Iterative pose computation from line correspondences *

Stéphane Christy and Radu Horaud

GRAVIR-IMAG & INRIA Rhône-Alpes

655, avenue de l'Europe

38330 Montbonnot Saint-Martin FRANCE

Running head: Pose from line correspondences

Corresponding author: Radu Horaud

e-mail: Radu.Horaud@inrialpes.fr

phone: +33 4 76 61 52 26

fax: +33 4 76 61 52 10

Revised, 7 may 1998

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

Abstract

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 non-linear 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 3 to 5 iterations). The rank analysis of the linear system to be solved at each iteration allows us to characterize geometric configurations which defeat the algorithm.

List of symbols

$\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$, $\boldsymbol{I}$, $\boldsymbol{J}$, $\boldsymbol{K}$, $\boldsymbol{I}_p$, $\boldsymbol{J}_p$, $\boldsymbol{I}_0$, $\boldsymbol{J}_0$, $\boldsymbol{P}_i$, $\boldsymbol{p}_i$, $\boldsymbol{\Omega}_i$, $\boldsymbol{D}_i$, $\boldsymbol{t}$, $\boldsymbol{b}$, $\boldsymbol{u}$, $\boldsymbol{x}$ ..vectors (bold italic)

$\mathbf{R}$, $\mathbf{T}$, $\mathbf{M}_p$, $\mathbf{M}_{wp}$, $\mathbf{M}_{pp}$, $\mathbf{A}$ ..... matrix (normal bold)

a_i, b_i, c_i, η_i, μ_i, λ, t_x, t_y, t_z, x_0, y_0 ..... scalars (lowercase italic)

P_0, P_i, p_0, p_i, Ω_i ..... geometric points (italic)

D_i, d_i ..... geometric lines (italic)

$\boldsymbol{I}^T, \mathbf{R}^T$ ..... the transpose of $\boldsymbol{I}$, $\mathbf{R}$

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 [5, 10, 13, 2, 8]. 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 end-points.

Nevertheless, the perspective camera model is non-linear and hence pose estimation from point and/or line correspondences is non-linear 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 non linear 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. Oberkampf 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 then the *iterative weak perspective method* both it terms of rate of convergence and of number of iterations.

In this paper we extend these methods, i.e., [2, 11, 8] 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 non-linear methods with line-based linear and non-linear 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 Ω_i and a direction $\boldsymbol{D}_i$. Let's denote by $(\boldsymbol{i},\ \boldsymbol{j},\ \boldsymbol{k})$ the rows of matrix $\mathbf{R}$ and by $\boldsymbol{t} = (t_x, t_y, t_z)$ the translation vector, both defining the rigid transformation between the object frame and the camera frame:

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

2.1 Perspective camera model

We assume that the camera is calibrated and hence image coordinates can be replaced by unit-less camera coordinates. The 3×4 matrix describing the projection of the 3-D Euclidean space onto the 2-D image is in this case:

$$\mathbf{M}_p = \begin{pmatrix} \mathbf{R} & \boldsymbol{t} \end{pmatrix}$$

With the following notations:

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

we obtain:

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

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:

$$\mathbf{M}_{wp} = \begin{pmatrix} \boldsymbol{I}^T & x_0 \\ \boldsymbol{J}^T & y_0 \\ \boldsymbol{0}^T & 1 \end{pmatrix} \tag{1}$$

2.3 Paraperspective camera model

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

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

with

$$\boldsymbol{I}_p = \boldsymbol{I} - x_0 \boldsymbol{K} \tag{3}$$

$$\boldsymbol{J}_p = \boldsymbol{J} - y_0 \boldsymbol{K} \tag{4}$$

3 Pose equations from line correspondences

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

3.1 Perspective model

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

$$\boldsymbol{P}_i = \boldsymbol{\Omega}_i + \lambda \boldsymbol{D}_i$$

The projection of this point onto the image verifies:

$$\begin{pmatrix} s\boldsymbol{p}_i \\ s \end{pmatrix} = \mathrm{M}_p \begin{pmatrix} \boldsymbol{P}_i \\ 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} \cdot \boldsymbol{\Omega}_i + x_0 \\ \boldsymbol{J} \cdot \boldsymbol{\Omega}_i + y_0 \\ 1 + \eta_i \end{pmatrix} + \lambda \begin{pmatrix} \boldsymbol{I} \cdot \boldsymbol{D}_i \\ \boldsymbol{J} \cdot \boldsymbol{D}_i \\ \mu_i \end{pmatrix} \tag{5}$$

with

$$\eta_i = \boldsymbol{K} \cdot \boldsymbol{\Omega}_i \quad \text{and} \quad \mu_i = \boldsymbol{K} \cdot \boldsymbol{D}_i$$

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:

$$a_i \boldsymbol{I} \cdot \boldsymbol{\Omega}_i + b_i \boldsymbol{J} \cdot \boldsymbol{\Omega}_i + a_i x_0 + b_i y_0 + c_i(1 + \eta_i) + \lambda(a_i \boldsymbol{I} \cdot \boldsymbol{D}_i + b_i \boldsymbol{J} \cdot \boldsymbol{D}_i + c_i \mu_i) = 0$$

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

$$a_i \boldsymbol{I} \cdot \boldsymbol{\Omega}_i + b_i \boldsymbol{J} \cdot \boldsymbol{\Omega}_i + a_i x_0 + b_i y_0 + c_i(1 + \eta_i) = 0 \tag{6}$$

$$a_i \boldsymbol{I} \cdot \boldsymbol{D}_i + b_i \boldsymbol{J} \cdot \boldsymbol{D}_i + c_i \mu_i = 0 \tag{7}$$

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

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

$$a_i \boldsymbol{I}_p \cdot \boldsymbol{\Omega}_i + b_i \boldsymbol{J}_p \cdot \boldsymbol{\Omega}_i + (a_i x_0 + b_i y_0 + c_i)(1 + \eta_i) = 0 \tag{8}$$

$$a_i \boldsymbol{I}_p \cdot \boldsymbol{D}_i + b_i \boldsymbol{J}_p \cdot \boldsymbol{D}_i + (a_i x_0 + b_i y_0 + c_i)\mu_i = 0 \tag{9}$$

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 equation (1) we obtain the following equations:

$$a_i \boldsymbol{I} \cdot \boldsymbol{\Omega}_i + b_i \boldsymbol{J} \cdot \boldsymbol{\Omega}_i + a_i x_0 + b_i y_0 + c_i = 0 \quad (10)$$

$$a_i \boldsymbol{I} \cdot \boldsymbol{D}_i + b_i \boldsymbol{J} \cdot \boldsymbol{D}_i = 0 \quad (11)$$

Therefore, by setting $\eta_i = \mu_i = 0$ in equations (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 equation (2) we obtain the following equations:

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

$$a_i \boldsymbol{I}_p \cdot \boldsymbol{D}_i + b_i \boldsymbol{J}_p \cdot \boldsymbol{D}_i = 0 \quad (13)$$

Therefore, by setting $\eta_i = \mu_i = 0$ in equations (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 $\boldsymbol{I}$, $\boldsymbol{J}$, x_0, and y_0. Since there are eight unknowns and each line correspondence provides two equations, a minimum of 4 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 $\boldsymbol{I}_p$, $\boldsymbol{J}_p$, 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 equations (6) and (7) (weak perspective iterations) or equations (8) and (9) (paraperspective iterations). In both cases the iterative algorithm is the following:

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

$$\eta_i = \frac{\boldsymbol{k} \cdot \boldsymbol{\Omega}_i}{t_z} \quad \text{and} \quad \mu_i = \frac{\boldsymbol{k} \cdot \boldsymbol{D}_i}{t_z}$$

  If the η_i and μ_i computed at this iteration are equal to the η_i and μ_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), (7) (weak perspective iterations) or eqs. (8), (9) (paraperspective iterations). In matrix form these equations can be written as:

$$\underbrace{\mathbf{A}}_{2n\times 8}\ \underbrace{\boldsymbol{x}}_{8\times 1} = \underbrace{\boldsymbol{b}}_{2n\times 1} \tag{14}$$

More precisely, this matrix equation is:

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

Or, equivalently:

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

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

The unknown vector $\boldsymbol{x}$ has 8 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) [1], 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 $\boldsymbol{\Omega}_i = \alpha \boldsymbol{D}_i$ and the 2-D line is reduced to a point.

[1] Equations (8) and (9) are strictly equivalent to (6) and (7)

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. Figure 1. Therefore two line correspondences have 4 independent equations associated with them.

Third, we consider three 3-D lines, Figure 2 (a) and (b). 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 co-planar 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. Figure 3 (a) and (b) shows a few examples of line configurations which can be used in conjunction with equation (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 3 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.

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

combination of a vector belonging to this plane and of vector $\boldsymbol{u}$, namely:

$$\boldsymbol{I} = \boldsymbol{I}_0 + \alpha\, \boldsymbol{u} \tag{15}$$

$$\boldsymbol{J} = \boldsymbol{J}_0 + \beta\, \boldsymbol{u} \tag{16}$$

The main idea (introduced in [11]) is to modify equation (14) by replacing the unknowns $\boldsymbol{I}$ and $\boldsymbol{J}$ with $\boldsymbol{I}_0$ and $\boldsymbol{J}_0$ and by adding to the linear system two additional constraints, namely $\boldsymbol{I}_0 \cdot \boldsymbol{u} = 0$ and $\boldsymbol{J}_0 \cdot \boldsymbol{u} = 0$. By substituting equations (15) and (16) into equations (6) and (7) and by noticing that $\boldsymbol{\Omega}_i \cdot \boldsymbol{u} = 0$ and $\boldsymbol{D}_i \cdot \boldsymbol{u} = 0$, we obtain:

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

or:

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

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

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

Clearly, due to the orthogonality between vector $\boldsymbol{u}$ 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 $\boldsymbol{x}'$. Figure 2 (c) and (d) show two examples of coplanar line configurations which defeat the method and Figure 3 (c) and (d) 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 $\boldsymbol{I}$ and $\boldsymbol{J}$ from the estimated $\boldsymbol{I}_0$ and $\boldsymbol{J}_0$, one must determine the scalars α and β. This can be easily done by combining equations (15) and (16) with the constraints $\|\boldsymbol{I}\| = \|\boldsymbol{J}\|$ and $\boldsymbol{I} \cdot \boldsymbol{J} = 0$. For more details concerning the estimation of α and β 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 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 (Figures 4 and 5);
- The convergence of the two iterative algorithms (Figure 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 3D 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 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] (Figures 4 and 5). However, we had not taken into account the fact that lines are extracted 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 towards 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 mili-seconds.

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 towards 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.,

Figure 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 non-linear method [12], and
- (f) – perspective pose from point correspondences obtained with the same non-linear method.

In order to asses these results quantitatively we computed the error between the pose computed with the non-linear methods and the pose computed with the iterative weak perspective methods. The results of this comparison are summarized on Table 1. In this table $\mathbf{R}_{opt}$ and $\boldsymbol{t}_{opt}$ correspond to the pose computed with a non-linear method (Figure 8 (e) and (f)): 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, 11, 8] 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 non coplanar 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 Transactions on Pattern Analysis and Machine Intelligence*, 13(6):530–541, June 1991.

[2] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, 1995.

[3] M. Dhome, M. Richetin, J.T. Lapreste, and G. Rives. Determination of the Attitude of 3D Objects from a Single Perspective View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

[4] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[5] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[6] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, July 1989.

[7] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. *IEEE Transactions on Robotics and Automation*, 1998. To appear.

[8] R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy. Object pose: The link between weak perspective, paraperspective, and full perspective. *International Journal of Computer Vision*, 22(2):173–189, March 1997.

[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 Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990.

[10] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[11] D. Oberkampf, D. F. DeMenthon, and L. S. Davis. Iterative pose estimation using coplanar feature points. *Computer Vision and Image Understanding*, 63(3):495–511, May 1996.

[12] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Object pose from 2-D to 3-D point and line correspondences. *International Journal of Computer Vision*, 15(3):225–243, July 1995.

[13] J. S.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Transactions on Robotics and Automation*, 5(2):129–142, April 1989.

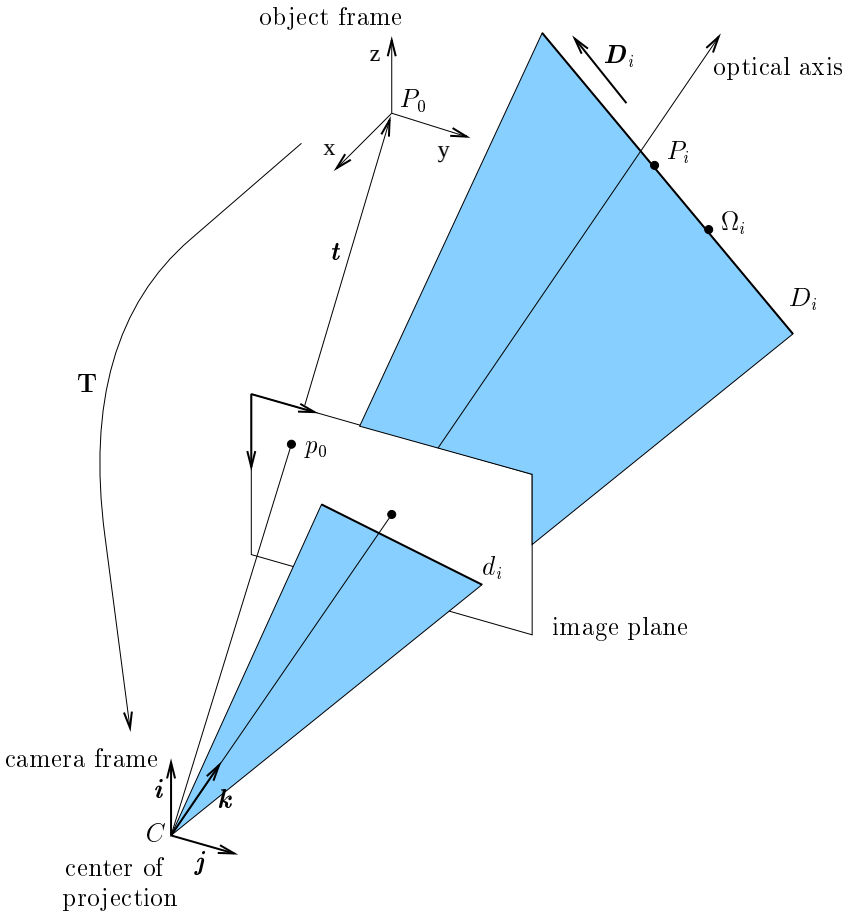

Figure 1: This figure shows the general setup. The point P_0 is the reference point of the object frame and is projection is $p_0(x_0, y_0)$. A 3-D line is represented by a reference point Ω_i and a direction vector $\boldsymbol{D}_i$ expressed in the object frame ($\boldsymbol{\Omega}_i \perp \boldsymbol{D}_i$). The projection of the 3-D line D_i is d_i.

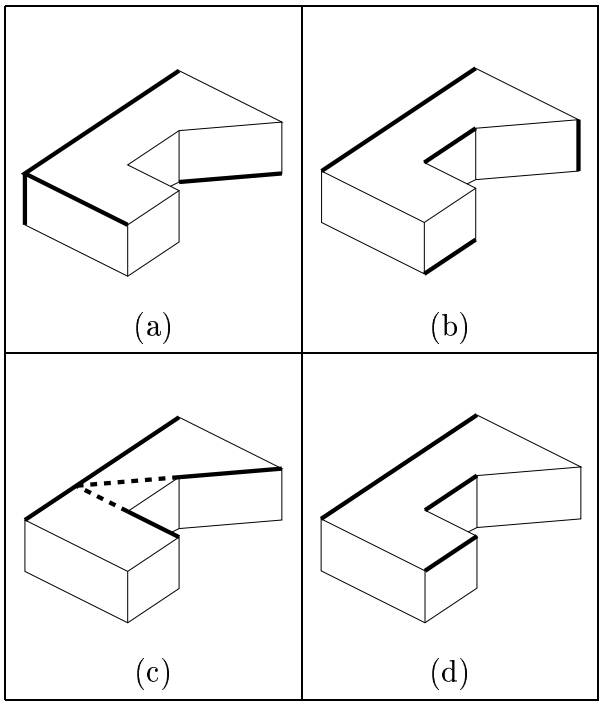

Figure 2: This figure shows all the geometric configurations which defeat the pose computation method described in this paper. In the case of a non coplanar configuration the only forbidden configuration is a pencil of three lines or more, whether they intersect in a common point (a) or they are parallel (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).

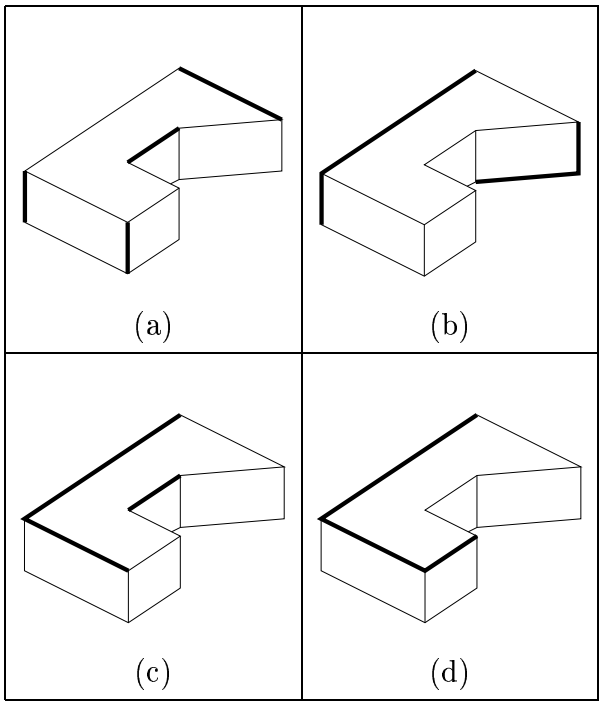

Figure 3: This figure shows a few examples of non-coplanar, (a) and (b) and coplanar (c) and (d) configurations which can be used to compute pose with either camera model described herein.

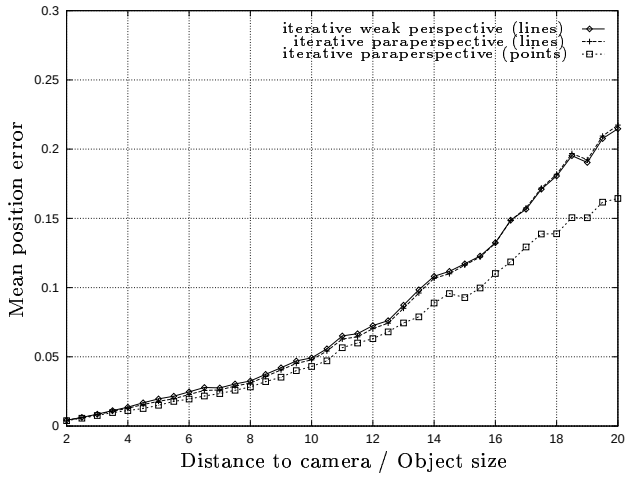

Figure 4: Position error as a function of the distance camera-object in presence of gaussian noise from line (and point) correspondences (the object is shifted in respect to the optical axis).

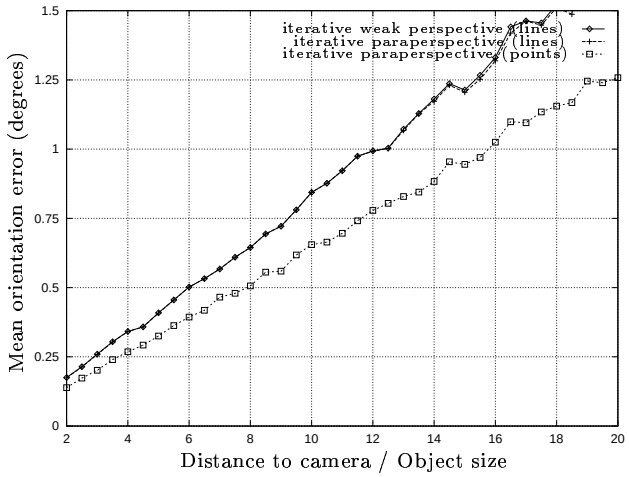

Figure 5: Orientation error as a function of the distance camera-object in presence of gaussian noise from line (and point) correspondences (the object is shifted in respect to the optical axis).

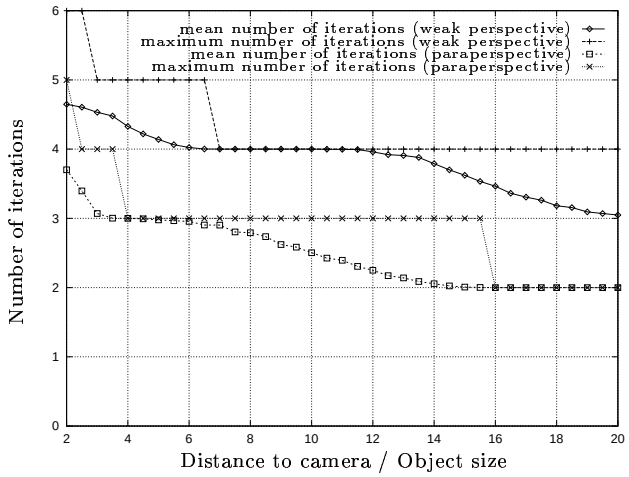

Figure 6: Number of iterations as a function of the distance camera-object in presence of gaussian noise from line correspondences (the object is shifted in respect to the optical axis).

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

Table 1: A comparison between the pose computed with point correspondences and the pose computed with line correspondences. The optimal pose parameters were estimated using a non-linear minimization method. The rotation error is the square root of the sum of the differences of the coefficients of the two matrix.

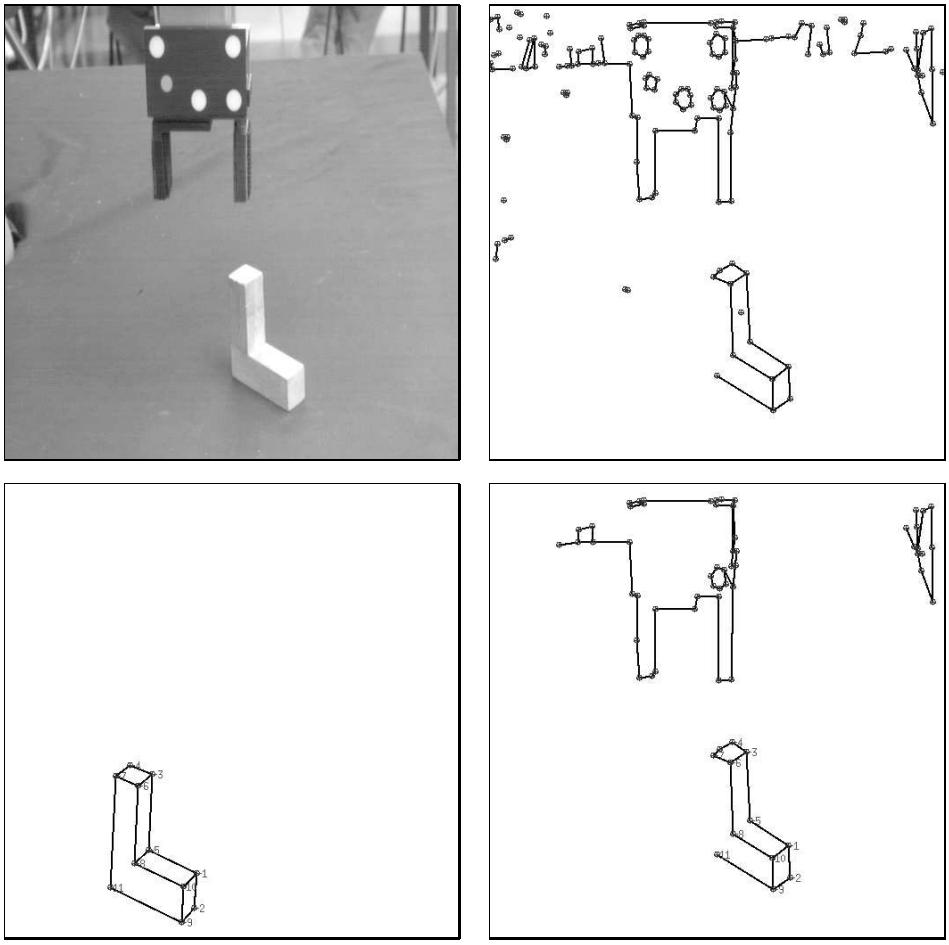

Figure 7: Straight lines and junctions are extracted from the raw data (top). A wireframe representation of the object is matched against the set of lines and junctions (bottom). The result is a set of line-to-line and junction-to-junction correspondences.

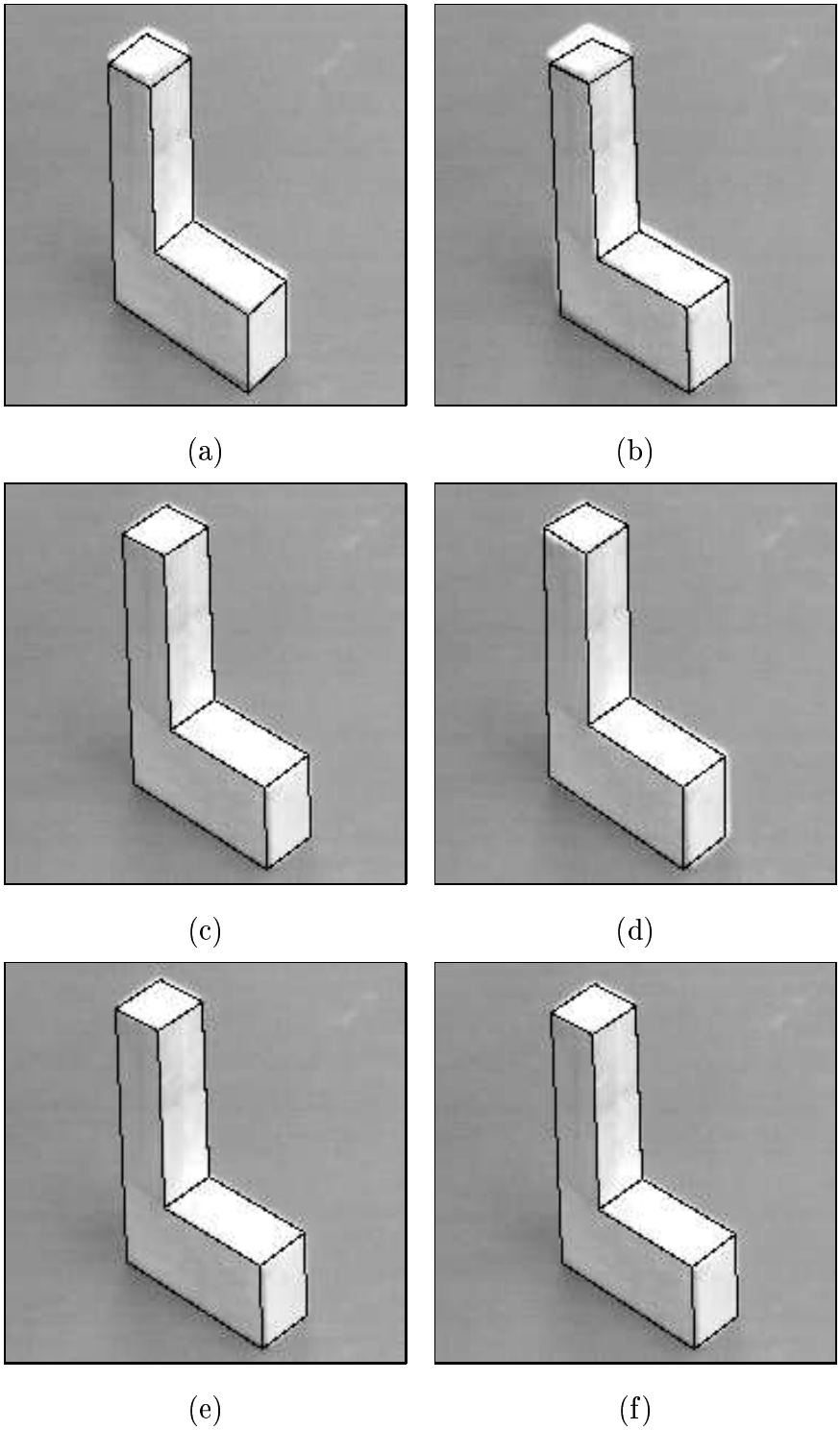

Figure 8: Pose computed with: (a) – weak perspective and lines, (b) – paraperspective and lines, (c) – iterative weak perspective and lines, (d) – iterative weak perspective and points, (e) – non-linear minimization and lines, (f) – non-linear minimization and points.