{u}$ , namely,

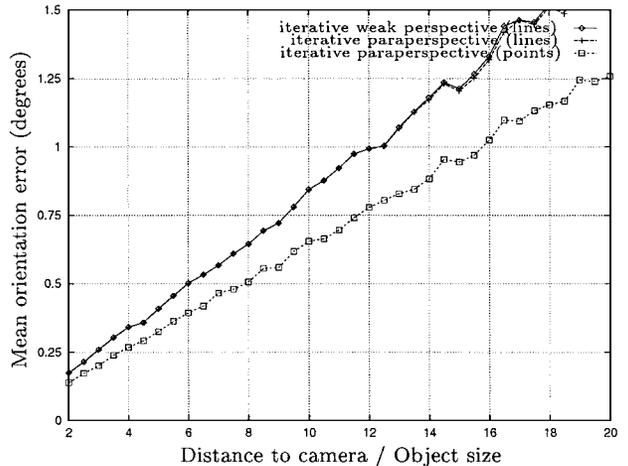
$$\mathbf{I} = \mathbf{I}_0 + \alpha \mathbf{u} \quad (15)$$

$$\mathbf{J} = \mathbf{J}_0 + \beta \mathbf{u}. \quad (16)$$

The main idea (introduced in [11]) is to modify Eq. (14) by replacing the unknowns  $\mathbf{I}$  and  $\mathbf{J}$  with  $\mathbf{I}_0$  and  $\mathbf{J}_0$  and by adding to the linear system two additional constraints, namely  $\mathbf{I}_0 \cdot \mathbf{u} = 0$  and  $\mathbf{J}_0 \cdot \mathbf{u} = 0$ . By substituting Eqs. (15) and (16) into Eqs. (6)



**FIG. 4.** Position error as a function of the distance camera-object in presence of gaussian noise from line (and point) correspondences (the object is shifted in respect to the optical axis).



**FIG. 5.** Orientation error as a function of the distance camera-object in presence of gaussian noise from line (and point) correspondences (the object is shifted in respect to the optical axis).

and (7) and by noticing that  $\mathcal{I}_i \cdot \mathbf{u} = 0$  and  $\mathcal{D}_i \cdot \mathbf{u} = 0$ , we obtain

$$\mathbf{A}' \mathbf{x}' = \mathbf{b}'$$

or

$$\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{u}^T & \mathbf{0}^T & 0 & 0 \\ \mathbf{0}^T & \mathbf{u}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_0 \\ \mathbf{J}_0 \\ x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \\ 0 \end{pmatrix}.$$

The vector  $\mathbf{u}$  is orthogonal to  $\mathbf{I}_0$  and  $\mathbf{J}_0$ , thus  $\mathbf{u} \cdot \mathbf{I}_0 = 0$  and  $\mathbf{u} \cdot \mathbf{J}_0 = 0$  are two more independent equations and the matrix  $\mathbf{A}'$  is of rank 8. Thus we obtain a solution for  $\mathbf{I}_0$ ,  $\mathbf{J}_0$ ,  $x_0$  and  $y_0$ :

$$\mathbf{x}' = (\mathbf{A}'^T \mathbf{A}')^{-1} \mathbf{A}'^T \mathbf{b}'.$$

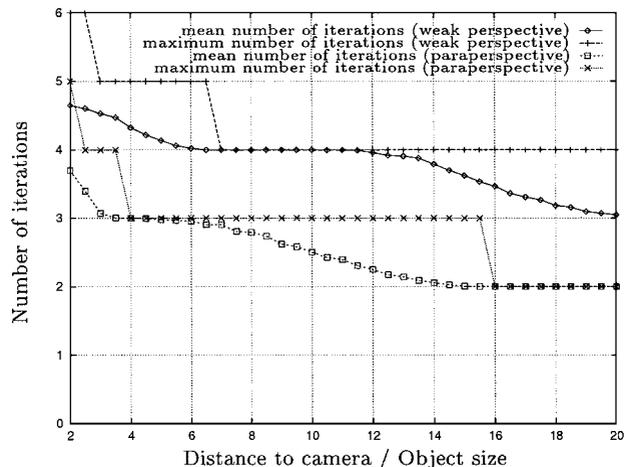
Clearly, due to the orthogonality between vector  $\mathbf{u}$  and the object lines, the two additional rows of matrix  $\mathbf{A}'$  are independent of the rows of matrix  $\mathbf{A}$ . Since  $\mathbf{A}'$  must have a rank equal to 8, it is sufficient that the rank of  $\mathbf{A}$  is equal to 6.

Therefore three lines (not mutually parallel and not intersecting at the same common point) are sufficient to solve for  $\mathbf{x}'$ . Figures 2c and 2d show two examples of coplanar line configurations which defeat the method and Figs. 3c and 3d show two examples of coplanar line configurations which can be used for pose computation using the method described in this section.

Finally, in order to estimate  $\mathbf{I}$  and  $\mathbf{J}$  from the estimated  $\mathbf{I}_0$  and  $\mathbf{J}_0$ , one must determine the scalars  $\alpha$  and  $\beta$ . This can be easily done by combining Eqs. (15) and (16) with the constraints  $\|\mathbf{I}\| = \|\mathbf{J}\|$  and  $\mathbf{I} \cdot \mathbf{J} = 0$ . For more details concerning the estimation of  $\alpha$  and  $\beta$  see [11] and [8].

## 6. EXPERIMENTS

In order to test the method we run a large number of simulations. Thus we tested both the iterative weak perspective and



**FIG. 6.** Number of iterations as a function of the distance camera-object in presence of gaussian noise from line correspondences (the object is shifted in respect to the optical axis).

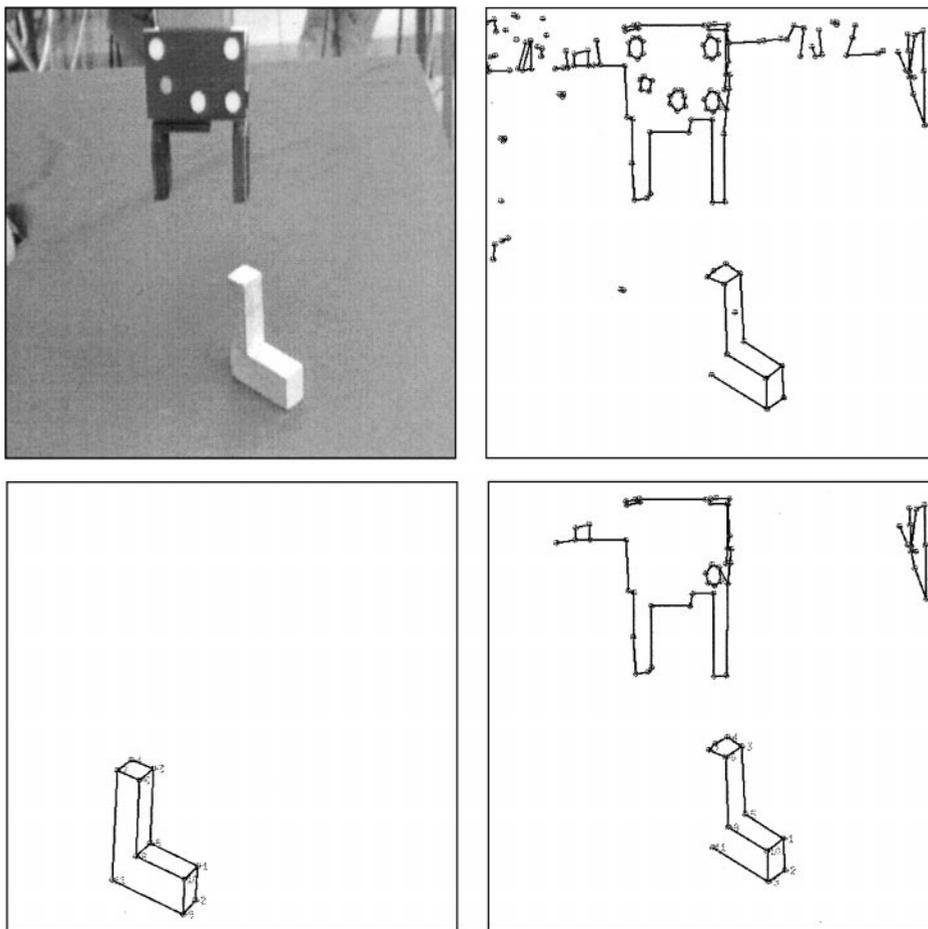
iterative paraperspective methods in the presence of image noise and as a function of the relative position and orientation of the 3-D data with respect to the camera. We studied:

- Object pose error (in position and orientation) as a function of the camera-object distance (Figs. 4 and 5);
- The convergence of the two iterative algorithms (Fig. 6).

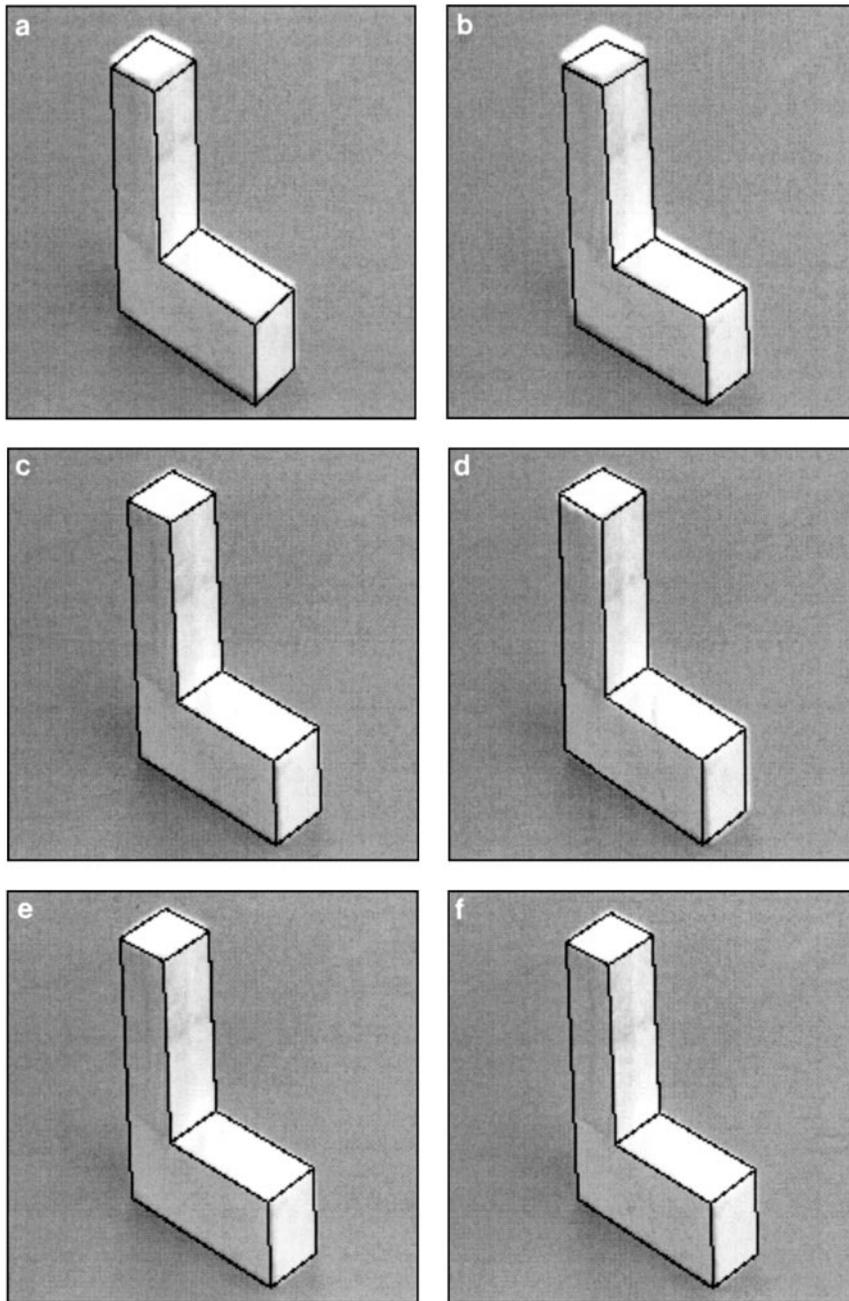
For the synthetic data, the parameters are the following:

- The intrinsic camera parameters are  $u_c = v_c = 256$ ,  $\alpha_u = \alpha_v = 1000$ .
- The 3-D model is made out of 18 lines.

We added gaussian noise (with standard deviation  $\sigma = 1$  pixel) to the image data, and we performed 500 random measures for each experiment. The rotation matrices for these 500 orientations are computed from Euler angles chosen in the range  $[0, 2\pi]$ . The object position is defined by a translation vector from the center of projection to the origin of the object frame. The rotation error is defined as being the rotation angle (in degrees) needed to align the object frame in the computed position with the theoretical



**FIG. 7.** Straight lines and junctions are extracted from the raw data (top). A wireframe representation of the object is matched against the set of lines and junctions (bottom). The result is a set of line-to-line and junction-to-junction correspondences.



**FIG. 8.** Pose computed with: (a) weak perspective and lines, (b) paraperspective and lines, (c) iterative weak perspective and lines, (d) iterative weak perspective and points, (e) nonlinear minimization and lines, (f) nonlinear minimization and points.

position. The position error is defined by the norm of the vector which represents the difference between the two translation vectors (the computed vector and the theoretical vector), divided by the object size. The abscissa of the graphic plots represents the  $z$  component of the translation vector divided by the object size.

The performance of these line-based algorithms is quite similar with the point-based ones thoroughly studied by Horaud *et al.* [8] and by Dementhon and Davis [2] (Figs. 4 and 5). However, we had not taken into account the fact that lines are ex-

tracted with a better accuracy than points. The iterative weak perspective method and the iterative paraperspective method converge to the same solution. This is explained because they rely on equivalent perspective equations. Nevertheless, the iterative paraperspective method converges faster than the iterative weak perspective algorithm because the pose obtained after the first iteration with the former is more accurate than the pose obtained with the latter algorithm. For all these configurations, convergence rate was 100%.

An example of application of pose computation is visual servoing. Within the framework of visual servoing, the camera is in the real-time feedback loop of a robot manipulator and the latter must be directed toward a target position in accordance with image changes. Variations in the image must therefore be transformed immediately into 3-D robot commands. If only one camera is being used, pose computation becomes crucial [4] and it must be performed in a few milliseconds.

An example of application of object pose within visual servoing is the grasping of a polyhedral object by a parallel-jaw gripper, [7]. A single camera observes both the object to be grasped and the gripper. In order to control the robot and guide the gripper toward the object, one must locate the object with respect to the camera, i.e., compute its pose. The raw image is segmented into junctions and lines and these lines and junctions are matched with a wireframe representation of the object. The result of matching is a list of point correspondences and a list of line correspondences, e.g., Fig. 7. Therefore pose computation can be performed either from point or from line correspondences. Figure 8 shows the results obtained with various algorithms:

- (a) weak perspective pose from line correspondences; that is, the result of the first iteration of the iterative weak perspective algorithm described in this paper;
- (b) paraperspective pose from line correspondences; that is, the result of the first iteration of the iterative paraperspective algorithm described in this paper;
- (c) perspective pose from line correspondences obtained with the iterative weak perspective algorithm;
- (d) perspective pose from point correspondences obtained with the iterative weak perspective algorithm;
- (e) perspective pose from line correspondences obtained with a nonlinear method [12], and
- (f) perspective pose from point correspondences obtained with the same nonlinear method.

In order to assess these results quantitatively we computed the error between the pose computed with the nonlinear methods and the pose computed with the iterative weak perspective methods. The results of this comparison are summarized in Table 1. In this table  $\mathbf{R}_{opt}$  and  $\mathbf{t}_{opt}$  correspond to the pose computed with a nonlinear method (Figs. 8e and 8f): the pose computed with

points is identical, in this case with the pose computed with lines. In this case the line-based iterative weak perspective algorithm performs better than the point-based one.

## 7. DISCUSSION

This paper extends the work on pose estimation from point matches [2, 8, 11] to line matches. Lines represent a good alternative to points whenever the vertices of a polyhedral object are occluded.

The methods described above (iterative weak perspective and iterative paraperspective) are able to deal with both noncoplanar and coplanar sets of 3-D lines. We studied the geometric configurations which defeat the method. The main advantage of the algorithm is that it is very fast regardless of the number of correspondences.

In theory, the paraperspective-based algorithm involves fewer iterations than the weak perspective one. However, because of the computational simplicity and ease of implementation, the latter should be preferred to the former.

## REFERENCES

1. H. Chen, Pose determination from line-to-plane correspondences: existence solutions and closed-form solutions, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(6), 1991, 530–541.
2. D. F. DeMenthon and L. S. Davis, Model-based object pose in 25 lines of code, *Int. J. Comput. Vision* **15**(1/2), 1995, 123–141.
3. M. Dhome, M. Richetin, J. T. Lapreste, and G. Rives, Determination of the attitude of 3D objects from a single perspective view, *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(12), 1989, 1265–1278.
4. B. Espiau, F. Chaumette, and P. Rives, A new approach to visual servoing in robotics, *IEEE Trans. Robotics Automat.* **8**(3), 1992, 313–326.
5. M. A. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Commun. Assoc. Comput. Mach.* **24**(6), 1981, 381–395.
6. R. Horaud, B. Conio, O. Le Boulleux, and B. Lacolle, An analytic solution for the perspective 4-point problem, *Comput. Vision Graphics Image Process.* **47**, 1989, 33–44.
7. R. Horaud, F. Dornaika, and B. Espiau, Visually guided object grasping, *IEEE Trans. Robotics Automat.* **14**(4), 1998, 525–532.
8. R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy, Object pose: The link between weak perspective, paraperspective, and full perspective, *Int. J. Comput. Vision* **22**(2), 1997, 173–189.
9. Y. Liu, T. S. Huang, and O. D. Faugeras, Determination of camera location from 2-d to 3-d line and point correspondences, *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(1), 1990, 28–37.
10. D. Lowe, Fitting parameterized three-dimensional models to images, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(5), 1991, 441–450.
11. D. Oberkampf, D. F. DeMenthon, and L. S. Davis, Iterative pose estimation using coplanar feature points, *Comput. Vision Image Understanding* **63**(3), 1996, 495–511.
12. T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham, Object pose from 2-D to 3-D point and line correspondences, *Int. J. Comput. Vision* **15**(3), 1995, 225–243.
13. J. S.-C. Yuan, A general photogrammetric method for determining object position and orientation, *IEEE Trans. Robotics Automat.* **5**(2), 1989, 129–142.

TABLE 1

A Comparison between the Pose Computed with Point Correspondences and the Pose Computed with Line Correspondences<sup>a</sup>

Iterative weak perspective	$\ \mathbf{R}_{opt} - \hat{\mathbf{R}}\ $	$\ \mathbf{t}_{opt} - \hat{\mathbf{t}}\ $
With points	0.031	10.2%
With lines	0.018	3.7%

<sup>a</sup> The optimal pose parameters were estimated using a nonlinear minimization method. The rotation error is the square root of the sum of the differences of the coefficients of the two matrices.