Euclidean Shape and Motion from Multiple Perspective Views by Affine Iterations

Stéphane Christy and Radu Horaud

Abstract—In this paper, we describe a method for solving the Euclidean reconstruction problem with a perspective camera model by incrementally performing Euclidean reconstruction with either a weak or a paraperspective camera model. With respect to other methods that compute shape and motion from a sequence of images with a calibrated camera, this method converges in a few iterations, is computationally efficient, and solves for the sign (reversal) ambiguity. We give a detailed account of the method, analyze its convergence, and test it with both synthetic and real data.

Index Terms—Perspective, weak perspective, paraperspective, Euclidean and affine reconstruction.

1 Introduction

THE problem of computing 3D 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 projective model or an affine model [1], [2]. With an uncalibrated camera the recovered shape is defined up to a projective transformation or up to an affine transformation [3], [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 one of its affine approximations—orthographic projection, weak perspective, or paraperspective.

The perspective model has associated with it, in general, non linear reconstruction techniques. This naturally leads to nonlinear minimization methods which require some form of initialization [5], [3]. 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 camera models lead, in general, to linear resolution methods [1], [2], but the solution is defined only up to a sign (reversal) ambiguity and, both these two solutions are just an approximation of the true solution.

The perspective projection can be modeled by a projective transformation from the 3D projective space to the 2D projective plane. Weak perspective and paraperspective are the most common affine approximations of perspective. Recently, in [6] a method has been proposed for determining the pose of a 3D shape with respect to a single view by iteratively improving the pose computed with a weak perspective camera model to converge, to the limit, to a pose estimation using a perspective camera model. To our knowledge, the method cited above, i.e., [6] is among one of the first computational paradigms that link linear techniques (associated with affine camera models) with a perspective model. In [7], 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, to the limit, to a set of 3D Euclidean shape and motion parameters that are consistent with a perspective model. The novelty of the method that we propose is twofold:

1) it extends the iterative pose determination algorithms described in [6] and in [7] to deal with the problem of shape and motion from multiple views and
2) it is a generalization to perspective of the factorization methods [1], [2] and of the affine-invariant methods [8].

More precisely, the *affine-iterative reconstruction* methods that we propose here have a number of interesting features:

- They solve the sign (or reversal) ambiguity that is inherent with affine reconstruction;
- They are fast because they converge in a few iterations (typically three to five iterations), each iteration involving singular value decomposition of a measurement matrix. In particular there is no matrix inversion being involved as is the case with any iterative non-linear optimization technique;
- They can be combined with almost any affine shape and motion algorithm. In particular we show how our method can be combined with factorization methods [1], [2].

1.1 Paper Organization

The remainder of this paper is organized as follows. Section 2 describes the relationship between full perspective, paraperspective, and weak perspective. Section 3 describes how to perform reconstruction with a perspective camera model by iterating either a weak perspective reconstruction or a paraperspective reconstruction algorithm. Section 4 outlines the well-known factorization algorithm. Section 5 describes how to solve for the reversal ambiguity. Section 6 outlines some useful features associated with the developed method. Finally, Section 7 provides a practical evaluation of the method using simulated data with various camera motions and Section 8 describes results obtained with real imagery.

2 Camera Models

Let us consider a pin hole camera model. We denote by P_i a 3D point lying onto a 3D object 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 ($\mathbf{P}_i$ is the vector $\overrightarrow{P_0P_i}$ from point P_0 to point P_i):

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

The first 3×4 matrix describes the affine transformation between the camera coordinates and the images coordinates combined with the perspective projection. The second 4×4 matrix describes the rigid transformation (rotation and translation) between the object frame and the camera frame: $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the three row vectors associated with the rotation matrix.

From now on we will be assuming that the intrinsic camera parameters are known and therefore we consider the relationship

• *The authors are with GRAVIR-IMAG and INRIA Rhône-Alpes, 655 avenue de l'Europe, 38330 Monbonnet, Saint -Martin, France. E-mail: Radu.Horaud@inriaalpes.fr.*

Manuscript received Jan. 30, 1995; revised Aug. 13, 1996. Recommended for acceptance by L. Shapiro.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96088.

between image points expressed in camera coordinates and in image coordinates: $u_i = \alpha_u x_i + u_c$ and $v_i = \alpha_v y_i + v_c$. 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. It will be shown that the quality of the reconstruction method described below depends strongly only on the ratio α_u/α_v and that the reconstruction obtained with our method is not sensitive to errors in u_c and v_c. The relationship between object points and image points expressed in camera coordinates can be written as:

$$x_i = \frac{\mathbf{i}\cdot\mathbf{P}_i + t_x}{\mathbf{k}\cdot\mathbf{P}_i + t_z} = \frac{\mathbf{I}\cdot\mathbf{P}_i + x_0}{1+\varepsilon_i} \tag{2}$$

$$y_i = \frac{\mathbf{j}\cdot\mathbf{P}_i + t_y}{\mathbf{k}\cdot\mathbf{P}_i + t_z} = \frac{\mathbf{J}\cdot\mathbf{P}_i + y_0}{1+\varepsilon_i} \tag{3}$$

We have introduced the following notations: $\mathbf{I} = \mathbf{i}/t_z$, $\mathbf{J} = \mathbf{j}/t_z$, $x_0 = t_x/t_z$ and $y_0 = t_y/t_z$ are the camera coordinates of p_0 which is the projection of P_0 (the origin of the object frame), and we denote by ε_i the following ratio:

$$\varepsilon_i = \frac{\mathbf{k}\cdot\mathbf{P}_i}{t_z} \tag{4}$$

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

Weak perspective assumes that the object points lie in a plane parallel to the image plane passing through the origin of the object frame, i.e., P_0. This is equivalent to a zero-order approximation: $\frac{1}{1+\varepsilon_i} \approx 1\ \forall i,\ i \in \{1 \ldots n\}$. With this approximation, (2) and (3) become:

$$x_i^w - x_0 = \mathbf{I}\cdot\mathbf{P}_i \tag{5}$$

$$y_i^w - y_0 = \mathbf{J}\cdot\mathbf{P}_i \tag{6}$$

In these two equations x_i^w and y_i^w are the camera coordinates of the weak perspective projection of the point P_i. By identification with (2) and (3) we obtain the relationship between the weak perspective and the perspective projections of P_i:

$$x_i^w = x_i(1+\varepsilon_i) \tag{7}$$

$$y_i^w = y_i(1+\varepsilon_i) \tag{8}$$

Paraperspective may be viewed as a first-order approximation of perspective. Indeed, with the approximation: $\frac{1}{1+\varepsilon_i} \approx 1-\varepsilon_i\ \forall i$, $i \in \{1 \ldots n\}$ and by neglecting the term in $1/t_z^2$ we obtain the paraperspective projection of P_i:

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

There is a similar expression for y_i^p. Finally, the paraperspective equations are:

$$x_i^p - x_0 = \frac{\mathbf{i} - x_0\mathbf{k}}{t_z}\cdot\mathbf{P}_i = \mathbf{I}_p\cdot\mathbf{P}_i \tag{9}$$

$$y_i^p - y_0 = \frac{\mathbf{j} - y_0\mathbf{k}}{t_z}\cdot\mathbf{P}_i = \mathbf{J}_p\cdot\mathbf{P}_i \tag{10}$$

Again, by identification with (2) and (3) we obtain the relationship between the paraperspective and the perspective projections of P_i:

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

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

3 Shape and Motion with a Perspective Camera

Let us consider again the perspective equations (2) and (3). These equations may also be written as:

$$x_i(1+\varepsilon_i) - x_0 = \mathbf{I}\cdot\mathbf{P}_i \tag{13}$$

$$y_i(1+\varepsilon_i) - y_0 = \mathbf{J}\cdot\mathbf{P}_i \tag{14}$$

Let us subtract the *paraperspective term* from both the left and right sides of (13) and (14). We obtain:

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

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

These equations can be written more compactly as:

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

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

To summarize, we have two different sets of equations that describe the *same* perspective camera model:

- The first set, i.e., (13) and (14) establish the link between perspective and weak perspective and
- The second set, i.e., (15) and (16) establish the link between perspective and paraperspective.

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

Let us consider now k views of the same scene points. We assume that image-to-image correspondences have already been established. Both (13) and (14) or (15) and (16) can be written as:

$$\underbrace{\mathbf{s}_{ij}}_{2\times1} = \underbrace{\mathbf{A}_j}_{2\times3}\underbrace{\mathbf{P}_i}_{3\times1} \tag{17}$$

In this formula the subscript i stands for the ith point and the subscript j for the jth image. The 2-vector $\mathbf{s}_{ij}$ is equal to (weak perspective):

$$\mathbf{s}_{ij} = \begin{pmatrix} x_{ij}(1+\varepsilon_{ij}) - x_{0j} \\ y_{ij}(1+\varepsilon_{ij}) - y_{0j} \end{pmatrix} \tag{18}$$

or to (paraperspective):

$$\mathbf{s}_{ij} = \begin{pmatrix} (x_{ij} - x_{0j})(1+\varepsilon_{ij}) \\ (y_{ij} - y_{0j})(1+\varepsilon_{ij}) \end{pmatrix} \tag{19}$$

In these equations ε_{ij}, i.e., (4) is defined for each point and for each image:

$$\varepsilon_{ij} = \frac{\mathbf{k}_j\cdot\mathbf{P}_i}{t_{z_j}} \tag{20}$$

The reconstruction problem is now the problem of simultaneously solving $2 \times n \times k$ equations of the form of (17). The unknowns of these equations are: The coordinates P_i, the matrices $\mathbf{A}_j$, and the *perspective corrections* ε_{ij}. We introduce a method that solves these equations by linear iterations. More precisely, this method can be summarized by the following algorithm:

Initialization $\forall i, i \in \{1 \ldots n\}$ and $\forall j, j \in \{1 \ldots k\}$ set: $\varepsilon_{ij} = 0$;
step 1 Update the values of $\mathbf{s}_{ij}$ according with (19);
step 2 Perform an Euclidean reconstruction with either a weak or a paraperspective camera;
step 3 $\forall i, i \in \{1 \ldots n\}$ and $\forall j, j \in \{1 \ldots k\}$ estimate new values for ε_{ij} according with (20);
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 1.

The keystone of this algorithm is **step 3**: estimate new values for ε_{ij}. This computation can be made explicit if one considers in some more detail **step 2** of the algorithm which can be further decomposed into:

1) affine reconstruction and
2) Euclidean reconstruction.

The problem of affine reconstruction is the problem of determining both $\mathbf{A}_j$ and $\mathbf{P}_i$, for all j and for all i, in (17), when some estimates of $\mathbf{s}_{ij}$ are provided. Such a reconstruction determines shape and motion up to a 3D affine transformation. Indeed, for any 3×3 invertible matrix $\mathbf{T}$ we have:

$$\mathbf{A}_j\mathbf{P}_i = \mathbf{A}_j\mathbf{T}\mathbf{T}^{-1}\mathbf{P}_i \tag{21}$$

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 weak or paraperspective [1], [2]. Therefore, **step 2** of the algorithm provides both Euclidean shape ($\mathbf{P}_1 \ldots \mathbf{P}_n$) and Euclidean motion (3D rotation and translation between each camera and a scene frame).

Various methods are available for estimating Euclidean structure from affine structure either with a calibrated camera [1], [8] (weak perspective), [2] (paraperspective), or with an uncalibrated camera [4]. 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 (20)—**step 3**.

The first iteration of the algorithm performs a 3D reconstruction using the initial image measurements and a weak or paraperspective camera model. This initial reconstruction allows the estimation of values for the ε_{ij}s which in turn allow the image vectors $\mathbf{s}_{ij}$ to be *modified* (**step 1** of the algorithm). The $\mathbf{s}_{ij}$s are modified according to (18) (for weak perspective) or according to (19) (for paraperspective) such that they better fit the approximated camera model being used.

The next iterations of the algorithm perform a 3D reconstruction using

1) image vectors that are incrementally modified and
2) a weak (or para) perspective camera model.

At convergence, (17) is equivalent with the perspective equations (13), (14) or (15), (16). In other terms, *this algorithm solves for Euclidean reconstruction with a perspective camera by iterations of an Euclidean reconstruction method with an affine 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 an affine camera model.

The iterative algorithm outlined in this paper is best illustrated in Fig. 1. At the first iteration, the algorithm considers the *true perspective projections* of P_i and attempts to reconstruct the 3D points as if they were projected in the image using weak perspective. 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 *weak perspective projections*.

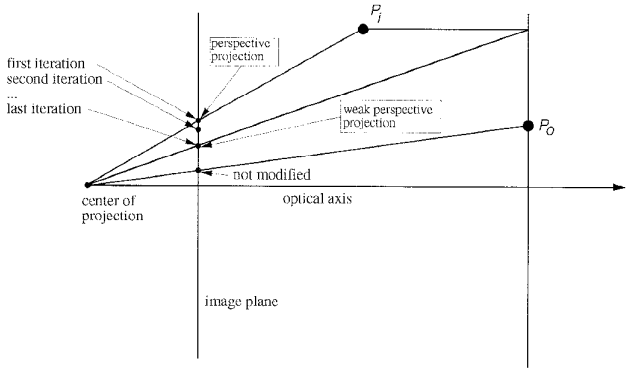

Fig. 1. The iterative algorithm described in this section modifies the projection of a 3D point from true perspective to weak perspective (see text).

4 Reconstruction with a Weak-Perspective or a Paraperspective Camera

In this section, we develop **step 2** of the algorithm outlined in the previous section. Methods that use a linear camera model provide a 3D affine reconstruction if at least two views of four noncoplanar points are available and if the motion is not a pure translation. However, three 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 [9], the factorization method is more convenient from a practical point of view.

The factorization method [1] computes shape and motion simultaneously by performing a singular value decomposition of the $2k \times n$ matrix σ which is formed by concatenating (17) for all i and j:

$$\sigma = \mathbf{A}\mathbf{S} \tag{22}$$

We refer to this formula as the *affine shape and motion equation*, or:

$$\begin{pmatrix} \mathbf{s}_{11} & \cdots & \mathbf{s}_{n1} \\ \vdots & & \vdots \\ \mathbf{s}_{1k} & \cdots & \mathbf{s}_{nk} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_k \end{pmatrix} \begin{pmatrix} \mathbf{S}_1 & \ldots & \mathbf{S}_n \end{pmatrix}$$

Tomasi and Kanade [1] noticed that $\mathbf{A}$ and $\mathbf{S}$ in (22) may be computed simultaneously by performing a singular value decomposition of the $2k \times n$ matrix: $\sigma = \mathbf{O}_1\Sigma\mathbf{O}_2$.

Obviously, the factorization method briefly outlined above does not provide a unique decomposition of the measurement matrix σ. Tomasi and Kanade [1] and Weinshall and Tomasi [8]} provide solutions for the case of a weak perspective camera. Poelman and Kanade [2] and Christy and Horaud [9] provide solutions for the case of a paraperspective camera.

One has to determine 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 $\mathbf{T}$ such that the affine shape $\mathbf{S}$ becomes Euclidean and the affine motion becomes rigid. Following [1], [8], [2], and [9] the constraints allowing the estimation of matrix $\mathbf{T}$ are nonlinear. With the substitution $\mathbf{Q} = \mathbf{T}\mathbf{T}^T$ these constraints become linear. Once $\mathbf{Q}$ is estimated, $\mathbf{T}$ can be recovered up to mirror-symmetric ambiguity with an affine camera and uniquely with a perspective camera, as it is explained below.

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 either a weak perspective or 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 $\mathbf{Q} = \mathbf{T}\mathbf{T}^T$. Clearly this decomposition is not unique and there are at least two ways to determine $\mathbf{T}$:

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

Let $\mathbf{H}$ be a nonsingular matrix such that $\mathbf{L} = \mathbf{K}\mathbf{H}$ and we have:
$$\mathbf{Q} = \mathbf{L}\mathbf{L}^T = \mathbf{K}\mathbf{H}\mathbf{H}^T\mathbf{K}^T = \mathbf{K}\mathbf{K}^T$$
We conclude that $\mathbf{H}$ is necessarily an orthogonal matrix. The orthogonality of $\mathbf{H}$ is also claimed in [8] but without any formal proof. Therefore $\mathbf{H}$ 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 $-\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Therefore the affine shape and motion equation can be written as:
$$\sigma - \mathbf{A}\mathbf{S} = (-\mathbf{A})(-\mathbf{S})$$
Because of this reversal ambiguity, there are two solutions for the ε_{ij}s at each iteration of the reconstruction algorithm described above.

We consider the case of paraperspective. A similar treatment is possible for weak perspective. The vectors $\mathbf{k}_j$ are computed according to [9]. This computation may use either $\mathbf{I}_p$ and $\mathbf{J}_p$ (the first solution) or $-\mathbf{I}_p$ and $-\mathbf{J}_p$ (the second solution). Therefore we obtain two distinct solutions, that is, $\mathbf{k}_j^1$ and $\mathbf{k}_j^2$. The two solutions for ε_{ij} correspond to $\mathbf{k}_j^1$ and $\mathbf{P}_i$ and to $\mathbf{k}_j^2$ and $-\mathbf{P}_i$:
$$\varepsilon_{ij}^{1,2} = \pm\frac{\mathbf{k}_j^{1,2}\cdot\mathbf{P}_i}{t_{z_j}}$$
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 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.

Let $S^{(1)}$ be the positive shape computed at the first iteration of the algorithm and $R^{(1)}$ be the negative shape ($R^{(1)} = -S^{(1)}$). At each one of the next iterations one has to deal with four shapes: $S_1^{(k)}$ and $S_2^{(k)}$ that are issued from the positive solution and $R_1^{(k)}$ and $R_2^{(k)}$ that are issued from the negative solution. The S-shape and the R-shape the most consistent with the shapes selected at the previous iterations are selected. Finally, a unique solution is selected on the basis of consistency with the image data.

6 Method Analysis

6.1 Sensitivity to Camera Calibration

So far we assumed that the camera is calibrated, which means that the intrinsic camera parameters are known, i.e., α_u, α_v, u_c, and v_c in (1). It is well known that camera calibration is difficult and that the camera parameters are not stable over time and space. It is also known that the only stable intrinsic camera parameter is the ratio between the horizontal and vertical pixel size: $\gamma = \alpha_u/\alpha_v$. We consider again the perspective equations (13) and (14). By combining them with $u_i = \alpha_u x_i + u_c$ and with $v_i = \alpha_v y_i + v_c$ we obtain:
$$u_i(1+\varepsilon_i) - u_0 - u_c\varepsilon_i = \gamma\alpha_v\mathbf{I}\cdot\mathbf{P}_i \tag{23}$$
$$v_i(1+\varepsilon_i) - v_0 - v_c\varepsilon_i = \alpha_v\mathbf{J}\cdot\mathbf{P}_i \tag{24}$$
By inspecting the above equations, it is straightforward to notice that the intrinsic parameter α_v acts as a scale factor on the 3D shape. *Since shape can be recovered only up to a similarity transformation, exact knowledge of the value of α_v does not affect the solution.*

Moreover, the remaining intrinsic parameters, u_c and v_c, are weighted by ε_i which is the ratio between the size of the object measured along the optical axis divided by the distance between the object and the optical center of the camera. The absolute value of this ratio (ε_i) varies between 0 (the object is very far) and a positive value that insures that the object doesn't "bump" into the lens of the camera. In practice a maximum value of ε_i may be 0.5 but more realistic values are in the range [0.1, 0.2]. Therefore, the calibration errors committed on u_c and v_c are scaled down by the same factor.

6.2 An Analysis of Convergence

In order to analyze the convergence of the iterative reconstruction algorithm outlined in Section 3 we consider separately the equations associated with a weak perspective camera model and with a paraperspective camera model. It is quite difficult to analyze the convergence of such algorithms from a theoretical point of view. Therefore we base our analysis on numerical considerations by analyzing the size of the neglected terms relative to those that are retained.

Consider (13), (14) and (15), (16). Both these sets of equations describe the *full* perspective projection with a calibrated camera. The first ones allow to express the perspective reconstruction problem in terms of an *iterative weak perspective* algorithm while the second ones allow to express the same problem in terms of an *iterative paraperspective* algorithm. If good estimates for the values of ε_i are available, then the perspective reconstruction problem is reduced to an affine reconstruction problem. Of course, in practice it seems difficult to provide such estimates. In this section we show that initializing ε_{ij} to zero provides sensible initial estimates even if the object is quite close to the camera. This explains why the iterative algorithms converge in a few iterations. We start by analyzing the weak perspective case and then we extend our analysis to paraperspective.

Let, for n points and k views, ε_{ij}^* be the true values that the iterative algorithm is supposed to eventually estimate. At the first iteration, the algorithm performs a standard reconstruction using a weak perspective camera model, i.e., it computes shape and motion using the affine shape and motion equation (22) followed by Euclidean normalization.

Therefore, the "reconstruction errors" are proportional to the "weak perspective errors" $\left|x_{ij}\varepsilon_{ij}^*\right|$ and $\left|y_{ij}\varepsilon_{ij}^*\right|$. If these errors are large, the weak perspective solution estimated at the first iteration

of the algorithm will be rather different than the solution being sought and convergence cannot be guaranteed. In their work on pose estimation, Dementhon and Davis [6] noticed that the iterative weak perspective algorithm converges even if the expected values for ε_{ij} are as large as 1, *provided that the scene points lie in the neighborhood of the optical axis*. Indeed, when the scene points are close to the optical axis, the camera coordinates of their projections, x_{ij} and y_{ij}, are small (the origin of the camera frame lies onto the optical axis) and they compensate for the large values of ε_{ij}^* (see Section 2.1). Moreover, as it has been discussed in the previous Section, the most realistic maximum value for ε_{ij}^* is 0.5.

The iterative paraperspective algorithm is able to deal with configurations where the weak perspective algorithm diverges. Indeed, in the case of paraperspective, the initial errors are $\left|(x_{ij} - x_{0j})\varepsilon_{ij}^*\right|$ and $\left|(x_{ij} - x_{0j})\varepsilon_{ij}^*\right|$. Whenever the point p_0 is properly chosen (typically, it should be the center of mass of the set of image points), then the differences $(x_{ij} - x_{0j})$ and $(y_{ij} - y_{0j})$ are small and they compensate for large values of ε_{ij}^*. Therefore the iterative paraperspective reconstruction algorithm is likely to converge over a wider range of configurations than the weak perspective one.

6.3 A Comparison with Nonlinear Minimization Methods

In the past, a number of authors have tried to solve the Euclidean structure and motion problem using nonlinear optimization techniques. It its most general form the problem is to minimize the following error function [5]:

$$f(X) = \sum_{ij}\left(\left(x_{ij} - \hat{x}_{ij}\right)^2 + \left(y_{ij} - \hat{y}_{ij}\right)^2\right)$$

where x_{ij} and y_{ij} are the coordinates of an image point and $\hat{x}_{ij}$ and $\hat{y}_{ij}$ are the coordinates of the projected 3D point using a perspective camera model: the latter coordinates are therefore given by (2) and (3).

For n points and k images, the error function above has $2 \times n \times k$ positive squared terms. The vector X encapsulates the unknowns of the problem: $3 \times n$ coordinates and $6 \times k$ motion parameters. We seek a value for X which minimizes the error function. The Jacobian of $f(X)$ is a $m \times p$ matrix and we have: $m = 2 \times n \times k$ and $p = 3 \times n + 6 \times k$. Any nonlinear minimization method searches the minimum incrementally and at each iteration the following linear system must be solved in order to compute dX and replace X by $X + dX$:

$$\mathbf{J}^T(X)\mathbf{J}(X)dX = \mathbf{b}$$

Therefore, the complexity of each iteration is dominated by the complexity of inverting a symmetric definite positive matrix—the Hessian. The size of the Hessian matrix is $p \times p$ and therefore it is a function of n and k. Furthermore, if one takes into account the fact that the Hessian is a banded matrix, the complexity of inverting the Hessian is of $p^3 + 8p^2 + p$ flops [10]. By replacing p with its expression we obtain the following complexity: $27n^3 + 162n^2k + 324nk^2 + 216k^3 + 72n^2 + 288nk + 288k^2 + 3n + 6k$. If we retain the third-order terms we obtain: $27n^3 + 162n^2k + 324nk^2 + 216k^3$.

In order to compare our iterative method with such a nonlinear method, let's compute the complexity of one iteration. The most time-consuming part of the algorithm is the singular value decomposition of the measurement matrix σ of size $2k \times n$. Therefore, the complexity of singular value decomposition is in our case [10]: $22n^3 + 8nk^2$.

The complexities of the factorization method and of the nonlinear minimization methods are shown in Table 1 for three cases: the number of images (k) is much smaller than the number of points (n), the number of images is approximately equal to the number of points, and the number of images is much larger than the number of points.

TABLE 1
THE NUMBER OF FLOAT OPERATIONS AS A FUNCTION OF THE NUMBER OF POINTS (N) AND OF THE NUMBER OF IMAGES (K)

Method	$k \approx n/10$	$k \approx n$	$k \approx 10n$
Factorization	$22n^3$	$30n^3$	$822n^3$
Nonlinear (one iteration)	$46n^3$	$729n^3$	$250000n^3$

The above comparison holds for one iteration. We can conclude that the method proposed in this paper is *intrinsically* more efficient than a non linear minimization method. Notice the dramatic increase in complexity of the nonlinear method when the number of images is larger than the number of points.

7 Simulation Experiments

In this section, we study the performance of the affine iterative algorithms and we compare them with the factorization method. Two types of performances are studied:

1) the accuracy of 3D reconstruction as a function of various types of motions and in the presence of image (pixel) noise and
2) the convergence of the affine iterative algorithms as a function of various types of motion.

In all the experiments described in this section we used 15 images and 42 tracked points and the following features:

- The intrinsic camera parameters are: $u_c = v_c = 256$, $\alpha_u = \alpha_v = 1{,}000$.
- The angular variation between each view is 2^0.
- Gaussian noise with maximum standard deviation $\sigma = 1$ has been added to the image measurements and there are 200 trials for each experiment.
- One important parameter for each experiment is the average distance between the 3D points and the camera. Let D be the distance from the center of gravity of the set of 3D points to the center of projection, *divided* by the diameter of the 3D point set—D is therefore unit-less and we call it a *relative distance*.
- For those experiments for which D varies, the maximum variation is limited to 5.
- The quality of a reconstruction result is evaluated by the mean and maximum of the Euclidean distance between the theoretical 3D points and the estimated 3D points as well as the mean and maximum of the difference between the theoretical angle values between pairs of 3D edges and the estimated angle values.

The experiments below compare the iterative paraperspective algorithm with the paraperspective factorization algorithm, the latter being simply the first iteration of the former. Figs. 2 and 3 show the quality of the 3D Euclidean reconstruction for various motion types. The behavior of the reconstruction algorithm does not seem to depend on the direction of motion.

Figs. 4 and 5 study the behavior of the two iterative algorithms (weak perspective and paraperspective) when the motion is parallel to the optical axis and when the center of gravity of the 3D point set is at a fixed offset away from the optical axis. On an average, the paraperspective iterative algorithm requires less iterations than the weak perspective iterative algorithm. The convergence rate of both algorithms is close to 100% for a relative distance greater than four. When the relative distance is equal to three, the convergence rate of both algorithms drops to 75%. It is worthwhile to notice that relative distances smaller than three are not realistic in practice, because, in this case, partial occlusion of the point set becomes predominant.

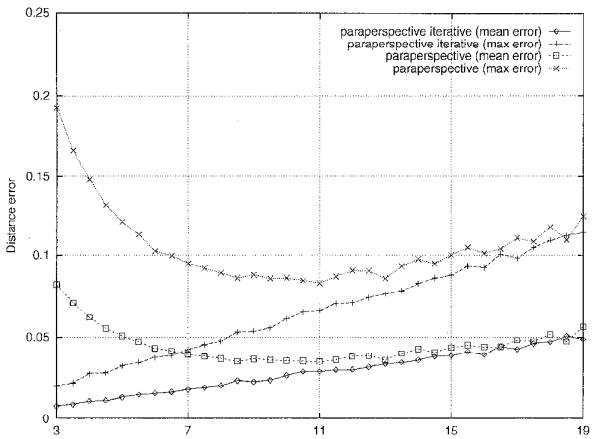

Fig. 2. The quality of reconstruction as a function of D (see text) when the motion direction is parallel to the image plane.

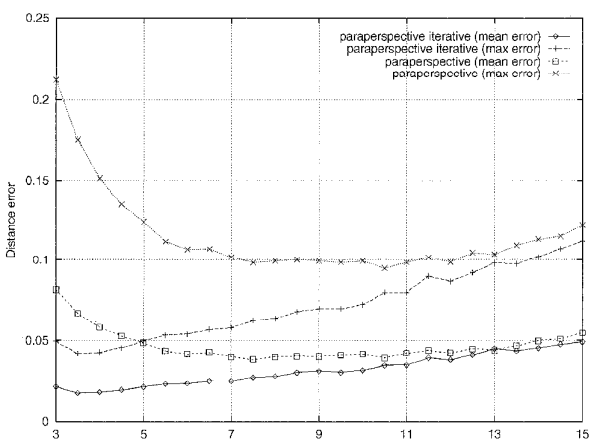

Fig. 3. The quality of reconstruction as a function of D when the motion is towards the camera, roughly parallel to the optical axis, and the center of gravity of the point set is at a fixed offset away from the optical axis.

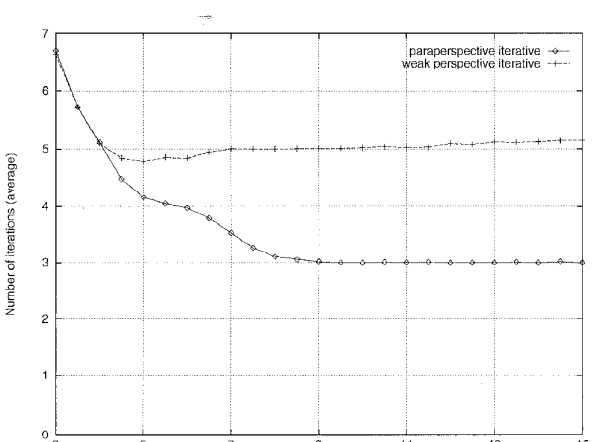

Fig. 4. Number of iterations (weak perspective and paraperspective) as a function of D where the motion is towards the camera and the center of gravity of the point set is at a fixed offset away from the optical axis.

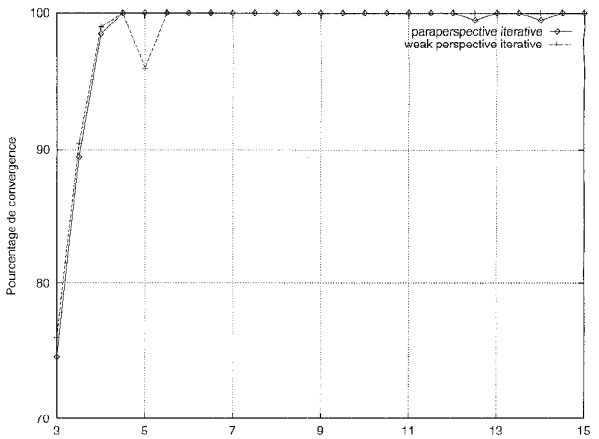

Fig. 5. The rate of convergence of the two iterative algorithms as a function of D where the motion is towards the camera and the center of gravity of the point set is at a fixed offset away from the optical axis.

8 Real Imagery Experiments

In this section, we consider two examples obtained with real images and with the paraperspective iterative algorithm:

- A sequence of five images of a cube where 38 points were tracked over the image sequence (Fig. 6) and
- A sequence of five images of a house with 46 tracked points (Fig. 7).

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 $\alpha_u = 1{,}500$ and $\alpha_v = 1{,}000$. The camera motion was a general motion. One may easily notice the large discrepancy between the reconstruction results obtained with the factorization method and with the affine iterative method—this discrepancy is visible especially when the data is shown from above such that the perspective effect associated with visualization vanishes. In both cases, the iterative paraperspective algorithm converged after five iterations and the computation time was of 0.3 seconds on a Sun/Sparc10-41/Sun-OS workstation.

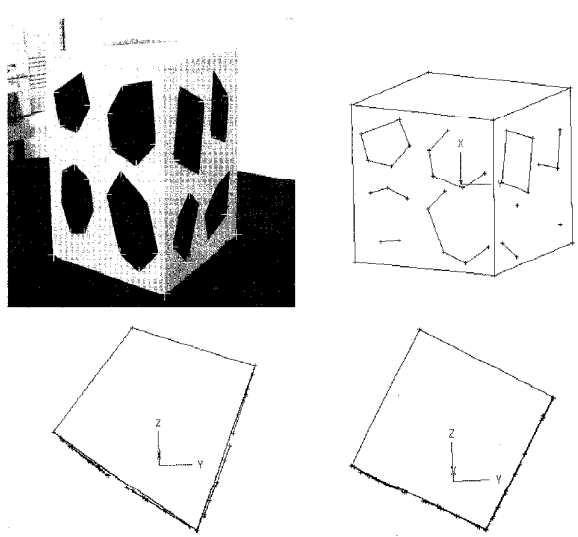

Fig. 6. This figure shows an image (top-left) out of a sequence of five images grabbed with a moving camera and the result of reconstructing 38 points with the iterative method (top-right). Top views of the reconstructed scene allow to compare more quantitatively the result of the factorization method (bottom-left) with the result of the iterative method (bottom-right) described in this paper.

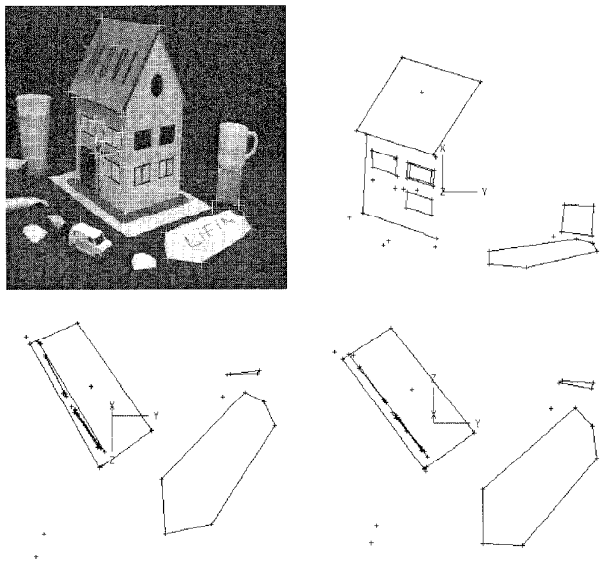

Fig. 7. Same as the previous figure for another sequence of five images and 46 points.

9 Discussion

In this paper, we described a method for solving the Euclidean reconstruction problem with a perspective camera by incrementally performing an Euclidean reconstruction with either a weak or a paraperspective camera model. The method converges—on an average—in five iterations, is computationally more efficient than non-linear minimization methods, 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 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.

Although the experimental results show that there are few 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. Indeed the experiments that we carried out and which are described in detail above (Section 8) show that convergence does not depend upon the motion that the camera undergoes with respect to the scene. The algorithm fails to converge when there are scene points very close to the camera. However such configurations are not desirable because they lead to occlusions. The paraperspective model has better convergence properties than the weak perspective one, mainly because it requires fewer iterations.

Although we have not performed such a comparison, it is clear that non-linear minimization algorithms provide more accurate results than our algorithm, simply because non linear methods are designed to minimize a least-squares error function at the cost, nevertheless, of a larger number of float operations (see Table 1). As already mentioned, our algorithm converges after five iterations (on an average) which compares favorably with the number of iterations associated with nonlinear minimization methods as it was reported by a number of authors [3], [5].

Therefore, the class of iterative algorithms described in this paper are an excellent compromise between linear resolution techniques (affine camera) and nonlinear minimization techniques (perspective camera), both in terms of quality of reconstruction and of computation time.

Acknowledgments

This work has been supported by Société Aérospatiale. Stéphane Christy acknowledges financial support from DRET.

References

[1] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams Under Orthography: A Factorization Method," *Int'l J. Computer Vision*, vol. 9, no. 2, pp. 137–154, Nov. 1992.

[2] C.J. Poelman and T. Kanade, "A Paraperspective Factorization Method for Shape and Motion Recovery," *Computer Vision—ECCV 94, Proc. Third European Conf. Computer Vision*, J.-O. Eklundh, ed., vol. 2, pp. 97–108. Stockholm: Springer Verlag, May 1994.

[3] R.I. Hartley, "Euclidean Reconstruction from Uncalibrated Views," *Applications of Invariance in Computer Vision*, M.Z. Forsyth, ed., pp. 237–256. Berlin Heidelberg: Springer Verlag, 1994.

[4] L. Quan, "Self-Calibration of an Affine Camera from Multiple Views," Technical Report RT 125, IMAG–LIFIA, Dec. 1994.

[5] R. Szelinski and S.B. Kang, "Recovering 3-D Shape and Motion from Image Streams Using Non-Linear Least Squares," *J. Visual Comm. and Image Representation*, vol. 5, no. 1, pp. 10–28, Mar. 1994.

[6] D.F. DeMenthon and L.S. Davis, "Model-Based Object Pose in 25 Lines of Code," *Int'l J. Computer Vision*, vol. 15, no. 1/2, pp. 123–141, 1995.

[7] R. Horaud, S. Christy, F. Dornaika, and B. Lamiroy, "Object Pose: Links Between Paraperspective and Perspective," *Proc. Fifth Int'l Conf. Computer Vision*, pp. 426–433, Cambridge, Mass., June 1995.

[8] D. Weinshall and C. Tomasi, "Linear and Incremental Acquisition of Invariant Shape Models from Image Sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 512–517, May 1995.

[9] S. Christy and R. Horaud, "Euclidean Shape and Motion from Multiple Perspective Views by Affine Iterations," Technical Report RR-2421, INRIA, Dec. 1994. Available by anonymous ftp on imag.fr, directory pub/MOVI, file RR-2421.ps.gz.

[10] G.H. Golub and C.F. Van Loan, *Matrix Computations*. Baltimore: The Johns Hopkins Univ. Press, 1989.