Object Pose: The Link between Weak Perspective, Paraperspective and Full Perspective

Radu Horaud, Fadi Dornaika, Bart Lamiroy, Stéphane Christy

Object Pose: The Link between Weak Perspective, Paraperspective, and Full Perspective [1]

To appear in *International Journal of Computer Vision*, volume 22, NO. 2, 1997

Radu Horaud, Fadi Dornaika,
Bart Lamiroy, and Stéphane Christy
GRAVIR–IMAG & INRIA Rhône-Alpes
655, avenue de l'Europe
38330 Montbonnot Saint-Martin, FRANCE

Abstract – *Recently, DeMenthon & Davis [4], [5] 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 give an algebraic derivation of DeMenthon and Davis' method and we show that it belongs to a larger class of methods where the perspective camera model is approximated either at zero order (weak perspective) or first order (paraperspective). We describe in detail an iterative paraperspective pose computation method for both non coplanar and coplanar object points. We analyse the convergence of these methods and we conclude that the iterative paraperspective method (proposed in this paper) has better convergence properties than the iterative weak perspective method. We introduce a simple way of taking into account the orthogonality constraint associated with the rotation matrix. We analyse the sensitivity to camera calibration errors and we define the optimal experimental setup with respect to imprecise camera calibration. We compare the results obtained with this method and with a non-linear optimization method.*

Keywords – *Perspective n-point problem, object pose, weak perspective, paraperspective, camera calibration, extrinsic camera parameters.*

[1] 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 fur Informatik und angewandte Mathematik, where he was a visiting scientist during July 1994.

Contents

1 Introduction

Recently, DeMenthon & Davis [4], [5] 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. To our knowledge, the method proposed by DeMenthon & Davis is among one of the first attempts to link linear techniques, associated with the weak perspective camera model, with non-linear techniques, associated with a perspective camera model. Indeed, 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.

One possibility could be to use a robust numerical method to perform the non linear minimization that is initialized with the solution obtained by a linear method. However, there are several drawbacks. First, such an approach does not take into account the simple mathematical link that exists between the perspective model and its linear approximations. Second, the linear approximation may be quite faraway from the true solution and a large number of iterations would be required before the non-linear minimization process converges to a stable solution. For example, as it will be described below, a non-linear method requires 10 times more iterations than the method of DeMenthon & Davis and than the method described in this paper.

The perspective projection is modelled by a projective transformation mapping 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 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 initial method proposed in [4] and [5] 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. We show that the case of a coplanar set of object points is somehow more complex than the case of a non coplanar set of object points and we describe a method which can deal with both cases. We show both theoretically and experimentally that our method has better convergence properties than the method proposed by DeMenthon & Davis. Moreover we introduce a simple computational way of taking into account the orthogonality constraint associated with the 3×3 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 matrix is orthogonal. The orthogonalization method that we describe below computes the best rotation in close form using unit quaternions. This orthogonalization method considerably increases the accuracy 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 [15], [16].

1.1 Background

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 literatures. Haralick et al. [10] cites a German dissertation from 1958 that surveys nearly 80 different solutions from the photogrammetry 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 [8], [11], [6]. Whenever the number of correspondences is larger than 4 then closed-form solutions are not efficient any more and iterative numerical solutions are necessary [19], [13], [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. [16] 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 available 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.

The method recently proposed by DeMenthon & Davis [4], [5] requires an arbitrary number of point correspondences. The 3-D points must be in general position (i.e., non coplanar) and there should be at least 4 point correspondences. The method is fast and it is robust with respect to image measurements and to camera calibration errors. However, the object pose algorithm as suggested by DeMenthon and Davis has convergence problems if the object is too faraway from the optical axis of the camera — a limitation that can be overcome as explained farther in this paper. The method of DeMenthon & Davis was extended to the case of a coplanar set of points by Oberkampf, DeMenthon, and Davis [14] and by DeMenthon [3].

1.2 Paper organization

The remainder of this paper is organized as follows. In Section 2 we briefly recall the perspective camera model and two possible approximations of this model, weak and paraperspective. Section 3 describes the iterative weak perspective algorithm as it has been proposed in [5]. Section 4 describes the new iterative paraperspective algorithm that we propose as well as how to compute the object pose from a paraperspective approximation. Section 5 provides details for solving the linear equations associated with pose computation in the case of both non coplanar and planar sets of object points. Section 6 analyses the convergence of both the iterative weak and paraperspective algorithms and Section 7 shows how to improve the pose algorithms using a simple and straightforward formulation of the orthogonality constraint. Section 8 analyses the sensitivity of both algorithms with respect to camera calibration errors, and Section 9 provides an experimental

comparison of the two methods described in this paper together with a comparison of these methods with a non-linear minimization method. Section 10 shows many examples of application of pose computation. Finally, Section 11 provides a discussion and gives directions for future work.

2 Camera models

We consider a simple setup, as depicted on Figure 1. 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:

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

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:

- $\mathbf{I} = \mathbf{i}/t_z$ is the first row of the rotation matrix scaled by the z-component of the translation vector;
- $\mathbf{J} = \mathbf{j}/t_z$ is the second row of the rotation matrix scaled by the z-component of the translation vector;
- $x_0 = t_x/t_z$ 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
- $\varepsilon_i = \mathbf{k} \cdot \mathbf{P}_i/t_z$.

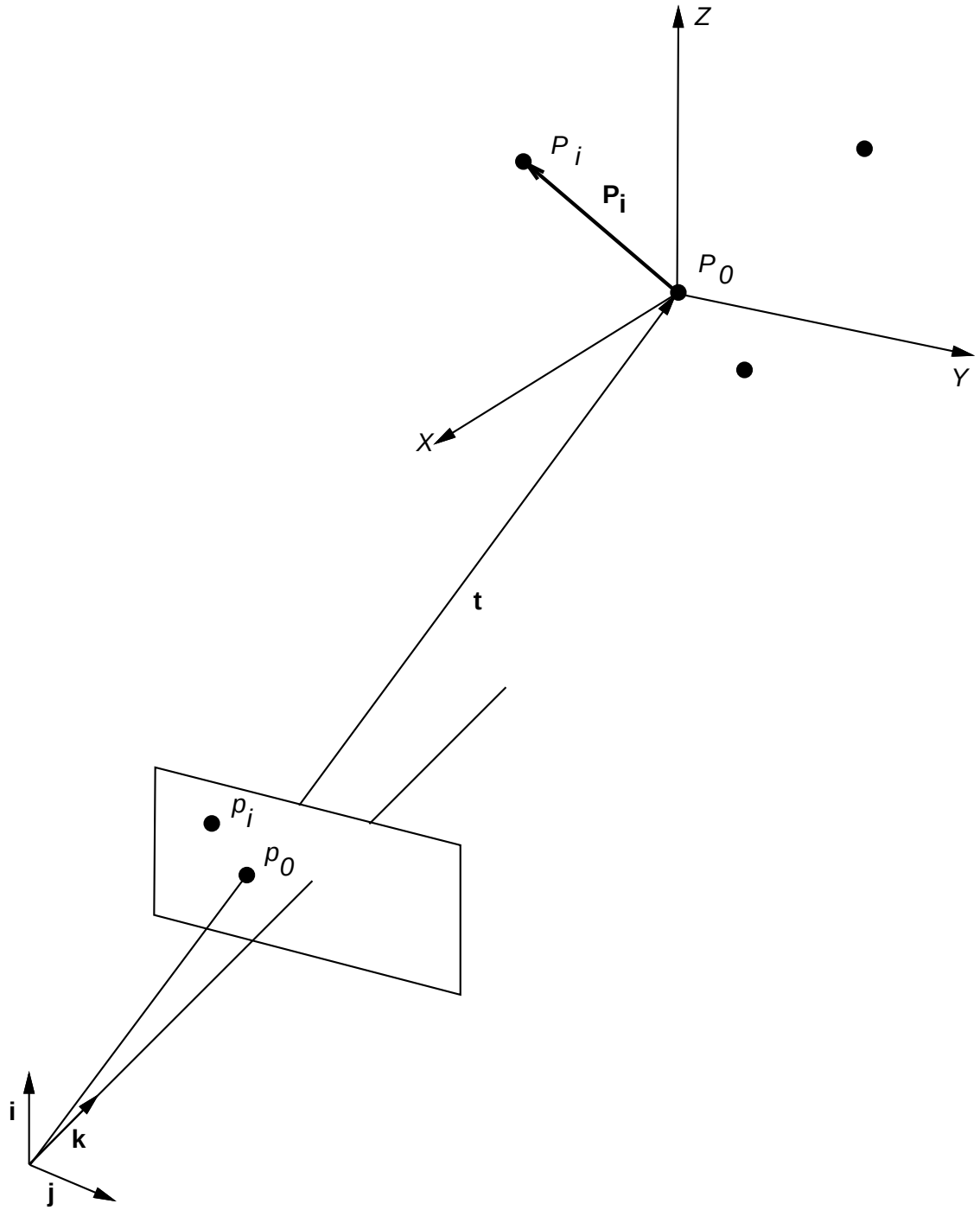

Figure 1: This figure shows the general setup. One of the object's point is selected as the origin of the object frame. Therefore, the pose parameters are the coordinates of this object point (P_0) in the camera frame and the orientation of the object frame with respect to the camera frame.

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 also be written as in [5]:

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

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

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. One can find a geometric description of these projections in [17].

2.1 Weak perspective

Weak perspective assumes that the object points lie in a plane parallel to the image plane passing through the origin of the object frame, i.e., P_0, e.g., Figure 2. This is equivalent to a zero-order approximation:

$$\frac{1}{1 + \varepsilon_i} \approx 1 \ \ \forall i, \ i \in \{1...n\}$$

With this approximation, eqs. (5) and (6) become:

$$x_i^w - x_0 = \mathbf{I} \cdot \mathbf{P}_i \quad (9)$$

$$y_i^w - y_0 = \mathbf{J} \cdot \mathbf{P}_i \quad (10)$$

In these two equations x_i^w and y_i^w are the camera coordinates of the weak perspective projection of the point P_i. By identification with eqs. (7) and (8) we obtain the relationship between the weak perspective and the perspective projections of P_i:

$$x_i^w = x_i(1 + \varepsilon_i) \quad (11)$$

$$y_i^w = y_i(1 + \varepsilon_i) \quad (12)$$

These equations allow us to determine the quality of the weak perspective approximation with respect to the perspective projection. Indeed the error between the weak perspective projection and the "true" projection is:

$$\Delta x^w = |x_i^w - x_i| = |x_i \varepsilon_i| \quad (13)$$

$$\Delta y^w = |y_i^w - y_i| = |y_i \varepsilon_i| \quad (14)$$

Hence the quality of the approximation depends both on the value of ε_i AND on the position of the point in the image. We consider for example a 512×512 image. Approximate values for the

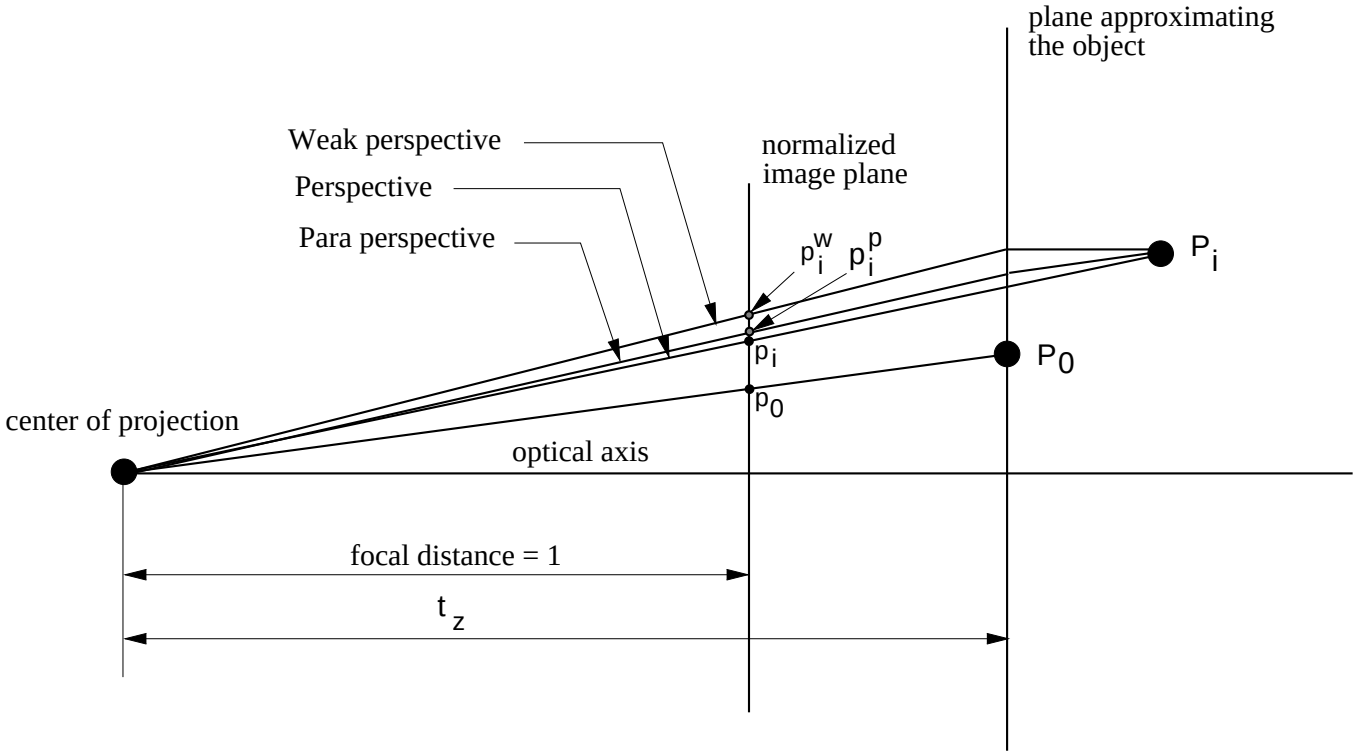

Figure 2: This figure shows p_0, p_i – the perspective projections of two object points, P_0 and P_i, and p_i^w, p_i^p – the weak and paraperspective projections of P_i. The quality of the paraperspective projection depends on the angle between the line of projection of P_i and the line of projection of P_0. Similarly, the quality of the weak perspective projection depends on the angle between the line of projection of P_i and the optical axis.

intrinsic parameters are: $\alpha_u = \alpha_v = 1000$ and $u_c = v_c = 256$. Therefore using eq. (3) and (4) we have:

$$0 \leq x_i \ , \ y_i \leq 0.25$$

The lower bound corresponds to the point of intersection of the optical axis with the image plane ($x_i = y_i = 0$). The upper bound corresponds to the image borders:

$$x_i = \frac{u_i - u_c}{\alpha_u} = \frac{512 - 256}{1000} \approx 0.25$$

We conclude that for small distance/size ratios, i.e., large values for ε_i, the weak perspective approximation is still valid *provided that the object lies in the neighbourhood of the optical axis.*

2.2 Paraperspective

Paraperspective may almost be viewed as a first-order approximation of perspective. Indeed, two approximations are needed in order to obtain the paraperspective camera model. With the approximation:

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

we obtain the paraperspective projection of P_i, e.g., Figure 2:

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

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

Finally, the paraperspective equations are:

$$x_i^p - x_0 = \frac{\mathbf{i} - x_0\,\mathbf{k}}{t_z} \cdot \mathbf{P}_i \tag{15}$$

$$y_i^p - y_0 = \frac{\mathbf{j} - y_0\,\mathbf{k}}{t_z} \cdot \mathbf{P}_i \tag{16}$$

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

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

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

$$x_i^p = x_i(1 + \varepsilon_i) - x_0 \varepsilon_i \tag{17}$$

$$y_i^p = y_i(1 + \varepsilon_i) - y_0 \varepsilon_i \tag{18}$$

As in the previous section, we can easily estimate the error between the paraperspective and perspective projections:

$$\Delta x^p = |x_i^p - x_i| = |(x_i - x_0)\varepsilon_i| \tag{19}$$

$$\Delta y^p = |y_i^p - y_i| = |(y_i - y_0)\varepsilon_i| \tag{20}$$

Whenever an object point P_i is far from the optical axis then 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 weak perspective to perspective

In order to solve the pose problem, DeMenthon & Davis [5] noticed that eqs. (9) and (10) are similar to eqs. (7) and (8) for which the ε_i are set to 0. They conclude that:

- Whenever the ε_i are fixed (not necessarily null) the pose equations (7) and (8) become linear in $\mathbf{I}$ and $\mathbf{J}$. A solution can be found if at least 4 object points are provided (see Section 5).

- It is possible to solve eqs. (7) and (8) *iteratively* by successive linear approximations.

In this case the pose algorithm proposed in [5] starts with a weak perspective camera model and computes an approximated pose. This approximated pose is improved iteratively as follows:

1. For all i, $i \in \{1...n\}$, $n \geq 3$, $\varepsilon_i = 0$;
2. Solve the overconstrained linear system of equations (7) and (8) which provides an estimation of vectors $\mathbf{I}$ and $\mathbf{J}$, i.e., Section 5;
3. Compute the position and orientation of the object frame with respect to the camera frame:

$$\begin{aligned} t_z &= \frac{1}{2}\left(\frac{1}{\|\mathbf{I}\|} + \frac{1}{\|\mathbf{J}\|}\right) \\ t_x &= x_0\, t_z \\ t_y &= y_0\, t_z \\ \mathbf{i} &= \frac{\mathbf{I}}{\|\mathbf{I}\|} \\ \mathbf{j} &= \frac{\mathbf{J}}{\|\mathbf{J}\|} \\ \mathbf{k} &= \mathbf{i} \times \mathbf{j} \end{aligned}$$

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.

While iterating, the above algorithm replaces the actual image coordinates x_i and y_i (which are perspective projections of a 3-D point) with $x_i(1+\varepsilon_i)$ and $y_i(1+\varepsilon_i)$, which are weak perspective projections of the same 3-D point. A geometric interpretation of this algorithm can be found in [3], [5].

4 From paraperspective to perspective

In this section we provide a generalization of the above algorithm to deal with the paraperspective case. 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:

$$\begin{aligned} x_i(1+\varepsilon_i) - x_0 - x_0 \underbrace{\frac{1}{t_z}\mathbf{k}\cdot\mathbf{P}_i}_{\varepsilon_i} &= \frac{1}{t_z}\mathbf{i}\cdot\mathbf{P}_i - x_0\frac{1}{t_z}\mathbf{k}\cdot\mathbf{P}_i \\ y_i(1+\varepsilon_i) - y_0 - y_0 \underbrace{\frac{1}{t_z}\mathbf{k}\cdot\mathbf{P}_i}_{\varepsilon_i} &= \frac{1}{t_z}\mathbf{j}\cdot\mathbf{P}_i - y_0\frac{1}{t_z}\mathbf{k}\cdot\mathbf{P}_i \end{aligned}$$

These equations can be written more compactly as:

$$(x_i - x_0)(1 + \varepsilon_i) = \mathbf{I}_p \cdot \mathbf{P}_i \quad (21)$$

$$(y_i - y_0)(1 + \varepsilon_i) = \mathbf{J}_p \cdot \mathbf{P}_i \quad (22)$$

with:

$$\mathbf{I}_p = \frac{\mathbf{i} - x_0\,\mathbf{k}}{t_z} \quad (23)$$

$$\mathbf{J}_p = \frac{\mathbf{j} - y_0\,\mathbf{k}}{t_z} \quad (24)$$

One may notice that when all the ε_i are null, the perspective equations above – eqs. (21) and (22) – become identical to the paraperspective equations – eqs. (15) and (16).

As in the weak perspective case, we have now a linear method for computing the pose with a paraperspective model and an iterative method to compute the object pose with a perspective model by successive paraperspective approximations. More precisely, the pose parameters can be derived from $\mathbf{I}_p$ and $\mathbf{J}_p$ as follows.

First, one may notice that:

$$\|\mathbf{I}_p\|^2 = \frac{(\mathbf{i} - x_0\,\mathbf{k}) \cdot (\mathbf{i} - x_0\,\mathbf{k})}{t_z^2} = \frac{1 + x_0^2}{t_z^2}$$

and:

$$\|\mathbf{J}_p\|^2 = \frac{1 + y_0^2}{t_z^2}$$

We therefore obtain:

$$t_z = \frac{1}{2}\left(\frac{\sqrt{1 + x_0^2}}{\|\mathbf{I}_p\|} + \frac{\sqrt{1 + y_0^2}}{\|\mathbf{J}_p\|}\right) \quad (25)$$

and:

$$t_x = x_0 t_z$$

$$t_y = y_0 t_z$$

Second, we derive the three orthogonal unit vectors $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$. From eqs. (23) and (24) we obtain:

$$\mathbf{i} = t_z\,\mathbf{I}_p + x_0\,\mathbf{k} \quad (26)$$

$$\mathbf{j} = t_z\,\mathbf{J}_p + y_0\,\mathbf{k} \quad (27)$$

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

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

Let $S(\mathbf{a})$ be the skew-symmetric matrix associated with a 3-vector $\mathbf{a}$ and $I_{3\times3}$ be the identity matrix. The previous expression can now be written as follows:

$$(I_{3\times3} - t_z y_0\, S(\mathbf{I}_p) + t_z x_0\, S(\mathbf{J}_p))\ \mathbf{k} = t_z^2\, \mathbf{I}_p \times \mathbf{J}_p \tag{28}$$

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

$$A = I_{3\times3} - t_z y_0\, S(\mathbf{I}_p) + t_z x_0\, S(\mathbf{J}_p)$$

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. (28) and $\mathbf{i}$ and $\mathbf{j}$ using eqs. (26) and (27).

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

1. For all i, $i \in \{1...n\}$, $n \geq 3$, $\varepsilon_i = 0$;
2. Solve the overconstrained linear system of equations (21) and (22) which provides an estimation of vectors $\mathbf{I}_p$ and $\mathbf{J}_p$, i.e., Section 5;
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.

5 Solving the linear equations

Both the weak perspective and paraperspective iterative algorithms need to solve an overconstrained linear system of equations, namely eqs. (7), (8) (weak perspective) and eqs. (21), (22) (paraperspective). In matrix form these equations can be written as:

$$\underbrace{P}_{n\times3}\ \underbrace{\mathbf{I}}_{3\times1} = \underbrace{\mathbf{x}}_{n\times1} \tag{29}$$

$$\underbrace{P}_{n\times3}\ \underbrace{\mathbf{J}}_{3\times1} = \underbrace{\mathbf{y}}_{n\times1} \tag{30}$$

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

5.1 Non coplanar object points

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

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

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

5.2 Coplanar object points

If the object points are coplanar then the rank of P is 2 and the above solution cannot be considered anymore. We consider the plane formed in this case by the object points and let $\mathbf{u}$ be the unit vector orthogonal to this plane. Vectors $\mathbf{I}$ and $\mathbf{J}$ (and equivalently $\mathbf{I}_p$ 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 (see [3], [14] and Figure 3):

$$\mathbf{I} = \mathbf{I}_0 + \lambda \mathbf{u} \tag{31}$$
$$\mathbf{J} = \mathbf{J}_0 + \mu \mathbf{u} \tag{32}$$

By substituting these expressions for $\mathbf{I}$ and $\mathbf{J}$ into eqs. (29) and (30) we obtain:

$$\begin{aligned} P\,\mathbf{I}_0 &= \mathbf{x} \\ P\,\mathbf{J}_0 &= \mathbf{y} \end{aligned}$$

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

$$\mathbf{u} \cdot \mathbf{I}_0 = 0$$

$$\mathbf{u} \cdot \mathbf{J}_0 = 0$$

Thus we obtain solutions for $\mathbf{I}_0$ and $\mathbf{J}_0$ as follows.

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

With P' defined by:

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

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

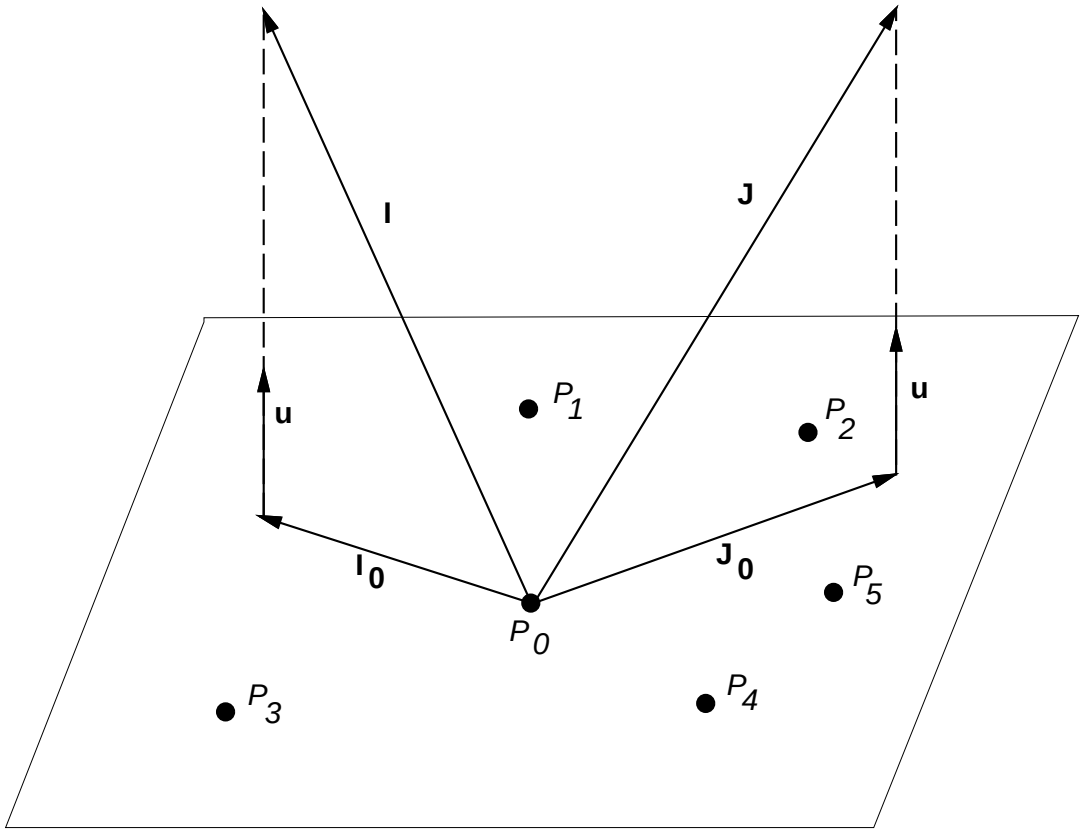

Figure 3: A coplanar configuration of object points.

In order to estimate $\mathbf{I}$ and $\mathbf{J}$ (and equivalently $\mathbf{I}_p$ and $\mathbf{J}_p$) one is left with the estimation of two scalars, λ and μ. In the case of weak perspective Oberkampf, DeMenthon, and Davis [14] provided a solution using the constraints $\|\mathbf{I}\| = \|\mathbf{J}\|$ and $\mathbf{I} \cdot \mathbf{J} = 0$.

In the case of paraperspective a similar solution can be obtained using the following constraints onto $\mathbf{I}_p$ and $\mathbf{J}_p$ (derived from eqs. (23) and (24)):

$$\|\mathbf{I}_p\|^2 = \frac{1 + x_0^2}{t_z^2}$$

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

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. (31) and (32) 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:

$$\mathcal{A}\,\lambda^4 + \mathcal{B}\,\lambda^2 + \mathcal{C} = 0 \tag{33}$$

With:

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

In order to study the number of real roots of eq. (33) we substitute λ^2 by t:

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

We examine the signs of the roots of this equation. Therefore we have to examine the sign of their product, i.e., $\mathcal{C}/\mathcal{A}$. We have:

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

Therefore, the value of $\mathcal{C}/\mathcal{A}$ is either 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.

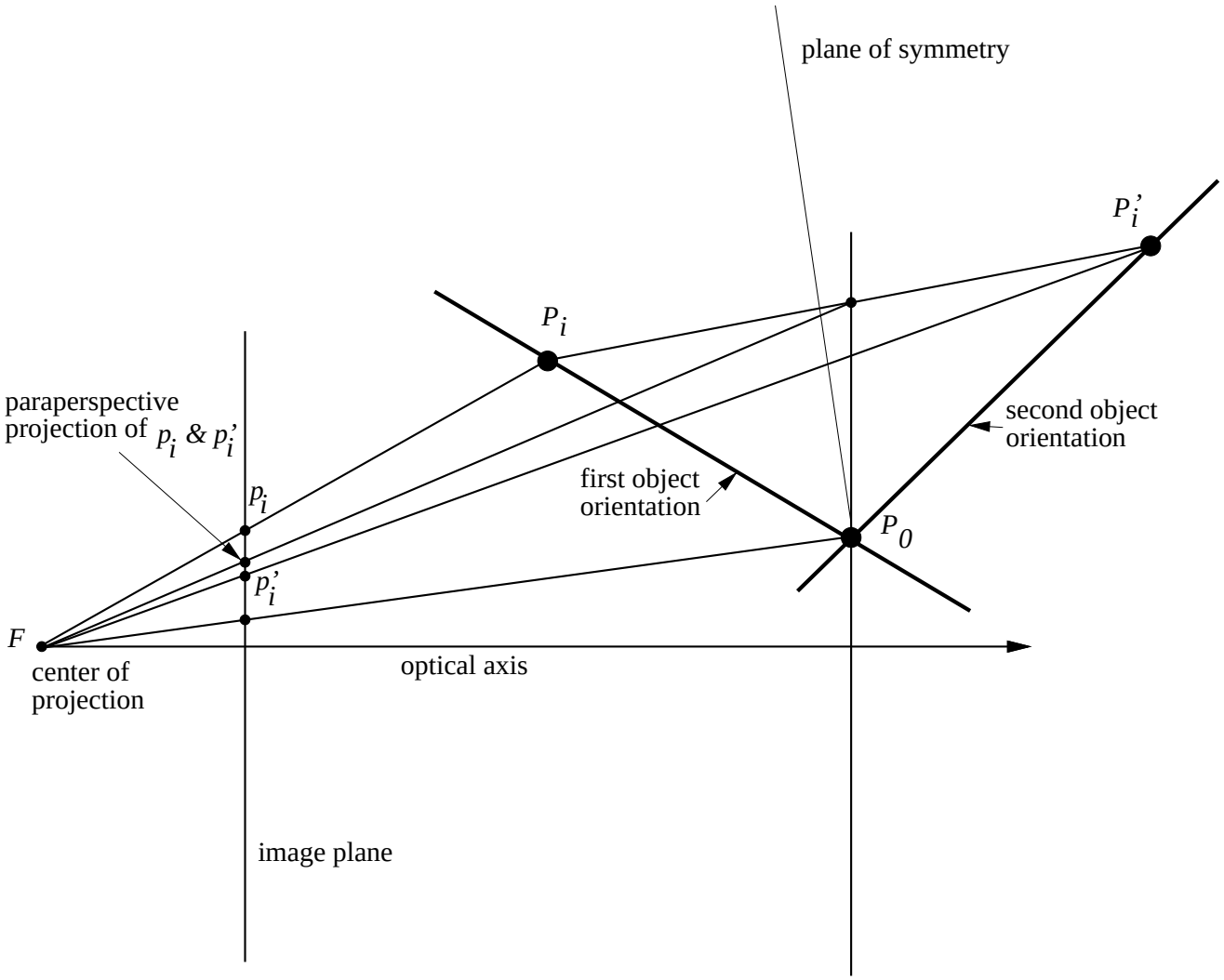

Figure 4: A planar object and a paraperspective camera model produces two orientations. However, these two object orientations have distinct perspective projections – p_i and p_i'.

The two real roots for λ provide two solutions for μ and hence there are two solutions for $\mathbf{I}_p$ and $\mathbf{J}_p$. These two solutions correspond to the well-known reversal ambiguity associated with an affine camera model. These two solutions are shown on Figure 4. The points P_i (lying on the planar object in one orientation) and P_i' (lying on the planar object in another orientation) have the same paraperspective projection but different perspective projections. Notice that these object orientations are symmetric with respect to a plane which is perpendicular to the line of sight passing through P_0.

Therefore, the iterative algorithm described in Section 4 produces two poses at each iteration. The two poses have the same translation vector but different orientations. Hence, after n iterations there will be 2^n solutions. In order to avoid this redundancy we proceed as follows. At the first

iteration we retain both solutions while at the next iterations we retain only one solution – the solution which is the most consistent with the data. At convergence we obtain two solutions for the orientation of the planar object: one solution associated with the "left" branch of the search tree and another solution associated with the "right" branch of the search tree (see Figure 5). Among the final two solutions, one of them is generally closer to the image data than the other. However, if the orientation of the planar object is close to the orientation of the plane of symmetry, the ambiguity remains and the algorithm provides two solutions.

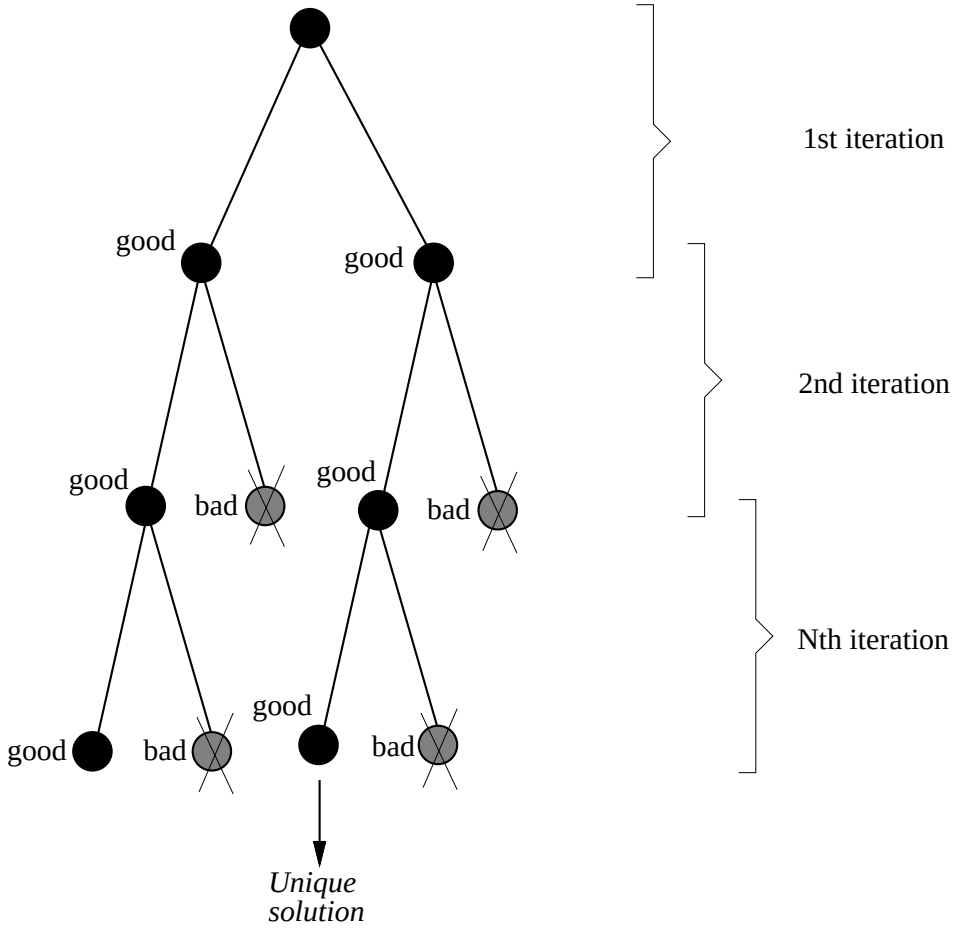

Figure 5: A binary tree search for selecting the best object pose when the object points are coplanar.

6 An analysis of convergence

In order to analyse the convergence of these algorithms (sections 3 and 4) we consider again eqs. (7) and (8):

$$\begin{aligned} x_i(1+\varepsilon_i) - x_0 &= \mathbf{I} \cdot \mathbf{P}_i \\ y_i(1+\varepsilon_i) - y_0 &= \mathbf{J} \cdot \mathbf{P}_i \end{aligned}$$

as well as eqs. (21) and (22):

$$\begin{aligned}(x_i - x_0)(1+\varepsilon_i) &= \mathbf{I}_p \cdot \mathbf{P}_i \\ (y_i - y_0)(1+\varepsilon_i) &= \mathbf{J}_p \cdot \mathbf{P}_i\end{aligned}$$

Both these sets of equations describe the *full* perspective projection. The first ones allows to express the pose problem in terms of an iterative weak perspective algorithm while the second ones allows to express the pose problem in terms of an iterative paraperspective algorithm. From Figure 2 and Section 2 we have:

- x_0 and y_0 are the camera coordinates of p_0, the perspective projection of P_0;
- x_i and y_i are the camera coordinates of p_i, the perspective projection of P_i;
- $x_i(1+\varepsilon_i)$ and $y_i(1+\varepsilon_i)$ are the camera coordinates of p_i^w, the weak perspective projection of P_i;
- $x_i(1+\varepsilon_i) - x_0\varepsilon_i$ and $y_i(1+\varepsilon_i) - y_0\varepsilon_i$ are the camera coordinates of p_i^p, the paraperspective projection of P_i;

Therefore, if good estimates for ε_i ($i \in \{1...n\}$) are available, then the perspective pose problem is reduced to solving an overconstrained set of linear equations. These equations provide a solution either for $\mathbf{I}$ and $\mathbf{J}$ or for $\mathbf{I}_p$ and $\mathbf{J}_p$ from which a pose may be easily calculated. However, in practice it is rather difficult to provide such estimates. Hence, one has to use a heuristic. The simplest heuristic is to initialize the values of ε_i to zero and to use either the iterative weak perspective or the iterative paraperspective algorithms.

We consider the first one of these two algorithms and let $\varepsilon_i^\star$ be the true values that the algorithm is supposed to compute. Therefore, at the first iteration, one attempts to solve in a least-square sense a linear set of equations of the form:

$$\left\{\begin{array}{rcl} & \vdots & \\ \mathbf{I} \cdot \mathbf{P}_i &=& x_i - x_0 \\ \mathbf{J} \cdot \mathbf{P}_i &=& y_i - y_0 \\ & \vdots & \end{array}\right.$$

The pose thus computed is just an approximation since one uses x_i and y_i instead of x_i^w and y_i^w. Therefore the "pose errors" are proportional to the following "projection errors":

$$\begin{aligned}\Delta_i^w &= \|p_i - p_i^w\|^2 \\ &= (x_i^2 + y_i^2)|\varepsilon_i^\star|^2\end{aligned}$$

If these projection errors are large, the linear least-square solution provided by the first iteration of the algorithm will be rather different than the true solution and convergence cannot be guaranteed. DeMenthon & Davis noticed that the iterative weak perspective algorithm converges with values as large for $\varepsilon_i^\star$ as 1, *provided that the object lies in the neighbourhood of the optical axis.*

Indeed, when the object is close to the optical axis, the camera coordinates of its projections, x_i and y_i, are small (the origin of the camera frame lies onto the optical axis) and they compensate for the large values of $\varepsilon_i^\star$.

In theory, the iterative paraperspective algorithm is able to deal with configurations where the weak perspective algorithm diverges. Indeed, in the case of paraperspective, the projection errors are:

$$\begin{aligned}\Delta_i^p &= \|p_i - p_i^p\|^2 \\ &= ((x_i - x_0)^2 + (y_i - y_0)^2)|\varepsilon_i^\star|^2 \\ &= \|p_i - p_0\|^2|\varepsilon_i^\star|^2\end{aligned}$$

Whenever one wants to use the paraperspective camera model, the choice for p_0 (and hence for P_0) is crucial. The best way to choose p_0 is to compute the center of gravity of all the image points and to select the image point which is the closest to this center of gravity. Hence, $x_i - x_0$ and $y_i - y_0$ will be small and they will compensate for large values of $\varepsilon_i^\star$. Therefore, the iterative paraperspective algorithm is likely to converge for those configurations where the weak perspective one diverges. Notice that if the point p_0 thus computed lies in the vicinity of the optical axis then the paraperspective and weak perspective models will yield the same results.

7 The orthogonality constraint

The algorithms described so far are linear and in the general case (non coplanar object points) they don't guarantee that the matrix describing the orientation of the object frame with respect to the camera frame is a rotation matrix. This 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 algorithms and we want to compute a rotation matrix from these three vectors. In other words, we seek a rotation matrix R which verifies:

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

This expression can be written as:

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

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:

$$\min_R \sum_{i=1}^{3} \|R\mathbf{v}_i - \mathbf{e}_i\|^2$$

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 [7]. 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. One may notice that two vectors, i.e., $\mathbf{i}$ and $\mathbf{j}$ are sufficient for computing the orthogonal matrix R.

8 Sensitivity to camera calibration

So far we assumed that the camera is calibrated, which means that the intrinsic camera parameters are known, i.e., eqs. (3) and (4). It is well known that camera calibration is difficult and that the camera parameters vary a lot. In this section we analyse the sensitivity of the pose algorithms that we just described with respect to variations of the intrinsic camera parameters.

We consider the equations associated with full perspective. By combining eqs. (3) and (4) with eqs. (7) and (8) we obtain:

$$u_i(1+\varepsilon_i) - u_0 - u_c\varepsilon_i = \alpha_u \mathbf{I} \cdot \mathbf{P}_i \tag{34}$$

$$v_i(1+\varepsilon_i) - v_0 - v_c\varepsilon_i = \alpha_v \mathbf{J} \cdot \mathbf{P}_i \tag{35}$$

When a camera is calibrated, relatively stable values may be obtained for the horizontal and vertical scale factors α_v and α_u while the image coordinates of the optical center, u_c and v_c can vary with up to 10% of their nominal values [18], [7]. By inspecting the previous equations it is easy to notice that the influence of u_c and v_c is weighted by ε_i. For any object point P_i, ε_i is the projection of the vector P_0P_i onto the optical axis, divided by the z-component of the displacement vector $\mathbf{t}$.

If the object is too far from the camera, the influence of u_c and v_c can be neglected but one needs to detect image points with great accuracy. If the object is too close to the camera, the influence of u_c and v_c becomes quite large. Therefore, in practice, the object should be at a distance from the camera that is roughly 5 to 10 times the size of the object. In that case the values of ε_i will be between 0.1 and 0.2 and the error on u_c and v_c will be scaled down by a factor of 5 to 10. These remarks are validated by experimental results as reported in the next section.

9 Experiments

In this section we study the performances of the iterative weak and para perspective 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 above and a non-linear algorithm [16]. 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), such that the three line segments joining the reference point to the other three points are equal and perpendicular to each other.

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 6 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 in this figure correspond to the iterative paraperspective algorithm (solid line), the iterative paraperspective algorithm with the orthogonal constraint described in Section 7 (dotted line) and to the non-linear algorithm described in [16] (dashed). The iterative weak perspective algorithm yields a curve that is identical to the paraperspective one. Figure 7 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 6 and Figure 7) the image data has been perturbed with gaussian noise with a standard deviation equal to 1 pixel.

Figure 8 shows the variation of the error in orientation as a function of the distance to the camera scaled by the object size. The difference between this figure and Figure 6 is that here we added uniformly distributed noise to the intrinsic camera parameters. Namely the coordinates of the optical axis in the image frame were perturbed with uniform noise in the interval $[-15\%, +15\%]$. The behaviour of the iterative paraperspective algorithm (solid line) is the one predicted by equations (34) and (35) – the effect of an inacurate center of projection vanishes as the object is farther away from the camera. For short distances the non-linear algorithm (dashed line) is more acurate than the iterative paraperspective algorithm.

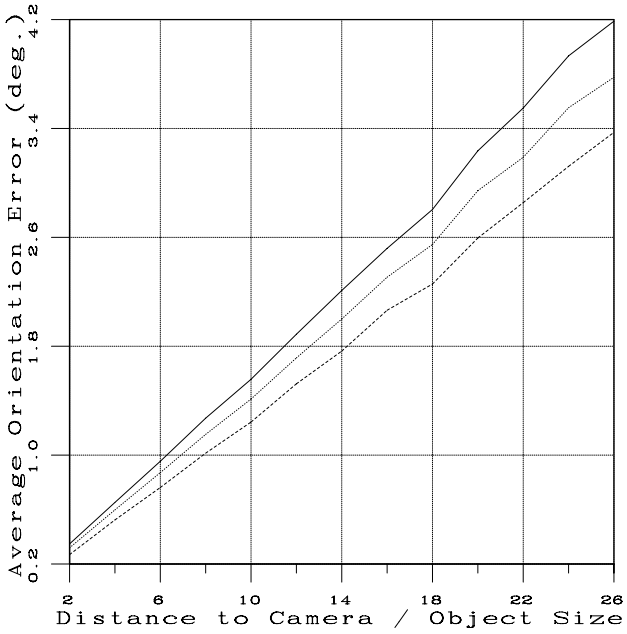

Figure 6: Error in orientation as a function of depth in the presence of image gaussian noise. The solid line corresponds to the iterative paraperspective algorithm, the dotted line corresponds to the iterative paraperspective algorithm combined with the orthogonal constraint, and the dashed line corresponds to a non linear algorithm.

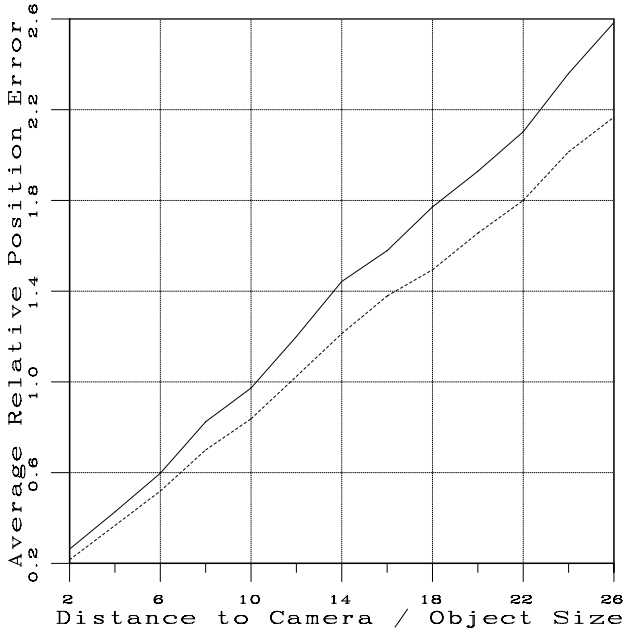

Figure 7: Error in position as a function of depth in the presence of image gaussian noise. The iterative weak and paraperspective algorithms have the same behaviour (solid line) while the non linear algorithm (dashed line) performs slightly better.

If we combine camera noise and image noise, then we obtain two other curves shown on Figure 9. Uniform pixel noise in the interval $[-0.5, +0.5]$ has been added to the data. The iterative paraperspective algorithm (solid line) has a minimum for a distance/size ratio equal to 10. This is consistent with the theoretical predictions of Section 8. We conclude that the optimal behaviour of the iterative pose algorithms occur when the distance between the object and the camera is 5 to 10 times the size of the object. In the presence of both camera noise and image noise, the non linear algorithm (dashed line) performs better than the linear algorithm.

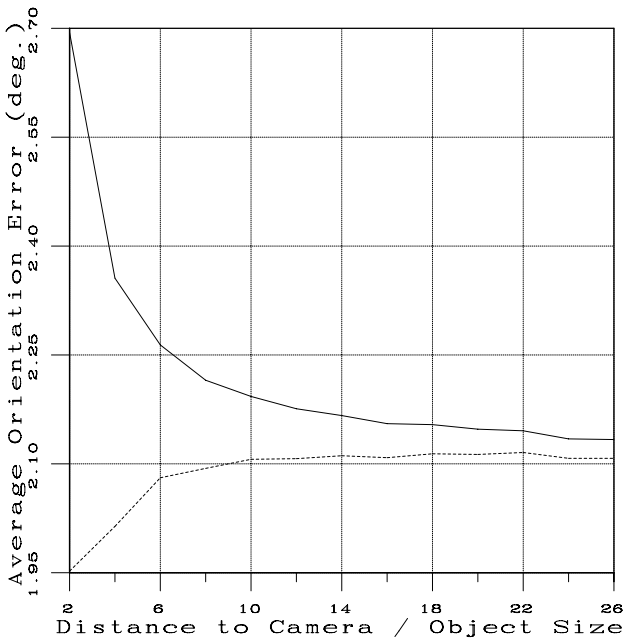

Figure 8: Error in orientation as a function of depth in the presence of camera uniform noise. The behaviour of the iterative paraperspective algorithm (solid line) is compared with a non linear algorithm (dashed line).

We compare now more thoroughly the two iterative pose algorithms. More precisely we study the convergence of these algorithms. We performed the following kinds of experiments. The first kind is meant to compare the number of iterations needed by each one of these algorithms to converge to the theoretical solution. In order to take into account the effect of the offset from the optical axis, we constrain the origin of the object frame to belong to a fixed line of sight. The object offset can now be defined as the angle between this line of sight and the optical axis. For each such offset and for each position (depth) we randomly selected 1000 different orientations and we run both algorithms for each such position and orientation. We plot the average value of the number of iterations over all the 1000 random orientations.

Figures 10 and 11 show the number of iterations as a function of depth - more precisely, the distance from the object to the camera scaled by the object size. The lines with squares correspond to the weak perspective algorithm, the lines with triangles correspond to the paraperspective algorithm. These figures correspond respectively to two offsets, 23^o and 30^o. On an average, the paraperspective algorithm converges 2 to 3 times faster than the weak perspective algorithm.

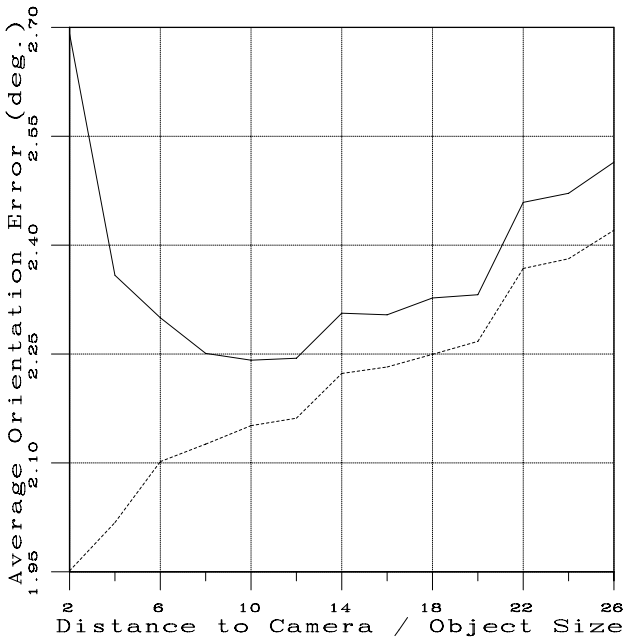

Figure 9: Error in orientation as a function of depth in the presence of camera and image uniform noise. The iterative paraperspective algorithm (solid line) has a behaviour that differs from the non-linear algorithm (dashed line).

The second kind is meant to compare the rate of convergence of each one of these algorithms for small distance/size ratios. Figures 12 and 13 show the percentage of the convergence of both algorithms as a function of depth. These figures correspond respectively to two offsets, 30^o and 35^o. As studied above in the paper, 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 the offset limitations (see figures 10, 11, 12 and 13). For example (Figure 13), 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 algorithm has converged only in 76 % of the configurations.

10 Examples

The iterative algorithms proposed in this paper have been applied to a number of images. For example, Figure 14 shows the image of a polyhedral object to be located (top-left) and its wireframe representation (top-right) whose position and orientation with respect to the camera has been roughly hypothesized. Both the image and the model are described by a network of straight lines and junctions which are matched using a method similar to the one described in [9]. 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).

The second example, i.e., Figure 15 illustrates a situation where only 4 junctions have been

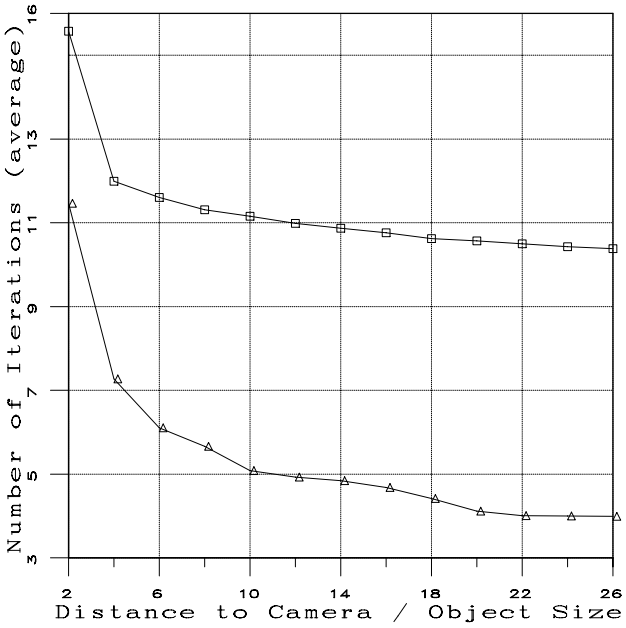

Figure 10: Speed of convergence as a function of depth, the offset is equal to 23^o (triangles: paraperspective, squares: weak perspective).

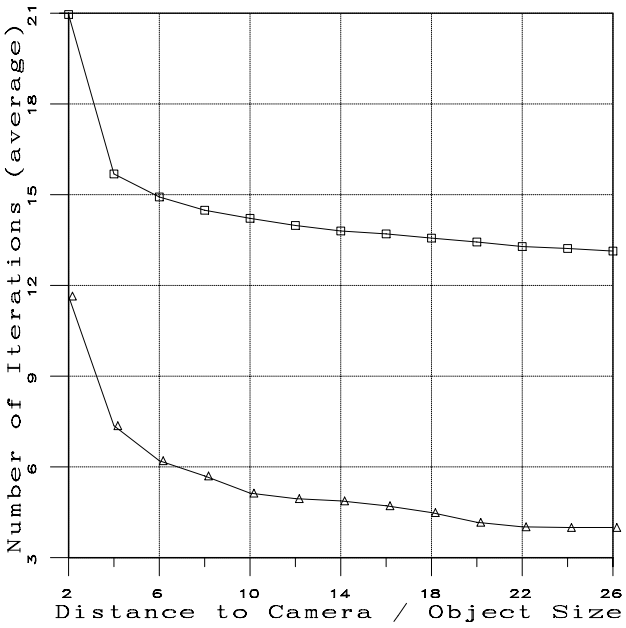

Figure 11: Speed of convergence as a function of depth, the offset is equal to 30^o (triangles: paraperspective, squares: weak perspective).

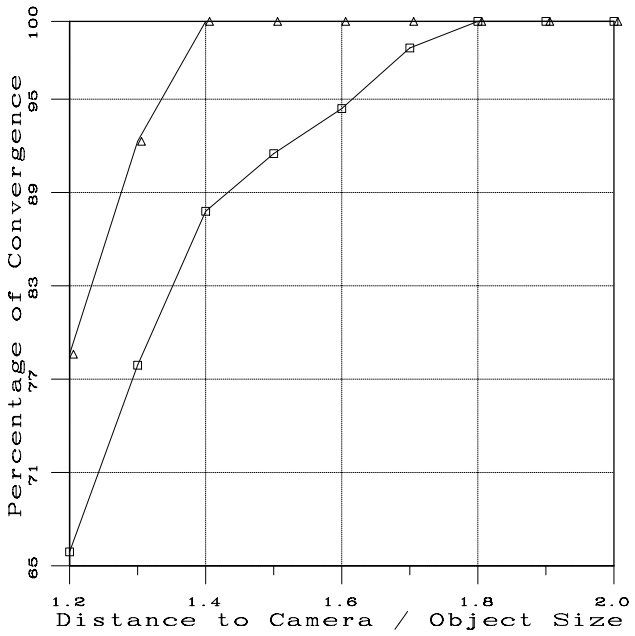

Figure 12: Rate of convergence as a function of depth, the offset is equal to 30^o (triangles: paraperspective, squares: weak perspective).

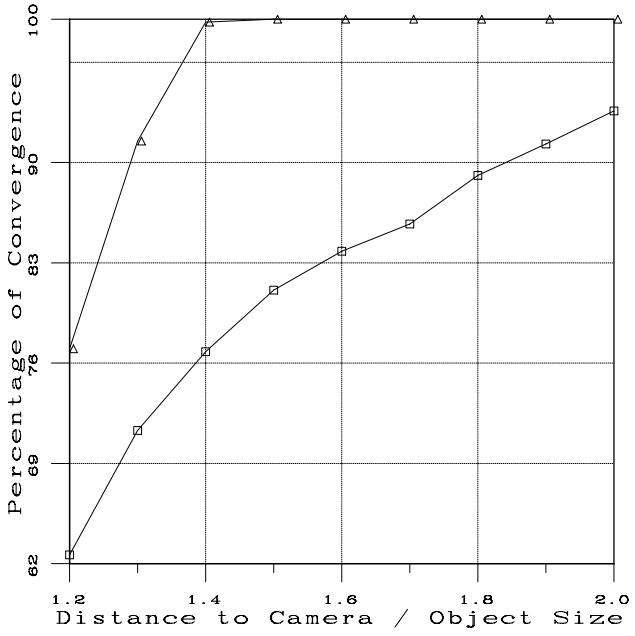

Figure 13: Rate of convergence as a function of depth, the offset is equal to 35^o (triangles: paraperspective, squares: weak perspective).

matched corresponding to coplanar model vertices (top). The first iteration found an approximated pose (bottom-left) and after 3 iterations the algortihm converged to the correct pose (bottom-right).

In both examples the iterative paraperspective algorithm converged after 0.7×10^{-3} seconds on a Sun/Sparc10.

The third example illustrates the behaviour of the linear and non linear algorithm with 4 coplanar object points, e.g., Figure 16 and Table 1. As one may easily notice, the iterative paraperspective algorithm is faster (both in term of number of iterations and CPU time) than the iterative weak perspective algorithm. The non-linear algorithm was initialized with the solution obtained by using a paraperspective model, i.e., the pose computed at the first iteration of the iterative paraperspective algorithm.

Method	Number of iterations	CPU time (ms)
It. weak persp.	10	4.3
It. parapersp.	5	2.9
Non linear	91	65.3

Table 1: A comparison of the performance of the three algorithms. The computer being used is a Sun Sparc10/51.

The fourth example illustrates the behaviour of the iterative paraperspective and non linear algorithms as a function of the number of point correspondences. As in the previous experiment we are interested by the speed of convergence of the algorithm and by the computation time. Figure 17-left shows the image of a scene and 11 points that correspond to the vertices of a polyhedral object. Figure 17-right shows the pose found for this object using these 11 points and the iterative paraperspective algorithm.

Table 2 summarizes the results obtained with a varying number of point correspondences.

11 Discussion

In this paper we proposed an extension to paraperspective of the iterative weak perspective pose algorithm developped by DeMenthon & Davis [4]. We established a link between perspective, weak perspective, and paraperspective camera models. 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, both theoretically and experimentally, the convergence of the iterative weak and paraperspective algorithms. We showed that, on an average, the latter algorithm requires 2.5 times less iterations than the former algorithm. Moreover, when the object is relatively close to the camera and at some distance from the optical axis, then the chance of convergence of the paraperspective algorithm is higher than the chance of convergence of the weak perspective one. Because it requires, on an average, less iterations, the iterative paraperspective algorithm is more time-efficient than the iterative weak perspective algorithm. Of course, if the experimental conditions are such that

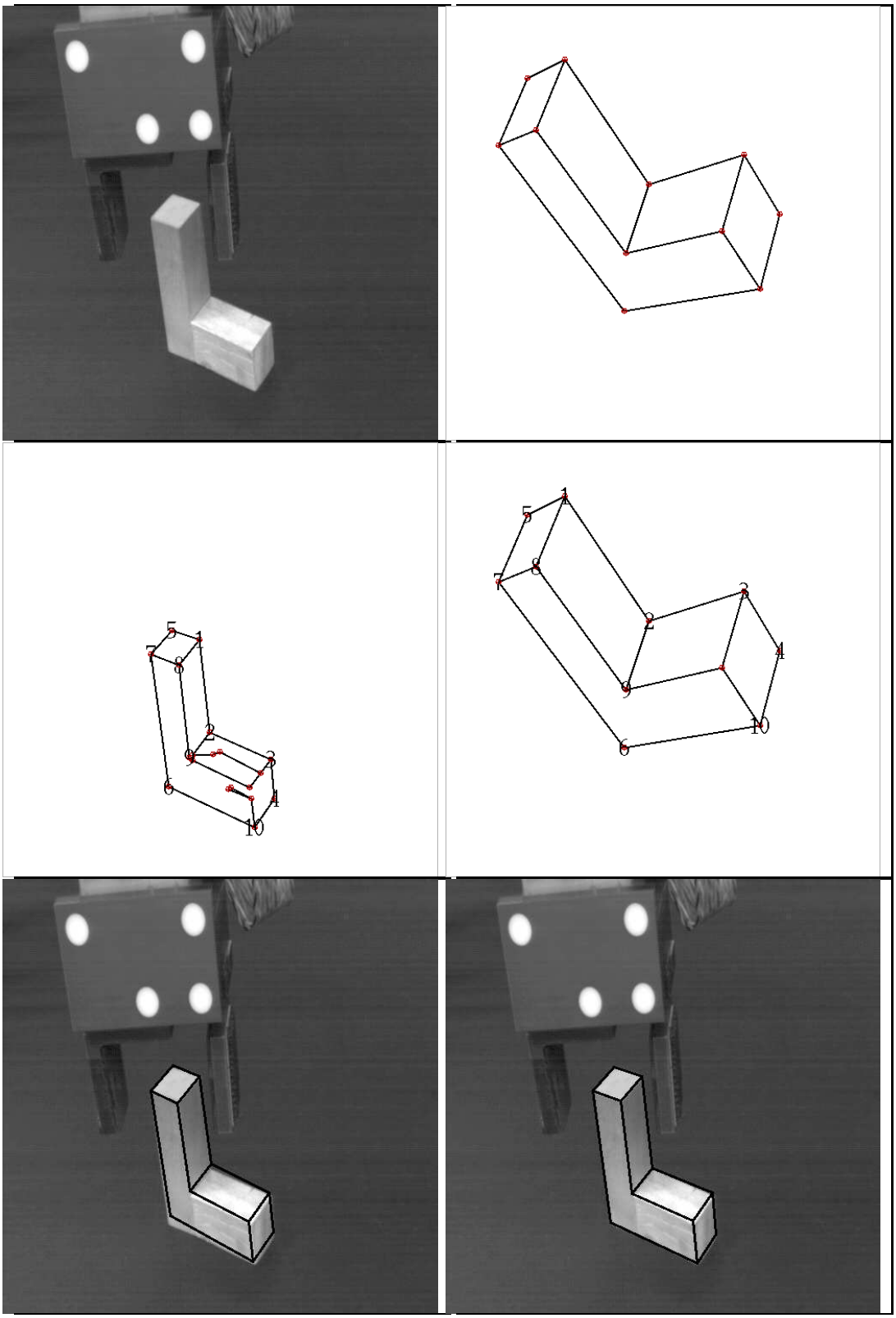

Figure 14: An example of applying the iterative paraperspective algorithm to a non coplanar set of point correspondences.

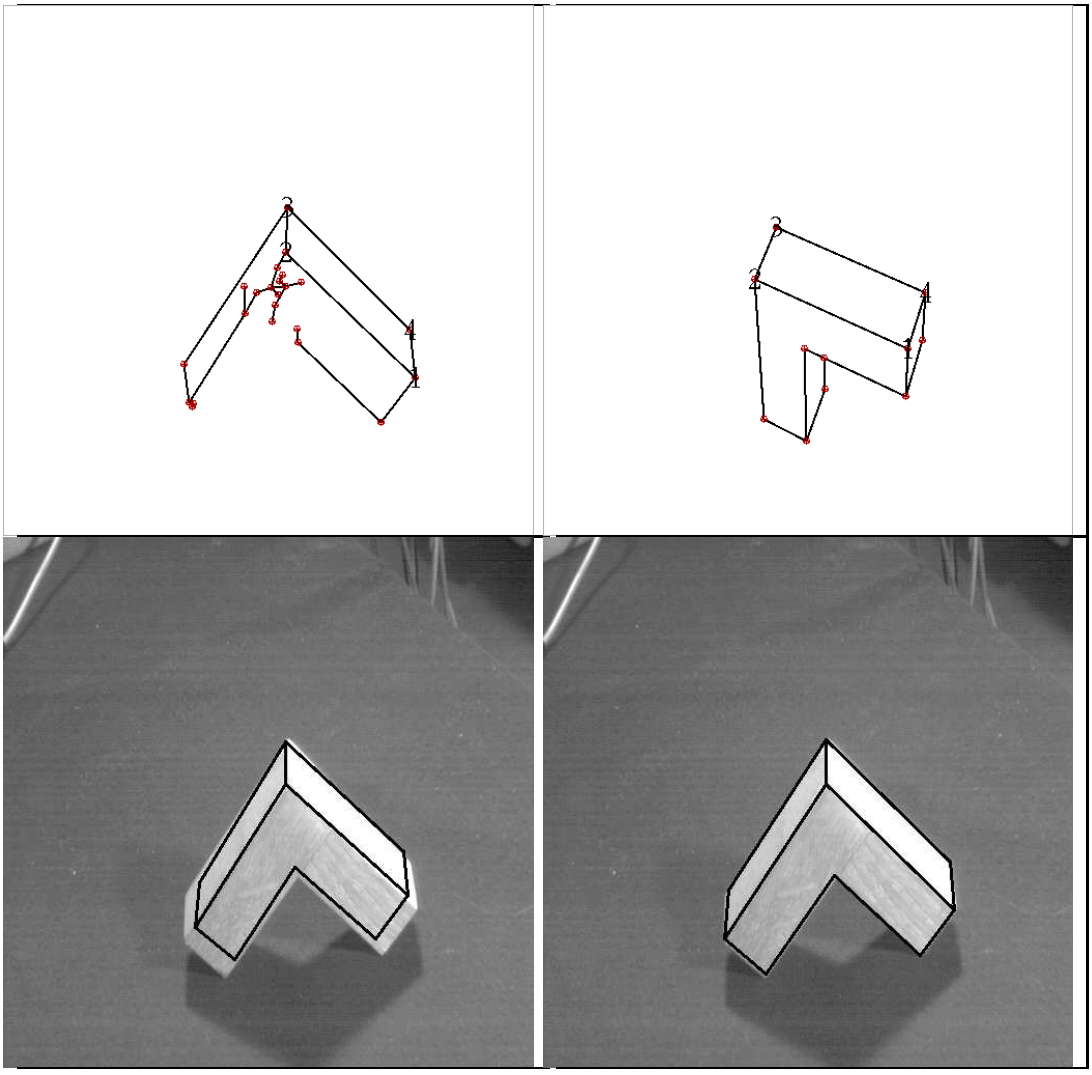

Figure 15: An example of applying the iterative paraperspective algorithm to a set of 4 coplanar point correspondences.

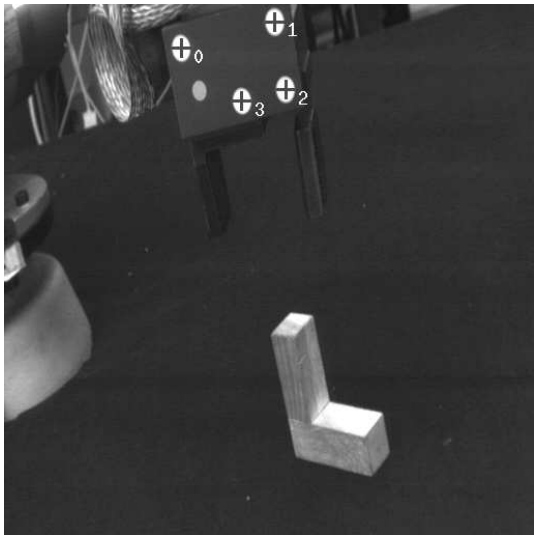

Figure 16: The four image dots correspond to a coplanar set of object points. Because the object is farther away from the optical axis, the iterative paraperspective method performs better in this case than the iterative weak perspective method.

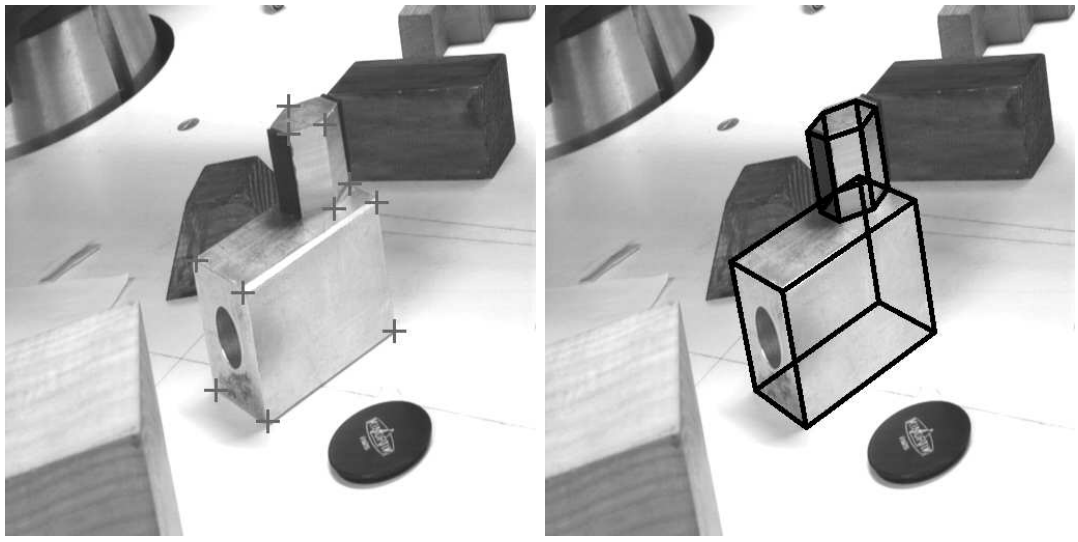

Figure 17: It takes 1.47 miliseconds on a Sun-Sparc10/51 to compute the pose in this case with the iterative paraperspective algorithm.

	Iterative paraperspective		Non linear	
Number of points	N. of iterations	CPU time (ms)	N. of iterations	CPU time (ms)
4	5	0.87	114	87.5
5	5	0.95	67	58.7
6	6	1.02	66	68.0
7	5	1.06	66	72.5
8	6	1.12	58	70.5
9	6	1.33	82	108.6
10	6	1.38	78	117.4
11	6	1.47	78	123.6

Table 2: A comparison of the iterative paraperspective and non linear methods as a function of the number of points. One may notice the large number of iterations required by the non linear method for 4 point correspondences.

the image center of gravity of the set of points is near the optical axis then the weak perspective model should be prefered.

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

Moreover, we compared the accuracy of the results obtained with these iterative linear algorithms with the results obtained with a non-linear one. The former algorithms are much faster and almost as precise as the latter, especially when the orthogonality constraint mentioned above is being used. However, in the presence of noise the performances of the iterative linear methods degrade faster with increasing noise amplitude than the non linear method. This is due to the fact that non-linear minimization numerical techniques are much more robust to noise than linear algebraic techniques.

Whenever speed is an important concern, iterative linear pose methods should however be prefered. Indeed, the iterative linear algorithms described by DeMenthon and Davis [5] and in this paper are one hundred times faster than the non-linear algorithms – this feature allows one to include these iterative linear techniques in real-time vision and robotics applications [12].

It is interesting to notice that the iterative linear techniques described here are able to deal with the problem of reconstruction from multiple views [2]. 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] S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. Technical Report RR-2421, INRIA, December 1994. Available by anonymous ftp on imag.fr, directory pub/MOVI, file RR-2421.ps.gz.

[3] D. F. DeMenthon. *De la Vision Artificielle à la Réalité Synthétique: Système d'interaction avec un ordinateur utilisant l'analyse d'images vidéo.* PhD thesis, Université Joseph Fourier - Grenoble I, Laboratoire TIMC/IMAG, October 1993.

[4] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 335–343. Springer Verlag, May 1992.

[5] 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.

[6] 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.

[7] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Boston, 1993.

[8] 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.

[9] 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.

[10] R. B. Haralick, H. Joo, C-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1445, 1989.

[11] 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.

[12] 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.*

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

[14] D. Oberkampf, D. F. DeMenthon, and L. S. Davis. Iterative pose estimation using coplanar feature points. In *Proceedings Computer Vision and Pattern Recognition*, pages 626–627, New York, June 1993. IEEE Computer Society Press, Los Alamitos, Ca.

[15] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Optimal estimation of object pose from a single perspective view. In *Proceedings Fourth International Conference on Computer Vision*, pages 534–539, Berlin, Germany, May 1993. IEEE Computer Society Press, Los Alamitos, Ca.

[16] 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.

[17] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In Jan-Olof Eklundh, editor, *Computer Vision - ECCV 94, Proceedings Third European Conference on Computer Vision, Stockholm, May 1994*, volume 2, pages 97–108. Springer Verlag, May 1994.

[18] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[19] 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.