Object Pose by Affine Iterations

Fadi Dornaika and Christophe Garcia

Institute For System Design Technology
GMD – German National Research Center for Information Technology
Sankt Augustin, Germany, E-mail: Christophe.Garcia@gmd.de

Abstract. The problem of a real-time pose estimation between a 3D scene and a 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 a fast method for estimating the 3D pose using 2D to 3D point and line correspondences. This method is inspired by 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, at convergence the computed pose corresponds to the perspective camera model. Our method is based on the iterative use of a paraperspective camera model which is a first order approximation of perspective. Experiments involving synthetic data as well as real range data indicate the feasibility and robustness of this method.

1 Introduction and motivation

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 2 distinct categories: closed-form solutions and non-linear solutions. Closed-form solutions may be applied only to a limited number of correspondences [2], [5]. Whenever the number of correspondences is larger than 4 then closed-form solutions are not efficient and iterative non-linear solutions are necessary [7]. 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, ...) [3]. To our knowledge, the method proposed by DeMenthon & Davis [1] 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 weak perspective model and after a few iterations converges towards a pose estimated under perspective.

In this paper we establish a link between paraperspective model and perspective model in order to estimate the pose using both points and straight lines. 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 occulted.

2 Background and notations

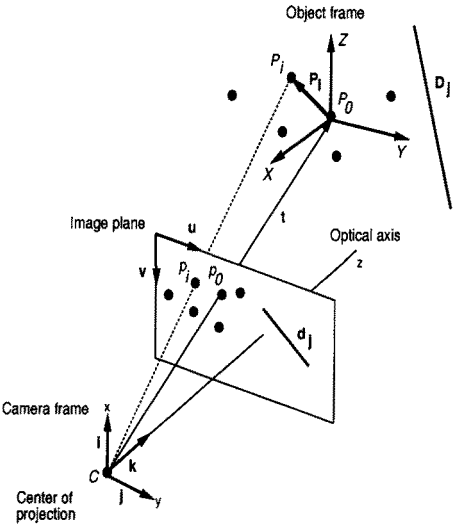


Fig. 1. The pin-hole camera model.

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. 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 (see Figure 1).

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 point 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:

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

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

$$u_i = \alpha_u \, x_i + u_c$$
$$v_i = \alpha_v \, y_i + v_c$$

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:

$$d_j : \; a_j \, x + b_j \, y + c_j = 0 \tag{3}$$

Where x and y are related to the pose parameters by these 2 equations:

$$x = \frac{\mathbf{i} \cdot \mathbf{P}_j + t_x}{\mathbf{k} \cdot \mathbf{P}_j + t_z} \tag{4}$$

$$y = \frac{\mathbf{j} \cdot \mathbf{P}_j + t_y}{\mathbf{k} \cdot \mathbf{P}_j + t_z} \tag{5}$$

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

We divide both the numerator and the denominator of eqs. (1), (2), (4), and (5) 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 (3) as:

$$x_i = \frac{\mathbf{I} \cdot \mathbf{P}_i + x_0}{1 + \epsilon_i} \tag{6}$$

$$y_i = \frac{\mathbf{J} \cdot \mathbf{P}_i + y_0}{1 + \epsilon_i} \tag{7}$$

$$a_j \, (\mathbf{I} \cdot \mathbf{P}_j + x_0) + b_j \, (\mathbf{J} \cdot \mathbf{P}_j + y_0) + c_j \, (1 + \mathbf{k} \cdot \mathbf{P}_j/t_z) = 0 \tag{8}$$

Each line $\mathcal{D}_j$ is described parametrically by its direction $\mathbf{V}_j$ and by a point vector $\mathbf{W}_j$. Thus, we can write:

$$\mathbf{P}_j = \mathbf{W}_j + \lambda_j \, \mathbf{V}_j \quad (\lambda_j \in I\!R)$$

By substituting this expression into eq. (8) and considering that this equation holds true for all λ_j, we obtain the following two constraints:

$$a_j \, \mathbf{W}_j \cdot \mathbf{I} + b_j \, \mathbf{W}_j \cdot \mathbf{J} + a_j \, x_0 + b_j \, y_0 + c_j \, (1 + \eta_j) = 0 \tag{9}$$

$$a_j \, \mathbf{V}_j \cdot \mathbf{I} + b_j \, \mathbf{V}_j \cdot \mathbf{J} + c_j \, \xi_j = 0 \tag{10}$$

where η_j and ξ_j are given by:

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

3 Pose by paraperspective iterations

3.1 Definition and equations

The notion of paraperspective projection was introduced by Ohta et al.[6]. Paraperspective may be viewed as a first-order approximation of perspective:

$$\frac{1}{1 + \epsilon_i} \approx 1 - \epsilon_i \quad \forall i, \; i \in \{1...n\}$$

By using this approximation in eqs. (6) and (7) we obtain the paraperspective projection of P_i:

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

where the term $1/t_z^2$ was neglected. There is a similar expression for y_i^p. By identification with eqs. (6) and (7) we obtain the relationship between the paraperspective and the perspective projections of P_i:

$$x_i^p = x_i \, (1 + \epsilon_i) - x_0 \epsilon_i \tag{11}$$

$$y_i^p = y_i \, (1 + \epsilon_i) - y_0 \epsilon_i \tag{12}$$

The paraperspective coordinates are related to the pose parameters by:

$$x_i^p - x_0 = \frac{\mathbf{i} - x_0 \, \mathbf{k}}{t_z} \cdot \mathbf{P}_i \tag{13}$$

$$y_i^p - y_0 = \frac{\mathbf{j} - y_0 \, \mathbf{k}}{t_z} \cdot \mathbf{P}_i \tag{14}$$

By substituting eqs. (11) and (12) in eqs. (13) and (14), we obtain:

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

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

with:

$$\mathbf{I}_p = \frac{\mathbf{i} - x_0 \, \mathbf{k}}{t_z} \quad \text{and} \quad \mathbf{J}_p = \frac{\mathbf{j} - y_0 \, \mathbf{k}}{t_z} \tag{17}$$

By using these relationships between vectors ($\mathbf{I} = \mathbf{i}/t_z$, $\mathbf{J} = \mathbf{j}/t_z$) and vectors ($\mathbf{I}_p$, $\mathbf{J}_p$) in eqs. (9) and (10), these ones become:

$$a_j\ \mathbf{W}_j \cdot \mathbf{I}_p + b_j\ \mathbf{W}_j \cdot \mathbf{J}_p + (a_j\ x_0 + b_j\ y_0 + c_j)\,(1 + \eta_j) = 0 \qquad (18)$$

$$a_j\ \mathbf{V}_j \cdot \mathbf{I}_p + b_j\ \mathbf{V}_j \cdot \mathbf{J}_p + (a_j\ x_0 + b_j\ y_0 + c_j)\,\xi_j = 0 \qquad (19)$$

In brief, each point correspondence provides the 2 constraints (15) and (16), and each line correspondence provides the 2 constraints (18) and (19). In matrix form these equations can be written as:

$$\underbrace{G}_{(2n+2m)\times 6} \begin{bmatrix} \mathbf{I}_p \\ \mathbf{J}_p \end{bmatrix} = \underbrace{\mathbf{z}_p}_{(2n+2m)\times 1} \qquad (20)$$

where G and $\mathbf{z}_p$ are a (2n+2m)x6 matrix and a (2n+2m) vector respectively:

$$G = \begin{bmatrix} \vdots & \vdots \\ \mathbf{P}_i^T & \mathbf{0}^T \\ \vdots & \vdots \\ \mathbf{0}^T & \mathbf{P}_i^T \\ \vdots & \vdots \\ a_j\ \mathbf{W}_j^T & b_j\ \mathbf{W}_j^T \\ \vdots & \vdots \\ a_j\ \mathbf{V}_j^T & b_j\ \mathbf{V}_j^T \\ \vdots & \vdots \end{bmatrix} \qquad \mathbf{z}_p = \begin{bmatrix} \vdots \\ (x_i - x_0)(1 + \epsilon_i) \\ \vdots \\ (y_i - y_0)(1 + \epsilon_i) \\ \vdots \\ -(a_j\ x_0 + b_j\ y_0 + c_j)\,(1 + \eta_j) \\ \vdots \\ -(a_j\ x_0 + b_j\ y_0 + c_j)\,\xi_j \\ \vdots \end{bmatrix}$$

3.2 Pose by successive approximations

One may notice if ϵ_i, η_j, and ξ_j are set to zero then (i) equation (20) 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 eqs. (11) and (12)). 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 (R and $\mathbf{t}$) 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:

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

Second, we derive the three orthogonal unit vectors $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$. From (17) we can write:

$$\mathbf{i} = t_z\ \mathbf{I}_p + x_0\ \mathbf{k} \tag{21}$$

$$\mathbf{j} = t_z\ \mathbf{J}_p + y_0\ \mathbf{k} \tag{22}$$

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

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

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

$$(I_{3\times 3} - t_z y_0\ \Omega(\mathbf{I}_p) + t_z x_0\ \Omega(\mathbf{J}_p))\ \mathbf{k} = t_z^2\ \mathbf{I}_p \times \mathbf{J}_p \tag{23}$$

This equation allows us to compute $\mathbf{k}$ since it has full rank. Therefore, one can easily determine $\mathbf{k}$ using eq. (23) and $\mathbf{i}$ and $\mathbf{j}$ using eqs. (21) and (22).

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

1. For all i and j, $i \in \{1...n\}$, $j \in \{1...m\}$, $(n+m) \geq 3$, $\epsilon_i = 0$, $\eta_j = 0$, $\xi_j = 0$.
2. Solve the overconstrained linear system (20) which provides an estimation of vectors $\mathbf{I}_p$ and $\mathbf{J}_p$:

$$\begin{bmatrix}\mathbf{I}_p \\ \mathbf{J}_p\end{bmatrix} = (G^T G)^{-1} G^T\ \mathbf{z}_p$$

3. Compute the pose parameters, i.e. the position (t_x, t_y, and t_z) and orientation ($\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$) as explained above;
4. For all i and j, compute:

$$\epsilon_i = \frac{\mathbf{k}\cdot\mathbf{P}_i}{t_z}, \quad \eta_j = \frac{\mathbf{k}\cdot\mathbf{W}_j}{t_z}, \quad \xi_j = \frac{\mathbf{k}\cdot\mathbf{V}_j}{t_z}$$

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. $(G^T G)^{-1} G^T$) can be computed once for all and hence it can be computed independently of the loop presented above. Therefore, the estimation of $\mathbf{I}_p$ and $\mathbf{J}_p$ is particularly efficient.

4 Experiments

Figure 2 shows an example of convergence of the paraperspective algorithm when it is applied to compute the pose of a cube. The first iteration of the algorithm found a paraperspective pose (left). After only six iterations the algorithm correctly determined the pose of the cube (right). This computation takes 6 iterations (3.1 ms on an Ultra-Sparc). Figure 3 illustrates the pose estimation of a cube (its size is 7 cm) and a gripper by the paraperspective algorithm. The gripper is identified by 5 vertices and 5 edges, the cube is identified by 6 vertices and 7 edges. By combining the 2 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 : $(20cm, 1.9cm, -5.9cm)^T$. The origins of the 2 coordinate systems are shown by big crosses (see Figure 3 (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 [4]. Table 1 gives the residual errors in the image plane between the true features and the projected 3D model associated with the 2 computed poses (gripper and cube).

5 Conclusion

In this paper we focused on the problem of pose computation from 2D to 3D point and line correspondences. We propose a fast method which establishes a link between paraperspective and perspective. The resulting method is very elegant, very fast, and quite accurate. It can be included in real-time vision and robotics applications. The iterative paraperspective method has better convergence properties than the iterative weak perspective method.

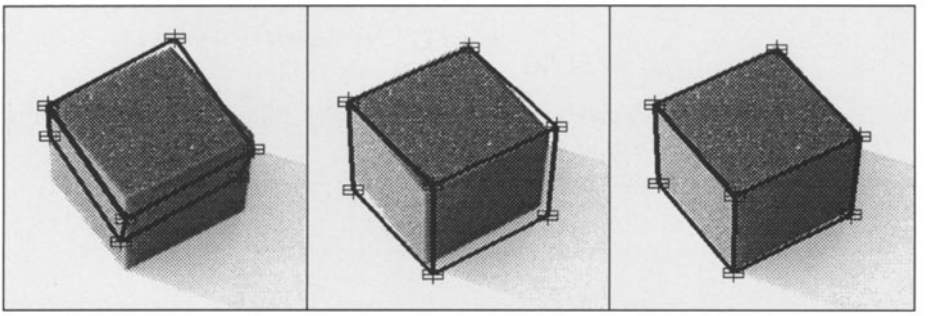
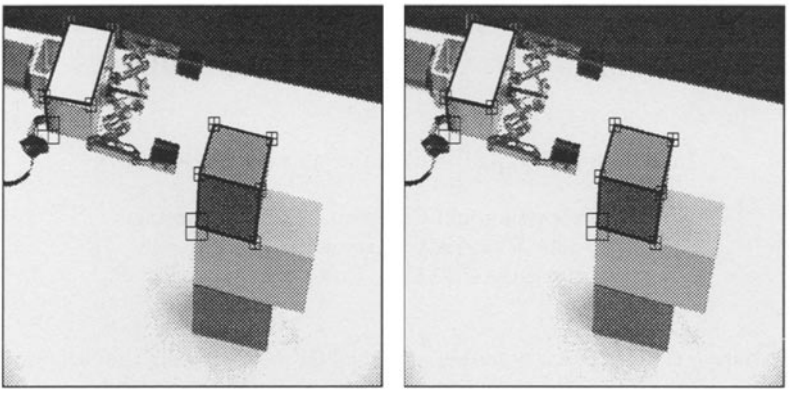


Fig. 2. An example of applying the iterative paraperspective algorithm to a cube using 7 vertices and 6 edges (peripheral). This computation takes 6 iterations (3.1 ms on an Ultra-Sparc).



first iteration third iteration

Fig. 3. An example of applying the paraperspective algorithm to both a gripper and a cube. The 2 obtained poses allow one to compute the relative position and orientation between them.

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)	2.1	2.5

Table 1. Pose estimation of both the gripper and the cube using the paraperspective algorithm, the computer being used is an Ultra-Sparc.

References

1. D. DeMenthon and L. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, June 1995.
2. M. Dhome, M. Richetin, J.T. Lapreste, and G. Rives. Determinition of the attitude of 3d objects from a single perspective view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, 1989.
3. B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.
4. N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. In *Proceedings of the Fourth British Machine vision Conference (BMVC 93)*, 1993.
5. 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.
6. Y. Ohta, K. Maenobu, and T. Sakai. Obtaining surface orientation from texels under perspective projection. In *Proceedings of the 7th IJCAI*, 1981.
7. J. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Transactions on Robotics and Automation*, 5(2):129–142, 1989.