



HAL
open science

Object Pose: Links Between Paraperspective and Perspective

Radu Horaud, Stéphane Christy, Fadi Dornaika, Bart Lamiroy

► **To cite this version:**

Radu Horaud, Stéphane Christy, Fadi Dornaika, Bart Lamiroy. Object Pose: Links Between Paraperspective and Perspective. 5th International Conference on Computer Vision (ICCV '95), Jun 1995, Cambridge, United States. pp.426–433, 10.1109/ICCV.1995.466908 . inria-00590052

HAL Id: inria-00590052

<https://inria.hal.science/inria-00590052v1>

Submitted on 16 Jun 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Object Pose: Links between Paraperspective and Perspective *

Radu Horaud, Stéphane Christy, Fadi Dornaika, and Bart Lamiroy
LIFIA & INRIA Rhône-Alpes
46, avenue Félix Viallet 38031 Grenoble FRANCE

Abstract

Recently, Dementhon & 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. In this paper we show that the method of Dementhon & Davis can be extended to paraperspective. The iterative paraperspective pose algorithm that we describe in detail in this paper has interesting properties both in terms of speed and rate of convergence. Moreover, we introduce a simple way of taking into account the orthogonality constraint associated with the rotation matrix and we define the optimal experimental setup to be used in the presence of camera calibration errors.

1 Introduction

The problem of object pose from 2-D to 3-D correspondences has received a lot of attention both in the photogrammetry and computer vision literature. Various approaches to the object pose (or external camera parameters) problem fall into 2 distinct categories: closed-form solutions and numerical solutions.

Closed-form solutions may be applied only to a limited number of correspondences [5], [7], [3]. Whenever the number of correspondences is larger than 4 then closed-form solutions are not efficient any more and iterative numerical solutions are necessary [12], [10]. These approaches are, in general, very robust but they converge towards the correct solution on the premise

that a good initial estimate of the true solution is provided. Phong et al. [11] describe a method that uses trust-region optimisation and that is less sensitive to initialisation than other minimization methods. However, the method of Phong et al. performs well for a relatively large number of correspondences. Whenever the number of correspondences is between 3 and 10, then the trust-region minimization method either requires a large number of iterations or doesn't converge towards the correct solution. From a practical point of view, it is important to have an object pose algorithm which doesn't necessarily require a large number of correspondences and which doesn't suffer from the limitations that are inherent to closed-form methods.

Recently, Dementhon & 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. At our knowledge, Dementhon & Davis method is among one of the first attempts to link linear techniques associated with the weak perspective camera model to non linear techniques associated with a perspective camera model. On one side, linear resolution methods can be used but the solution thus obtained is just an approximation and, on the other side, non linear resolution methods lead to a very accurate solution but proper initialization is required.

The perspective projection is modelled by a projective transformation from the 3-D projective space to the 2-D projective plane. Weak perspective is just an affine approximation of full perspective. More precisely, it may well be viewed as a zero-order approximation: $1/(1+\varepsilon) \approx 1$. Paraperspective [1] is a first order approximation of full perspective: $1/(1+\varepsilon) \approx 1-\varepsilon$. The method proposed by Dementhon & Davis starts with computing the pose of an object using weak perspective and after a few iterations converges towards a pose estimated under perspective. The method is very elegant, very fast, and quite accurate. It is however

*This work has been supported by the Esprit programme through the Basic Research Project "SECOND" (Esprit-BRA No. 6769). R. Horaud acknowledges partial support from the University of Bern, Institut für Informatik und angewandte Mathematik, where he was a visiting scientist during July 1994. S. Christy acknowledges support from DRET and from Société Aérospatiale.

limited to situations where the weak perspective approximation is valid. If the object is close to the camera and/or at some distance away from the optical axis then the pose algorithm of Dementhon & Davis either converges very slowly (100 iterations rather than 5 to 10) or it doesn't converge at all.

In this paper we show how the method proposed in [2] may be extended to paraperspective. More precisely, we describe a method for computing object pose with a paraperspective model and we establish the link between paraperspective pose and perspective pose. Moreover we introduce a simple computational way of taking into account the orthogonality constraint associated with the 3×3 rotation matrix describing the orientation between the 3-D object and the camera. Indeed, the linear pose algorithms using weak perspective and paraperspective do not guarantee that this rotation matrix is orthogonal. The orthogonalization method that we describe below computes the best rotation in closed form using unit quaternions. This orthogonalization method considerably increases the robustness of the method at the cost of very few extra computations. We characterize the best experimental setup that allows one to compute a precise pose even in the presence of camera calibration errors. Finally we provide a comparison between the results obtained with this method and the results obtained with a non linear minimization method [11].

2 Camera models

We denote by P_i a 3-D point with coordinates X_i , Y_i , and Z_i in a frame that is attached to the object – the object frame. The origin of this frame is the object point P_0 . An object point P_i projects onto the image in p_i with camera coordinates x_i and y_i and we have (\mathbf{P}_i is the vector from point P_0 to point P_i):

$$x_i = \frac{\mathbf{i} \cdot \mathbf{P}_i + t_x}{\mathbf{k} \cdot \mathbf{P}_i + t_z} \quad (1)$$

$$y_i = \frac{\mathbf{j} \cdot \mathbf{P}_i + t_y}{\mathbf{k} \cdot \mathbf{P}_i + t_z} \quad (2)$$

These equations describe the classical perspective camera model where the rigid transformation from the object frame to the camera frame is:

$$T = \begin{pmatrix} \mathbf{i}^T & t_x \\ \mathbf{j}^T & t_y \\ \mathbf{k}^T & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

The relationship between the camera coordinates and the image coordinates may be easily obtained by introducing the intrinsic camera parameters:

$$u_i = \alpha_u x_i + u_c \quad (3)$$

$$v_i = \alpha_v y_i + v_c \quad (4)$$

In these equations α_u and α_v are the vertical and horizontal scale factors and u_c and v_c are the image coordinates of the intersection of the optical axis with the image plane.

We divide both the numerator and the denominator of eqs. (1) and (2) by t_z . We introduce the following notations:

- $\mathbf{I} = \mathbf{i}/t_z$ and $\mathbf{J} = \mathbf{j}/t_z$;
- $x_0 = t_x/t_z$ and $y_0 = t_y/t_z$ are the camera coordinates of p_0 which is the projection of P_0 – the origin of the object frame, and
- $\varepsilon_i = \mathbf{k} \cdot \mathbf{P}_i/t_z$.

We may now rewrite the perspective equations as:

$$x_i = \frac{\mathbf{I} \cdot \mathbf{P}_i + x_0}{1 + \varepsilon_i} \quad (5)$$

$$y_i = \frac{\mathbf{J} \cdot \mathbf{P}_i + y_0}{1 + \varepsilon_i} \quad (6)$$

These equations may be also written as:

$$x_i(1 + \varepsilon_i) - x_0 = \mathbf{I} \cdot \mathbf{P}_i \quad (7)$$

$$y_i(1 + \varepsilon_i) - y_0 = \mathbf{J} \cdot \mathbf{P}_i \quad (8)$$

Whenever the object is at some distance from the camera, the ε_i are small compared to 1. We may therefore introduce two approximations of the perspective equations: weak and paraperspective. Here we deal with the latter approximation.

2.1 Paraperspective

Paraperspective may be viewed as a first-order approximation of perspective. Indeed, with the approximation:

$$\frac{1}{1 + \varepsilon_i} \approx 1 - \varepsilon_i \quad \forall i, i \in \{1 \dots n\}$$

we obtain the paraperspective projection of P_i :

$$\begin{aligned} x_i^p &= (\mathbf{I} \cdot \mathbf{P}_i + x_0)(1 - \varepsilon_i) \\ &\approx \mathbf{I} \cdot \mathbf{P}_i + x_0 - x_0 \varepsilon_i \\ &= \frac{\mathbf{i} \cdot \mathbf{P}_i}{t_z} + x_0 - x_0 \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z} \end{aligned}$$

where the term in $1/t_z^2$ was neglected. There is a similar expression for y_i^p .

Finally, the paraperspective equations are:

$$x_i^p - x_0 = \frac{\mathbf{i} - x_0 \mathbf{k}}{t_z} \cdot \mathbf{P}_i \quad (9)$$

$$y_i^p - y_0 = \frac{\mathbf{j} - y_0 \mathbf{k}}{t_z} \cdot \mathbf{P}_i \quad (10)$$

In order to obtain the relationship between the paraperspective and the perspective projections of P_i we can write these equations as follows:

$$\begin{aligned} x_i^p &= x_0 + \mathbf{I} \cdot \mathbf{P}_i - x_0 \varepsilon_i \\ y_i^p &= y_0 + \mathbf{J} \cdot \mathbf{P}_i - y_0 \varepsilon_i \end{aligned}$$

By identification with eqs. (7) and (8) we obtain the relationship between the paraperspective and the perspective projections of P_i :

$$x_i^p = x_i(1 + \varepsilon_i) - x_0 \varepsilon_i \quad (11)$$

$$y_i^p = y_i(1 + \varepsilon_i) - y_0 \varepsilon_i \quad (12)$$

We can easily express the error between the paraperspective and perspective projections:

$$\Delta x_i^p = |x_i^p - x_i| = |(x_i - x_0)\varepsilon_i| \quad (13)$$

$$\Delta y_i^p = |y_i^p - y_i| = |(y_i - y_0)\varepsilon_i| \quad (14)$$

Whenever an object point P_i is far from the optical axis the weak perspective model is a poor approximation. However, a proper choice of the origin, i.e. P_0 , and the use of the paraperspective model can compensate and provide a good approximation even if ε_i is not small.

3 From paraperspective to perspective

We consider again the perspective equations (7) and (8) and let us subtract the *paraperspective term* from both the left and right sides of these equations. We obtain:

$$x_i(1 + \varepsilon_i) - x_0 - x_0 \underbrace{\frac{1}{t_z} \mathbf{k} \cdot \mathbf{P}_i}_{\varepsilon_i} = \frac{1}{t_z} \mathbf{i} \cdot \mathbf{P}_i - x_0 \frac{1}{t_z} \mathbf{k} \cdot \mathbf{P}_i$$

$$y_i(1 + \varepsilon_i) - y_0 - y_0 \underbrace{\frac{1}{t_z} \mathbf{k} \cdot \mathbf{P}_i}_{\varepsilon_i} = \frac{1}{t_z} \mathbf{j} \cdot \mathbf{P}_i - y_0 \frac{1}{t_z} \mathbf{k} \cdot \mathbf{P}_i$$

These equations can be written more compactly as:

$$(x_i - x_0)(1 + \varepsilon_i) = \mathbf{I}_p \cdot \mathbf{P}_i \quad (15)$$

$$(y_i - y_0)(1 + \varepsilon_i) = \mathbf{J}_p \cdot \mathbf{P}_i \quad (16)$$

with:

$$\mathbf{I}_p = \frac{\mathbf{i} - x_0 \mathbf{k}}{t_z} \quad (17)$$

$$\mathbf{J}_p = \frac{\mathbf{j} - y_0 \mathbf{k}}{t_z} \quad (18)$$

It is worthwhile to notice that when all the ε_i are null, the perspective equations above – eqs. (15) and (16) – become identical to the paraperspective equations – eqs. (9) and (10).

In order to solve the pose problem linearly and iteratively one may notice that:

- Whenever the ε_i are fixed (not necessarily null) the pose equations (15) and (16) associated with the *perspective camera model* become linear in \mathbf{I}_p and \mathbf{J}_p . This linear solution will be detailed below (section 4);
- It is possible to solve the equations (15) and (16) iteratively by successive approximations of the variables ε_i .

The pose parameters can be derived from \mathbf{I}_p and \mathbf{J}_p as follows. First, one may notice that:

$$\|\mathbf{I}_p\|^2 = \frac{1 + x_0^2}{t_z^2} \quad \text{and} \quad \|\mathbf{J}_p\|^2 = \frac{1 + y_0^2}{t_z^2}$$

We therefore obtain:

$$t_z = \frac{1}{2} \left(\frac{\sqrt{1 + x_0^2}}{\|\mathbf{I}_p\|} + \frac{\sqrt{1 + y_0^2}}{\|\mathbf{J}_p\|} \right) \quad (19)$$

and:

$$t_x = x_0 t_z \quad \text{and} \quad t_y = y_0 t_z$$

Second, we derive the three orthogonal unit vectors \mathbf{i} , \mathbf{j} , and \mathbf{k} . From eqs. (17) and (18) we obtain:

$$\mathbf{i} = t_z \mathbf{I}_p + x_0 \mathbf{k} \quad (20)$$

$$\mathbf{j} = t_z \mathbf{J}_p + y_0 \mathbf{k} \quad (21)$$

The third vector, \mathbf{k} is the cross-product of these two vectors:

$$\begin{aligned} \mathbf{k} &= \mathbf{i} \times \mathbf{j} \\ &= t_z^2 \mathbf{I}_p \times \mathbf{J}_p + t_z y_0 \mathbf{I}_p \times \mathbf{k} - t_z x_0 \mathbf{J}_p \times \mathbf{k} \end{aligned}$$

Let $S(\mathbf{a})$ be the skew-symmetric matrix associated with a 3-vector \mathbf{a} and $I_{3 \times 3}$ be the identity matrix. The previous expression can now be written as follows:

$$(I_{3 \times 3} - t_z y_0 S(\mathbf{I}_p) + t_z x_0 S(\mathbf{J}_p)) \mathbf{k} = t_z^2 \mathbf{I}_p \times \mathbf{J}_p \quad (22)$$

This equation allows us to compute \mathbf{k} , provided that the linear system above has a full rank. Indeed, the 3×3 matrix A :

$$A = I_{3 \times 3} - t_z y_0 S(\mathbf{I}_p) + t_z x_0 S(\mathbf{J}_p)$$

is of the form:

$$A = \begin{pmatrix} 1 & c & -b \\ -c & 1 & a \\ b & -a & 1 \end{pmatrix}$$

Its determinant is always strictly positive:

$$\det(A) = 1 + a^2 + b^2 + c^2$$

Therefore, one can easily determine \mathbf{k} using eq. (22) and \mathbf{i} and \mathbf{j} using eqs. (20) and (21).

The ε_i can now be easily computed and the pose algorithm becomes:

1. For all i , $i \in \{1 \dots n\}$, $n \geq 3$, $\varepsilon_i = 0$;
2. Solve the over constrained linear system of equations (15) and (16) which provides an estimation of vectors \mathbf{I}_p and \mathbf{J}_p , i.e., Section 4;
3. Compute the position (t_x , t_y , and t_z) and orientation (\mathbf{i} , \mathbf{j} , and \mathbf{k}) of the object frame with respect to the camera frame as explained above in this Section;
4. For all i , compute:

$$\varepsilon_i = \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z}$$

If the ε_i computed at this iteration are equal to the ε_i computed at the previous iteration then stop the procedure, otherwise go to step 2.

4 Solving the linear equations

The paraperspective iterative algorithm needs to solve an overconstrained linear system of equations, namely eqs. (15), (16). In matrix form these equations can be written as:

$$\underbrace{P}_{n \times 3} \underbrace{\mathbf{I}_p}_{3 \times 1} = \underbrace{\mathbf{x}}_{n \times 1} \quad (23)$$

$$\underbrace{P}_{n \times 3} \underbrace{\mathbf{J}_p}_{3 \times 1} = \underbrace{\mathbf{y}}_{n \times 1} \quad (24)$$

where P is a $n \times 3$ matrix formed by the 3-D coordinates of n vectors $\mathbf{P}_1 \dots \mathbf{P}_n$. Since the point P_0 is the

origin of the object frame, this matrix can be written as :

$$P = \begin{pmatrix} X_1 & Y_1 & Z_1 \\ \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n \end{pmatrix}$$

In order to solve for these linear equations one has to distinguish two cases: non coplanar and coplanar sets of object points.

4.1 Non coplanar object points

If the object points are not coplanar, the rank of P is 3 and therefore the solutions for \mathbf{I}_p and \mathbf{J}_p are simply given by:

$$\begin{aligned} \mathbf{I}_p &= (P^T P)^{-1} P^T \mathbf{x} \\ \mathbf{J}_p &= (P^T P)^{-1} P^T \mathbf{y} \end{aligned}$$

One may notice that the pseudo-inverse of P can be computed off-line and hence the estimation of \mathbf{I}_p and \mathbf{J}_p is particularly efficient.

4.2 Coplanar object points

If the object points are coplanar then the rank of P is 2 and the above solution cannot be envisaged anymore. In this case, we consider the plane formed by the object points and let \mathbf{u} be the unit vector orthogonal to this plane. Vectors \mathbf{I}_p and \mathbf{J}_p can be written as a sum of a vector belonging to this plane and a vector perpendicular to this plane, namely :

$$\mathbf{I}_p = \mathbf{I}_0 + \lambda \mathbf{u} \quad (25)$$

$$\mathbf{J}_p = \mathbf{J}_0 + \mu \mathbf{u} \quad (26)$$

By substituting these expressions for \mathbf{I}_p and \mathbf{J}_p into eqs. (23) and (24) we obtain:

$$P \mathbf{I}_0 = \mathbf{x} \quad \text{and} \quad P \mathbf{J}_0 = \mathbf{y}$$

These linear equations can be solved provided that the following additional linear constraints are used:

$$\mathbf{u} \cdot \mathbf{I}_0 = 0 \quad \text{and} \quad \mathbf{u} \cdot \mathbf{J}_0 = 0$$

Thus we obtain solutions for \mathbf{I}_0 and \mathbf{J}_0 :

$$\mathbf{I}_0 = (P'^T P')^{-1} P'^T \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix}$$

$$\mathbf{J}_0 = (P'^T P')^{-1} P'^T \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}$$

With P' defined by:

$$P' = \begin{pmatrix} P \\ \mathbf{u} \end{pmatrix}$$

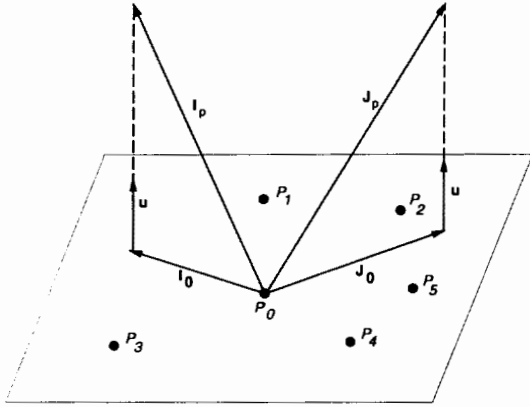


Figure 1: A coplanar configuration of object points.

Obviously, the rank of P' is equal to 3.

In order to estimate \mathbf{I}_p and \mathbf{J}_p one is left with the estimation of two scalars, λ and μ . A solution can be obtained using the following constraints onto derived from eqs. (17) and (18):

$$\begin{aligned}\|\mathbf{I}_p\|^2 &= \frac{1+x_0^2}{t_z^2} \\ \|\mathbf{J}_p\|^2 &= \frac{1+y_0^2}{t_z^2} \\ \mathbf{I}_p \cdot \mathbf{J}_p &= \frac{x_0 y_0}{t_z^2}\end{aligned}$$

By eliminating t_z we obtain two constraints:

$$\begin{aligned}\mathbf{I}_p \cdot \mathbf{J}_p &= \frac{x_0 y_0}{1+x_0^2} \|\mathbf{I}_p\|^2 \\ \mathbf{I}_p \cdot \mathbf{J}_p &= \frac{x_0 y_0}{1+y_0^2} \|\mathbf{J}_p\|^2\end{aligned}$$

By substituting in these expressions \mathbf{I}_p and \mathbf{J}_p given by eqs. (25) and (26) we obtain:

$$\begin{aligned}\mathbf{I}_0 \cdot \mathbf{J}_0 + \lambda \mu &= \frac{x_0 y_0}{1+x_0^2} (\|\mathbf{I}_0\|^2 + \lambda^2) \\ \mathbf{I}_0 \cdot \mathbf{J}_0 + \lambda \mu &= \frac{x_0 y_0}{1+y_0^2} (\|\mathbf{J}_0\|^2 + \mu^2)\end{aligned}$$

And finally, by eliminating μ we obtain a biquadratic equation in one unknown:

$$\mathcal{A} \lambda^4 + \mathcal{B} \lambda^2 + \mathcal{C} = 0 \quad (27)$$

With:

$$\begin{aligned}\mathcal{A} &= a^2 - g \\ \mathcal{B} &= 2a^2 d - g d + e - 2ac \\ \mathcal{C} &= a^2 d^2 + c^2 - 2acd\end{aligned}$$

$$\begin{aligned}a &= \frac{x_0 y_0}{1+x_0^2} \\ b &= \frac{x_0 y_0}{1+y_0^2} \\ c &= \mathbf{I}_0 \cdot \mathbf{J}_0 \\ d &= \|\mathbf{I}_0\|^2 \\ e &= \|\mathbf{J}_0\|^2 \\ g &= \frac{1+y_0^2}{1+x_0^2}\end{aligned}$$

In order to study the number of real roots of eq. (27) we substitute λ^2 by t :

$$\mathcal{A} t^2 + \mathcal{B} t + \mathcal{C} = 0$$

The signs of the roots of this equation are given by the sign of \mathcal{C}/\mathcal{A} :

$$\begin{aligned}\frac{\mathcal{C}}{\mathcal{A}} &= f(x_0, y_0, c, d) \\ &= -\frac{(x_0 y_0 d - (1+x_0^2) c)^2}{1+x_0^2+y_0^2}\end{aligned}$$

The value of \mathcal{C}/\mathcal{A} is always negative or null. Therefore there are one positive (or null) root and one negative (or null) root for t . Hence there are two real roots for λ – a positive one and a negative one – and two imaginary roots.

The two real roots for λ provide two solutions for μ as well and hence there are two solutions for \mathbf{I}_p and \mathbf{J}_p . These two solutions correspond to the wellknown reversal ambiguity associated with an affine camera model.

Thus each iteration described in Section 3 produces two poses which have the same translation vector. Theoretically, after n iterations we must have 2^n solutions. In order to avoid this redundancy we proceed as follows. At the first iteration we retain the two solutions, while at the other iterations we only retain the solution that is the most consistent with the image data. At convergence, we therefore obtain two poses. Generally, it is easy to distinguish between these two poses since the true solution is more consistent with the image data than the other solution.

5 The orthogonality constraint

The algorithm described so far is linear and in the general case (non coplanar object points) it doesn't guarantee that the rotation matrix describing the orientation of the object frame with respect to the camera frame is orthogonal. This rotation matrix is

formed by the three row vectors \mathbf{i}^T , \mathbf{j}^T , and \mathbf{k}^T . These vectors are updated at each step of the pose algorithm and we want to compute a *true* rotation matrix from these three vectors. In other terms, we seek a rotation matrix R_{\perp} which verifies:

$$R_{\perp} \begin{pmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{pmatrix}^T = I_{3 \times 3}$$

This expression can be written as:

$$\begin{aligned} R_{\perp} \mathbf{i} &= \mathbf{e}_1 \\ R_{\perp} \mathbf{j} &= \mathbf{e}_2 \\ R_{\perp} \mathbf{k} &= \mathbf{e}_3 \end{aligned}$$

where \mathbf{e}_i is the i^{th} column of the identity matrix. The solution is given by the rotation matrix which minimizes the following criterion:

$$\min_{R_{\perp}} \sum_{i=1}^3 \|R_{\perp} \mathbf{v}_i - \mathbf{e}_i\|^2$$

where \mathbf{v}_i is either \mathbf{i} , \mathbf{j} , or \mathbf{k} . It is well known that this minimization problem has a closed-form solution [4]. Indeed, if the unknown rotation is represented by a unit quaternion \mathbf{q} , then the minimization problem can be written as a quadratic form:

$$\min_{\mathbf{q}} (\mathbf{q}^T B \mathbf{q})$$

where B is a 4×4 symmetric, semi definite, and positive matrix. Therefore, the quaternion \mathbf{q} which minimizes this quadratic form is the eigen vector associated with the smallest eigen value of B . It is worthwhile to notice that two vectors, i.e., \mathbf{i} and \mathbf{j} are sufficient for computing the orthogonal matrix R_{\perp} .

6 Experiments

In this section we study the performances of the iterative weak and paraperspective algorithms. Two types of performances are studied:

- the precision of pose as a function of position and orientation of the object with respect to the camera in the presence of image and/or camera noise, and
- the convergence of the iterative pose algorithms as a function of position and orientation of the object with respect to the camera.

In the first class of experiments (precision) we compare the results obtained with three algorithms: the two linear algorithms described in [2] and above, and a non-linear algorithm [11]. In the second class of experiments (convergence) we compare the two linear algorithms. In both classes, the simulated object is a configuration of four points (tetrahedron).

For each experiment we fix a number of positions of the object with respect to the camera and for each such position the object is rotated at 1000 random orientations. These experiments are repeated for various levels of image and/or camera noise. The rotation matrices defining these 1000 orientations are computed from Euler angles chosen by a random number generator in the range $[0, 2\pi]$. The position of an object with respect to the camera is described by the translation vector from the center of projection of the camera to the origin of the object frame. More quantitatively, we compute the error between the theoretical pose and the pose computed by an algorithm. For each position we plot the average of this error over all the 1000 orientations. The pose errors are: orientation error and position error. The *orientation error* is defined as the rotation angle in degrees required to align the coordinate system of the object in its computed orientation with the coordinate system of the object in its theoretical orientation. The *position error* is defined as the norm of the vector which represents the difference between the two translation vectors: the computed one and the theoretical one, divided by the norm of the second vector. The horizontal coordinates of all the following plots represents the z-component of the translation vector scaled by the object size.

Figure 2 shows the variation of the error in orientation in the presence of image Gaussian noise as a function of the ratio between the distance to the camera and the object size, where the object size has been fixed to a constant value. The three curves onto this figure correspond to the iterative paraperspective algorithm (solid line), the iterative paraperspective algorithm with the orthogonal constraint described in Section 5 (dotted line or the middle curve) and to the non-linear algorithm described in [11] (dashed). The iterative weak perspective algorithm yields a curve that is identical to the paraperspective one. Figure 3 shows the variation of the relative error in position in the presence of image Gaussian noise as a function of the ratio between the distance to the camera and the object size. The solid line corresponds to the iterative weak and paraperspective algorithms (identical behaviour) and the dashed line corresponds to the non-linear algorithm. In both cases (Figure 2 and Fig-

ure 3) the image data has been perturbed with Gaussian noise with a standard deviation equal to 1 pixel.

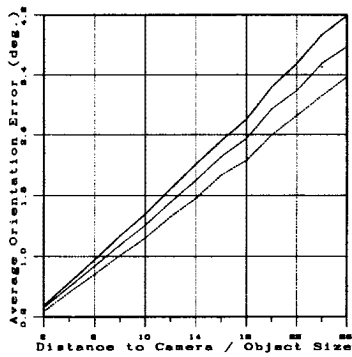


Figure 2: Error in orientation as a function of depth in the presence of image Gaussian noise (see text).

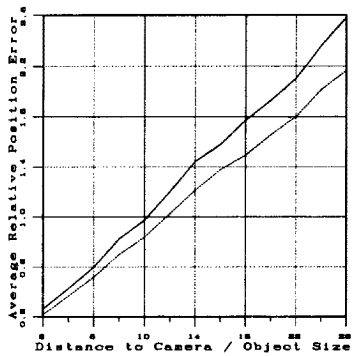


Figure 3: Error in position as a function of depth in the presence of image Gaussian noise (see text).

The second class of experiments is meant to compare the rate of convergence of each one of these algorithms when the object is quite close to the camera. Figure 4 shows the percentage of the convergence of both algorithms as a function of depth. In this experiment the angle between the “direction of view” towards the object and the optical axis of the camera is of 35° . Indeed, as mentioned above, the weak perspective algorithm has an optimal behaviour when the object is either quite far from the camera or close to the optical axis. In theory, the paraperspective algorithm does not suffer from such limitations. For example (Figure 4), when the distance from the object to the camera is 1.4 times the size of the object, the paraperspective algorithm has always converged (solid line with triangles) while the weak perspective algo-

rithm has converged only in 76 % of the configurations (solid line with squares).

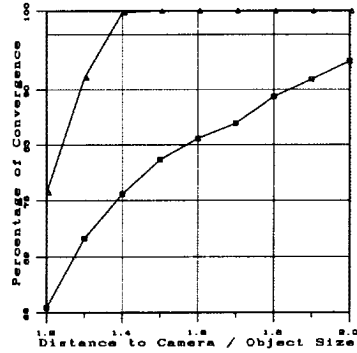


Figure 4: Rate of convergence as a function of depth. The angle between the direction of view towards the object and the optical axis of the camera is equal to 35° (see text).

The iterative algorithm proposed in this paper has been applied to a number of images. For example, Figure 5 shows the image of a polyhedral object to be located (top-left) in order to be grasped by a parallel-jaw gripper [8] and the wireframe representation of this object (top-right). Both the image and the model are described by a network of straight lines and junctions which are matched using a method described in [6]. Using this technique, 10 junctions were correctly matched (middle). The first iteration of the algorithm found a “paraperspective pose” (bottom-left). After only three iterations the algorithm correctly determined the position and orientation of the polyhedral object with respect to the camera (bottom-right).

7 Discussion

In this paper we proposed an extension to paraperspective of the iterative weak perspective pose algorithm developed by Dementhon & Davis. In an extended version of this paper we established the link between perspective, weak perspective, and paraperspective camera models [9]. We described a linear method for computing object pose with a paraperspective model and an iterative algorithm which computes pose with a perspective model by successive paraperspective approximations.

We studied, experimentally, the convergence of the weak and paraperspective algorithms. We showed that, on an average, the latter algorithm requires 2.5

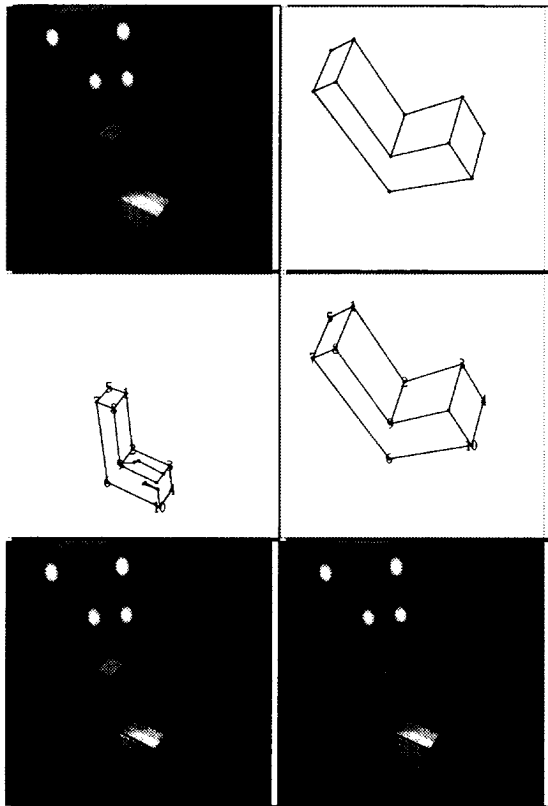


Figure 5: An example of applying the iterative paraperspective algorithm to a non coplanar set of point correspondences (see text).

times less iterations than the former algorithm. Moreover, when the object is relatively closed to the camera, and at some distance from the optical axis then the rate of convergence of the paraperspective algorithm is higher than the rate of convergence of the weak perspective one.

We showed that both algorithms (weak and paraperspective) can take advantage of a simple to implement orthogonality constraint associated with the rotation matrix of the object pose.

Moreover, we compared the accuracy of the results obtained with these linear algorithms with the results obtained with a non-linear one. The linear algorithms are almost as precise as the non-linear one, especially when the orthogonality constraint mentioned above is being used.

In the near future we plan to extend the iterative linear techniques described here to deal with the problem of reconstruction from multiple views. Indeed, attempts have been made to solve the multiple view reconstruction problem using either an approximated (linear) camera model or a projective (non-linear)

camera model, but no attempts have been made to properly establish a link between these two classes of algorithms.

References

- [1] Y. Aloimonos. Perspective approximations. *Image and Vision Computing*, 8(3):177–192, August 1990.
- [2] D. F. Dementhon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 1995. In press.
- [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] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Boston, 1993.
- [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] P. Gros. Matching and clustering: two steps towards automatic model generation in computer vision. In *Proceedings of the AAAI Fall Symposium Series: Machine Learning in Computer Vision: What, Why, and How?*, Raleigh, North Carolina, USA, pages 40–44, October 1993.
- [7] R. Horaud, B. Conio, O. Le Boulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, July 1989.
- [8] R. Horaud, F. Dornaika, C. Bard, and B. Espiau. Visually guided object grasping. Technical report, INRIA, March 1995. Submitted to *IEEE Trans. on Robotics & Automation*.
- [9] R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy. Object pose: The link between weak perspective, paraperspective, and full perspective. Technical Report RR-2356, INRIA, September 1994.
- [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] 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*, 14, 1995.
- [12] 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.