Real-Time Imaging **5**, 215–230 (1999)
Article No. rtim.1997.0117, available online at http://www.idealibrary.com on IDEAL

Pose Estimation using Point and Line Correspondences

The problem of a real-time pose estimation between a 3D scene and a single camera is a fundamental task in most 3D computer vision and robotics applications such as object tracking, visual servoing, and virtual reality. In this paper we present two fast methods for estimating the 3D pose using 2D to 3D point and line correspondences. The first method is based on the iterative use of a weak perspective camera model and forms a generalization of DeMenthon's method (1995) which consists of determining the pose from point correspondences. In this method the pose is iteratively improved with a weak perspective camera model and at convergence the computed pose corresponds to the perspective camera model. The second method is based on the iterative use of a paraperspective camera model which is a first order approximation of perspective. We describe in detail these two methods for both non-planar and planar objects. Experiments involving synthetic data as well as real range data indicate the feasibility and robustness of these two methods. We analyse the convergence of these methods and we conclude that the iterative paraperspective method has better convergence properties than the iterative weak perspective method. We also introduce a non-linear optimization method for solving the pose problem.

Fadi Dornaika and Christophe Garcia

RWCP Theorical Foundation GMD Laboratory, Institute For System Design Technology
GMD – German National Research Center for Information Technology, 53754 Sankt Augustin, Germany
Email: christophe.garcia@gmd.de

Introduction

The problem of object pose from 2D to 3D correspondences has received a lot of attention both in the photogrammetry and computer vision literatures. Various approaches to the object pose (or external camera parameters) problem fall into two distinct categories: closed-form solutions and non-linear solutions. Closed-form solutions may be applied only to a limited number of correspondences [1,2]. Whenever the number of correspondences is larger than four, closed-form solutions are not efficient and iterative non-linear solutions are necessary [3,4]. The latter approaches have two drawbacks: (i) they need a good initial estimate of the true solution, and (ii) they are time consuming. Therefore, such approaches can not be used in tasks that require high speed performance (visual servoing, object tracking, …) [5,6,7,8]. To our knowledge, the method proposed by DeMenthon and Davis [9] is among the first attempts to use linear techniques, associated with the weak perspective camera model in order to obtain the pose that is associated with the perspective camera model. The method starts with computing the object pose using a weak perspective model and after a few iterations converges towards a pose estimated under perspective.

In this paper we show how the initial method proposed by DeMenthon [9] (iterative weak perspective algorithm) can be used with both points and straight lines. Moreover,

1077-2014/99/030215 + 16 $30.00/0

we establish a link between paraperspective pose and perspective pose (iterative paraperspective algorithm). It has been argued that since features like straight lines are determined by a large number of pixels, the redundancy makes it possible to locate them accurately in the image. Furthermore, lines can be extracted even if they are partially hidden.

The remainder of this paper is organized as follows. In the following section we recall the fundamental equations relating image features (points and lines) to the pose parameters associated with the perspective camera model. The subsequent sections then describe the iterative weak perspective algorithm, and the iterative paraperspective algorithm, and provide details for solving the linear equations associated with pose computation in the case of planar objects. Then there are sections which describe a non-linear optimization method for determining the pose parameters, analyse the convergence of both the iterative weak and paraperspective algorithms, and provide an experimental comparison of the two methods described in this paper together with a comparison with the non-linear method. The final section provides some examples of application of pose estimation.

Background and Notations

We denote by P_i a 3D point with coordinates (X_i, Y_i, Z_i) in a frame that is attached to the object — the object frame (Figure 1). The origin of this frame is the object point P_0. We denote by $\mathcal{D}_j$ a 3D line that is described parametrically by its direction $\mathbf{V}_j$ and by a point vector $\mathbf{W}_j$. We suppose that the observed scene contains n points (P_1, …, P_n) (in addition to the reference point P_0) and m straight lines ($\mathcal{D}_1$, …, $\mathcal{D}_m$). These points and lines are expressed in the object frame.

An object point P_i projects onto the image in p_i with normalized camera coordinates x_i and y_i. An object line $\mathcal{D}_j$ projects onto the image in d_j with normalized coefficient (a_j, b_j, c_j). We denote by $\mathbf{P}_i$ the vector from point P_0 to P_i. The normalized camera coordinates of p_i are given by:

$$x_i = \frac{X_{ci}}{Z_{ci}} = \frac{\mathbf{i}\cdot\mathbf{P}_i + t_x}{\mathbf{k}\cdot\mathbf{P}_i + t_z} \tag{1}$$

$$y_i = \frac{Y_{ci}}{Z_{ci}} = \frac{\mathbf{j}\cdot\mathbf{P}_i + t_y}{\mathbf{k}\cdot\mathbf{P}_i + t_z} \tag{2}$$

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

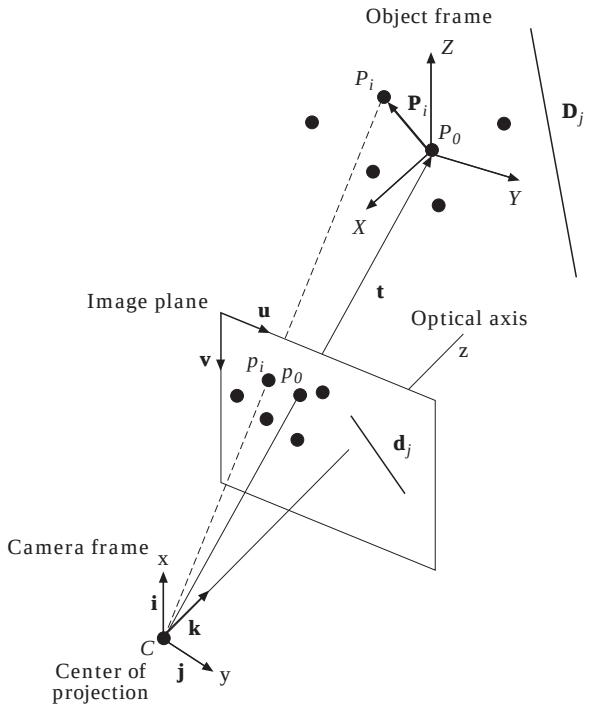


Figure 1. The pin-hole camera model.

$$T = \begin{bmatrix} \mathbf{i}^T & t_x \\ \mathbf{j}^T & t_y \\ \mathbf{k}^T & t_z \\ 0\ 0\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{3}$$

Vectors $\mathbf{i}^T$, $\mathbf{j}^T$, and $\mathbf{k}^T$ represent the three rows of the rotation matrix R.

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

$$u_i = \alpha_u x_i + u_c \tag{4}$$

$$v_i = \alpha_v y_i + v_c \tag{5}$$

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

Similarly one can express the normalized perspective projection of the straight line $\mathcal{D}_j$ as:

$$d_j : a_j x + b_j y + c_j = 0 \tag{6}$$

where x and y are related to the pose parameters by these two equations:

$$x = \frac{\mathbf{i} \cdot \mathbf{P}_j + t_x}{\mathbf{k} \cdot \mathbf{P}_j + t_z} \tag{7}$$

$$y = \frac{\mathbf{j} \cdot \mathbf{P}_j + t_y}{\mathbf{k} \cdot \mathbf{P}_j + t_z} \tag{8}$$

with $\mathbf{P}_j$ being a point on the line $\mathcal{D}_j$.

We divide both the numerator and the denominator of Eqns (1), (2), (7), and (8) 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 normalized camera coordinates of p_0 which is the projection of P_0 (the origin of the object frame);
- $\epsilon_i = \mathbf{k} \cdot \mathbf{P}_i / t_z$.

One can notice that $\mathbf{I}$ and $\mathbf{J}$ encapsulate the pose parameters (R and $\mathbf{t}$). We now rewrite the perspective equations (1), (2), and (6) as:

$$x_i = \frac{\mathbf{I} \cdot \mathbf{P}_i + x_0}{1 + \varepsilon_i} \tag{9}$$

$$y_i = \frac{\mathbf{J} \cdot \mathbf{P}_i + y_0}{1 + \varepsilon_i} \tag{10}$$

$$a_j\left(\mathbf{I} \cdot \mathbf{P}_j + x_0\right) + b_j\left(\mathbf{J} \cdot \mathbf{P}_j + y_0\right) + c_j\left(1 + \mathbf{k} \cdot \mathbf{P}_j / t_z\right) = 0 \tag{11}$$

Pose by Weak Perspective Iterations

Definition and equations

Weak perspective assumes that the object lies in a plane parallel to the image plane passing through the origin of the object frame (P_0). Geometrically the weak perspective projection is performed as follows (Figure 2). The object point P_i is first projected to the plane passing through the reference point P_0, which is parallel to the image plane. This projection is performed by using a ray that is parallel to the optical axis. Then the obtained point is perspectively projected to the image plane. We therefore obtain the point p_i^w. This is equivalent to a zero-order approximation:

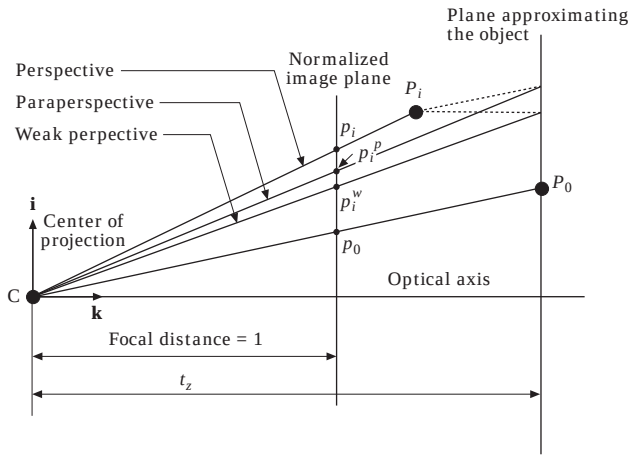


Figure 2. Weak perspective and paraperspective projections.

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

Using this approximation we can rewrite Eqns (9) and (10) as:

$$x_i^w = \mathbf{I} \cdot \mathbf{P}_i + x_0$$

$$y_i^w = \mathbf{J} \cdot \mathbf{P}_i + y_0$$

In these equations x_i^w and y_i^w are the camera coordinates of the weak perspective projection of P_i. By identification with Eqns (9) and (10) we obtain the relationship between the weak perspective and the perspective projections of P_i:

$$x_i^w = x_i\left(1 + \varepsilon_i\right) \tag{12}$$

$$y_i^w = y_i\left(1 + \varepsilon_i\right) \tag{13}$$

From Eqns (9) and (10) one can conclude that each point correspondence provides the following two constraints:

$$\mathbf{P}_i \cdot \mathbf{I} = x_i\left(1 + \varepsilon_i\right) - x_0 \tag{14}$$

$$\mathbf{P}_i \cdot \mathbf{J} = y_i\left(1 + \varepsilon_i\right) - y_0 \tag{15}$$

Each line $\mathcal{D}_j$ is described parametrically by its direction $\mathbf{V}_j$ and by a point vector $\mathbf{W}_j$. Thus, the equation of the line $\mathcal{D}_j$ in the object frame will be given by:

$$\mathbf{P}_j = \mathbf{W}_j + \lambda_j \mathbf{V}_j \left(\lambda_j \in \mathbb{R}\right) \tag{16}$$

where $\mathbf{P}_j$ is a vector from the origin frame P_0 to any point belonging to the line $\mathcal{D}_j$. By substituting this expression into Eqn (11) we obtain:

$$a_j\mathbf{W}_j\cdot\mathbf{I}+b_j\mathbf{W}_j\cdot\mathbf{J}+a_jx_0+b_jy_0+c_j\left(1+\mathbf{k}\cdot\mathbf{W}_j/t_z\right)+$$
$$\lambda_j\left(a_j\mathbf{V}_j\cdot\mathbf{I}+b_j\mathbf{V}_j\cdot\mathbf{J}+c_j\mathbf{k}\cdot\mathbf{V}_j/t_z\right)=0$$

Since this equation holds true for all λ_j, we obtain the following two constraints:

$$a_j\mathbf{W}_j\cdot\mathbf{I}+b_j\mathbf{W}_j\cdot\mathbf{J}+a_jx_0+b_jy_0+c_j\left(1+\eta_j\right)=0 \quad (17)$$
$$a_j\mathbf{V}_j\cdot\mathbf{I}+b_j\mathbf{V}_j\cdot\mathbf{J}+c_j\xi_j=0 \quad (18)$$

where η_j and ξ_j are given by:

$$\eta_j=\mathbf{k}\cdot\mathbf{W}_j/t_z$$
$$\xi_j=\mathbf{k}\cdot\mathbf{V}_j/t_z$$

In brief, each point correspondence provides the two constraints (14) and (15), and each line correspondence provides the two constraints (17) and (18). In matrix form these equations can be written as (we have *n* points and *m* lines):

$$\underbrace{G}_{(2n+2m)\times 6}\begin{bmatrix}\mathbf{I}\\ \mathbf{J}\end{bmatrix}=\underbrace{\mathbf{z}}_{(2n+2m)\times 1} \quad (19)$$

where G and $\mathbf{z}$ are a $(2n + 2m) \times 6$ matrix and a $(2n + 2m)$ vector, respectively:

$$G=\begin{bmatrix}\vdots & \vdots\\ \mathbf{P}_i^T & \mathbf{0}^T\\ \vdots & \vdots\\ \mathbf{0}^T & \mathbf{P}_i^T\\ \vdots & \vdots\\ a_j\mathbf{W}_j^T & b_j\mathbf{W}_j^T\\ \vdots & \vdots\\ a_j\mathbf{V}_j^T & b_j\mathbf{V}_j^T\\ \vdots & \vdots\end{bmatrix}\quad \mathbf{z}=\begin{bmatrix}\vdots\\ x_i\left(1+\varepsilon_i\right)-x_0\\ \vdots\\ y_i\left(1+\varepsilon_i\right)-y_0\\ \vdots\\ -a_jx_0-b_jy_0-c_j\left(1+\eta_j\right)\\ \vdots\\ -c_j\xi_j\\ \vdots\end{bmatrix}$$

Pose by successive approximations

In order to estimate the pose parameters one is left with the estimation of the vectors **I** and **J**. One can notice that if ϵ_i, η_j, and ξ_j are set to zero then (i) the matrix equation (19) becomes linear in **I** and **J** and (ii) the image features are supposed to be obtained with a weak perspective camera model [see Eqns (12) and (13)]. Once the pose parameters have been derived from **I** and **J**, one can update the value of ϵ_i, η_j, and ξ_j (this in turn will update the entries of the vector **z**). Thus it is possible to solve equation (19) by successive linear approximations. In this case the pose algorithm starts with a weak perspective camera model and computes an approximated pose. This approximated pose is improved iteratively as follows:

1. For all i and $j, i\in\{1\cdots n\}, j\in\{1\cdots m\}, (n+m)\geq 3$, $\varepsilon_i=0, \eta_j=0, \xi_j=0$.
2. Solve the overconstrained linear system (19) which provides an estimation of vectors **I** and **J**:
$$\begin{bmatrix}\mathbf{I}\\ \mathbf{J}\end{bmatrix}=\left(G^TG\right)^{-1}G^T\mathbf{z}$$
3. Compute the pose parameters, i.e. the position and orientation of the object frame with respect to the camera frame:
$$t_z=\frac{1}{2}\left(\frac{1}{\|\mathbf{I}\|}+\frac{1}{\|\mathbf{J}\|}\right)$$
$$t_x=x_0t_z$$
$$t_y=y_0t_z$$
$$\mathbf{i}=\frac{\mathbf{I}}{\|\mathbf{I}\|}$$
$$\mathbf{i}=\frac{\mathbf{J}}{\|\mathbf{J}\|}$$
$$\mathbf{k}=\mathbf{i}\times\mathbf{j}$$
4. For all i and j, compute:
$$\varepsilon_i=\frac{\mathbf{k}\cdot\mathbf{P}_i}{t_z}, \eta_j=\frac{\mathbf{k}\cdot\mathbf{W}_j}{t_z}, \xi_j=\frac{\mathbf{k}\cdot\mathbf{V}_j}{t_z}$$

If the changes in ϵ_i, η_j, and ξ_j in two consecutive iterations are below a fixed threshold then stop the procedure, otherwise go to step 2.

The matrix G has full rank since it is assumed that the observed scene is non-coplanar. One may notice that the

pseudo-inverse of G [i.e. $(G^TG)^{-1}\ G^T$] can be computed once and for all, and hence it can be computed independently of the loop presented above. Therefore the estimation of **I** and **J** is particularly efficient. The initial value of ϵ_i, η_j, and ξ_j can be fixed with any value that is not necessarily null. However, the null value is a heuristic choice since ϵ_i, η_j, and ξ_j have algebraic values (generally, the scene features can lie on either side of the plane passing through the reference point P_0).

Pose by Paraperspective Iterations

Definition and equations

The notion of paraperspective projection was introduced by Ohta *et al.* [10] and named by Aloimonos [11]. Geometrically paraperspective is performed as follows (Figure 2). The point P_i is first projected to the plane passing through the reference point P_0, which is parallel to the image. This projection is performed by using a ray that is parallel to the ray going through the projection center C and the reference point P_0. The obtained point is then perspectively projected to the image plane. We therefore obtain the point y_i^p.

Paraperspective may be viewed as a first-order approximation of perspective:

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

By using this approximation in Eqns (9) and (10) we obtain the paraperspective projection of P_i:

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

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

Thus, the paraperspective equations are:

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

By identification with Eqns (9) and (10) we obtain the relationship between the paraperspective and the perspective projections of P_i:

$$x_i^p = x_i\left(1+\varepsilon_i\right) - x_0\varepsilon_i \quad (20)$$

$$y_i^p = y_i\left(1+\varepsilon_i\right) - y_0\varepsilon_i \quad (21)$$

The paraperspective coordinates are related to the pose parameters by:

$$x_i^p - x_0 = \frac{\mathbf{i}-x_0\mathbf{k}}{t_z}\cdot\mathbf{P}_i \quad (22)$$

$$y_i^p - y_0 = \frac{\mathbf{j}-y_0\mathbf{k}}{t_z}\cdot\mathbf{P}_i \quad (23)$$

By substituting Eqns (20) and (21) in Eqns (22) and (23), we obtain:

$$\mathbf{P}_i\cdot\mathbf{I}_p = \left(x_i - x_0\right)\left(1+\varepsilon_i\right) \quad (24)$$

$$\mathbf{P}_i\cdot\mathbf{J}_p = \left(y_i - y_0\right)\left(1+\varepsilon_i\right) \quad (25)$$

with:

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

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

The relationship between vectors (**I**, **J**) and vectors ($\mathbf{I}_p$, $\mathbf{J}_p$) can be derived from (26) and (27):

$$\mathbf{I} = \mathbf{I}_p + \frac{x_0\mathbf{k}}{t_z}$$

$$\mathbf{J} = \mathbf{J}_p + \frac{y_0\mathbf{k}}{t_z}$$

By substituting these expressions into Eqns (17) and (18), these become:

$$a_j\mathbf{W}_j\cdot\mathbf{I}_p + b_j\mathbf{W}_j\cdot\mathbf{J}_p + \left(a_jx_0 + b_jy_0 + c_j\right)\left(1+\eta_j\right) = 0 \quad (28)$$

$$a_j\mathbf{V}_j\cdot\mathbf{I}_p + b_j\mathbf{V}_j\cdot\mathbf{J}_p + \left(a_jx_0 + b_jy_0 + c_j\right)\xi_i = 0 \quad (29)$$

where η_j and ξ_j are given by:

$$\eta_j = \mathbf{k} \cdot \mathbf{W}_j / t_z$$
$$\xi_j = \mathbf{k} \cdot \mathbf{V}_j / t_z$$

In brief, each point correspondence provides the two constraints (24) and (25), and each line correspondence provides the two constraints (28) and (29). In matrix form these equations can be written as:

$$\underbrace{G}_{(2n+2m)\times 6}\begin{bmatrix}\mathbf{I}_p \\ \mathbf{J}_p\end{bmatrix} = \underbrace{\mathbf{z}_p}_{(2n+2m)\times 1} \tag{30}$$

where G has the expression given in a previous section and $\mathbf{z}_p$ is a $(2n + 2m)$ vector:

$$\begin{aligned}&\left[(x_i - x_0)(1+\varepsilon_i),\cdots,(y_i - y_0)(1+\varepsilon_i),\cdots,\right.\\ &-(a_j x_0 + b_j y_0 + c_j)(1+\eta_j),\cdots,\\ &\left.-(a_j x_0 + b_j y_0 + c_j)\xi_j,\cdots\right]^T\end{aligned}$$

Pose by successive approximations

As in the weak perspective case, one may notice that if ϵ_i, η_j, and ξ_j are set to zero then (i) Eqn (30) becomes linear in $\mathbf{I}_p$ and $\mathbf{J}_p$; and (ii) the image features are supposed to be obtained with a paraperspective camera model [see Eqns (20) and (21)]. Therefore, it is possible to solve this equation by successive linear approximations. In the following, we show how the pose parameters can be computed from $\mathbf{I}_p$ and $\mathbf{J}_p$.

Pose parameters. The pose parameters can be derived from $\mathbf{I}_p$ and $\mathbf{J}_p$ as follows.

First, one may notice that:

$$\|\mathbf{I}_p\|^2 = \frac{(\mathbf{i} - x_0\mathbf{k})\cdot(\mathbf{i} - x_0\mathbf{k})}{t_z^2} = \frac{1 + x_0^2}{t_z^2}$$
$$\|\mathbf{J}_p\|^2 = \frac{1 + y_0^2}{t_z^2}$$

We obtain:

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

Second, we derive the three orthogonal unit vectors **i**, **j**, and **k**. From Eqns (26) and (27) we can write:

$$\mathbf{i} = t_z\mathbf{I}_p + x_0\mathbf{k} \tag{31}$$
$$\mathbf{j} = t_z\mathbf{J}_p + y_0\mathbf{k} \tag{32}$$

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

$$\left(I_{3\times3} - t_z y_0\Omega(\mathbf{I}_p) + t_z x_0\Omega(\mathbf{J}_p)\right)\mathbf{k} = t_z^2\mathbf{I}_p\times\mathbf{J}_p \tag{33}$$

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

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

is of the form:

$$A = \begin{bmatrix}1 & \gamma & -\beta\\ -\gamma & 1 & \alpha\\ \beta & -\alpha & 1\end{bmatrix}$$

Its determinant is always strictly positive:

$$det(A) = 1 + \alpha^2 + \beta^2 + \gamma^2$$

Therefore, one can easily determine **k** using Eqn (33) and **i** and **j**, using Eqns (31) and (32).

Pose by successive approximations. The algorithm can be written as follows.

1. For all i and $j, i \in \{1 \cdots n\}, j \in \{1 \cdots m\}, (n+m) \geq 3$,
   $\varepsilon_i = 0, \eta_j = 0, \xi_j = 0.$
2. Solve the overconstrained linear system (30) which provides an estimation of vectors and $\mathbf{I}_p$ and $\mathbf{J}_p$:
$$\begin{bmatrix} \mathbf{I}_p \\ \mathbf{J}_p \end{bmatrix} = \left(G^T G\right)^{-1} G^T \mathbf{z}_p$$
3. Compute the pose parameters, i.e. the position (t_x, t_y, and t_z) and orientation (**i**, **j**, and **k**) as explained above;
4. For all i and j, compute:
$$\varepsilon_i = \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z}, \eta_j = \frac{\mathbf{k} \cdot \mathbf{W}_j}{t_z}, \xi_j = \frac{\mathbf{k} \cdot \mathbf{V}_j}{t_z}$$

If the changes in ϵ_i, η_j, and ξ_j in two consecutive iterations are below a fixed threshold then stop the procedure, otherwise go to step 2.

Planar Objects

If the object is planar then the matrix G is not of full rank. Therefore, step 2 of both the weak perspective and paraperspective algorithms cannot be considered anymore. In this case, we consider the plane of the object and let **u** be the unit vector orthogonal to this plane. Vectors $\mathbf{I}_p$ and $\mathbf{J}_p$ (and equivalently **I** and **J**) can be written as a sum of a vector belonging to this plane and a vector perpendicular to this plane (Figure 3).

$$\mathbf{I}_p = \mathbf{I}_0 + \lambda \mathbf{u} \tag{34}$$
$$\mathbf{J}_p = \mathbf{J}_0 + \mu \mathbf{u} \tag{35}$$

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

$$G \begin{bmatrix} \mathbf{I}_0 \\ \mathbf{J}_0 \end{bmatrix} = \mathbf{z}_p$$

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

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

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

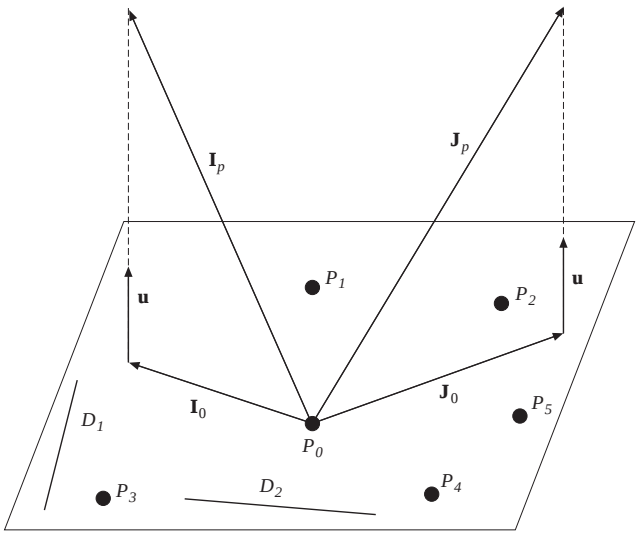


Figure 3. A planar object.

$$\begin{bmatrix} \mathbf{I}_0 \\ \mathbf{J}_0 \end{bmatrix} = \left(G'^T G'\right)^{-1} G'^T \begin{bmatrix} \mathbf{z}_p \\ 0 \\ 0 \end{bmatrix}$$

With G' defined by:

$$G' = \begin{bmatrix} G \\ \mathbf{u}^T, \mathbf{0}^T \\ \mathbf{0}^T, \mathbf{u}^T \end{bmatrix}$$

Obviously, the rank of G' is equal to 6. In order to estimate $\mathbf{I}_p$ and $\mathbf{J}_p$ (and equivalently **I** and **J**) one is left with the estimation of two scalars, λ and μ. In the case of weak perspective one can determine a solution using the constraints $\|\mathbf{I}\| = \|\mathbf{J}\|$ and $\mathbf{I} \cdot \mathbf{J} = 0$. In the case of paraperspective we use the following constraints onto $\mathbf{I}_p$ and $\mathbf{J}_p$ [derived from Eqns (26) and (27)]:

$$\left\|\mathbf{I}_p\right\|^2 = \frac{1 + x_0^2}{t_z^2}$$
$$\left\|\mathbf{J}_p\right\|^2 = \frac{1 + y_0^2}{t_z^2}$$
$$\mathbf{I}_p \cdot \mathbf{J}_p = \frac{x_0 y_0}{t_z^2}$$

By eliminating t_z we obtain two constraints:

$$\mathbf{I}_p \cdot \mathbf{J}_p = \frac{x_0 y_0}{1+x_0^2}\left\|\mathbf{I}_p\right\|^2$$
$$\mathbf{I}_p \cdot \mathbf{J}_p = \frac{x_0 y_0}{1+y_0^2}\left\|\mathbf{J}_p\right\|^2$$

By substituting in these expressions $\mathbf{I}_p$ and $\mathbf{J}_p$ given by Eqns (34) and (35) we obtain:

$$\mathbf{I}_0 \cdot \mathbf{J}_0 + \lambda\mu = \frac{x_0 y_0}{1+x_0^2}\left(\left\|\mathbf{I}_0\right\|^2 + \lambda^2\right)$$
$$\mathbf{I}_0 \cdot \mathbf{J}_0 + \lambda\mu = \frac{x_0 y_0}{1+y_0^2}\left(\left\|\mathbf{J}_0\right\|^2 + \mu^2\right)$$

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

$$\mathcal{A}\lambda^4 + \mathcal{B}\lambda^2 + C = 0 \qquad (36)$$

With:

$$\mathcal{A} = a^2 - g$$
$$\mathcal{B} = 2a^2 d - gd + e - 2ac$$
$$C = a^2 d^2 + c^2 - 2acd$$
$$a = \frac{x_0 y_0}{1+x_0^2}$$
$$b = \frac{x_0 y_0}{1+y_0^2}$$
$$c = \mathbf{I}_0 \cdot \mathbf{J}_0$$
$$d = \left\|\mathbf{I}_0\right\|^2$$
$$e = \left\|\mathbf{J}_0\right\|^2$$
$$g = \frac{1+y_0^2}{1+x_0^2}$$

In order to study the number of real roots of Eqn (36) we substitute λ^2 by t:

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

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

$$\frac{C}{\mathcal{A}} = f(x_0, y_0, c, d)$$
$$= \frac{-x_0^2 y_0^2 d^2 - \left(1+x_0^2\right)^2 c^2 + 2x_0 y_0\left(1+x_0^2\right)cd}{1+x_0^2+y_0^2}$$
$$= -\frac{\left[x_0 y_0 d - \left(1+x_0^2\right)c\right]^2}{1+x_0^2+y_0^2}$$

Therefore, the value of $C/\mathcal{A}$ is either negative or null. Thus there is 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 and hence there are two solutions for $\mathbf{I}_p$ and $\mathbf{J}_p$. 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 the previous section 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 iteration 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 (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.

Non-linear Optimization

In this section we propose to estimate the pose parameters associated with the perspective model of the camera. For this purpose, we rewrite the perspective Eqns (1) and (2) associated with the point P_i as:

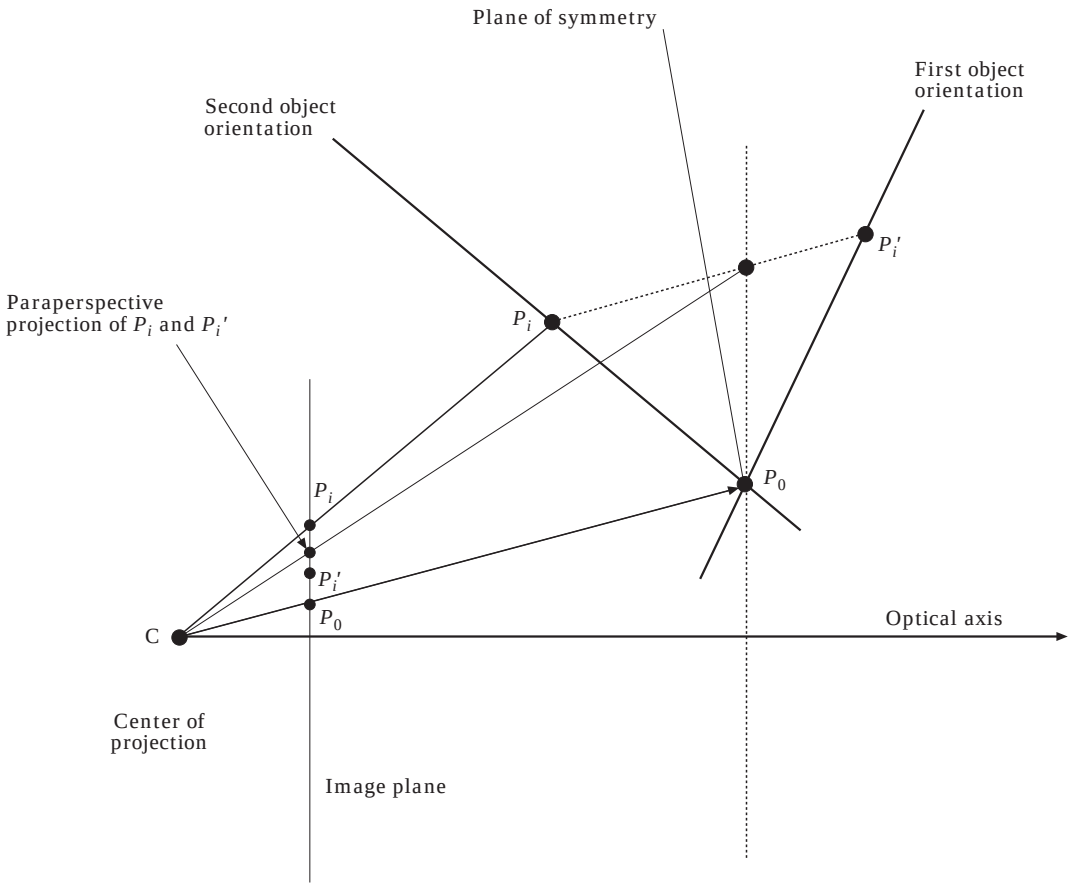


Figure 4. A planar object and a paraperspective camera model produces two orientations. However, these two object orientations have distinct perspective projections – p_i and p_i'.

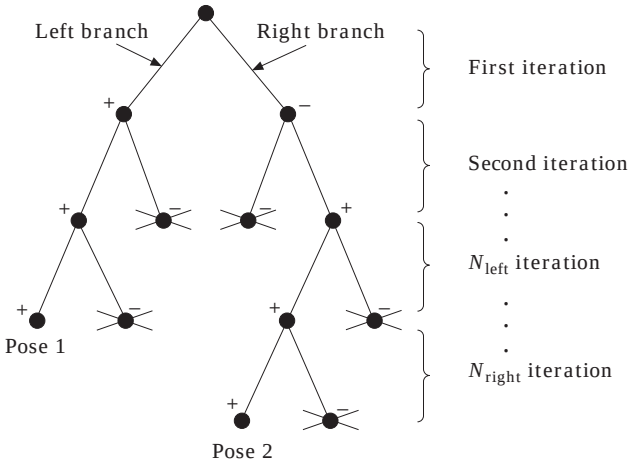


Figure 5. A binary tree search for selecting the best object pose when the object is planar. The algorithm produces two poses at each iteration (+: a good solution, −: a bad solution).

$$r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_x - x_i\left(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z\right) = 0$$
$$r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_y - y_i\left(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z\right) = 0$$

where (X_i, Y_i, Z_i) are the 3D coordinates of P_i.

Let $\mathbf{q} = (q_0, q_x, q_y, q_z)^T$ be the unit quaternion associated with the rotation R [12], therefore the matrix R can be written as:

$$R = \begin{bmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{bmatrix}$$
$$= \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2\left(q_x q_y - q_0 q_z\right) & 2\left(q_x q_z + q_0 q_y\right) \\ 2\left(q_x q_y + q_0 q_z\right) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2\left(q_y q_z - q_0 q_x\right) \\ 2\left(q_x q_z - q_0 q_y\right) & 2\left(q_y q_z + q_0 q_x\right) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

From point correspondences one can build a positive error function f_{points}:

$$f_{points} = \sum_{i=0}^{n} \Big(r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_x - x_i\big(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z\big)\Big)^2 + \Big(\big(r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_y - y_i\big(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z\big)\Big)^2$$

We now rewrite the constraints (17) and (18) associated with the line $\mathcal{D}_j$ as:

$$a_j\mathbf{W}_j \cdot \mathbf{i} + b_j\mathbf{W}_j \cdot \mathbf{j} + c_j\mathbf{W}_j \cdot \mathbf{k} + a_j t_x + b_j t_y + c_j t_z = 0$$
$$a_j\mathbf{V}_j \cdot \mathbf{i} + b_j\mathbf{V}_j \cdot \mathbf{j} + c_j\mathbf{V}_j \cdot \mathbf{k} = 0$$

These constraints can be written more compactly as:

$$\mathbf{N}_j \cdot \big(R\mathbf{W}_j + \mathbf{t}\big) = 0$$
$$\mathbf{N}_j \cdot \big(R\mathbf{V}_j\big) = 0$$

with $\mathbf{N}_j = (a_j, b_j, c_j)^T$.

Similarly one can build a positive error function f_{lines}:

$$f_{lines} = \sum_{j=1}^{m}\left[\Big(\mathbf{N}_j \cdot \big(R\mathbf{V}_j\big)\Big)^2 + \Big(\mathbf{N}_j \cdot \big(R\mathbf{W}_j + \mathbf{t}\big)\Big)^2\right]$$

Thus the problem of pose estimation from point and line correspondences can be stated in terms of the following minimization problem:

$$\min_{\mathbf{q},\mathbf{t}} \left(\lambda_p\, f_{points} + \lambda_l\, f_{lines} + \lambda\big(1 - \mathbf{q}^T\mathbf{q}\big)^2\right) \qquad (37)$$

which has the form of sum of squares of non-linear functions, and $\lambda\,(1 - \mathbf{q}^T\mathbf{q})^2$ is a penalty function that guarantees that $\mathbf{q}$ has a module equal to 1. λ_p and λ_l are two weights, and λ is a real positive number. High values for λ ensure that the module of the quaternion is close to 1.

In order to minimize this function we have used the Levenberg-Marquardt non-linear minimization method as described in Press *et al.* [13]. In all our experiments we have:

$$\begin{cases}\lambda_p = \lambda_l = 1\\ \lambda = 2.10^7\end{cases}$$

An Analysis of Convergence

In order to analyse the convergence of the two algorithms (see previous sections) we consider Figure 6. On this figure are depicted the three kinds of projection of the point P_i. The two iterative procedures start with the perspective projection p_i to automatically compute locations p_i^w and p_i^p. Therefore the convergence properties (i.e. the rate of convergence and the number of iterations) of the two procedures will depend on distances $\|p_i - p_i^w\|$ and $\|p_i - p_i^p\|$, respectively.

The projection error in the case of the iterative weak perspective algorithm will be given by [see Eqns (12) and (13)]:

$$\Delta_i^w = \big\|p_i - p_i^w\big\| = \sqrt{x_i^2 + y_i^2}\,\big|\varepsilon_i^*\big|$$

where ϵ_i^* is the true value that the algorithm is supposed to compute. Thus, this error is proportional to the distance between p_i and the image center.

The projection error in the case of the iterative paraperspective algorithm will be given by [see Eqns (20) and (21)]:

$$\Delta_i^p = \big\|p_i - p_i^p\big\| = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}\,\big|\varepsilon_i^*\big| = \big\|p_i - p_0\big\|\,\big|\varepsilon_i^*\big|$$

This error is proportional to the distance $\|p_i\, p_0\|$. We can conclude that the convergence properties of the iterative weak perspective method depend on the translational offset of the object from the optical axis. On the other hand the convergence properties of the paraperspective method do not depend on this offset. Whenever one wants to use the paraperspective method, the choice for p_0 is crucial. The best way to choose p_0 is to compute the center of gravity of all image features and to select the image point which is the closest to this center of gravity.

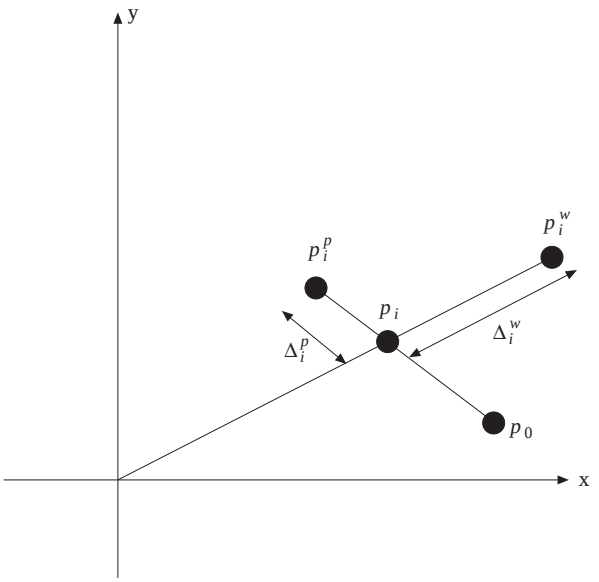


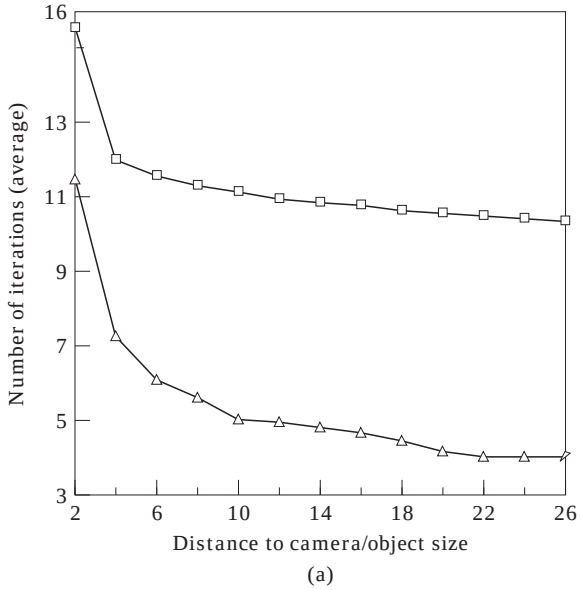
Figure 6. Projection errors $\triangle_i^w$ and $\triangle_i^p$ which are associated with the weak perspective and paraperspective camera models, respectively.

Experiments

Convergence comparison

In this section we compare the convergence of the iterative weak and paraperspective algorithms as a function of position and orientation of the object with respect to the camera. We consider a simulated object formed by a configuration of four points, such that the three line segments joining the reference point to the other three points are equal and perpendicular to each other. We performed the following kinds of experiments. The first kind is meant to compare the number of iterations needed by each algorithm 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 and the optical axis. For each such offset and for each depth we randomly selected 1000 different orientations and ran both algorithms for each such position and orientation. We plot the average value of the number of iterations over all the 1000 orientations.

Figure 7 shows the number of iterations as a function of relative depth. The lines with squares correspond to the weak perspective algorithm, the lines with triangles correspond to the paraperspective algorithm. Figures (7a) and (7b) correspond to two offsets, 23° and 30°, respectively.



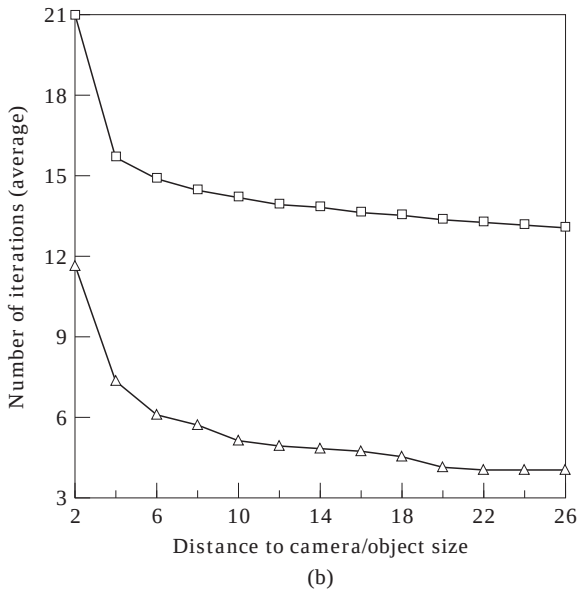


Figure 7. Speed of convergence as a function of depth, the offset is equal to 23˚ (a), and to 30˚ (b), (triangles: paraperspective, squares: weak perspective).

On average, the paraperspective algorithm is up to three times faster than the weak perspective algorithm.

The second kind is meant to compare the rate of convergence of each one of these algorithms for small distance/size ratios. Figures 8(a) and 8(b) show the percentage of the convergence of both algorithms as a function of depth. One can verify that the convergence properties of the paraperspective algorithm (number of iterations and rate of

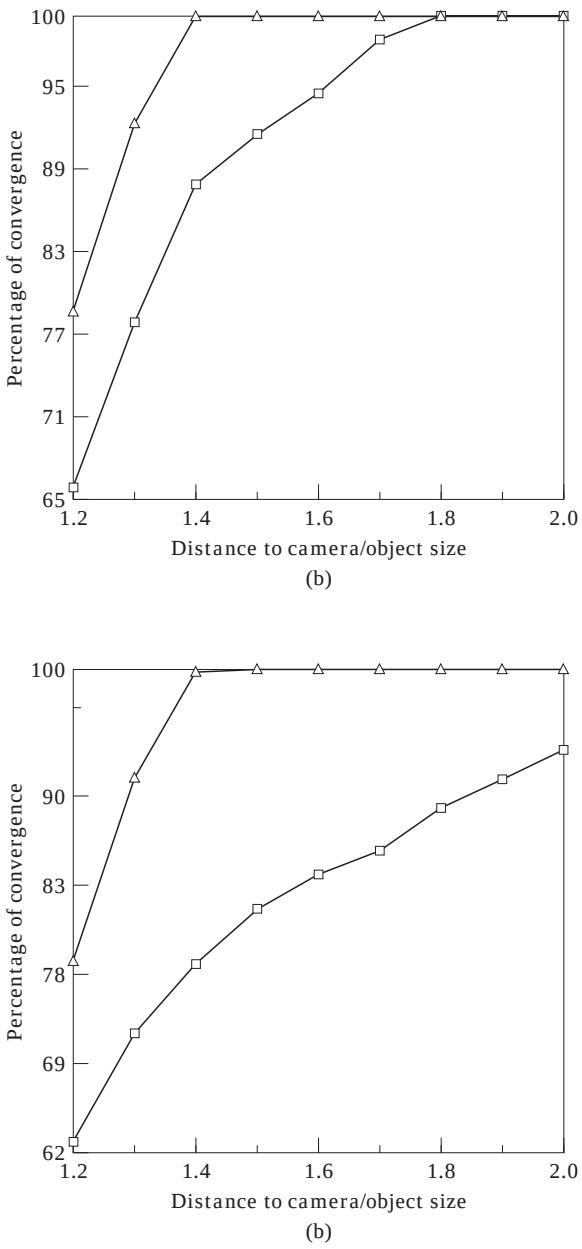


Figure 8. Rate of convergence as a function of depth, the offset is equal to 30° (a), and to 35° (b), (triangles: paraperspective, squares: weak perspective).

convergence) do not depend on the object offset as expected in the previous section (Figures 7 and 8).

Stability comparison

In this section we study the precision of pose as a function of image noise. More quantitatively, we compute the error between the theoretical pose and the pose computed by an algorithm. The pose errors are: rotation error and translation error. The rotation error is defined as the rotation angle in degrees required to align the coordinate system in its computed orientation with the coordinate system in its theoretical orientation. The translation 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.

Table 1 shows the average of pose errors as a function of image uniform noise. The second and the third columns correspond to the paraperspective algorithm while the fourth and the fifth columns correspond to the non-linear method. The iterative weak perspective algorithm yields the same errors as the paraperspective algorithm. The simulated object is a cube (with seven vertices and four edges). Its distance to the camera divided by its size is equal to 7. The number of trials for each noise level is equal to 100. The uniform noise has been added to the 2D lines.

Examples

Figure 9 shows an example of convergence of the paraperspective algorithm when it is applied to compute the pose of a cube. We extract intensity edges from the image using the optimal Deriche edge detector [14]. Edge pixels are labeled using a simple hysteresis threshold method. Then the equation of each straight line is obtained by applying the Hough Transform to the set of edge pixels.

The left column shows the algorithm behavior using seven vertices and six edges, the right column shows the behavior using seven vertices and nine edges. The first iteration of the algorithm found a paraperspective pose (top-right). After only three iterations the algorithm correctly determined the pose of the cube (bottom-right). This computation (right column) takes three iterations (3.3 ms on an Ultra-Sparc). As one can easily notice, the pose computation may become more accurate by increasing the number of lines [Figure 9 (bottom left) and (bottom right)]. The application of the weak perspective algorithm to the same cube gives the same behavior as the paraperspective algorithm since the cube is centered in the image.

Table 2 illustrates the speed of convergence and the computation time of the iterative paraperspective and the non-linear algorithms as a function of line correspondences (Figure 9). The non-linear algorithm was initialized with the solution obtained at the first iteration of the paraperspective algorithm.

Table 1. Average error in rotation and translation as a function of image uniform noise. The simulated object is a cube (with seven points and four lines). Its distance to the camera divided by its size is equal to 7. The noise is added to the 2D line parameters. The number of trials for each noise level is equal to 100.

	Iterative paraperspective		Non-linear	
Noise level (%)	Rot. error (deg.)	Trans. error (%)	Rot. error (deg.)	Trans. error (%)
1	0.45	0.24	0.08	0.06
2	0.88	0.50	0.16	0.13
3	1.32	0.75	0.24	0.19
4	1.72	0.99	0.32	0.26
5	2.20	1.25	0.41	0.32
6	2.63	1.49	0.49	0.41

Table 2. A comparison of the iterative paraperspective and the non-linear method as a function of the number of correspondences, the computer being used is an Ultra-Sparc.

	Iterative paraperspective		Non-linear
No. of correspondences	No. of iterations	CPU time (ms)	CPU time (ms)
7 points	3	1.64	12.8
7 points + 1 edge	3	1.86	14.2
7 points + 2 edges	3	2.04	15.6
7 points + 3 edges	4	2.39	17.0
7 points + 4 edges	4	2.58	18.3
7 points + 5 edges	4	2.76	19.8
7 points + 6 edges	6	3.34	21.6
7 points + 7 edges	3	2.95	22.6
7 points + 8 edges	3	3.13	24.0
7 points + 9 edges	3	3.31	25.6

Figure 10 illustrates the pose estimation of a cube (its size is 7 cm) and a gripper by the paraperspective algorithm. The gripper is identified by five vertices and five edges, the cube is identified by six vertices and seven edges. By combining the obtained poses one can obtain the relative position and orientation of the gripper with respect to the cube. For example, the relative position which is given by the translation vector gripper-cube has been found to be: $(20\ cm, 1.9\ cm, -5.9\ cm)^T$. The origins of the two coordinate systems are shown by large crosses [Figure 10 (right)]. Therefore, by tracking the gripper location in the image, one can apply visual servoing approaches in order to guide the gripper such that it can grasp the cube [8,15,16]. Table 3 gives the residual errors in the image plane between the true features and the projected 3D model associated with the two computed poses (gripper and cube).

We now consider the plane formed by the upper face of the gripper (four vertices and three edges). We apply the three algorithms presented in this paper: the weak perspective algorithm, the paraperspective algorithm and the non-linear algorithm. The corresponding computation times are: 4.5 ms, 3.2 ms and 17 ms, respectively. Since this object is not near the optical axis then it is evident that the paraperspective algorithm is faster than the weak perspective algorithm.

Table 3. Pose estimation of both the gripper and the cube using the paraperspective algorithm, the computer being used is an Ultra-Sparc.

Residual error (image space)	Gripper	Cube
Vertices locations (pixels)	0.8	0.9
Edges orientations (deg.)	0.38	1.4
Edges locations (pixels)	0.72	6.0
Number of iterations	3	3
CPU time (ms)	2.3	2.8

Figure 11 illustrates the application of the paraperspective algorithm to two planar objects: a screen and a table. The screen is identified by four points and four lines. The table is identified by four points and three lines. Using the two obtained poses one can compute the relative orientation between the two planes. This orientation has been found to be 27°.

Conclusion

In this paper we focused on the problem of pose computation from 2D to 3D point and line correspondences. We proposed two fast methods. The first one is a generalization of DeMenthon's algorithm (the weak perspective algorithm).

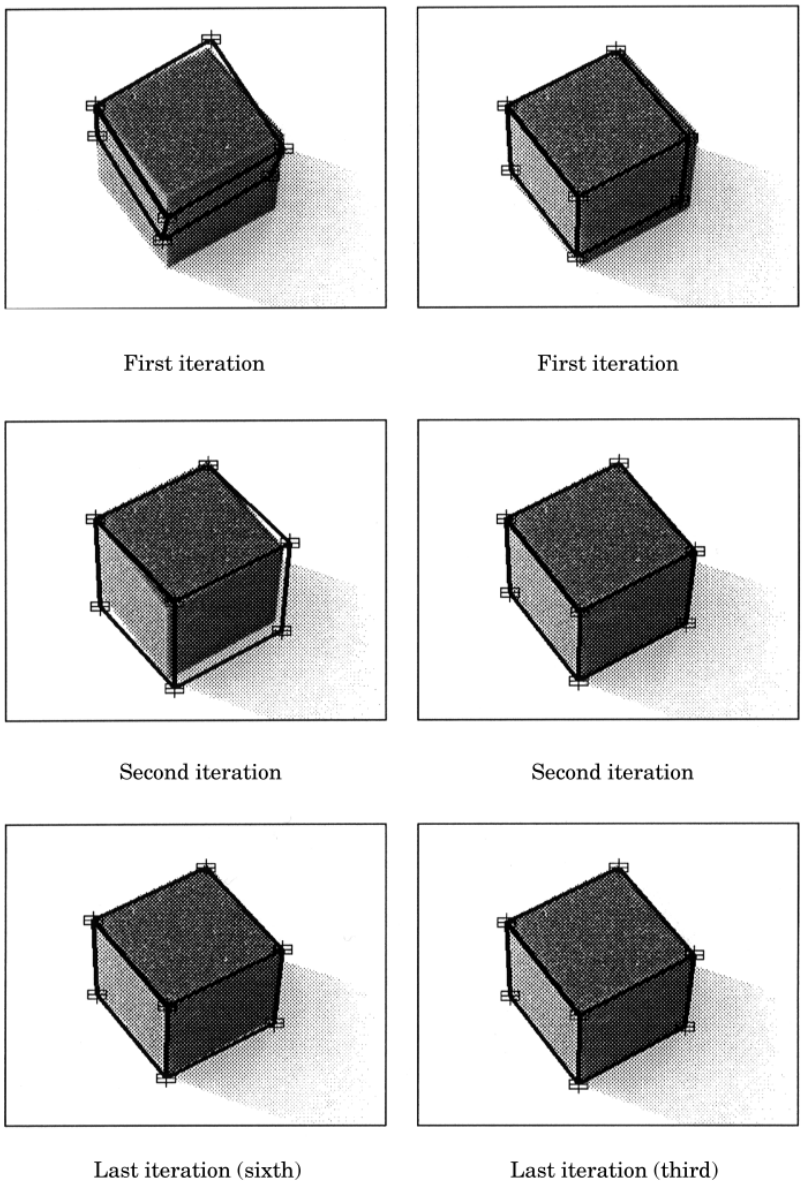


Figure 9. An example of applying the iterative paraperspective algorithm to a cube. The left column corresponds to seven vertices and six edges (peripheral), the right column corresponds to seven vertices and nine edges. This computation takes three iterations (3.3 ms on an Ultra-Sparc) (right column).

The second one establishes a link between paraperspective and perspective. The resulting methods are very elegant, very fast, and quite accurate. It seems that these two methods do not fail for complex scenes. We studied, both theoretically and experimentally, the convergence of the iterative weak and paraperspective algorithms. We showed that the convergence properties of the second algorithm do not depend on the distance of the object with respect to the optical axis. We showed that, for compact objects, the second algorithm requires 2.5 times less iterations than the first 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 the weak perspective algorithm. However, these two algorithms always converge for scenes that are not too close to the camera.

We also described a non-linear method for computing object pose with a perspective camera model. We showed that, in the presence of noise, the performances of the itera-

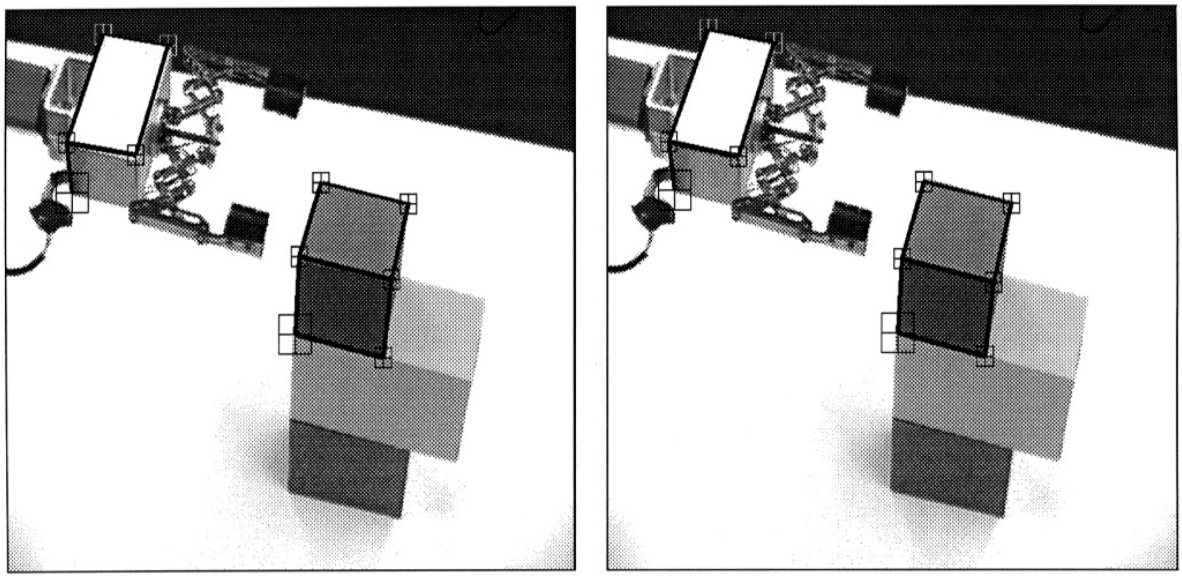


Figure 10. An example of applying the paraperspective algorithm to both a gripper and a cube. The two obtained poses allow one to compute the relative position and orientation between them.

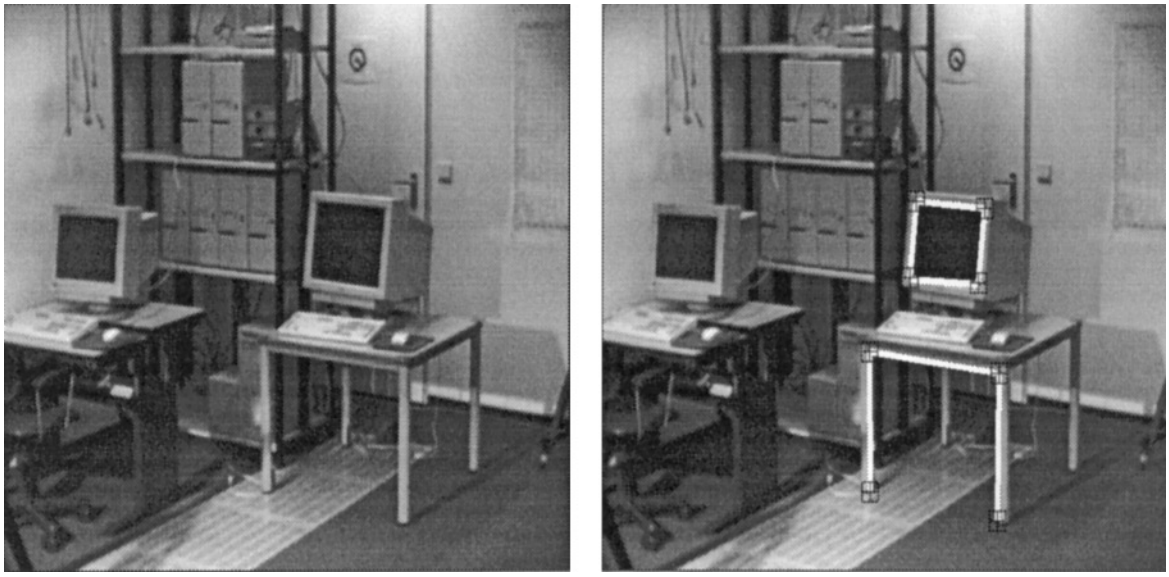


Figure 11. Pose estimation of two planar objects with the paraperspective algorithm: a screen and a table. The screen is identified by four points and four lines. The table is identified by four points and three lines. The two computed poses allow the computation of the relative orientation between the two planes. In this case, the relative orientation has been found to be 27˚.

tive linear methods degrade faster with increasing noise amplitude than the non-linear method. Whenever speed is an important concern, iterative linear methods should however be preferred. They can be included in real-time vision and robotics applications.

References

1. Dhome, M., Richetin, M., Lapreste, J.T. & Rives, G. (1989) Determinition of the attitude of 3D objects from a single perspective view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11**: 1265–1278.

2. Horaud, R., Conio, B., Leboulleux, O. & Lacolle, B. (1989) An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, **47**: 33–44.
3. Yuan, J. (1989) A general photogrammetric method for determining object position and orientation. *IEEE Transactions on Robotics and Automation*, **5**: 129–142.
4. Phong, T.Q., Horaud, R., Yassine, A. & Pham, D.T. (1993) Optimal estimation of object pose from a single perspective view. In: *Proceedings of the Fourth International Conference on Computer Vision*.
5. Papanikolopoulos, N., Khosla, P. & Kanade, T. (1993) Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *IEEE Transactions on Robotics and Automation*, **9**: 14–35.
6. Denker, A., Sabanovic, A. & Kaynak, O. (1994) Vision-controlled robotic tracking and acquisition. In: *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, **3**: 2000–2006.
7. Espiau, B., Chaumette, F. & Rives, P. (1992) A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, **8**: 313–326.
8. Hager, G.D. (1994) Real-time feature tracking and projective invariance as a basis for hand-eye coordination. In: *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 533–539
9. DeMenthon, D. & Davis, L. (1995) Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, **15**: 123–141.
10. Ohta, Y., Maenobu, K. & Sakai, T. (1981) Obtaining surface orientation from texels under perspective projection. In: *Proceedings of the 7th IJCAI*.
11. Aloimonos, J. (1989) Shape from texture. *Biological Cybernetics*, **58**: 345–360.
12. Horn, B.K.P. (1987) Closed-form solution of absolute orientation using unit quaternions. *J Opt Soc Amer A*, **4**: 629–642.
13. Press, W.H., Flannery, B.P., Teukolsky, S.A. & Wetterling, W.T. (1988) *Numerical Recipes in C: The Art of Scientific Computing*. New York: Cambridge University Press.
14. Deriche, R. (1987) Using Canny's criteria to derive an optimal edge detector recursively implemented. *International Journal of Computer Vision*, **2**: 167–187.
15. Hollinghurst, N. & Cipolla, R. (1993) Uncalibrated stereo hand-eye coordination. In: *Proceedings of the Fourth British Machine Vision Conference (BMVC 93)*.
16. Horaud, R., Dornaika, F., Bard, C. & Laugier, C. (1995) Integrating grasp planning and visual servoing for automatic grasping. In: *Proceedings of the Fourth International Symposium on Experimental Robotics*.