Euclidean Reconstruction: from Paraperspective to Perspective [⋆]

Stéphane Christy and Radu Horaud

GRAVIR-IMAG & INRIA Rhône-Alpes
46, avenue Félix Viallet 38031 Grenoble FRANCE

ECCV'96, volume II, pages 129–140

Abstract. In this paper we describe a method to perform Euclidean reconstruction with a perspective camera model. It incrementally performs reconstruction with a paraperspective camera in order to converge towards a perspective model. With respect to other methods that compute shape and motion from a sequence of images with a calibrated perspective camera, this method converges in a few iterations, is computationally efficient, and does not suffer from the non linear nature of the problem. Moreover, the behaviour of the algorithm may be simply explained and analysed, which is an advantage over classical non linear optimization approaches. With respect to 3-D reconstruction using an approximated camera model, our method solves for the sign (reversal) ambiguity in a very simple way and provides much more accurate reconstruction results.

1 Introduction and background

The problem of computing 3-D shape and motion from a long sequence of images has received a lot of attention for the last few years. Previous approaches attempting to solve this problem fall into several categories, whether the camera is calibrated or not, and/or whether a projective or an affine model is being used. With a calibrated camera one may compute Euclidean shape up to a scale factor using either a perspective model [8], or a linear model [9], [10], [6]. With an uncalibrated camera the recovered shape is defined up to a projective transformation or up to an affine transformation [4]. One can therefore address the problem of either Euclidean, affine, or projective shape reconstruction. In this paper we are interested in Euclidean shape reconstruction with a calibrated camera. In that case, one may use either a perspective camera model or an affine approximation – orthographic projection, weak perspective, or paraperspective.

The perspective model has associated with it, in general, non linear reconstruction techniques. This naturally leads to non-linear minimization methods

[⋆] This work has been supported by "Société Aérospatiale" and by DRET.

which require some form of initialization [8], [4]. If the initial "guess" is too faraway from the true solution then the minimization process is either very slow or it converges to a wrong solution. Affine models lead, in general, to linear resolution methods [9], [10], [6], but the solution is defined only up to a sign (reversal) ambiguity, i.e., there are two possible solutions. Moreover, an affine solution is just an approximation of the true solution.

One way to combine the perspective and affine models could be to use the linear (affine) solution in order to initialize the non-linear minimization process associated with perspective. However, there are several drawbacks with such an approach. First, such a resolution technique does not take into account the simple link that exists between the perspective model and its linear approximations. Second, there is no mathematical evidence that a non-linear least-squares minimization method is "well" initialized by a solution that is obtained linearly. Third, there are two solutions associated with the affine model and it is not clear which one to choose.

The perspective projection can be modelled by a projective transformation from the 3-D projective space to the 2-D projective plane. Weak perspective and paraperspective are the most common affine approximations of perspective. Weak perspective may well be viewed as a zero-order approximation: $1/(1+\varepsilon) \approx 1$. Paraperspective is a first order approximation of full perspective: $1/(1+\varepsilon) \approx 1-\varepsilon$. Recently, in [3] a method has been proposed for determining the pose of a 3-D shape with respect to a single view by iteratively improving the pose computed with a weak perspective camera model to converge, at the limit, to a pose estimation using a perspective camera model. At our knowledge, the method cited above, i.e., [3] is among one of the first computational paradigms that link linear techniques (associated with affine camera models) with a perspective model. In [5] an extension of this paradigm to paraperspective is proposed. The authors show that the *iterative paraperspective* pose algorithm has better convergence properties than the *iterative weak perspective* one.

In this paper we describe a new Euclidean reconstruction method that makes use of affine reconstruction in an iterative manner such that this iterative process converges, at the limit, to a set of 3-D Euclidean shape and motion parameters that are consistent with a perspective model. The novelty of the method that we propose is twofold: (i) it extends the iterative pose determination algorithms described in [3] and in [5] to deal with the problem of shape and motion from multiple views and (ii) it is a generalization to perspective of the factorization methods [9], [6] and of the affine-invariant methods [10]. More precisely, the *affine-iterative reconstruction* method that we propose here has a number of interesting features:

- It solves the sign (or reversal) ambiguity that is inherent with affine reconstruction;
- It is fast because it converges in a few iterations (3 to 5 iterations), each iteration involving simple linear algebra computations;

– We show that the quality of the Euclidean reconstruction obtained with our method is only weakly influenced by camera calibration errors;
– It allows the use of either weak perspective [1] or paraperspective camera models (paraperspective in this paper) which are used iteratively, and
– It can be combined with almost any affine shape and motion algorithm. In particular we show how our method can be combined with the factorization method [9], [6].

2 Camera models

Let us consider a pin hole camera model. We denote by P_i a 3-D point with Euclidean 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 may well be the object point P_0. An object point P_i projects onto the image in p_i with image coordinates u_i and v_i and we have ($\boldsymbol{P}_i$ is the vector $\overrightarrow{P_0P_i}$ from point P_0 to point P_i):

$$\begin{pmatrix} su_i \\ sv_i \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{i}^T & t_x \\ \boldsymbol{j}^T & t_y \\ \boldsymbol{k}^T & t_z \\ \boldsymbol{0} & 1 \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

The first matrix describes the projective transformation between the 3-D camera frame and the image plane. The second matrix describes the rigid transformation (rotation and translation) between the object frame and the camera frame.

From now on we will be assuming that the intrinsic camera parameters are known and therefore we can compute camera coordinates from image coordinates: $x_i = (u_i - u_c)/\alpha_u$ and $y_i = (v_i - v_c)/\alpha_v$

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.

The relationship between object points and camera points can be written as:

$$x_i = (\boldsymbol{i} \cdot \boldsymbol{P}_i + t_x)/(\boldsymbol{k} \cdot \boldsymbol{P}_i + t_z) \tag{1}$$
$$y_i = (\boldsymbol{j} \cdot \boldsymbol{P}_i + t_y)/(\boldsymbol{k} \cdot \boldsymbol{P}_i + t_z) \tag{2}$$

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

– $\boldsymbol{I} = \boldsymbol{i}/t_z$ is the first row of the rotation matrix scaled by the z-component of the translation vector;
– $\boldsymbol{J} = \boldsymbol{j}/t_z$ 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
– We denote by ε_i the following ratio:

$$\varepsilon_i = \boldsymbol{k} \cdot \boldsymbol{P}_i / t_z \tag{3}$$

We may now rewrite the perspective equations as:

$$x_i = (\boldsymbol{I} \cdot \boldsymbol{P}_i + x_0)/(1 + \varepsilon_i) \tag{4}$$
$$y_i = (\boldsymbol{J} \cdot \boldsymbol{P}_i + y_0)/(1 + \varepsilon_i) \tag{5}$$

Whenever the object is at some distance from the camera, the ε_i are small compared to 1. We may therefore introduce the paraperspective model as a first order approximation of the perspective equations, Figure 1. Indeed, with the approximation: $1/(1+\varepsilon_i) \approx 1 - \varepsilon_i \ \forall i,\ i \in \{1...n\}$ we obtain x_i^p and y_i^p which are the coordinates of the paraperspective projection of P_i:

$$x_i \approx (\boldsymbol{I} \cdot \boldsymbol{P}_i + x_0)(1 - \varepsilon_i) \approx \boldsymbol{I} \cdot \boldsymbol{P}_i + x_0 - x_0 \varepsilon_i = \frac{\boldsymbol{i} \cdot \boldsymbol{P}_i}{t_z} + x_0 - x_0 \frac{\boldsymbol{k} \cdot \boldsymbol{P}_i}{t_z} = x_i^p \tag{6}$$

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

Finally, the paraperspective equations are:

$$x_i^p - x_0 = \frac{\boldsymbol{i} - x_0\, \boldsymbol{k}}{t_z} \cdot \boldsymbol{P}_i \tag{7}$$
$$y_i^p - y_0 = \frac{\boldsymbol{j} - y_0\, \boldsymbol{k}}{t_z} \cdot \boldsymbol{P}_i \tag{8}$$

3 Reconstruction with a perspective camera

Let us consider again the perspective equations (4) and (5). These equations may be also written as (there is a similar expression for y_i):

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

These equations can be written more compactly as:

$$(x_i - x_0)(1 + \varepsilon_i) = \boldsymbol{I}_p \cdot \boldsymbol{P}_i \tag{9}$$
$$(y_i - y_0)(1 + \varepsilon_i) = \boldsymbol{J}_p \cdot \boldsymbol{P}_i \tag{10}$$

with: $\boldsymbol{I}_p = \frac{\boldsymbol{i} - x_0\, \boldsymbol{k}}{t_z}$ and $\boldsymbol{J}_p = \frac{\boldsymbol{j} - y_0\, \boldsymbol{k}}{t_z}$ Equations (9) and (10) can be interpreted in two different ways: (i) we can consider x_i and y_i as the perspective projection of P_i or (ii) we can consider $x_i(1+\varepsilon_i) - x_0\varepsilon_i$ and $y_i(1+\varepsilon_i) - y_0\varepsilon_i$ as the paraperspective projection of P_i.

The basic idea of our method is to estimate values for ε_i *incrementally* such that one can compute the paraperspective projections of the 3-D points from the perspective projections which are the *true image measurements*. Therefore, the perspective reconstruction problem is reduced to the problem of iterative paraperspective reconstruction.

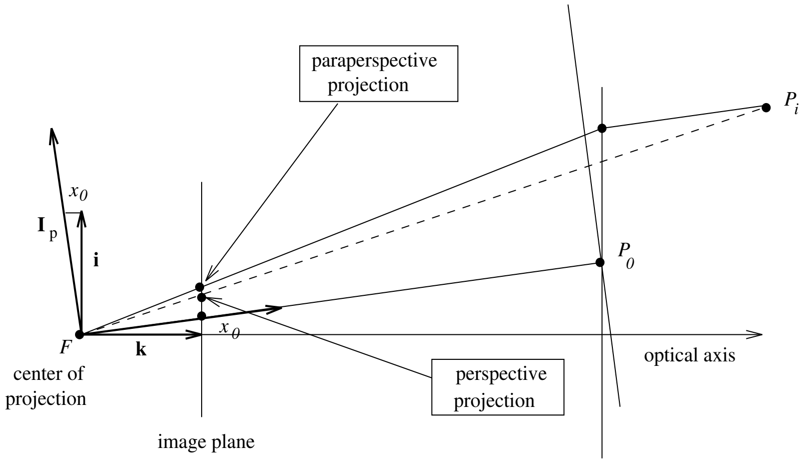

Fig. 1. This figure shows the principle of projection with a paraperspective camera model. We consider the plane through P_0 parallel to the image plane. A 3-D point P_i first projects onto this plane along the direction of FP_0 and then is projected onto the image along a line passing through F. Notice that the two vectors $\boldsymbol{I}_p$ and $\boldsymbol{J}_p$ are orthogonal to the direction of projection FP_0 ($\boldsymbol{I}_p$ only is depicted here).

Let us consider now k views of the same scene points. We assume that image-to-image correspondences have already been established. Equations (9) and (10) can be written as:

$$\boldsymbol{s}_{ij} = A_j \boldsymbol{P}_i \tag{11}$$

In this formula the subscript i stands for the i^{th} point and the subscript j for the j^{th} image. The 2-vector $\boldsymbol{s}_{ij}$ is equal to:

$$\boldsymbol{s}_{ij} = \begin{pmatrix} (x_{ij} - x_{0j})(1+\varepsilon_{ij}) \\ (y_{ij} - y_{0j})(1+\varepsilon_{ij}) \end{pmatrix} \tag{12}$$

In these equations ε_{ij} (see eq. (3)) is defined for each point and for each image:

$$\varepsilon_{ij} = \boldsymbol{k}_j \cdot \boldsymbol{P}_i / t_{z_j} \tag{13}$$

The reconstruction problem is now the problem of solving simultaneously $2 \times n \times k$ equations of the form of eq. (11). We introduce a method that solves these equations by affine iterations. More precisely, this method can be summarized by the following algorithm:

1. $\forall i,\ i \in \{1...n\}$ and $\forall j,\ j \in \{1...k\}$ set: $\varepsilon_{ij} = 0$ (initialisation);
2. Update the values of $\boldsymbol{s}_{ij}$ according with eq. (12) and using the newly computed values for ε_{ij};
3. Perform an Euclidean reconstruction with a paraperspective camera;
4. $\forall i,\ i \in \{1...n\}$ and $\forall j,\ j \in \{1...k\}$ estimate new values for ε_{ij} according with eq.(13);
5. Check the values of ε_{ij}:
   if $\forall(i, j)$ the values of ε_{ij} just estimated at this iteration are identical with the values estimated at the previous iteration, *then* stop;
   else go to step 2.

The most important step of this algorithm is step 4: estimate new values for ε_{ij}. This computation can be made explicit if one considers into some more detail step 3 of the algorithm which can be further decomposed into: (i) Affine reconstruction and (ii) Euclidean reconstruction.

The problem of affine reconstruction is the problem of determining both A_j and $\boldsymbol{P}_i$, for all j and for all i. It is well known that affine reconstruction determines shape and motion up to a 3-D affine transformation. Indeed, for any 3×3 invertible matrix T we have: $A_j \boldsymbol{P}_i = A_j T T^{-1} \boldsymbol{P}_i$. In order to convert affine shape and motion into Euclidean shape and motion one needs to consider some Euclidean constraints associated either with the motion of the camera or with the shape being viewed by the camera. Since we deal here with a calibrated camera, we may well use rigid motion constraints in conjunction with paraperspective [6]. See [7] for the case of an uncalibrated affine camera. Therefore, step 3 of the algorithm provides both Euclidean shape $(\boldsymbol{P}_1 ... \boldsymbol{P}_n)$ and Euclidean motion. Based on the parameters of the Euclidean shape and motion thus computed one can estimate ε_{ij} for all i and for all j using eq. (13) – step 4.

The above algorithm can be easily interpreted as follows. The first iteration of the algorithm performs a 3-D reconstruction using the initial image measurements and a paraperspective camera model. This first reconstruction allows an estimation of values for the ε_{ij}'s which in turn allow the image vectors $\boldsymbol{s}_{ij}$ to be *modified* (step 2 of the algorithm). The $\boldsymbol{s}_{ij}$'s are modified according to eq. (12) such that they better fit the approximated camera model being used.

The next iterations of the algorithm perform a 3-D reconstruction using (i) image vectors that are incrementally modified and (ii) a paraperspective camera model.

At convergence, the equations (11) are equivalent with the perspective equations (9), (10). In other terms, this algorithm solves for Euclidean reconstruction with a perspective camera by iterations of Euclidean reconstruction with a paraperspective camera. Therefore, before we proceed further in order to understand some important features of this iterative algorithm, it is necessary to have insights into the problem of Euclidean reconstruction with a paraperspective camera.

The iterative algorithm outlined in this paper is best illustrated on Figure 2. At the first iteration, the algorithm considers the *true* perspective projections of P_i and attempts to reconstruct the 3-D points as if they were projected in the image using paraperspective. At the second iteration the algorithm considers modified image point positions. At the last iteration, the image point positions were modified such that they fit the paraperspective projections.

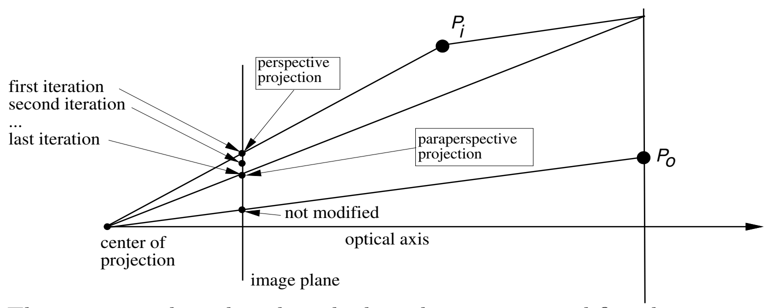

Fig. 2. The iterative algorithm described in this section modifies the projection of a 3-D point from true perspective to paraperspective (see text).

4 Reconstruction with a paraperspective camera

In this section we develop step 3 of the algorithm outlined in the previous section. Methods that use a linear camera model provide a 3-D affine reconstruction if at least 2 views of 4 non-coplanar points are available and if the motion is not a pure translation. However, 3 views are necessary in order to convert this affine reconstruction into an Euclidean one. While the affine-invariant method allows a more direct analysis of the problem, [10] the factorization method is more convenient from a practical point of view.

The factorization method, [9] computes shape and motion simultaneously by performing a singular value decomposition of the $2k \times n$ matrix σ which is formed by concatenating eq. (11) for all i and j: $\sigma = AS = O_1 \Sigma O_2$. Affine shape and motion, i.e., the $2k \times 3$ matrix A and the $3 \times n$ matrix S, can be computed only if the rank of the *measurement* matrix σ is equal to 3.

The rank of σ is equal to the rank of the $n \times n$ diagonal matrix Σ. Even if the rank condition stated above is satisfied, the rank of Σ may be greater than 3 because of numerical instability due to noise. Tomasi & Kanade [9] suggested to solve the rank problem by truncating the matrix Σ such that only the 3 largest diagonal values are considered. They claim that this truncation amounts to removing noise present in the measurement matrix. Therefore, one can write the singular value decomposition of the measurement matrix as $\sigma = O_1' \Sigma' O_2' + O_1'' \Sigma'' O_2''$ where Σ' is a 3×3 diagonal matrix containing the 3 largest diagonal terms of Σ.

Finally, affine shape and affine motion are given by $S = (\Sigma')^{1/2} O_2'$ and $A = O_1' (\Sigma')^{1/2}$.

4.1 From affine to Euclidean

Obviously, the factorization method described above does not provide a unique decomposition of the measurement matrix σ. The method that we describe here for recovering Euclidean shape and motion with a paraperspective camera is an alternative approach to the method described in [6].

One has to determine now Euclidean shape and motion by combining the affine reconstruction method just described and the Euclidean constraints available with the camera model being used. As already mentioned, one has to determine a 3×3 invertible matrix T such that the affine shape S becomes Euclidean: $\left(\boldsymbol{P}_1 \ldots \boldsymbol{P}_n \right) = T^{-1} \left(\boldsymbol{S}_1 \ldots \boldsymbol{S}_n \right)$ and the affine motion becomes rigid: $\left(R_1 \ldots R_k \right)^T = \left(A_1 \ldots A_k \right)^T T$. Indeed, in order to avoid confusion we denote by S and A affine shape and affine motion and by P and R Euclidean shape and rigid motion. The matrices R_j are given by:

$$R_j = \begin{pmatrix} \boldsymbol{I}_{p_j} \\ \boldsymbol{J}_{p_j} \end{pmatrix}$$

The Euclidean constraints allowing the computation of T are the following [6]:

$$\|\boldsymbol{I}_{p_j}\|^2/(1 + x_{0_j}^2) = \|\boldsymbol{J}_{p_j}\|^2/(1 + y_{0_j}^2)$$

and

$$\boldsymbol{I}_{p_j} \cdot \boldsymbol{J}_{p_j} = x_{0_j} y_{0_j}/2 \left(\|\boldsymbol{I}_{p_j}\|^2/(1 + x_{0_j}^2) + \|\boldsymbol{J}_{p_j}\|^2/(1 + y_{0_j}^2) \right)$$

We denote by $\boldsymbol{a}_j$ and $\boldsymbol{b}_j$ the row vectors of matrix A_j. Using the constraints above, for k images one obtains $2k$ constraints for the matrix T:

$$\boldsymbol{a}_j^T T T^T \boldsymbol{a}_j/(1 + x_{0_j}^2) - \boldsymbol{b}_j^T T T^T \boldsymbol{b}_j/(1 + y_{0_j}^2) = 0 \tag{14}$$

$$\boldsymbol{a}_j^T T T^T \boldsymbol{b}_j = x_{0_j} y_{0_j}/2 \left(\boldsymbol{a}_j^T T T^T \boldsymbol{a}_j/(1 + x_{0_j}^2) + \boldsymbol{b}_j^T T T^T \boldsymbol{b}_j/(1 + y_{0_j}^2) \right) \tag{15}$$

These constraints are homogeneous and non linear in the coefficients of T. In order to avoid the trivial null solution the scale factor must be fixed in advance. For example, one may choose $\|\boldsymbol{I}_{p_1}\|^2 = 1 + x_{0_1}^2$ or $\|\boldsymbol{J}_{p_1}\|^2 = 1 + y_{0_1}^2$. Hence we obtain one additional constraint such as:

$$\boldsymbol{a}_1^T T T^T \boldsymbol{a}_1 = 1 + x_{0_1}^2 \tag{16}$$

These constraints are non linear in the coefficients of T. With the substitution $Q = TT^T$ equations (14), (15), and (16) become linear and there are 6 unknowns because, by definition, Q is a 3×3 symmetric positive matrix. Since we have $2k+1$ independent equations and 6 unknowns, at least 3 views are necessary to estimate Q. Finally T can be derived from Q using a factorization of Q. As it will be explained later in section 5 there is an ambiguity associated with the factorization of the symmetric semi-definite positive matrix Q and this ambiguity is the origin of the reversal ambiguity associated with any affine camera model.

Next we determine the parameters of the Euclidean motion by taking explicitly into account the paraperspective camera model. The method presented below is an alternative to the method proposed in [6] and it is equivalent to the problem of computing pose with a paraperspective camera [5].

First we determine the translation vector. From the formulae above we have:

$$t_{z_j} = 1/2 \left((\sqrt{1 + x_{0_j}^2})/(\|\boldsymbol{I}_{p_j}\|) + (\sqrt{1 + y_{0_j}^2})/(\|\boldsymbol{J}_{p_j}\|) \right)$$

and $t_{x_j} = x_{0_j} t_{z_j}$, $t_{y_j} = y_{0_j} t_{z_j}$.

Second, we derive the three orthogonal unit vectors $\boldsymbol{i}_j$, $\boldsymbol{j}_j$, and $\boldsymbol{k}_j$ as follows. $\boldsymbol{I}_p$ and $\boldsymbol{J}_p$ may be written as:

$$\boldsymbol{i}_j = t_{z_j}\, \boldsymbol{I}_{p_j} + x_{0_j}\, \boldsymbol{k}_j \tag{17}$$

$$\boldsymbol{j}_j = t_{z_j}\, \boldsymbol{J}_{p_j} + y_{0_j}\, \boldsymbol{k}_j \tag{18}$$

The third vector, $\boldsymbol{k}_j$ is the cross-product of these two vectors $\boldsymbol{k}_j = \boldsymbol{i}_j \times \boldsymbol{j}_j$. Let's, for convenience, drop the subscript j. We obtain for $\boldsymbol{k}$:

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

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

$$\underbrace{(I_{3\times3} - t_z y_0\, S(\boldsymbol{I}_p) + t_z x_0\, S(\boldsymbol{J}_p))}_{B}\, \boldsymbol{k} = t_z^2\, \boldsymbol{I}_p \times \boldsymbol{J}_p \tag{19}$$

This equation allows us to compute $\boldsymbol{k}$, provided that the linear system above has full rank. Indeed, one may notice that the 3×3 matrix B is of the form:

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

Its determinant is strictly positive and therefore, B has full rank and one can easily determine $\boldsymbol{k}_j$ using eq. (19) and $\boldsymbol{i}_j$ and $\boldsymbol{j}_j$ using eqs. (17) and (18). As a consequence, it is possible to compute the rigid motion between each camera position and the 3-D scene, i.e., $\boldsymbol{i}_j$, $\boldsymbol{j}_j$, $\boldsymbol{k}_j$, t_{x_j}, t_{y_j}, and t_{z_j} and to estimate ε_{ij} for each image and for each point (eq. (13)).

5 Solving the reversal ambiguity

The algorithm outlined in Section 3 solves for Euclidean reconstruction with a perspective camera by iterations of an Euclidean reconstruction method with a paraperspective camera. In this section we show how this iterative algorithm has to be modified in order to solve the reversal ambiguity problem which is inherent with any affine camera model. Indeed, let us consider again the affine shape and motion recovery method outlined in the previous section. A key step of this method consists of computing a transformation T that converts affine structure into Euclidean structure. This transformation must be computed by decomposition of a symmetric semi-definite positive matrix Q: $Q = TT^T$. There are at least two ways to determine T:

1. Q can be written as $Q = ODO^T$, where O is an orthogonal matrix containing the eigenvectors of Q and D is a diagonal matrix containing the eigenvalues of Q. Since the eigenvalues of a symmetric semi-definite positive matrix are all real and positive, one may write Q as: $Q = (OD^{1/2})(OD^{1/2})^T = KK^T$.
2. Alternatively one may use the Cholesky decomposition of Q: $Q = LL^T$ where L is a lower triangular matrix.

Let H be a non singular matrix such that $L = KH$ and we have:

$$Q = LL^T = KHH^TK^T = KK^T \tag{20}$$

We conclude that H is necessarily an orthogonal matrix. The orthogonality of H is also claimed in [11] but without any formal proof. Therefore H represents either a rotation or a mirror transformation (its determinant is either $+1$ or -1) and there are two classes of shapes that are possible:

– a *direct* shape which is defined up to a rotation and
– a *reverse* shape which is obtained from the direct shape by applying a mirror transformation.

Since shape is defined up to rotation and without loss of generality we choose the mirror transformation to be $-I$ where I is the identity matrix. Therefore the affine shape and motion equation can be written as: $\sigma = AS = (-A)(-S)$. Because of this reversal ambiguity, there are two solutions for the ε_{ij}'s at each iteration of the reconstruction algorithm described above.

The vectors $\boldsymbol{k}_j$ are computed using eq. (19). This equation may use either $\boldsymbol{I}_p$ and $\boldsymbol{J}_p$ (the first solution) or $-\boldsymbol{I}_p$ and $-\boldsymbol{J}_p$ (the second solution). Therefore we obtain two distinct solutions, that is, $\boldsymbol{k}_j^1$ and $\boldsymbol{k}_j^2$. The two solutions for ε_{ij} correspond to $\boldsymbol{k}_j^1$ and $\boldsymbol{P}_i$ and to $\boldsymbol{k}_j^2$ and $-\boldsymbol{P}_i$: $\varepsilon_{ij}^{1,2} = \pm\boldsymbol{k}_j^{1,2} \cdot \boldsymbol{P}_i / t_{z_j}$.

At each iteration of the perspective reconstruction algorithm two values for ε_{ij} are thus estimated. Therefore, after N iterations there will be 2^N possible

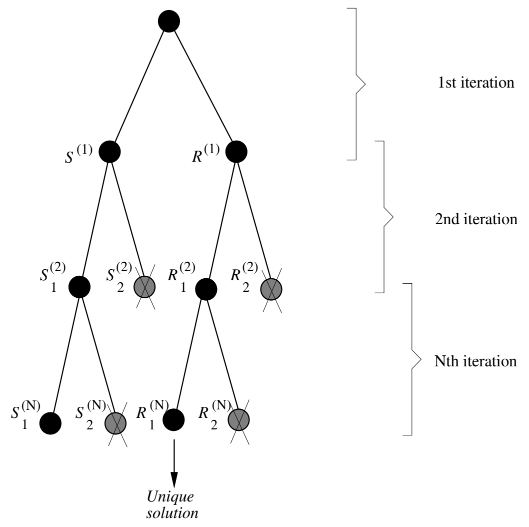

Fig. 3. A strategy for selecting a unique solution (see text).

solutions. All these solutions are not, however, necessarily consistent with the image data and a simple verification technique allows to check this consistency and to avoid the explosion of the number of solutions. Finally, a unique solution is obtained.

The first iteration of the algorithm makes available two solutions – a "positive" shape S and a "negative" shape $(-S)$ – that are both considered. At the next iterations of the algorithm two shapes are maintained: one shape consistent with S and another shape consistent with $-S$. Therefore, at convergence, one obtains two solutions, each one of these solutions being consistent with one or the other of the initial shapes. Finally, the solution that best fits the image data is selected as the unique solution. This solution selection process is best illustrated on Figure 3.

6 Experimental results and discussion

In this section we describe two types of experiments: (i) experiments with synthetic data which allow us to study both the accuracy of the 3-D reconstruction and the behaviour of the iterative algorithm, and (ii) experiments with real data.

Let us consider some synthetic data. We designate by D the distance between the center of these data and the camera center of projection divided by the size of the data's diameter – D is therefore a relative distance. Hence, D is approximatively equal to $1/\overline{\varepsilon_{ij}}$ where $\overline{\varepsilon_{ij}}$ is the average value of ε_{ij} for all i and j. For a fixed value of D we consider 10 camera motions, each motion being composed of 15 images. Each such motion is farther characterized by a translation vector and a rotation axis and angle. The directions of the translation vector and rotation axis are randomly chosen. The angle of rotation between two images is equal to 2^0. Moreover, the image data obtained by projecting this object onto the image plane is perturbed by adding Gaussian noise with a standard deviation equal to 1.

The accuracy of the reconstruction is measured by the difference between the theoretical 3-D points and the reconstructed 3-D points. We compute the mean and the maximum values of these differences over all motions at a fixed relative distance D. Figure 4 summarizes the results.

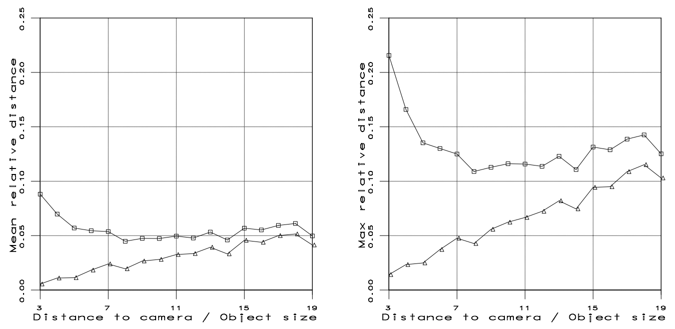

Fig. 4. The behaviour of the factorization method (small squares) is compared with the behaviour of the iterative method described in this paper (small triangles) as a function of the relative distance between the object and the camera (see text). The left side shows the mean value of the distance between object points and reconstructed points and the right side shows the maximum value of this distance.

Finally, we consider one experiment performed with real images: A sequence of 13 images of a wood piece with 10 tracked points (figure 5); In all these experiments the camera center was fixed to $u_c = v_c = 256$ and the horizontal and vertical scale factors were fixed to $\alpha_u = 1500$ and $\alpha_v = 1000$.

In this paper we described a method for solving the Euclidean reconstruction problem with a perspective camera by incrementally performing Euclidean reconstruction with a paraperspective camera model. The method converges, on an average, in 5 iterations, is computationally efficient, and it produces accurate results even in the presence of image noise and/or camera calibration errors. The method may well be viewed as a generalization to perspective of shape and motion computation using factorization and/or affine-invariant methods. It is well known that with a linear camera model, shape and motion can be recovered only up to a sign (reversal) ambiguity. The method that we propose in this paper solves for this ambiguity and produces a unique solution even if the camera is at some distance from the scene.

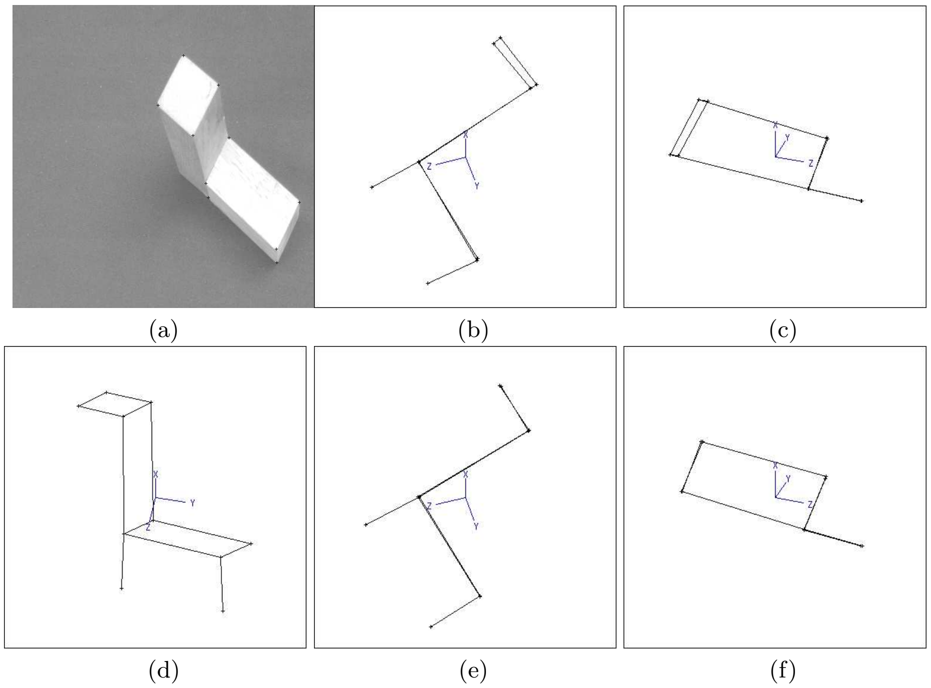

Fig. 5. This figure shows one image (a) out of a sequence of 13 images where only 10 points were tracked and reconstructed. The first row (b) and (c) shows the result of reconstruction using the factorization method with a paraperspective model, while the second row (d), (e), and (f) shows the result of reconstruction with the iterative method and a perspective model. In this example the iterative algorithm converged in 4 iterations.

Although the experimental results show that there are little convergence problems, we have been unable to study the convergence of the algorithm from a theoretical point of view. We studied its convergence based on some numerical and practical considerations which allow one to determine in advance the optimal experimental setup under which convergence can be guaranteed [2]. In the future we plan to study more thoroughly the convergence of this type of algorithms.

References

1. S. Christy and R. Horaud. A quasi linear reconstruction method from multiple perspective views. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 374–380, Pittsburgh, Pennsylvania, USA, August 1995. IEEE Computer Society Press, Los Alamitos, Ca.
2. S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104, November 1996.

3. 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.
4. R. I. Hartley. Euclidean reconstruction from uncalibrated views. In Mundy Zisserman Forsyth, editor, *Applications of Invariance in Computer Vision*, pages 237–256. Springer Verlag, Berlin Heidelberg, 1994.
5. R. Horaud, S. Christy, F. Dornaika, and B. Lamiroy. Object pose: Links between paraperspective and perspective. In *Proceedings Fifth International Conference on Computer Vision*, pages 426–433, Cambridge, Mass., June 1995. IEEE Computer Society Press, Los Alamitos, Ca.
6. 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*, volume 2, pages 97–108. Springer Verlag, Stockholm, Sweden, May 1994.
7. L. Quan and R. Mohr. Self-calibration of an affine camera from multiple views. In *6th International Conference CAIP'95, Computer Analysis of Images and Patterns*, pages 448–455, Prague, September 1995.
8. R. Szelinski and S. B. Kang. Recovering 3-D shape and motion from image streams using non-linear least squares. Technical Report CRL 93/3, Digital – Cambridge Research Laboratory, March 1993.
9. C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.
10. D. Weinshall. Model-based invariants for 3-d vision. *International Journal of Computer Vision*, 10(1):27–42, February 1993.
11. D. Weinshall and C. Tomasi. Linear and incremental acquisition of invariant shape models from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):512–517, May 1995.