Chapter 1

Using Geometric Algebra in Optical Motion Capture

Joan Lasenby & Adam Stevenson

1.1 Introduction

Optical motion capture refers to the process by which accurate 3D data from a moving subject is reconstructed from the images in two or more cameras. In order to achieve this reconstruction it is necessary to know how the cameras are placed relative to each other, the internal characteristics of each camera and the matching points in each image. The goal is to carry out this process as automatically as possible. In this paper we will outline a series of calibration techniques which use all of the available data simultaneously and produce accurate reconstructions with no complicated calibration equipment or procedures. These techniques rely on the use of geometric algebra and the ability therein to differentiate with respect to multivectors and linear functions.

Optical motion capture involves the use of multiple cameras to observe a moving subject. From the 2D data in each camera the goal is to obtain a moving 3D reconstruction of our subject. This process has applications in medicine, biomechanics, sports training and animation. The whole motion capture process starts by *calibrating* the cameras – i.e. determining their relative positions and orientations and the internal camera characteristics. In any practical system, we require this process to be easy to accomplish and the results to be accurate. This paper will look in detail at this initial stage of the motion capture process, in particular the determination of the relative orientations and positions of any number of cameras given no special calibration object. The algorithms developed for this purpose involve the use of geometric algebra and result in an iterative scheme which does not require any non-linear minimization stage. There are already many examples of the use of geometric algebra in other computer vision applications a few of which are given in [1, 8, 9]. During the m-camera calibration process we shall see that two very useful algorithms emerge: firstly, a straightforward, analytic means of estimating the relative translations between cameras (not simply their directions) given that the relative rotations are known, is presented. Secondly, given any number of cameras and their relative rotations and translations, we show how to produce a robust, optimal (in a least squares sense) estimate of the world coordinates. Both techniques could be useful in a variety of applications and are each programmed in just a few lines of code.

The setup we use consists of three 50Hz monochrome CCD cameras each connected to the inputs of a framegrabber card located in a PC – a synch signal is fed into the cameras so that the digitised data comes from simultaneous frames, see figure 1.1. The system will shortly be extended to 6 cameras.

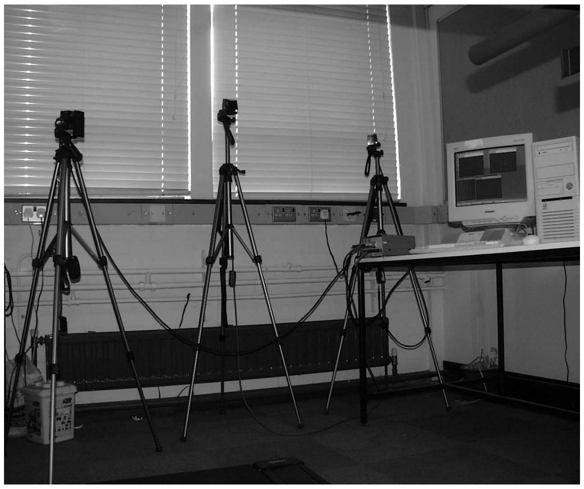

FIGURE 1.1. 3-camera motion capture system

Retroreflective markers are placed on the moving subject and these are illuminated with IR radiation directed from each of the cameras. Image sequences of bright blobs are then captured – one for each camera. Storing only the locations of the bright blobs dispenses with the need for expensive frame-stores. In the subsequent processing, the bright blobs in each frame are reduced to single points by an algorithm which attempts to find the 'centre of mass' of each blob. We are therefore left with a list of the pixel coordinates for the points seen in each frame for each image. Assuming we are able to reconstruct 3D data from matched image points, it is essential that we are able to track and match the points through the sequences. For complicated motions, tracking can be the hardest part of the whole process; points crossing, being occluded, performing abrupt changes of direction, all add to the difficulties. Experience has shown that one reliable means of tracking is to track the points in space, i.e. to track the 3D motion – this enables one to use rigidity and length constraints (i.e. information from a model) in a simple fashion to improve the prediction process. Therefore, for reliable tracking it is very important that we have a good initial calibration of the system, otherwise the reconstructions will be poor and the tracking may experience problems. This is one of the main incentives for developing an accurate user-friendly means of obtaining the system calibration parameters. The following section will explain what the calibration parameters are and how we can estimate these using geometric algebra (GA) techniques. This will be followed by some results showing the accuracy of the calibrations via simulations and tests on real data. Throughout the paper we will assume that the readers are familiar with basic GA manipulations – for simple introductions to GA see [4, 7, 3, 5]. In this paper we will use the convention that where indices are repeated in the contravariant and covariant positions, i.e. $a^i b_i$, they are summed over unless explicitly stated otherwise.

1.2 External and Internal Calibration

In this section we will explain what is meant by external and internal calibration and show how we can use GA techniques to determine the unknown calibration parameters.

1.2.1 External Calibration

Suppose that we have m cameras which we label 1 to m – these cameras are placed about the field of view. The aim is to place the cameras such that at any point in the image sequence, any given world point will always be visible in at least two of the cameras – this may not always be possible, but the tracking software can often make sensible predictions based on the rest of the tracked sequence when no prediction from the data is possible. Let us take the first camera, 1, as our reference camera. Then the position and orientation of camera j will be completely specified by a rotor R_j and a translation $\boldsymbol{t}_j$ as shown in figure 1.2.

Part of the calibration process will therefore be to determine, as accurately as possible, the $m-1$ rotors and the $m-1$ translations.

1.2.2 Internal Calibration

A world point $\boldsymbol{X} = (X, Y, Z)$ is projected onto an image plane to give an image point $\boldsymbol{x} = (x, y, f)$ where f is the focal length of the camera (pinhole camera model), see figure 1.3

However, from the image we will measure pixel coordinates $\boldsymbol{u} = (u, v, 1)$. In order to move between pixel and image coordinates it is easy to show that there exists a 3×3 matrix, C which takes $\boldsymbol{x}$ to $\boldsymbol{u}$:

$$\boldsymbol{u} = C(\boldsymbol{x}/f), \qquad \boldsymbol{x}/f = C^{-1}\boldsymbol{u}$$

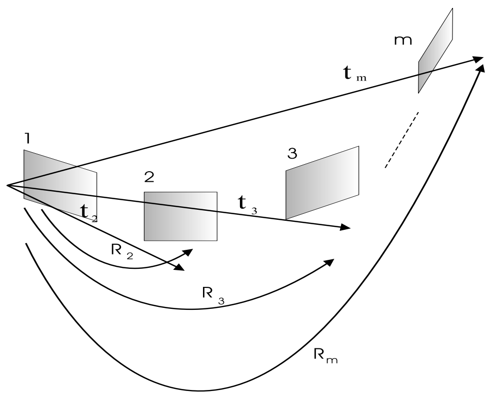

FIGURE 1.2. Rotations and translations of cameras relative to reference, chosen as camera 1

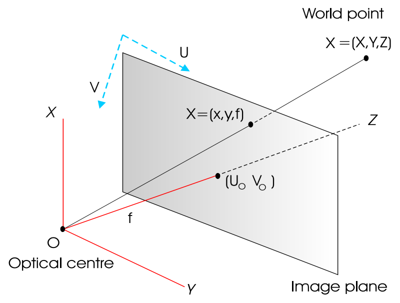

FIGURE 1.3. Factors determining the internal calibration parameters

where C is of the form

$$C = \begin{pmatrix} \alpha & \beta & u_0 \\ \gamma & \delta & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

(u_0, v_0) is known as the principal point – it is where the optical axis of the camera cuts the image plane. $\alpha, \beta, \gamma, \delta$ depend upon the possible scaling and skewing of the pixel axes and f is the focal length (distance along the optical axis from the optical centre to the image plane).

The remainder of the calibration process will therefore be to determine the internal camera parameters. The internal parameters can be found via a variety of techniques and, once found, are unlikely to vary over reasonable timescales. In this paper we will mainly focus on how to accurately estimate the external parameters given knowledge of the internal parameters (in this case we say we are working with *calibrated cameras*, although a later section will indicate how we can include estimation of the internal parameters in the estimation procedure.)

1.3 Estimating the External Parameters

Suppose first that we know internal calibration matrices C_j for each camera, $j = 1, .., m$. Let the N world points that we observe with our cameras be $\boldsymbol{X}_i, i = 1, .., N$, and define an *occlusion field* O_{ij} such that $O_{ij} = 1$ if $\boldsymbol{X}_i$ is visible in camera j and 0 if it is not visible in camera j. In practice, we would like to be able to do this external calibration without having to track points (recall the tracking *uses* the calibration information). This is done by waving a single marker or light source over the viewing area (usually a volume of around 2m^3 should be covered for adequate calibration). In this way each camera will see no points or only one point and there is no tracking or matching problem. It is of course possible that some cameras will see more than one point due to the presence of spurious sources – if this occurs the frame is not used in the calibration process.

Let $\boldsymbol{u}_{ij}$ be the observed pixel coordinates (of the form $(u, v, 1)$) of the projection of world point $\boldsymbol{X}_i$ in camera j. Since we know the internal calibration parameters of each camera, we can recover the image coordinates, $\boldsymbol{x}_{ij}$, for this point via $\boldsymbol{x}_{ij} = C_j^{-1}\boldsymbol{u}_{ij}$ (from hereon we will take it that $\boldsymbol{x}_{ij} \equiv \boldsymbol{x}_{ij}/f$ to reduce the complication). If R_j and $\boldsymbol{t}_j$ are the rotor and translation which relate the frame and position of camera j to the reference frame of camera 1 then the following relation holds

$$\boldsymbol{X}_{ij} = \tilde{R}_j(\boldsymbol{X}_i - \boldsymbol{t}_j)R_j \tag{1.1}$$

where $\boldsymbol{X}_{ij}$ is world point i in the coordinate frame of camera j, see [7].

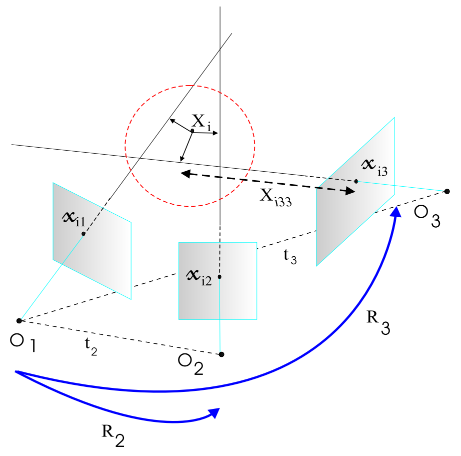

FIGURE 1.4. Geometric depiction of the meaning of cost function S_2

If the noise occurs in the image planes, we might expect that our estimates of the Rs and $\boldsymbol{t}$s would best be found via minimization of the following cost function

$$S_1 = \sum_{j=1}^{m} \sum_{i=1}^{N} \left[\boldsymbol{x}_{ij} - \frac{\tilde{R}_j(\boldsymbol{X}_i - \boldsymbol{t}_j)R_j}{[\tilde{R}_j(\boldsymbol{X}_i - \boldsymbol{t}_j)R_j]\cdot \boldsymbol{e}_3} \right]^2 O_{ij} \tag{1.2}$$

This is effectively minimizing the sum of the squared distances in the image planes between the observed image points and the projected points. We should note here that $R_1 \equiv \mathcal{I}$ (the identity), $\boldsymbol{t}_1 \equiv \mathbf{0}$, and the presence of the O_{ij} ensures that if the point $\mathbf{X}_i$ is not visible in camera j then there is no contribution from this term. However, we can see immediately that the presence of the parameters we are trying to estimate in the denominator of the right-hand term makes this equation a difficult one – we would certainly have to find the minimum via some non-linear optimization technique.

Now, suppose that instead we consider the following cost function:

$$S_2 = \sum_{j=1}^{m} \sum_{i=1}^{N} \left[X_{ij3}\boldsymbol{x}_{ij} - \tilde{R}_j(\boldsymbol{X}_i - \boldsymbol{t}_j)R_j \right]^2 O_{ij} \tag{1.3}$$

Here X_{ij3} is the distance we have to move out along the ray joining the optical centre of camera j to image point $\boldsymbol{x}_{ij}$ in order to minimize the distance between the world point $\boldsymbol{X}_i$ and the point $X_{ij3}\boldsymbol{x}_{ij}$. The above cost function is therefore the sum of squared distances between the world points and their closest points on the camera rays projecting out from the observed image points. Thus, while S_1 represents a cost function in the image planes, S_2 represents a cost function in the world, see figure (1.4). Our observations are the image points in the m cameras, therefore the noise on our observations occurs in these image planes – if one assumes Gaussian noise one might therefore want to minimize the cost function S_1. However, it is also true that minimizing S_1 does not ensure that the reconstructed world points are in some way 'as close as possible' to the observed rays – which one might also deem desirable. In fact, both cost functions are likely to give good results and we choose to optimize S_2 in order to obtain avoid a non-linear minimization.

Now, let us try to optimize S_2 over our parameters $R_j, \boldsymbol{t}_j, X_{ij3}, \boldsymbol{X}_i$. Although for external calibration purposes we are only interested in the relative rotations and translations of the cameras, here we shall adopt a maximum likelihood approach and differentiate with respect to all of our unknown parameters. We shall show in the following sections that it is possible to obtain an iterative solution to this minimization problem and that this procedure converges reliably provided the data is not very poor. This differentiation will involve differentiation with respect to scalars, vectors and rotors.

In the following sections we will frequently use the quantities defined below:

$$n_j = \sum_{i=1}^{N} O_{ij} \qquad \text{for} \qquad j = 1, 2, .., m \tag{1.4}$$

$$m_i = \sum_{j=1}^{m} O_{ij} \qquad \text{for} \qquad i = 1, 2, .., N \tag{1.5}$$

Here, n_j is the number of points visible in camera j and m_i is the number of cameras that can see world point i.

1.3.1 Differentiation w.r.t. $\boldsymbol{t}_k$

When we take the derivative, ∂_a, with respect to (w.r.t.) a vector quantity $\boldsymbol{a}$ we use the fact that the differential operator ∂_a can be written (in terms of a basis $\{\boldsymbol{e}_i\}$) as

$$\partial_a = \boldsymbol{e}^i \frac{\partial}{\partial a^i} \qquad \text{where} \qquad \boldsymbol{a} = a^i \boldsymbol{e}_i \tag{1.6}$$

Here $\{\boldsymbol{e}^i\}$ is the **reciprocal frame** to $\{\boldsymbol{e}_i\}$, and is defined by $\boldsymbol{e}_i \cdot \boldsymbol{e}^j = \delta_i^j$, for $i, j = 1, 2, 3$. Note that we do not write vectors in bold when they appear as subscripts in the vector derivative. We now want to differentiate S_2 w.r.t. $\boldsymbol{t}_k$, where k can take values $2, 3, ..., m$. Consider first differentiating a vector squared, $\boldsymbol{x}^2$, w.r.t. $\boldsymbol{t} = t^j \boldsymbol{e}_j$. Taking out a factor of $\boldsymbol{e}^j$ on the left and using the fact that $\boldsymbol{uv} + \boldsymbol{vu}$ is equivalent to the inner product of the two vectors, we have that

$$\begin{aligned}\partial_t(\boldsymbol{x}\boldsymbol{x}) &= \boldsymbol{e}^j\frac{\partial \boldsymbol{x}}{\partial t^j}\boldsymbol{x} + \boldsymbol{e}^j\boldsymbol{x}\frac{\partial \boldsymbol{x}}{\partial t^j} \\ &= 2\boldsymbol{e}^j\left\{\frac{\partial \boldsymbol{x}}{\partial t^j}\cdot\boldsymbol{x}\right\} \end{aligned} \tag{1.7}$$

Thus, if we let $\{X_{ik3}\boldsymbol{x}_{ik} - \tilde{R}_k(\boldsymbol{X}_i - \boldsymbol{t}_k)R_k\} \equiv \boldsymbol{Y}_{ik}$, then $\partial_{t_k}S_2 = 0$ gives

$$\begin{aligned}\partial_{t_k}S_2 &= 2\sum_{i=1}^{N}\boldsymbol{e}^j\frac{\partial}{\partial t_k^j}\{\tilde{R}_k\boldsymbol{t}_kR_k\}\cdot\boldsymbol{Y}_{ik}O_{ik} \\ &= 2\sum_{i=1}^{N}\boldsymbol{e}^j[(\tilde{R}_k\boldsymbol{e}_jR_k)\cdot\boldsymbol{Y}_{ik}]O_{ik} \\ &= 2\sum_{i=1}^{N}\boldsymbol{e}^j\left(\boldsymbol{e}_j\cdot R_k\boldsymbol{Y}_{ik}\tilde{R}_k\right) \\ &= 2\sum_{i=1}^{N}R_k\boldsymbol{Y}_{ik}\tilde{R}_k = 2R_k\left[\sum_{i=1}^{N}\boldsymbol{Y}_{ik}\right]\tilde{R}_k = 0 \\ \Longrightarrow \quad & \sum_{i=1}^{N}\boldsymbol{Y}_{ik} = 0 \end{aligned} \tag{1.8}$$

where we have used the fact that $(R\boldsymbol{a}\tilde{R})\cdot\boldsymbol{b} = \boldsymbol{a}\cdot(\tilde{R}\boldsymbol{b}R)$. Since $\sum_{i=1}^{N}\boldsymbol{Y}_{ik}$ is linear in $\boldsymbol{t}_k$, it is straightforward to solve equation (1.8) for $\boldsymbol{t}_k$ to give

$$\boldsymbol{t}_k = \frac{1}{n_k}\sum_{i=1}^{N}\left[\boldsymbol{X}_i - X_{ik3}R_k\boldsymbol{x}_{ik}\tilde{R}_k\right]O_{ik} \tag{1.9}$$

We have $m-1$ such equations as k goes from 2 to m. Thus, if we have the data and have estimates for the world points, the rotors and the X_{ik3} values, we can solve for each of the translations.

1.3.2 Differentiation w.r.t. R_k

In geometric algebra we can differentiate w.r.t. any element of the algebra (for more details on multivector differentiation see [7, 6, 3]) and therefore w.r.t. rotors. Let us write

$$\partial_{R_k}S_2 = \partial_{R_k}\sum_{i=1}^{N}(\boldsymbol{v}_{ik} - \tilde{R}_k\boldsymbol{u}_{ik}R_k)^2O_{ik}$$

where $\boldsymbol{v}_{ik} = X_{ik3}\boldsymbol{x}_{ik}$ and $\boldsymbol{u}_{ik} = \mathbf{X}_i - \boldsymbol{t}_k$. The RHS has now been put in a standard form for which the solution (see [7] for details) is as follows

$$\partial_{R_k}S_2 = 4\tilde{R}_k\sum_{i=1}^{N}\boldsymbol{v}_{ik}\wedge(\tilde{R}_k\boldsymbol{u}_{ik}R_k)O_{ik} \tag{1.10}$$

For a minimum we require $\partial_{R_k} S_2 = 0$, and therefore the R_k must satisfy

$$\sum_{i=1}^{N} \boldsymbol{v}_{ik}\wedge(\tilde{R}_k \boldsymbol{u}_{ik} R_k) O_{ik} = \sum_{i=1}^{N}\{X_{ik3}\boldsymbol{x}_{ik}\wedge\tilde{R}_k(\mathbf{X}_i - \boldsymbol{t}_k)R_k\}O_{ik} = 0 \qquad (1.11)$$

or, substituting for $\boldsymbol{t}_k$ from equation (1.9)

$$\begin{aligned}
\text{LHS} &= \sum_{i=1}^{N}\{X_{ik3}\boldsymbol{x}_{ik}\wedge\tilde{R}_k\left[\mathbf{X}_i - \frac{1}{n_k}\sum_{j=1}^{N}\left[\boldsymbol{X}_j - X_{jk3}R_k\boldsymbol{x}_{jk}\tilde{R}_k\right]O_{jk}\right]R_k\}O_{ik} \\
&= \sum_{i=1}^{N}\{X_{ik3}\boldsymbol{x}_{ik}\wedge\tilde{R}_k\left(\mathbf{X}_i - \frac{1}{n_k}\sum_{j=1}^{N}\boldsymbol{X}_j O_{jk}\right)R_k\}O_{ik} \\
&\equiv \sum_{i=1}^{N}(\tilde{\boldsymbol{v}}_{ik}\wedge\tilde{R}_k\tilde{\boldsymbol{u}}_{ik}R_k) = 0 \qquad (1.12)
\end{aligned}$$

where we now have $\tilde{\boldsymbol{v}}_{ik} = X_{ik3}\boldsymbol{x}_{ik}O_{ik}$ and $\tilde{\boldsymbol{u}}_{ik} = \mathbf{X}_i - \frac{1}{n_k}\sum_{j=1}^{N}\mathbf{X}_j O_{jk}$. The second line in the set of equations (1.12) is obtained by noting that $\sum_{i=1}^{N}[O_{ik}X_{ik3}\boldsymbol{x}_{ik}]\wedge\frac{1}{n_k}\sum_{j=1}^{N}X_{jk3}\boldsymbol{x}_{jk}O_{jk} = 0$. We can now solve for R_k via SVD as outlined in [7] – i.e.

$$\begin{aligned}
\tilde{R}_k &= VU^T \qquad \text{where} \qquad F^k = USV^T \\
\text{with} \qquad F^k_{\alpha\beta} &= \sum_{i=1}^{N}(\boldsymbol{e}_\alpha\cdot\tilde{\boldsymbol{u}}_{ik})(\boldsymbol{e}_\beta\cdot\tilde{\boldsymbol{v}}_{ik}) \qquad (1.13)
\end{aligned}$$

This can be done for each k. Thus, we see from the above that provided we have the data, the world points and the X_{ij3} values, we can make an estimate of the rotations using the maximum likelihood estimator for the translations.

1.3.3 Differentiation w.r.t. the X_{pq3}

Next we would like to differentiate w.r.t. the scalars X_{pq3} – recall these represent the distance along the ray we have to move to bring us 'as close as possible' to the world point.

For each X_{pq3} we have

$$\begin{aligned}
\partial_{X_{pq3}} S_2 &= \partial_{X_{pq3}}\left\{X_{pq3}\boldsymbol{x}_{pq} - \tilde{R}_q(\mathbf{X}_p - \boldsymbol{t}_q)R_q\right\}^2 O_{pq} \\
&= 2\left\{X_{pq3}\boldsymbol{x}_{pq} - \tilde{R}_q(\mathbf{X}_p - \boldsymbol{t}_q)R_q\right\}\cdot\boldsymbol{x}_{pq}O_{pq} = 0
\end{aligned} \qquad (1.14)$$

For $O_{pq} \neq 0$ we therefore have

$$X_{pq3} = \frac{[\tilde{R}_q(\mathbf{X}_p - \boldsymbol{t}_q)R_q]\cdot\boldsymbol{x}_{pq}}{\boldsymbol{x}_{pq}^2} \equiv \frac{(\mathbf{X}_p - \boldsymbol{t}_q)\cdot[R_q\boldsymbol{x}_{pq}\tilde{R}_q]}{\boldsymbol{x}_{pq}^2} \qquad (1.15)$$

This equation tells us how to estimate the values of the $\{X_{ij3}\}$ given we know the data, world points, rotations and translations.

1.3.4 Differentiation w.r.t. the $\mathbf{X}_k$

If we expand S_2 it is easy to see that the derivative w.r.t. $\mathbf{X}_k$ (for k from 1 to N) is given by

$$\begin{aligned}\partial_{X_k} S_2 &= \partial_{X_k} \sum_{j=1}^{m} \left\{ -2X_{kj3}(R_j \boldsymbol{x}_{kj} \tilde{R}_j)\cdot\mathbf{X}_k + (\mathbf{X}_k - \boldsymbol{t}_j)^2 \right\} O_{kj} \\ &= 2\sum_{j=1}^{m} \left[-X_{kj3} R_j \boldsymbol{x}_{kj} \tilde{R}_j + (\mathbf{X}_k - \boldsymbol{t}_j) \right] O_{kj} = 0 \qquad (1.16)\end{aligned}$$

where we have used the fact that $\partial_a(\boldsymbol{a}\cdot\boldsymbol{b}) = \boldsymbol{b}$. The above expression can then be rearranged to give

$$\mathbf{X}_k = \frac{1}{m_k} \sum_{j=1}^{m} [\boldsymbol{t}_j + X_{kj3} R_j \boldsymbol{x}_{kj} \tilde{R}_j] O_{kj} \qquad (1.17)$$

if $m_k \neq 0$. k can take the values 1 to N. Thus, we are able to estimate the world points given values of the rotations, translations and the $\{X_{ij3}\}$.

1.3.5 Refining the estimates of $\boldsymbol{t}_j$ *and* $\mathbf{X}_k$

From our data (consisting of one point in many frames viewed by each camera) it is relatively straightforward to obtain an initial guess at the R_j – this can be done by taking two cameras at a time and applying some standard algorithm (e.g. decomposing the Essential matrix [10], Weng et al's algorithm [12], etc.). Of course, this will not give a *consistent* set of rotations (e.g. $R_{23}R_2 \neq R_3$, where R_{23} is the rotor which takes the frame at camera 2 to the frame at camera 3), but it will give a reasonable starting point for the algorithm. Now, it would then be nice if we were able to estimate a consistent set of translations from these rotations and the data – but currently equation (1.9) gives $\boldsymbol{t}$ in terms of the other unknown parameters as well as the rotations. In addition, for reconstruction purposes, we would like to have an expression for the world points, $\{\boldsymbol{X}_k\}$, in terms of just the rotations and translations. This is clearly also going to be essential when we have calibrated our cameras and we are wanting to reconstruct in an optimal fashion, points in the world from all of our m-camera data. We will deal with the case of reconstruction first.

1.3.6 Optimal reconstruction from calibrated data

If we substitute equation (1.15) into equation (1.17) to eliminate the $\{X_{pq3}\}$ values, we have

$$\mathbf{X}_k = \frac{1}{m_k}\sum_{j=1}^{m}\left[\boldsymbol{t}_j + \frac{1}{\boldsymbol{x}_{kj}^2}\left\{(\mathbf{X}_k - \boldsymbol{t}_j)\cdot R_j\boldsymbol{x}_{kj}\tilde{R}_j\right\} R_j\boldsymbol{x}_{kj}\tilde{R}_j\right] O_{kj} \qquad (1.18)$$

To simplify the notation we write $\boldsymbol{w}_{ij} = R_j\boldsymbol{x}_{ij}\tilde{R}_j$. If we take the inner product of the above equation with $\boldsymbol{e}_i$, $i = 1,2,3$, we can rearrange to give

$$\mathbf{X}_k \cdot \left[\boldsymbol{e}_i - \frac{1}{m_k}\sum_{j=1}^{m}\frac{1}{\boldsymbol{x}_{kj}^2}(\boldsymbol{w}_{kj}\cdot\boldsymbol{e}_i)\boldsymbol{w}_{kj}O_{kj}\right] = \frac{1}{m_k}\sum_{j=1}^{m}\left[\boldsymbol{t}_j\cdot\boldsymbol{e}_i - \frac{1}{\boldsymbol{x}_{kj}^2}(\boldsymbol{w}_{kj}\cdot\boldsymbol{t}_j)(\boldsymbol{w}_{kj}\cdot\boldsymbol{e}_i)\right] O_{kj} \qquad (1.19)$$

We have $3 \times N$ such equations ($k = 1,..,N$ and $i = 1,2,3$). For each k we can construct a matrix equation for $\mathbf{X}_k$

$$A_k\mathbf{X}_k = \boldsymbol{b}_k \qquad \Longrightarrow \qquad \mathbf{X}_k = A_k^{-1}\boldsymbol{b}_k \qquad (1.20)$$

where the matrix A_k and vector $\boldsymbol{b}_k$ are given by

$$\begin{aligned} A_{ip}^k &= \delta_{ip} - \frac{1}{m_k}\sum_{j=1}^{m}\frac{1}{\boldsymbol{x}_{kj}^2}(\boldsymbol{w}_{kj}\cdot\boldsymbol{e}_i)(\boldsymbol{w}_{kj}\cdot\boldsymbol{e}_p)O_{kj} & (1.21)\\ \boldsymbol{b}_k\cdot\boldsymbol{e}_i = b_i^k &= \frac{1}{m_k}\sum_{j=1}^{m}\boldsymbol{t}_j\cdot\left[\boldsymbol{e}_i - \frac{1}{\boldsymbol{x}_{kj}^2}(\boldsymbol{w}_{kj}\cdot\boldsymbol{e}_i)\boldsymbol{w}_{kj}\right] O_{kj} & (1.22)\end{aligned}$$

Thus, if we have a knowledge of the calibration (Rs and $\boldsymbol{t}$s), we see that via equation (1.20) we can very quickly reconstruct the 3D world points with a method that uses *all* of the available data at once in a sensible way. More generally the SVD can be used to solve $A_k\mathbf{X}_k = \boldsymbol{b}_k$ to avoid possible degeneracy.

1.3.7 An initial estimate for the translations

Suppose we substitute for X_{ik3} from equation (1.15) into equation (1.9) (still using $\boldsymbol{w}_{ij} = R_j\boldsymbol{x}_{ij}\tilde{R}_j$)

$$\boldsymbol{t}_k = \frac{1}{n_k}\sum_{i=1}^{N}\left[\mathbf{X}_i - \frac{1}{\boldsymbol{x}_{ik}^2}\{(\mathbf{X}_i - \boldsymbol{t}_k)\cdot\boldsymbol{w}_{ik}\}\boldsymbol{w}_{ik}\right] O_{ik} \qquad (1.23)$$

for $n_k \neq 0$. If we now take the inner product of the above equation with $\boldsymbol{e}_j$ we have

$$\boldsymbol{t}_k \cdot \left[\boldsymbol{e}_j - \frac{1}{n_k} \sum_{i=1}^{N} \boldsymbol{y}_{ijk} O_{ik} \right] = \frac{1}{n_k} \sum_{i=1}^{N} \mathbf{X}_i \cdot \left[\boldsymbol{e}_j - \boldsymbol{y}_{ijk} \right] O_{ik} \tag{1.24}$$

with $\boldsymbol{y}_{ijk} = \frac{1}{\boldsymbol{x}_{ik}^2}(\boldsymbol{w}_{ik} \cdot \boldsymbol{e}_j)\boldsymbol{w}_{ik}$. Now, writing $\boldsymbol{a}_{ijk} = [\boldsymbol{e}_j - \boldsymbol{y}_{ijk}]O_{ik}$ and $\boldsymbol{p}_{jk} = \boldsymbol{e}_j - \frac{1}{n_k}\sum_{i=1}^{N} \boldsymbol{y}_{ijk} O_{ik}$ the above equation can be written more concisely as

$$\boldsymbol{t}_k \cdot \boldsymbol{p}_{jk} = \frac{1}{n_k} \sum_{i=1}^{N} \mathbf{X}_i \cdot \boldsymbol{a}_{ijk} \tag{1.25}$$

Recall from the previous section that we can write $\mathbf{X}_i = A_i^{-1}\boldsymbol{b}_i$ where A_i is a matrix which is a function of the Rs only and $\boldsymbol{b}_i$ is a vector which is a function of both the Rs and the $\boldsymbol{t}$s. Let us therefore write $\mathbf{X}_i = \underline{f}_i(\boldsymbol{b}_i)$, where $\underline{f}_i$ is the linear function corresponding to A_i^{-1}. Using the fact that $\underline{f}(\boldsymbol{c}) \cdot \boldsymbol{d} = \boldsymbol{c} \cdot \overline{f}(\boldsymbol{d})$, we can now rewrite equation (1.25) as

$$\boldsymbol{t}_k \cdot \boldsymbol{p}_{jk} = \frac{1}{n_k} \sum_{i=1}^{N} \boldsymbol{b}_i \cdot \overline{f}_i(\boldsymbol{a}_{ijk}) \tag{1.26}$$

The next step is to note that we can write equation (1.22) as

$$\boldsymbol{b}_k \cdot \boldsymbol{e}_i = \frac{1}{m_k} \sum_{j=1}^{m} \boldsymbol{t}_j \cdot \boldsymbol{a}_{kij} \tag{1.27}$$

Letting $\overline{f}_i(\boldsymbol{a}_{ijk}) = \tilde{a}_{ijk}^s \boldsymbol{e}_s$ we can write $\boldsymbol{b}_i \cdot \overline{f}_i(\boldsymbol{a}_{ijk})$ as

$$\boldsymbol{b}_i \cdot \overline{f}_i(\boldsymbol{a}_{ijk}) = \frac{1}{m_i} \tilde{a}_{ijk}^s \sum_{l=1}^{m} (\boldsymbol{t}_l \cdot \boldsymbol{a}_{isl}) \tag{1.28}$$

From this equation we can see that it will now be possible to use equation (1.26) in order to form a linear equation in the $\boldsymbol{t}$s. With some manipulation is it possible to obtain, for given j and k, the following expression

$$\sum_{l=1}^{m} \boldsymbol{t}_l \cdot \left\{ \frac{1}{n_k} \sum_{i=1}^{N} \frac{1}{m_i} \tilde{a}_{ijk}^s O_{ik} \boldsymbol{a}_{isl} - \boldsymbol{p}_{jk} \delta_{lk} \right\} = 0 \tag{1.29}$$

This can be written as a matrix equation of the form $Q\boldsymbol{T} = 0$ where $\boldsymbol{T} = [t_{21}, t_{22}, t_{23}, t_{31}, \ldots\ldots, t_{m3}]^T$ (since $\boldsymbol{t}_1 = 0$) and can therefore be solved by assigning to $\boldsymbol{T}$ the eigenvector corresponding to the smallest eigenvalue of the matrix $Q^T Q$ (alternatively use SVD). Thus, given only an estimate of the Rs we have been able to formulate an estimate of the $\boldsymbol{t}$s – again, using all of the available data simultaneously.

1.3.8 The iterative calibration scheme

Having worked out all of the necessary steps in the previous sections, we are now in a position to outline the iterative scheme by which the external calibration is carried out.

1. Guess an initial set of Rs given only the data (use standard 2-camera algorithms)
2. Estimate a set of $\boldsymbol{t}$s given these Rs
3. Estimate the world points $\{\mathbf{X}_i\}$ given these Rs and $\boldsymbol{t}$s
4. Estimate the $\{X_{pq3}\}$s given all of the above
5. Obtain a new estimate of the Rs using values from (2),(3),(4) and start the next iteration by returning to step (2).

In practice each step of the procedure can be performed quickly and convergence is achieved within a few tens of iterations. In estimating the $\boldsymbol{t}$s we should note that we are only able to do this up to scale. One may therefore set a value to unity (say $\boldsymbol{t}_2 \cdot \boldsymbol{e}_3$) and evaluate the other values relative to this – when doing this however, checks must be made that the signs of the estimated $\boldsymbol{t}$s do not produce negative depths (if they do, we will need to take $\boldsymbol{t}_2 \cdot \boldsymbol{e}_3 = -1$).

The above external calibration routine requires a very simple initial data gathering stage (waving a single point over a volume representative of where the world points will be) and utilises all of the image data simultaneously in order to produce optimal estimates of the relative rotations and translations of the cameras. In addition the formula for reconstruction given in equation (1.17) is very simple and gives accurate and robust 3D reconstructions. The value of the cost function (S_2) can also be monitored throughout the iterations; a final value of S_2 which is too large is usually indicative of poor data and a new calibration should be performed.

1.4 Examples and Results

In order to illustrate this calibration procedure we will present some results on both simulated and real data. While the procedure is routinely used in the tracking and subsequent reconstruction of *real* motion capture data, a quantitative evaluation of its behaviour is more easily obtained from simulations. The real data presented attempts to evaluate the performance of the calibration by checking that rays from the image planes, from which we reconstruct the world point, do indeed cross approximately at a single point in space. Real multiple camera data together with example reconstructions can be downloaded from `http://www.sig-proc.eng.cam.ac.uk/vision`.

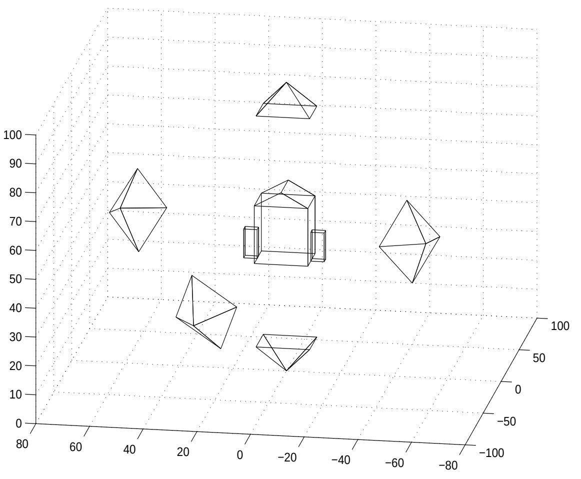

FIGURE 1.5. Wireframe house (26 points) viewed from 5 cameras – the optical centre and 4 defining points of the image planes are shown. The position and orientation of the cameras are such that the house is in view in each camera.

We use 5 cameras, the first camera placed with its optical centre at the origin $[0, 0, 0]$ and with its optical axis along the z-direction, viewing a wireframe house which is placed about 50 units in z away from the origin. Cameras 2 to 4 are rotated and translated from camera 1 as shown in figure 1.5.

R_j is the rotor which takes the frame at camera 1 to the frame at camera j, and the axes, $\hat{\boldsymbol{n}}_j$, and angles, θ_j, which characterise R_j (since $R_j = \exp -\frac{I\hat{\boldsymbol{n}}_j\theta_j}{2}$) are given in table 1.

Rotor	$\hat{\boldsymbol{n}}$	θ	$\boldsymbol{t}$
R_2	$[0.7071, -0.7071, 0]$	$51.498°$	$[40, 40, 5]$
R_3	$[0.7593, -0.6508, 0]$	$83.810°$	$[60, 70, 40]$
R_4	$[-0.7071, 0.7071, 0]$	$96.721°$	$[-60, -60, 60]$
R_5	$[-1, 0, 0]$	$180°$	$[0, 0, 100]$

Table 1 Table showing true values of the rotations and translations of the cameras

In order to calibrate the cameras we used 30 points generated at random from a cube centred at $[0, 0, 50]$ with side length 30 – these points simulated the calibration process whereby one bright marker is moved around the scene over a number of frames. Here we will assume that each of the 30 points is visible in all cameras. The 30 points were projected into the 5 cameras and the image points from each image plane were the only data given to the calibration routines. In the image planes Gaussian noise was added. Three different levels of noise were tested having standard deviation, $\sigma = 0.001, 0.005, 0.01$ – with the image plane coordinates ranging roughly from -0.45 to +0.45, at a resolution of 1000×1000 this would correspond to standard deviations ranging from 1 pixel to 10 pixels. To initialise the algorithm, an initial set of Rs and $\boldsymbol{t}$s were found by taking two cameras at a time and performing some simple method to determine the parameters, e.g. the algorithm of Weng *et al.* [12] – call these R_k^0 and $\boldsymbol{t}_k^0$. 20 iterations of the algorithm were allowed in each case, although generally fewer were needed to achieve adequate convergence. Let the final estimated values be R_k^f and $\boldsymbol{t}_k^f$.

Using R_k^f and $\boldsymbol{t}_k^f$ we can then reconstruct the wireframe house. We use a realistic set of data which consists of the image points in each camera of those points from the house that were visible in that camera (i.e. we include the relevant occlusion field). For the case depicted in figure (1.5), we can see, for example, that the uppermost camera will not see any of the vertices on the lower side of the house. For these simulations it was the case that every vertex was visible in at least two cameras. Also the same data and occlusion field were used to perform the 3D reconstruction using the initial guesses R_k^0 and $\boldsymbol{t}_k^0$. The 3D reconstruction was carried out using equation (1.20) in both cases.

Figure (1.6) shows 6 different 3D views of the true wireframe house – the azimuth and elevation ([az,el], in degrees) of the viewpoint for each of the views is as follows (from top left to bottom right)

$$[-38, 30],\ [-15, 5],\ [-110, 20],\ [80, -25],\ [-90, 90],\ [-90, 0]$$

Figure (1.7) shows the reconstructions obtained for the case of added noise, $\sigma = 0.001$ – the left column shows the results from the iterative scheme (20 iterations), while the right column shows the results for reconstruction from the two-camera estimates. The top, middle and bottom views have azimuth and elevation as for the left column of figure (1.6). Figures (1.8) and (1.9) show similar plots for $\sigma = 0.005$ and $\sigma = 0.01$. We see that with little noise the reconstruction is very good for both cases. However, as the noise gets more severe, we see that the iterative scheme tends to give better reconstructions. Even under higher noise levels the reconstruction remains acceptable.

As well as comparing the reconstructions it is also instructive to see how the estimated rotors compare with the true rotors in each of the above cases. If a rotor R, is written as $R = \exp(-I\hat{\boldsymbol{n}}\theta/2)$, then the bivector describing the rotation is $I\hat{\boldsymbol{n}}\theta/2$, so that a good way of comparing rotors is to compare the bivector components: i.e. $n_1\theta,\ n_2\theta,\ n_3\theta$, with $n_i = \hat{\boldsymbol{n}}\cdot\boldsymbol{e}_i$. Figure (1.10) compares these components for the true rotors, and the two sets of rotors described above for four noise values, $\sigma = 0.001, 0.005, 0.007, 0.01$. Similar comparisons for the translations are shown in figure (1.11).

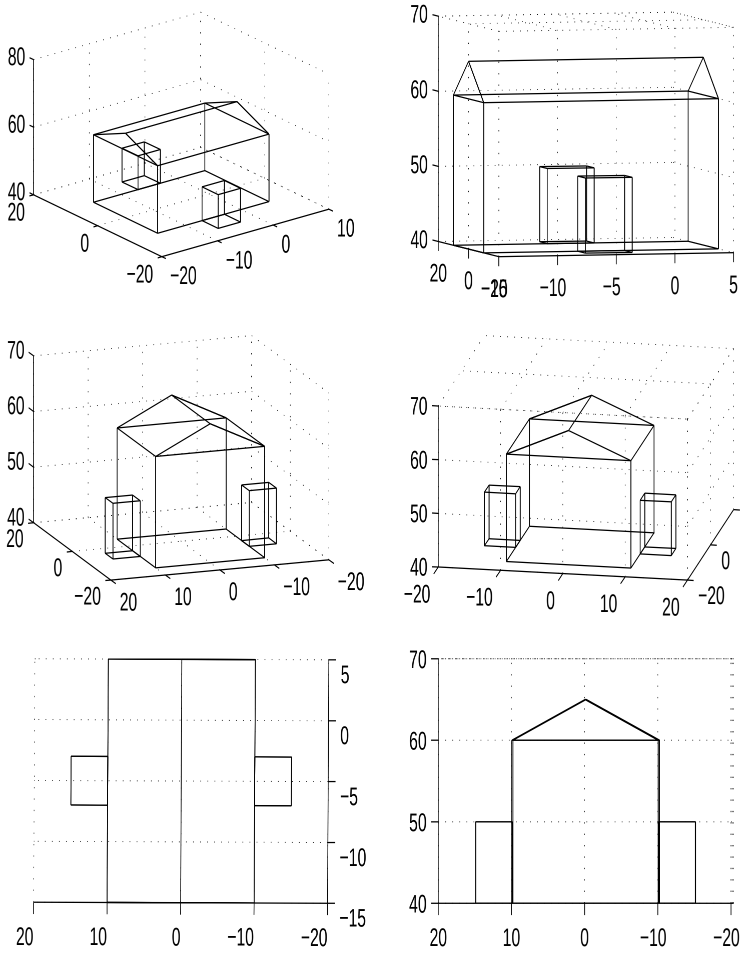

FIGURE 1.6. Six views of the true vertices of the simulated house

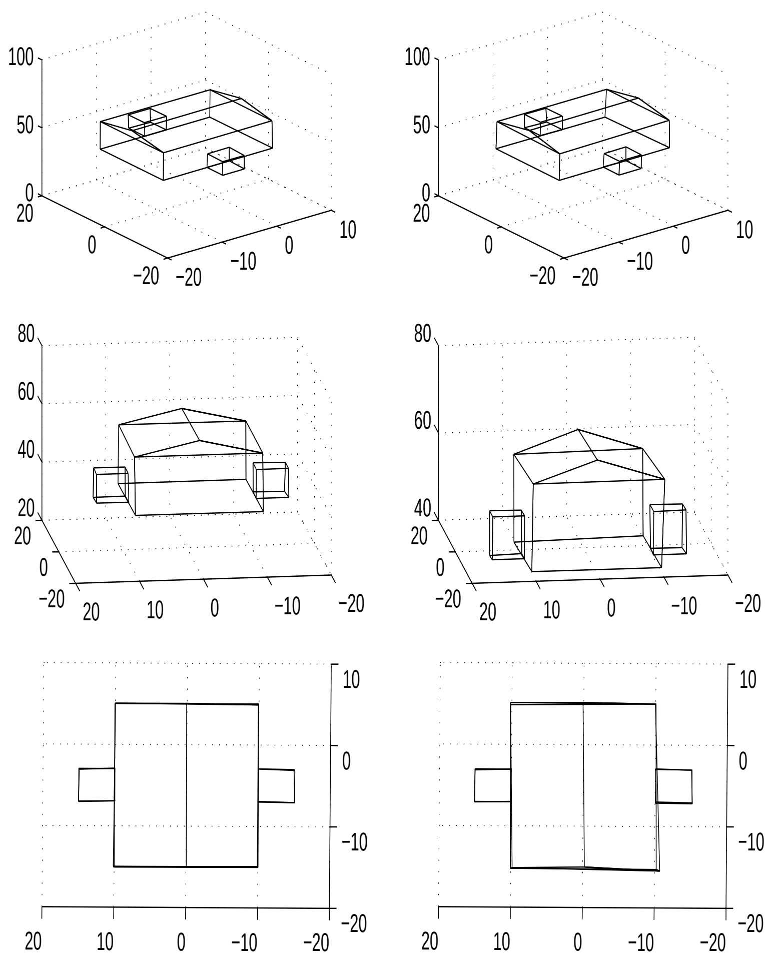

FIGURE 1.7. Results of the reconstruction with $\sigma = 0.001$. The left column shows results of iterative algorithm; right column shows results from taking two-camera estimates.

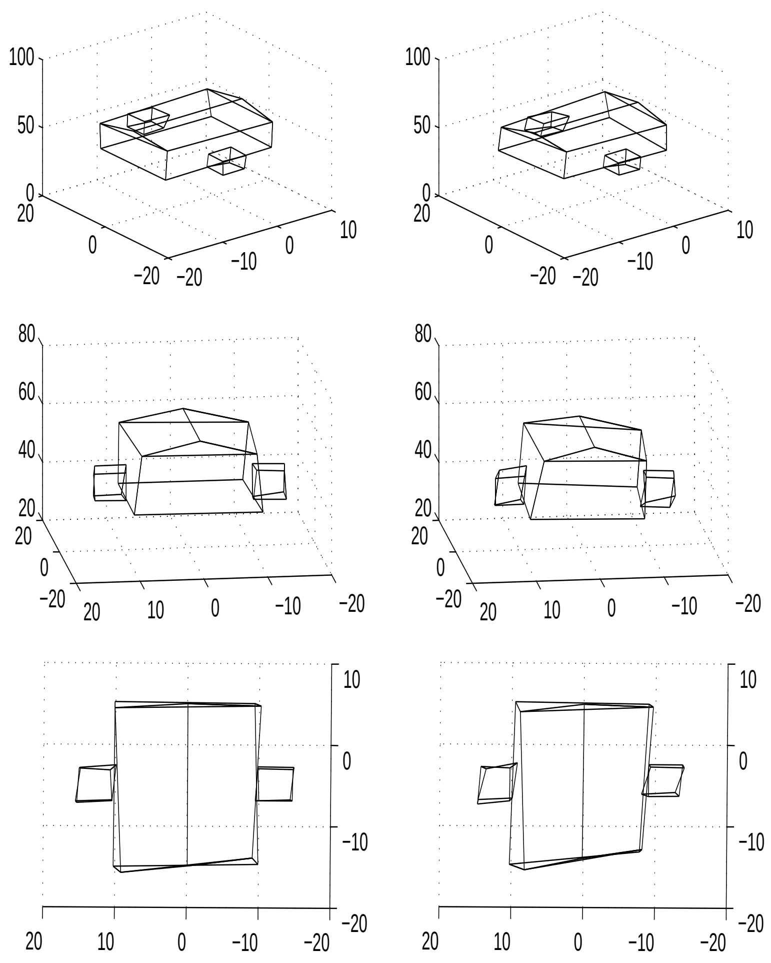

FIGURE 1.8. Results of the reconstruction with $\sigma = 0.005$. The left column shows results of iterative algorithm; right column shows results from taking two-camera estimates.

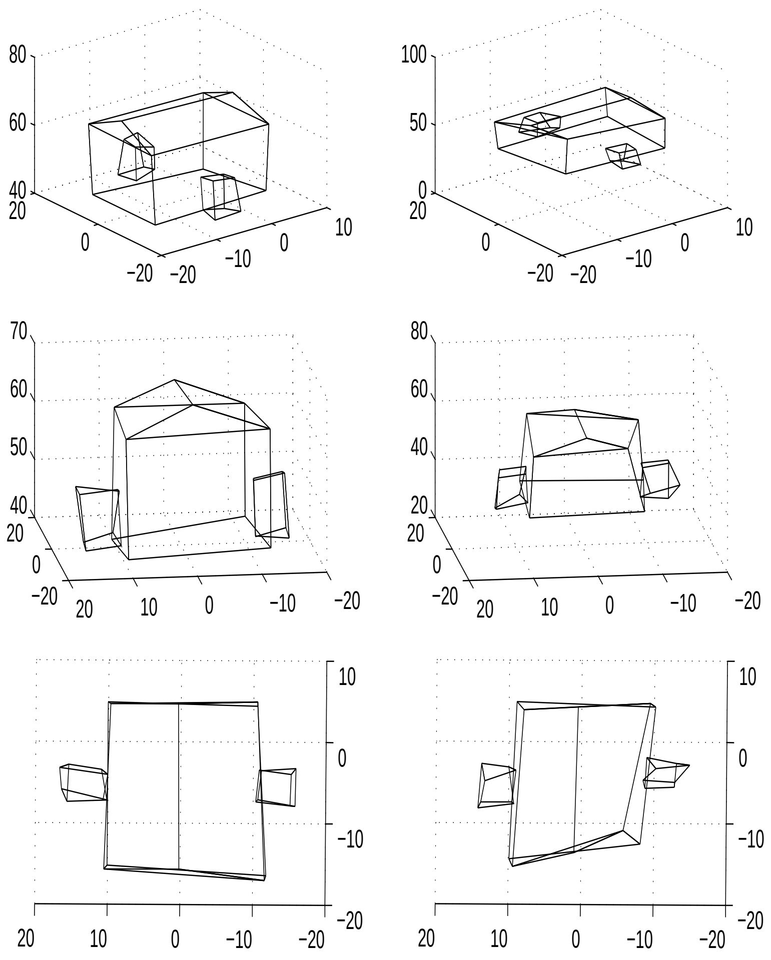

FIGURE 1.9. Results of the reconstruction with $\sigma = 0.01$. The left column shows results of iterative algorithm; right column shows results from taking two-camera estimates.

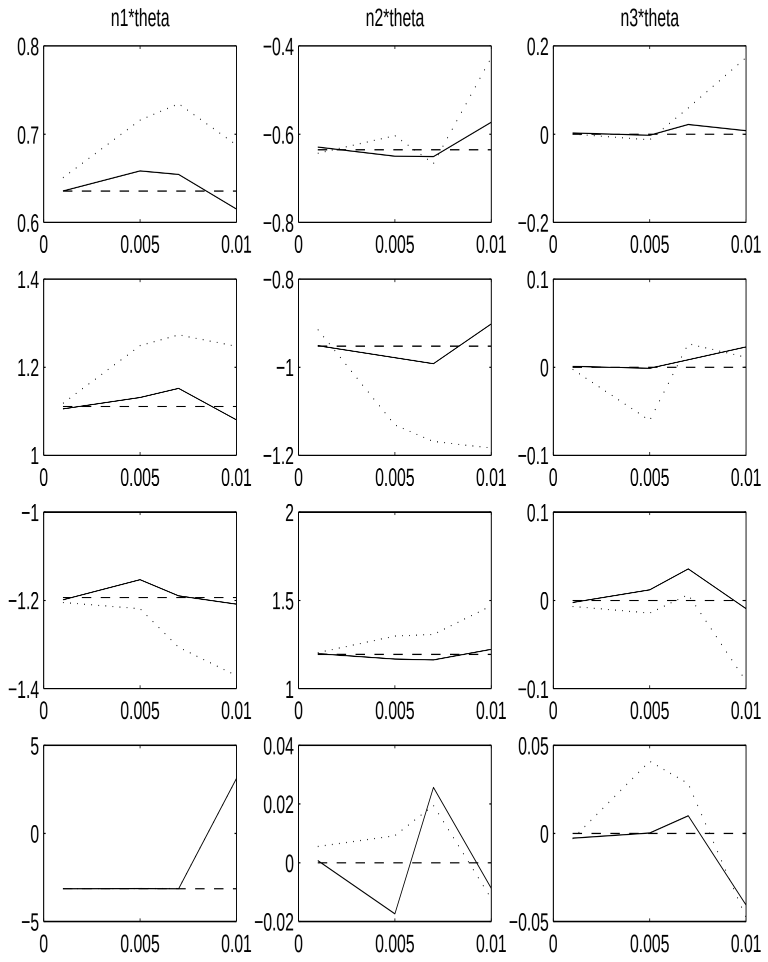

FIGURE 1.10. Moving left to right in columns shows results for $n_1\theta$, $n_2\theta$, $n_3\theta$, while moving down rows from top to bottom shows results for R_2, R_3, R_4, R_5. In each plot the dashed line gives the true value of the bivector component, the solid line gives the bivector component from the iterative algorithm and the dotted line gives that from the two-camera estimate. The x-axis in each case gives the standard deviation of the Gaussian noise added.

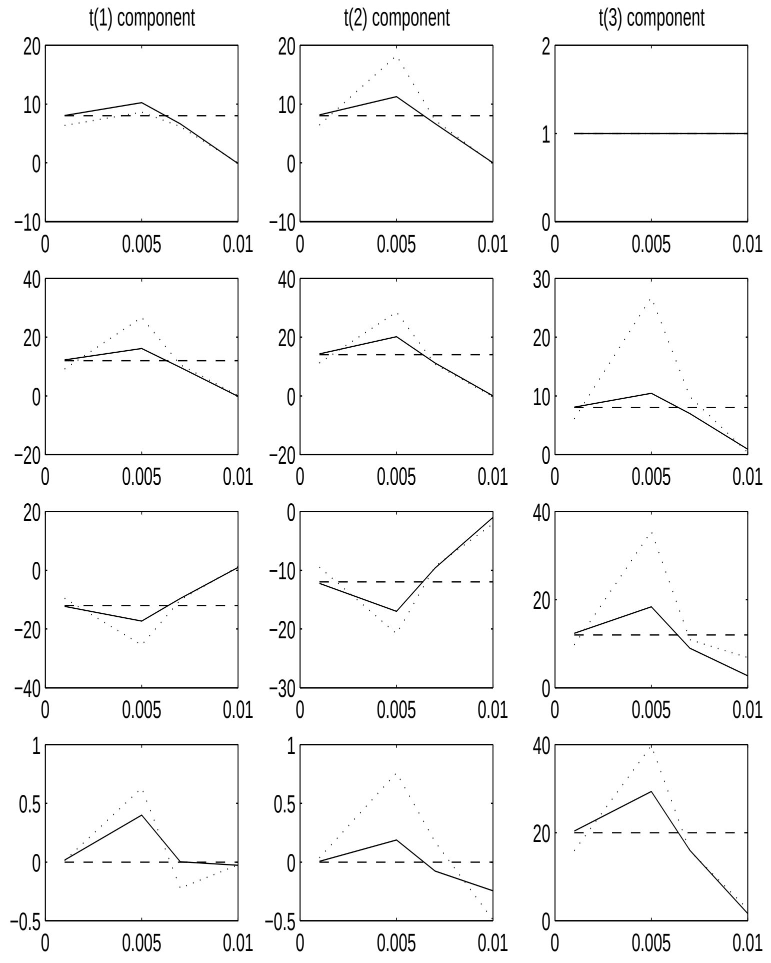

FIGURE 1.11. Moving left to right in columns shows results for the first, second and third components of the translation vectors, while moving down rows from top to bottom shows results for cameras 2 to 4. In each plot the dashed line gives the true value of the translation component, the solid line gives the component from the iterative algorithm and the dotted line gives that from the two-camera estimate. The x-axis in each case gives the standard deviation of the Gaussian noise added. Note that the translations are normalised so that $t_2{\cdot}e_3 = 1$, hence the graph in the upper right.

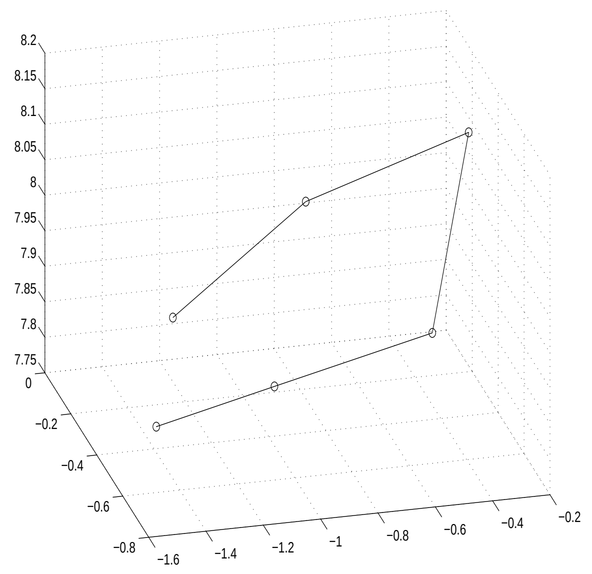

FIGURE 1.12. 3D snapshot at frame 3 of the shoulders, elbows and wrists of the golfer

In order to show the performance of the calibration algorithms on real data we used three cameras to take a sequence of 300 frames of a person performing a golf swing, with markers placed on shoulders, elbows and wrists. The cameras were calibrated prior to taking the data by waving a single bright marker over a representative volume and applying the algorithms outlined in section 1.3. Figure 1.12 shows an example of the reconstruction by showing the linked points for frame 3 of the sequence. Although this plot does not tell us much without detailed information of the real subject, figure 1.13 gives some idea of the accuracy of the calibration by plotting the rays from the matching image points (four such points were taken) through the optical centres of the cameras. The positions of the cameras are obtained from the calibration. If the calibration is good, we would expect all matching image points to intersect more or less at a single point in space. From figure 1.12 we can see that this is indeed the case for the particular frame chosen, and is also the case throughout the rest of the sequence.

We can see that on the whole, the iterative algorithm described in this paper produces good estimates of the bivectors and of the translations over a wide range of noise cases. The two-camera estimates that we have compared the algorithm with are, of course, not something that would be routinely used in practice. However, most calibration schemes would start with some such estimate and generally proceed via non-linear minimization. Such minimizations use gradient descent methods and as such are

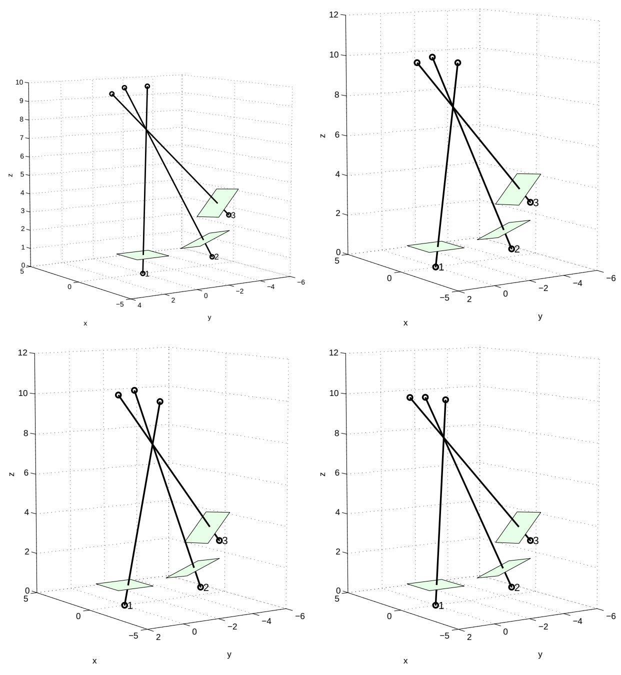

FIGURE 1.13. Reconstructed rays of 4 points (shoulders, one elbow and one wrist) in randomly chosen frames from a 300 frame sequence of a golf swing. The reconstruction was carried out using calibration data determined by the iterative scheme described in section 1.3

crucially dependent on the initial guess as they will tend to find the local minimum in the vicinity of this initial guess. Other methods of calibration involve building up the external calibration parameters camera by camera; such methods have to ensure that the final estimates are independent of the particular order of estimation and form a self-consistent set. Some calibration schemes in the literature are given in [11, 2], however, code is generally not available to compare such algorithms with those discussed here.

1.5 Extending to include internal calibration

The discussion in this paper has assumed that we have the internal calibration of the cameras. Typically, for the motion capture system, this is done every few weeks or so, and the values are assumed not to change significantly on this timescale. However, it is possible to adjust the algorithm presented here to include determination of the internal parameters. If we return to equation 1.3, but replace $\boldsymbol{x}_{ij}$ by $\boldsymbol{u}_{ij}$,

$$\boldsymbol{u_{ij}} = f_j(\boldsymbol{x}_{ij}) \equiv C_j \boldsymbol{x}_{ij} \tag{1.30}$$

where the linear function f_j represents the 3×3 camera matrix C_j, our cost function S_2 in terms of the observations $\boldsymbol{u}$ and the internal calibration, becomes

$$S_2 = \sum_{j=1}^{m} \sum_{i=1}^{N} \left[X_{ij3} f_{cj}(\boldsymbol{u}_{ij}) - \tilde{R}_j(\boldsymbol{X}_i - \boldsymbol{t}_j) R_j \right]^2 O_{ij} \tag{1.31}$$

where $f_{cj} \equiv f_j^{-1}$. We can now minimize over the $\{f_{cj}\}$ as well as the other parameters using the ability in GA to carry out functional differentiation. One must note, however, that the f_cs take a particular form (which can be made equivalent to an upper triangular matrix), so this constraint must be allowed for. A detailed description of this self-calibration procedure will be presented elsewhere.

1.6 Conclusions

A means of determining the external calibration parameters (relative rotations and translations) for any number of cameras observing a scene has been presented. Using geometric algebra to differentiate with respect to the the unknowns in the problem, we are able to build up an iterative estimation scheme. In the process, we also produce an efficient and robust reconstruction algorithm which can be used for estimating the world points once the calibration has been achieved. The method is essentially a maximum likelihood technique in which we substitute maximum likelihood estimators in order to eliminate the parameters we do not want to estimate (e.g. the world points and the $\{X_{ij3}\}$s). Another technique which can be employed is a Bayesian approach, which marginalises over these parameters (nuisance parameters) prior to estimating the Rs and $\boldsymbol{t}$s – a review of the geometric algebra approach to this procedure is given in this volume **??**. Indeed, if the parameters in question have a multivariate Gaussian distribution then the two techniques should give the same results. Preliminary tests indicate that, even though the noise is unlikely to be multivariate Gaussian in real data, the two approaches produce very similar results on good data.

The calibration scheme presented here is currently used on an optical motion capture system. The algorithms are used with data from a single moving marker to produce the external calibration. This calibration is then used in the tracking and reconstruction of subsequent data taken from the subject. The algorithm is relatively quick, robust and is easily effected, meaning that the cameras can be moved and the system speedily recalibrated.

In summary, we have presented a technique for external camera calibration which used the ease of expressing geometric entities in geometric algebra and the ability to differentiate with respect to any element of the algebra. Using rotors provides a very efficient way of optimizing over a rotation manifold; it is a minimally parameterized system, does not have the singularities associated with Euler angles and is less cumbersome and more easily extendable than quaternions (in the sense that rotors can rotate any geometric object, not just vectors and have the same form in any dimension). The results presented here can be used alone or used to initialize algorithms which employ minimization techniques and different cost functions. The intermediate steps of determining the best estimate of the world points from known data points and given calibration, and of determining the relative translations between cameras given the rotations and data points, are also useful in many reconstruction and tracking scenarios.

Acknowledgements
JL would like to thank the Royal Society of London for their support in the form of a University Research Fellowship. AXSS acknowledges the support of the EPSRC in the form of a Studentship. The authors would like to thank Ken Yam and Anthony Lasenby for useful comments on the text, Maurice Ringer for the tracking of the golf data and the referees for helpful suggestions.

1.7 References

[1] E.D. Bayro-Corrochano and J. Lasenby. Geometric techniques for the computation of projective invariants using n uncalibrated cameras. Proceedings of *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, New Delhi, India, pp. 95-100. 21,22 Dec. 1998.

[2] P.E. Debevec, C.J. Taylor, J. and Malik. Facade: Modeling and Rendering Architecture from Photographs. *SIGGRAPH96*. 1996
Also, see: `http://www.cs.berkeley.edu/~debevec/Research`

[3] C.J.L. Doran and A.N. Lasenby. Lecture Notes to accompany 4th year undergraduate course on *Physical Applications of Geometric Algebra*. 2000. Available at `http://www.mrao.cam.ac.uk/~clifford/ptIIIcourse/`.

[4] L. Dorst, S. Mann and T. Bouma. GABLE: A Matlab Tutorial for Geometric Algebra. 2000. Available at `http://www.carol.wins.uva.nl/~leo/clifford/gablebeta.html`.

[5] D. Hestenes. New Foundations for Classical Mechanics – second edition. *D. Reidel*, Dordrecht. 1999.

[6] D. Hestenes and G. Sobczyk, Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics. *D. Reidel*, Dordrecht. 1984.

[7] J. Lasenby, W.J. Fitzgerald, A.N. Lasenby and C.J.L. Doran. New geometric methods for computer vision – an application to structure and motion estimation. *International Journal of Computer Vision*, 26(3), 191-213. 1998.

[8] J. Lasenby and A.N. Lasenby. Estimating tensors for matching over multiple views. *Phil.Trans.Roy.Soc.-A*, 356, 1267-1282. Ed. J. Lasenby, A. Zisserman, H.C. Longuet-Higgins and R. Cipolla. 1998.

[9] J. Lasenby and E.D. Bayro-Corrochano. Analysis and Computation of Projective Invariants from Multiple Views in the Geometric Algebra Framework. *International Journal of Pattern Recognition and Artificial Intelligence – Special Issue on Invariants for Pattern Recognition and Classification*, Vol.13, No.8, 1105-1121. Ed. M.A. Rodrigues. 1999.

[10] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293: 133–138. 1981.

[11] D.P. Robertson and R. Cipolla. Photobuilder. 2000. available at `http://svr-www.eng.cam.ac.uk/photobuilder/`.

[12] J. Weng, T.S. Huang and N. Ahuja. Motion and Structure from Two Perspective Views: Algorithms, Error Analysis and Error Estimation. *IEEE Trans.Pattern Anal.Mach.Intelligence*, 11(5): 451–476. 1989.