# Bayesian inference and geometric algebra: an application to camera localization

AUTHORS
Chris Doran

In E. Bayro and G. Sobczyk eds.
*Geometric algebra: a geometric approach to computer vision, neural and quantum computing, robotics and engineering*
Birkhauser, 172 (2000)

## Abstract

Geometric algebra provides a number of powerful tools for the treatment of rotations in three dimensions. Rotations are parameterised by rotors, which are normalised multivectors in a 4-d subalgebra of the 3-d geometric algebra. This parametrisation can be exploited to simplify both extrapolation and optimisation problems involving rotations. A number of these are considered in the context of computer vision.

## 1 Introduction

Geometric algebra is an extremely powerful language for solving complex geometric problems in engineering [4, 8]. Its advantages are particularly clear in the treatment of rotations. Rotations of a vector are performed by the double-sided application of a *rotor*, which is formed from the *geometric product* of an even number of unit vectors. In three dimensions a rotor is simply a normalised element of the even subalgebra of $\mathcal{G}_3$, the geometric algebra of three dimensional space. In this paper we are solely interested in rotations in space, and henceforth all reference to rotors can be assumed to refer to the 3-d case. Rotors have a number of useful features. They can be easily parameterised in terms of the bivector representing the plane of rotation. Their product is a very efficient way of computing the effect of compound rotations, and is numerically very stable.

Rotors are normalised elements in a 4-d algebra (the even subalgebra of $\mathcal{G}_3$), so they can be represented by points on the unit sphere in 4-d. This is called a 3-sphere, and is the rotor *group manifold* [2, 5]. The simple structure of this manifold makes it very easy to extrapolate between rotations, which is useful in many fields including finite element analysis and rigid body dynamics. The extrapolation method can be easily understood in terms of relaxing the normalisation constraint and working with unnormalised rotors, and normalising the result at the end of a computation. This is also the key to simplifying the problem of differentiating with respect to rotations. Ordinarily, a function of a rotation is viewed as taking its value on the group manifold. Derivatives of this function take their values in the tangent space to the group manifold. This is mathematically rigorous, but rather cumbersome computationally. A better idea is to move off the group manifold and work in the 4-d linear space, where the rules of calculus are much simpler [3, 4, 8]. Used properly, this trick can significantly simplify optimisation problems involving rotations.

The main applications considered here are to variations of the camera localization problem in computer vision [7, 8, 10, 11, 13, 14]. Suppose that a number of cameras are placed in unknown positions and they observe the same scene. In order to reconstruct the scene, we need to determine the relative positions and orientations of the cameras. Given a sufficient number of point matches between the cameras, this information can be accurately recovered without any external measurements. For most cases this problem can be reduced to a least squares minimisation over a set of rotations and translations, and this can be simplified considerably using the techniques of rotor calculus. The least squares likelihood functions used here are derived from a simple Bayesian probabilistic model, which helps to expose some of the underlying assumptions in the choice of likelihood function [12]. This is useful in pointing the way to constructing improved models. In this paper we assume a projective camera model, and will further assume that the internal camera parameters are all known. A preliminary discussion of how geometric algebra can be used to estimate these internal parameters is contained in [9]. The basic techniques described here can be generalised in a number of ways to deal with more complex situations and at various points we discuss how one might exploit this. In particular, the extension beyond two cameras is straightforward. This is an area where more traditional tensor-based approaches run into difficulties.

## 2 Geometric algebra in three dimensions

The geometric algebra of three-dimensional space is generated by a right-handed orthonormal set of vectors $\{e_1, e_2, e_3\}$. Their geometric product satisfies

$$e_i e_j = \delta_{ij} + I \epsilon_{ijk} e_k \tag{2.1}$$

where $I$ is the pseudoscalar

$$I = e_1 \wedge e_2 \wedge e_3 = e_1 e_2 e_3. \tag{2.2}$$

The full algebra is spanned by

$$\begin{array}{cccc} 1 & \{e_i\} & \{Ie_i\} & I \\ 1 \text{ scalar} & 3 \text{ vectors} & 3 \text{ bivectors} & 1 \text{ trivector.} \end{array} \tag{2.3}$$

The dot and wedge symbols have their usual meaning as inner and outer products, and for vectors

$$a \cdot b = \frac{1}{2}(ab + ba) \qquad a \wedge b = \frac{1}{2}(ab - ba). \tag{2.4}$$

The geometric product for general multivectors is denoted simply by juxtaposition, and throughout inner and outer products take precedence over geometric products. Angled brackets $\langle \, \rangle_n$ are used for the projection onto grade operation, and the scalar part of a multivector $A$ is denoted simply by $\langle A \rangle$. The scalar part satisfies the cyclic reordering property

$$\langle AB \cdots C \rangle = \langle B \cdots CA \rangle. \tag{2.5}$$

The *reverse* of a multivector is formed by reversing the order of geometric products of vectors in the multivector and is denoted with a tilde. An arbitrary multivector $M$ can be decomposed as

$$M = \alpha + a + B + \beta I, \tag{2.6}$$

where $\alpha$ and $\beta$ are scalars, $a$ is a vector and $B$ is a bivector. The reverse of $M$ is

$$\tilde{M} = \alpha + a - B - \beta I. \tag{2.7}$$

## 3 Rotors and Rotations

A *rotor* is a normalised element of the even subalgebra,

$$R = \alpha + B, \tag{3.8}$$

where $\alpha$ is a scalar and $B$ is a bivector. The normalisation condition is that

$$R\tilde{R} = \tilde{R}R = \alpha^2 - B^2 = 1. \tag{3.9}$$

Rotors generate rotations of vectors via the double-sided transformation law

$$a \mapsto a' = Ra\tilde{R}. \tag{3.10}$$

This same law holds for bivectors, since

$$\begin{aligned} (Ra\tilde{R}) \wedge (Rb\tilde{R}) &= \tfrac{1}{2}\Big(Ra\tilde{R}Rb\tilde{R} - Rb\tilde{R}Ra\tilde{R}\Big) \\ &= \tfrac{1}{2}R(ab - ba)\tilde{R} \\ &= R\, a \wedge b\, \tilde{R}. \end{aligned} \tag{3.11}$$

It is also simple to check that rotors leave inner products invariant,

$$a' \cdot b' = \langle Ra\tilde{R}Rb\tilde{R} \rangle = \langle ab \rangle = a \cdot b. \tag{3.12}$$

The rotor transformation law $a \mapsto Ra\tilde{R}$ also leaves trivectors invariant, so has determinant $+1$ and must be a rotation.

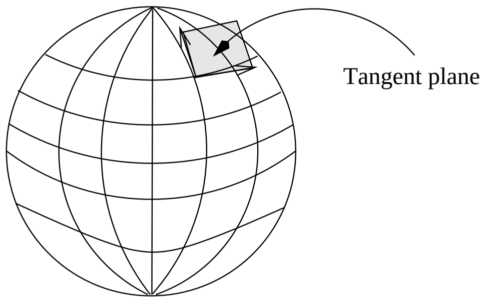

Figure 1: *Tangent Space.* At each point on the sphere one can attach a tangent plane.

Rotors can be parameterised directly in terms of the plane of rotation by writing

$$R = \exp(-B/2). \tag{3.13}$$

The rotor R now generates a rotation through an angle $|B|$ in the plane specified by $B$, with the same orientation as $B$. In three dimensions we can also write

$$R = \exp(-\theta I\hat{n}/2) \tag{3.14}$$

where $\theta = |B|$, and $\hat{n} = -IB/|B|$ is the unit vector representing the rotation axis. The map between a vector $n$ and the bivector $In$ is called a *duality transformation*. Bivectors can only be dualised to vectors in three dimensions, so the concept of an axis of rotation only exists for three-dimensional space.

## 3.1 The Group Manifold

Rotors are elements of a four-dimensional space, normalised to 1. They can be represented as points on a *3-sphere* — the set of unit vectors in four dimensions. This is the rotor *group manifold*. At any point on the manifold, the *tangent space* is three-dimensional. This is the analog of the tangent plane to a sphere in three dimensions (see Figure 1).

Rotors require three parameters to specify them uniquely. One common parameterisation is in terms of the *Euler angles* $(\theta, \phi, \psi)$,

$$R = \exp(-e_1e_2\phi/2)\exp(-e_2e_3\theta/2)\exp(-e_1e_2\psi/2). \tag{3.15}$$

But often it is more convenient to use the set of bivector generators, with

$$|B^2| \leq \pi. \tag{3.16}$$

The rotors $R$ and $-R$ generate the *same* rotation, because of their double-sided action. It follows that the *rotation* group manifold is more complicated than the rotor group manifold — it is a projective 3-sphere with points $R$ and $-R$ identified. This is one reason why it is usually easier to work with rotors.

## 3.2 Extrapolating Between Rotations

Suppose we are given two estimates of a rotation, $R_0$ and $R_1$, how do we find the mid-point? With rotors this is remarkably easy! We first make sure sure they have smallest angle between them in four dimensions. This is done by ensuring that

$$\langle R_0 \tilde{R}_1 \rangle = \cos\theta > 0. \tag{3.17}$$

If this inequality is not satisfied, then the sign of one of the rotors should be flipped. The 'shortest' path between the rotors on the group manifold is defined by

$$R(\lambda) = R_0 \exp(\lambda B), \tag{3.18}$$

where

$$R(0) = R_0, \quad R(1) = R_1. \tag{3.19}$$

It follows that we can find $B$ from

$$\exp(B) = \tilde{R}_0 R_1. \tag{3.20}$$

The path defined by $\exp(\lambda B)$ is an invariant construct. If both endpoints are transformed, the path transforms in the same way. The midpoint is

$$R_{1/2} = R_0 \exp(B/2), \tag{3.21}$$

which therefore generates the midpoint rotation. This is quite general — it works for any rotor group (or any *Lie group*). For rotations in three dimensions we can do even better. $R_0$ and $R_1$ can be viewed as two unit vectors in a four-dimensional space. The path $\exp(\lambda B)$ lies in the plane specified by these vectors (see Figure 2).

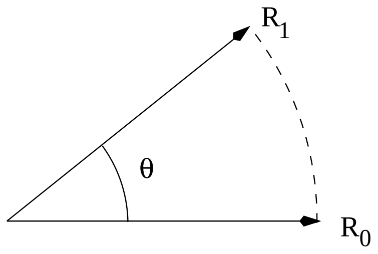

Figure 2: *The path between two rotors* The rotors can be treated as unit vectors in four dimensions. The path between them lies entirely in the plane of the two rotors, and therefore defines a segment of a circle.

The rotor path between $R_0$ and $R_1$ can be written as

$$R(\lambda) = R_0\Big(\cos\lambda\theta + \sin\lambda\theta\,\hat{B}\Big), \tag{3.22}$$

where we have used $B = \theta\hat{B}$. But we know that

$$\exp(B) = \cos\theta + \sin\theta\,\hat{B} = \tilde{R}_0 R_1. \tag{3.23}$$

It follows that

$$\begin{aligned} R(\lambda) &= \frac{R_0}{\sin\theta}\Big(\sin\theta\,\cos\lambda\theta + \sin\lambda\theta(\tilde{R}_0R_1 - \cos\theta)\Big) && (3.24)\\ &= \frac{1}{\sin\theta}\Big(\sin(1-\lambda)\theta\,R_0 + \sin\lambda\theta\,R_1\Big), && (3.25)\end{aligned}$$

which satisfies $R(\lambda)\tilde{R}(\lambda) = 1$ for all $\lambda$. The midpoint rotor is therefore simply

$$R_{1/2} = \frac{\sin(\theta/2)}{\sin\theta}(R_0 + R_1). \tag{3.26}$$

This gives us a remarkably simple prescription for finding the midpoint: *add the rotors and normalise the result.* By comparison, the equivalent matrix is quadratic in $R$, and so is much more difficult to express in terms of the two endpoint rotation matrices.

Suppose now that we have a number of estimates for a rotation and wanted to find the average. Again the answer is simple. First one chooses the sign of the rotors so that they are all in the 'closest' configuration. This will normally be easy if the rotations are all roughly equal. If some of the rotations are quite different then one might have to search around for the closest configuration, though in these cases the average of such rotations is not a useful concept. Once one has all of the rotors chosen, one simply adds them up and normalises the result to obtain the average. This sort of calculation can be useful in computer vision problems where one has a number of estimates of the relative rotations between cameras, and their average is required.

The lesson here is that problems involving rotations can be simplified by working with rotors and relaxing the normalisation criteria. This enables us to work in a four-dimensional linear space and is the basis for a simplified calculus for rotations.

## 4 Rotor Calculus

Any function of a rotation can be viewed as taking its values over the group manifold. In most of what follows we are interested in scalar functions, though there is no reason to restrict to this case. The derivative of the function with respect to a rotor defines a vector in the tangent space at each point on the group manifold. The vector points in the direction of steepest increase of the function. This can all be made mathematically rigorous and is the subject of *differential geometry*. The problem is that much off this is over-complicated for the relatively simple minimisation problems encountered in computer vision. Working intrinsically on the group manifold involves introducing local coordinates (such as the Euler angles) and differentiating with respect to each of these in turn. The resulting calculations can be long and messy and often hide the simplicity of the answer.

Geometric algebra provides us with a more elegant and simpler alternative. We relax the rotor normalisation constraint and replace $R$ by $\psi$ — a general element of the even subalgebra. There is a very simple derivative operator associated with $\psi$. We first decompose $\psi$ in terms of the $\{e_i\}$ basis as

$$\psi = \psi_0 + \sum_{k=1}^{3} \psi_k I e_k \tag{4.27}$$

where the $\{\psi_0, \ldots, \psi_3\}$ are a set of scalar components. We now define the *multivector derivative* $\partial_\psi$ by

$$\partial_\psi = \frac{\partial}{\partial \psi_0} - \sum_{k=1}^{3} I e_k \frac{\partial}{\partial \psi_k}. \tag{4.28}$$

This derivative is independent of the chosen frame. It satisfies the basic result

$$\partial_\psi \langle \psi A \rangle = A \tag{4.29}$$

where $A$ is a constant, even-grade multivector. All further results for $\partial_\psi$ are built up from this basic result and Leibniz' rule for the derivative of a product.

The basic trick now is to re-write a rotation as

$$Ra\tilde{R} = \psi a \psi^{-1}. \tag{4.30}$$

This works because any even multivector $\psi$ can be written as

$$\psi = \rho^{1/2} R \tag{4.31}$$

where $R$ is a rotor, $\rho = \psi\tilde{\psi}$ and $\rho = 0$ if and only if $\psi = 0$. The inverse of $\psi$ is then

$$\psi^{-1} = \rho^{-1/2} \tilde{R} \tag{4.32}$$

so that

$$\psi\psi^{-1} = R\tilde{R} = 1. \tag{4.33}$$

The equality of equation (4.30) follows immediately. If one imagines a function over a sphere in three dimensions, one can extend this to a function over all space by attaching the same value to all points on each line from the origin. The extension $R \mapsto \psi$ does precisely this, but in a four dimensional space.

We are now able to differentiate functions of the rotation quite simply. The typical application is to a scalar of the type

$$(Ra\tilde{R}) \cdot b = \langle Ra\tilde{R}b \rangle = \langle \psi a \psi^{-1} b \rangle. \tag{4.34}$$

We now have

$$\partial_\psi \langle \psi a \psi^{-1} b \rangle = a\psi^{-1} b + \dot{\partial}_\psi \langle \psi a \dot{\psi}^{-1} b \rangle \tag{4.35}$$

where the overdot denotes the scope of the differential operator (*i.e.* the term being differentiated). We next require a formula for the inverse term. We start by letting $M$ be a constant multivector, and derive

$$0 = \partial_\psi \langle \psi \psi^{-1} M \rangle = \psi^{-1} M + \dot{\partial}_\psi \langle \psi \dot{\psi}^{-1} M \rangle. \tag{4.36}$$

It follows that

$$\dot{\partial}_\psi \langle \dot{\psi}^{-1} M \psi \rangle = -\psi^{-1} M. \tag{4.37}$$

But in this formula we can now let $M$ become a function of $\psi$, as only the first term, $\psi^{-1}$, is acted on by the differential operator. We can therefore replace $M$ by $M\psi^{-1}$ to obtain the useful formula

$$\dot{\partial}_\psi \langle \dot{\psi}^{-1} M \rangle = -\psi^{-1} M \psi^{-1}. \tag{4.38}$$

We can now complete the derivation started at (4.35) to find

$$\partial_\psi \langle \psi a \psi^{-1} b \rangle = a\psi^{-1} b - \psi^{-1} b \psi a \psi^{-1}. \tag{4.39}$$

It is convenient to premultiply this expression by $\psi$ to get

$$\psi \partial_\psi \langle \psi a \psi^{-1} b \rangle = \psi a \psi^{-1} b - b \psi a \psi^{-1} = 2(Ra\tilde{R}) \wedge b. \tag{4.40}$$

The fact that the *geometric product* is formed between $\psi$ and $\partial_\psi$ is important. This product is *invertible*, so no information is lost. The fact that a bivector is formed here is sensible. Bivectors belong to a three-dimensional space — the same number of dimensions as the tangent space to the group manifold. The big advantage of the approach used here is that one never leaves the geometric algebra of space, and the resultant bivector is evaluated in the same space, rather than in some abstract tangent space on the group manifold. The result (4.40) is also sensible if one thinks about varying $R$ in $(Ra\tilde{R}) \cdot b$ while keeping the vectors $a$ and $b$ constant. This function clearly has a maximum when $Ra\tilde{R}$ is parallel to $b$, which is precisely where the derivative vanishes.

This simple derivation turns out to be very useful in a range of applications, including rigid body dynamics and point-particle models for fermions. Here we have chosen to illustrate its use with some applications in computer vision.

## 5 Computer Vision

The main problem of interest in this paper is that of camera localization. Suppose that we have different camera views of the same scene. Given point matches with added noise, we want to find the relative translation and rotation between the cameras. Once the camera geometry has been calculated like this, it is possible to reconstruct the three-dimensional scene. Applications of this basic idea include

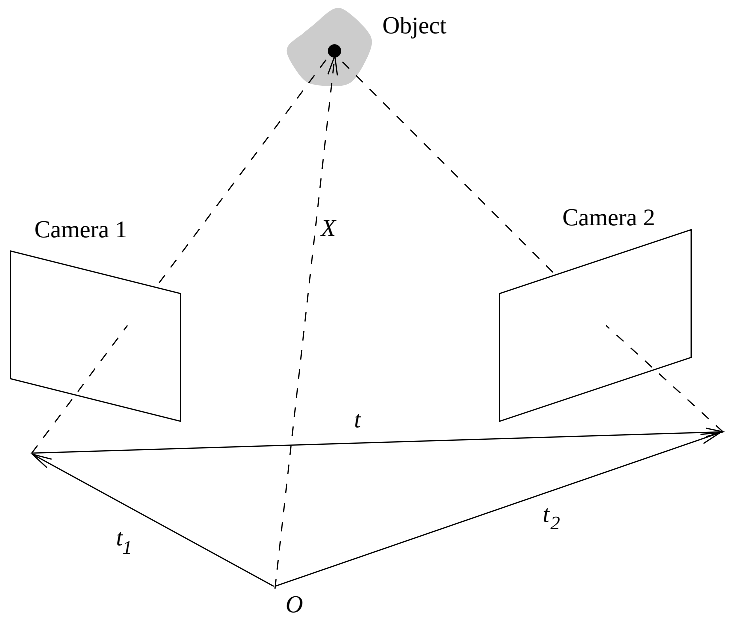

Figure 3: *The basic two camera setup.* The same object is viewed from two different directions. The cameras are related by a translation and a rotation. All vectors are expressed relative to some arbitrary origin $O$. The relative vector between the camera centres, $t = t_2 - t_1$, is independent of the origin.

fields such as motion analysis, reaching and neurocontrol, and robot control. Before studying the more realistic case of a projective camera model (see Section 6) we first study a simpler, toy problem whose solution is well known. This is the case where the full 3-d position is measured for the point matches, including the range data. This enables us to introduce some of the tools of Bayesian inference in a simplified setting.

## 5.1 Known Range Data

Suppose that we know the full three-dimensional coordinates of each point match (which is not very common in practice). The basic solution in this case is well known for the two camera case and has been discussed by many authors [1, 6, 7, 11]. The derivation presented here is slightly different, however, in being based on an underlying probabilistic model for the data, with the rotations and translation recovered via a Bayesian argument. Relative to an arbitrary origin, $O$, the camera centres are located at positions $t_1$ and $t_2$, and the point matches at positions $X^k$

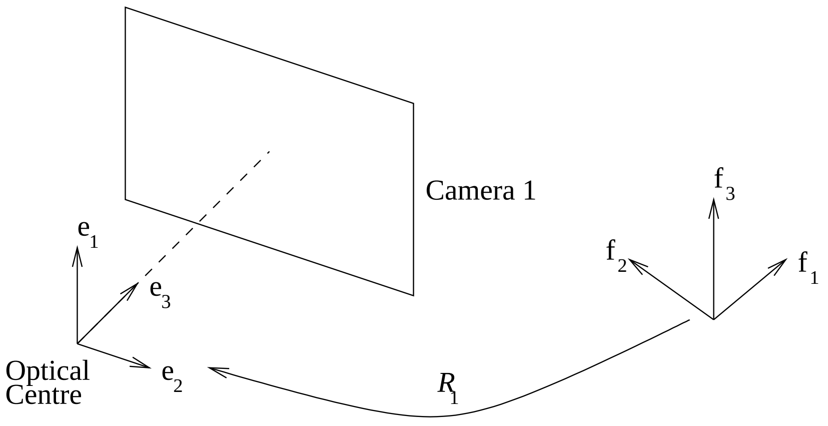

Figure 4: *The camera frame.* Each camera has a frame $\{e_i\}$ attached to it, with the 3-axis representing the optical axis. The camera frame is related to an arbitrary global frame $\{f_i\}$ by a rotor, with a separate rotor required for each camera. The rotor taking the camera 1 frame onto the camera 2 frame is then $R_2\tilde{R}_1$, and this is what we aim to find.

(see Figure 3). Throughout we use superscript indices to label the point matches, and subscript Latin indices to label frame vectors, $\{e_i\}$, or components of a vector, $x_i$. Which of these is intended should be obvious, as we only use $e_i$ and $f_i$ for frame vectors. At various points, subscript Greek indices are used to label the cameras.

If we write the two camera frames as $\{e_{1i}\}$ and $\{e_{2i}\}$ respectively, then the data we assume that we can record are a set of coordinates for the point matches,

$$x^k_{1i} = e_{1i}\cdot(X-t_1) \tag{5.41}$$

$$x^k_{2i} = e_{2i}\cdot(X-t_2). \tag{5.42}$$

We now introduce a third, arbitrary reference frame $\{f_i\}$, which is related to the two camera frames by

$$e_{1i} = R_1 f_i \tilde{R}_1, \quad e_{2i} = R_2 f_i \tilde{R}_2. \tag{5.43}$$

(See Figure 4). The advantage of working with separate rotors for the camera frames, instead of the mutual rotation between them, is that it keeps all formulae symmetric in the choice of frame, and ensures that the equations generalise easily to the $n$-camera case. This also provides a useful check on the formalism — we should only obtain equations for the mutual rotation between the camera frames, and not the absolute rotations between the camera frames and the $\{f_i\}$. In terms of storing and manipulating the data, everything is done in terms of the $\{f_i\}$ frame, which is usually chosen to coincide with the camera 1 frame. We next define the vectors

$$x_1^k = x_{1i}^k f_i, \quad x_2^k = x_{2i}^k f_i, \tag{5.44}$$

which should be related by

$$X^k = R_1 x_1^k \tilde{R}_1 + t_1 = R_2 x_2^k \tilde{R}_2 + t_2, \tag{5.45}$$

for all point matches $k$.

When we measure the position coordinates for a point match the measurements will be subject to various forms of noise due to discretisation (from the conversion to digital pixel coordinates), camera wobble, inexact point matches and many other effects. We will assume that all of this noise can be modeled with a simple Gaussian distribution, centred on the exact value. This is an enormous simplification and is almost certainly incorrect. The main advantage in assuming Gaussian noise is that the various marginalisation integrals can be performed analytically and usually return simple, least squares functions to minimise. The point of adopting a Bayesian framework is that these (often hidden) assumptions are brought out clearly. This in turn suggests various improvements which can lead to more accurate reconstruction.

Our assumed probability density function (pdf) is (ignoring the normalisation)

$$P(x_{1i}^k) \propto \exp\Big(\frac{-1}{2\sigma^2}\big(x_{1i}^k - e_{1i}\cdot(X^k - t_1)\big)^2\Big) \tag{5.46}$$

$$P(x_{2i}^k) \propto \exp\Big(\frac{-1}{2\sigma^2}\big(x_{2i}^k - e_{2i}\cdot(X^k - t_2)\big)^2\Big). \tag{5.47}$$

The pdf for the vector $x_1^k$ is therefore simply

$$P(x_1^k) \propto \exp\Big(\frac{-1}{2\sigma^2}(R_1 x_1^k \tilde{R}_1 + t_1 - X^k)^2\Big), \tag{5.48}$$

with a similar result holding for $x_2^k$. The full joint probability distribution over all point matches is therefore

$$\begin{aligned} &P(\{x_1^k, x_2^k\}|\{X^k\}, R_1, R_2, t_1, t_2) \propto \\ &\qquad \exp\Big(\frac{-1}{2\sigma^2}\sum_k (R_1 x_1^k \tilde{R}_1 + t_1 - X^k)^2 + (R_2 x_2^k \tilde{R}_2 + t_2 - X^k)^2\Big). \end{aligned} \tag{5.49}$$

Bayes' theorem [12] states that

$$P(X|Y,I) = \frac{P(Y|X,I) \times P(X|I)}{P(Y|I)} \propto P(Y|X,I) \times P(X|I). \tag{5.50}$$

This follows immediately from the product rule of probability theory. The final term $P(X|I)$ is called the *prior* and is chosen to reflect any knowledge we might have about the quantity to be determined prior to any measurements being made. In our case we have no such knowledge, so we assume uniform priors for the camera frames and centres, and for the positions of the point matches. We can therefore use Bayes' theorem to invert our pdf to obtain

$$P(\{x_1^k, x_2^k\}|\{X^k\}, R_1, R_2, t_1, t_2) \propto P(R_1, R_2, t_1, t_2, \{X^k\}|\{x_1^k, x_2^k\}), \tag{5.51}$$

where we continue to ignore normalisation factors. The next step is to *marginalise* over the actual positions $X^k$ to get the pdf for the rotors $R_i$ and positions $t_i$ in terms of the data. This marginalisation process is performed by simply integrating out the unwanted degrees of freedom,

$$\begin{aligned} &P(R_1, R_2, t_1, t_2|\{x_1^k, x_2^k\}) \\ &\qquad \propto \int d^3X^1\, d^3X^2 \cdots d^3X^n\, P(R_1, R_2, t_1, t_2, \{X^k\}|\{x_1^k, x_2^k\}). \end{aligned} \tag{5.52}$$

The marginalisation integrals are straightforward once one employs the result

$$(X-a)^2 + (X-b)^2 = 2\Big(X - \frac{1}{2}(a+b)\Big)^2 + \frac{1}{2}(a-b)^2. \tag{5.53}$$

All that remains after the integral is therefore

$$\begin{aligned} &P(R_1, R_2, t_1, t_2|\{x_1^k, x_2^k\}) \propto \\ &\qquad \exp\Big(\frac{-1}{2\sigma^2}\sum_k (R_1 x_1^k \tilde{R}_1 - R_2 x_2^k \tilde{R}_2 + t_1 - t_2)^2\Big). \end{aligned} \tag{5.54}$$

Maximising this function therefore reduces to minimising the least squares difference

$$S = \sum_k (R_1 x_1^k \tilde{R}_1 - R_2 x_2^k \tilde{R}_2 + t_1 - t_2)^2, \tag{5.55}$$

as has been discussed by many authors [1, 6, 7, 11].

## 5.2 Solution

The first point to note is that $S$ of equation (5.55) is a function of $t_1 - t_2$ only, and hence is independent of the absolute origin. This is precisely the behaviour we expect. It follows that minimisation of $S$ with respect to either $t_1$ or $t_2$ lead to the same equation, which is simply that

$$t_2 - t_1 = R_1 \bar{x}_1 \tilde{R}_1 - R_2 \bar{x}_2 \tilde{R}_2 \tag{5.56}$$

where

$$\bar{x}_1 = \frac{1}{n} \sum_{k=1}^{n} x_1^k, \quad \bar{x}_2 = \frac{1}{n} \sum_{k=1}^{n} x_2^k. \tag{5.57}$$

The vector $t_2 - t_1$ is simply the difference in the two centroids of the data, and depends on the rotors $R_i$.

Now that we have found $t_2 - t_1$ we can substitute its value back into $S$ to express $S$ as a function of the rotors only:

$$S = \sum_k \Big( R_1 (x_1^k - \bar{x}_1) \tilde{R}_1 - R_2 (x_2^k - \bar{x}_2) \tilde{R}_2 \Big)^2. \tag{5.58}$$

On squaring this only the cross terms remain with any rotor dependence, and we are left to maximise

$$S' = \sum_k \langle (x_1^k - \bar{x}_1) \tilde{R}_1 R_2 (x_2^k - \bar{x}_2) \tilde{R}_2 R_1 \rangle. \tag{5.59}$$

This is a function of the relative rotor $\tilde{R}_1 R_2$ only, again as expected. The same equation is obtained if we differentiate $S'$ with respect to $R_1$ or $R_2$. Using the result of equation (4.40) we see that the equation to solve is

$$\sum_k (R_1 (x_1^k - \bar{x}_1) \tilde{R}_1) \wedge (R_2 (x_2^k - \bar{x}_2) \tilde{R}_2) = 0. \tag{5.60}$$

Taking the inner product with the bivector $e_{1i} \wedge e_{1j}$ produces the equation

$$\mathsf{F}_{ij} - \mathsf{F}_{ji} = 0 \tag{5.61}$$

where

$$\mathsf{F}_{ij} = \sum_k f_i \cdot (x_1^k - \bar{x}_1)\, f_j \cdot (\tilde{R}_1 R_2 (x_2^k - \bar{x}_2) \tilde{R}_2 R_1). \tag{5.62}$$

This is easily solved with a singular-value decomposition of $\mathsf{F}_{ij}$, as has been discussed elsewhere [8].

## 5.3 Adding more cameras

The generalisation to $n$ cameras is quite straightforward. Instead of two terms in the pdf of equation (5.49) there are now $n$ of them. The marginalisation integral simply involves completing the square as follows:

$$\sum_{\alpha=1}^{n}(X-a_\alpha)^2 = n\Big(X-\frac{1}{n}\sum_{\alpha=1}^{n}a_\alpha\Big)^2+\frac{1}{n}\sum_{\alpha<\beta}(a_\alpha-a_\beta)^2. \tag{5.63}$$

The least squares expression to minimise therefore involves the sum over all $n(n-1)/2$ combinations of different cameras,

$$S=\sum_{\alpha<\beta}\sum_k(R_\alpha x_\alpha^k\tilde{R}_\alpha - R_\beta x_\beta^k\tilde{R}_\beta + t_\alpha - t_\beta)^2, \tag{5.64}$$

where the $k$ sum runs over point matches, and $\alpha, \beta$ run over the camera pairs. This result is sensible as it is totally symmetric on the camera labels and does not depend on relating everything back to a preferred reference camera.

Minimising $S$ with respect to each of the $t_\alpha$ vectors gives the simple solution for the relative translations

$$t_\alpha - t_\beta = R_\alpha\bar{x}_\alpha\tilde{R}_\alpha - R_\beta\bar{x}_\beta\tilde{R}_\beta. \tag{5.65}$$

Again, the total vector $t_1+\cdots+t_n$ is unspecified. Substituting the values for the relative vectors into $S$, we are left with the function

$$S=\sum_{\alpha<\beta}\sum_k\Big(R_\alpha(x_\alpha^k-\bar{x}_\alpha)\tilde{R}_\alpha - R_\beta(x_\beta^k-\bar{x}_\beta)\tilde{R}_\beta\Big)^2, \tag{5.66}$$

which we want to minimise with respect to the $n$ rotors $R_\alpha$. As before, one only obtains equations for the relative rotations between two cameras, and not the absolute rotation from the global $\{f_i\}$ frame.

One can get the general feel of this equation structure considering three cameras (Figure 5). The three equations from the three rotors reduce to

$$\sum_k\Big(R_1(x_1^k-\bar{x}_1)\tilde{R}_1\Big)\wedge\Big(R_2(x_2^k-\bar{x}_2)\tilde{R}_2+R_3(x_3^k-\bar{x}_3)\tilde{R}_3\Big)=0 \tag{5.67}$$

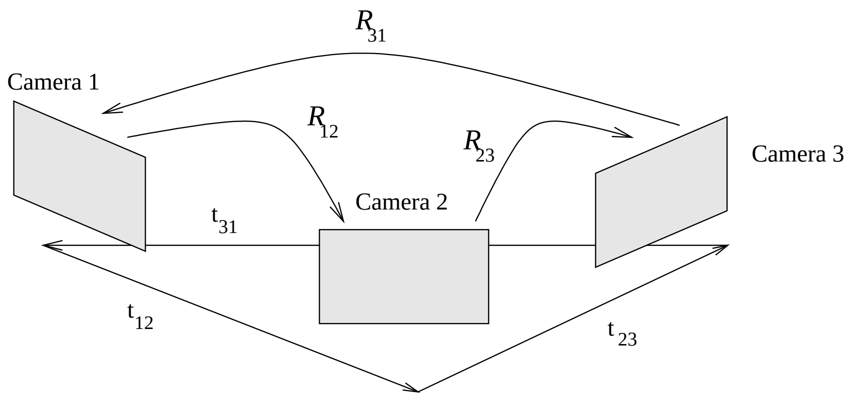

Figure 5: *The three camera setup.* The relative vectors between the cameras are given by $t_{ij} = t_j - t_i$. The relative rotations are $R_{ij} = R_j\tilde{R}_i$. These satisfy $t_{12} + t_{23} + t_{31} = 0$ and $R_{31}R_{23}R_{12} = 1$.

and

$$\sum_k \Big(R_2(x_2^k - \bar{x}_2)\tilde{R}_2\Big) \wedge \Big(R_3(x_3^k - \bar{x}_3)\tilde{R}_3 + R_1(x_1^k - \bar{x}_1)\tilde{R}_1\Big) = 0. \qquad (5.68)$$

The final equation is just the sum of the first two and contains no further information. Again, this is to be expected as there are always $n-1$ relative rotations to solve for.

This equation structure is more complicated that the 2-camera case, and cannot by solved simply with a singular-value decomposition. Rather than removing the anti-symmetric component of a single tensor, one has to minimise the anti-symmetric components of 3 independent tensors, using 2 independent rotors. This problem should be numerically quite straightforward to solve, either at the level of the equations, or through direct numerical minimisation of the $S$ of equation (5.66). This latter approach is simplified by the fact that the individual pairwise minimisers for two of the pairs provide good starting points for any minimisation routine.

## 6 Unknown Range Data

In most computer vision applications we do not have access to the third coordinate giving the direction to a point. Instead what we measure are pixel coordinates in the camera plane (see Figure 6). Placing the origin at the camera centre, a world point $X$ has coordinates $(X_1, X_2, X_3)$ expressed in the camera frame. Adopting

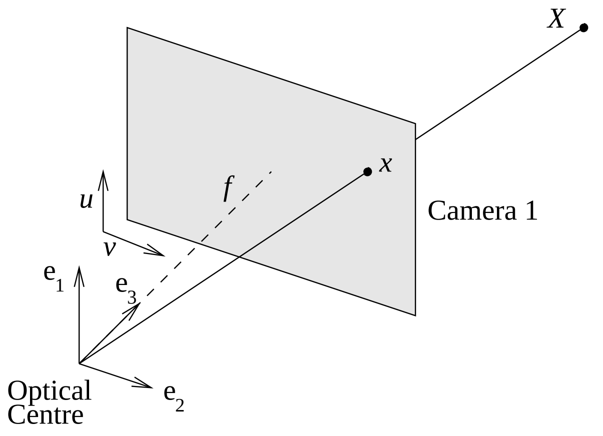

Figure 6: *Pixel Coordinates.* In most applications in computer vision one only measures the pixel coordinates of a point in the camera plane. Provided the camera is calibrated, these can be converted to the image coordinates of $x$.

the projective pinhole camera model, the image point $x$ has coordinates $(x_1, x_2, f)$, where $f$ is the focal length. The pixel coordinates $u = (u_1, u_2, 1)$ are related to the image coordinates by a $3 \times 3$ camera matrix $C$,

$$u = C(x/f), \quad x/f = C^{-1}u. \tag{6.69}$$

(See [9] for more details). Provided the matrix $C$ is known, we can recover the vector $x/f$. For a projective pinhole camera, the components of this are simply the *homogeneous coordinates* $(X_1/X_3, X_2/X_3, 1)$ of the world point $X$.

For the 2-camera setup of Figure 3, the two coordinates we measure in the Camera 1 system are

$$x_{1i}^k = \frac{e_{1i} \cdot (X^k - t_1)}{e_{13} \cdot (X^k - t_1)}. \quad i = 1, 2. \tag{6.70}$$

A simple model would be to assume is that the observed data is taken from a Gaussian distribution centred on these values. The problem with this is that the resulting marginalisation integral over the $X^k$ cannot be performed analytically. Instead we will use a different model in which the marginalisation integrals can be performed. The result is a likelihood function which can be minimised very quickly and efficiently. The results of this turn out to be reasonable, and geometrically quite sensible.

Our choice of a simplified model, including modeling the combined effects of the various sources of noise with a simple Gaussian distribution, is one of a number of simplifying assumptions we will make in order to find a simple function to minimise. Each of these assumptions can be challenged and modified to construct more realistic models and give better reconstruction. This approach is quite different from the standard alternative, based on the epipolar geometry and the fundamental matrix [10, 14]. In this approach an assortment of least-squares optimisers are considered, none with any underlying justification from a probabilistic model, and an assortment of linear algebra techniques are used to find the mutual translation and rotation. Many of these do not properly account for the structure of the rotation group, which limits their accuracy. They do have some value, however, in providing some fast algorithms to give initial points for the nonlinear schemes developed here.

Our starting point is the pdf of equation (5.49). That is, we start by treating all three coordinates in the same way. Again, we marginalise over the positions $X^k$ to get the 2-camera joint pdf, but this time we view the range data as an unknown parameter and assign it a uniform prior. We therefore arrive at the distribution

$$\begin{aligned} &P(R_1, R_2, t_1, t_2, \{z_1^k, z_2^k\} | \{x_{1i}^k, x_{2i}^k\}) \propto \\ &\qquad \exp\Big(\frac{-1}{2\sigma^2} \sum_k (R_1 z_1^k x_1^k \tilde{R}_1 - R_2 z_2^k x_2^k \tilde{R}_2 + t_1 - t_2)^2\Big), \end{aligned} \tag{6.71}$$

where $i$ runs over the two coordinates in the camera plane, $z_\alpha^k$ is the unknown range ($\alpha$ denotes the camera), and the vectors $x_1^k, x_2^k$ are formed directly from the measured data by

$$x_\alpha^k = \sum_{i=1}^{2} x_{\alpha i}^k f_i + f_3. \tag{6.72}$$

The next step is to marginalise over the unknown ranges $z_1$ and $z_2$. Here we make one final simplification by taking the range of the integrals from $-\infty \ldots \infty$. This allows for points behind the camera to be considered, so is clearly unjustified, but has the advantage that the integrals can be performed analytically. The integral we require has the form

$$I = \int_{-\infty}^{\infty} dz_1\, dz_2 \exp\Big(-(z_1 a_1 - z_2 a_2 + t)^2\Big) \tag{6.73}$$

where $a_1 = R_1 x_1^k \tilde{R}_1$, *etc.* and $t = t_1 - t_2$. To carry out this integral we need the result that

$$\int d^n x \exp\Big(-x_i x_j \mathsf{T}_{ij} + 2x_i b_i\Big) = N \exp\Big(b_i b_j \mathsf{T}^{-1}_{ij}\Big) \tag{6.74}$$

where $\mathsf{T}_{ij}$ is an $n \times n$ symmetric matrix, $b_i$ is an $n$-component vector and $N$ is a normalisation constant. For the integral (6.73) the matrix $\mathsf{T}_{ij}$ is given by

$$\mathsf{T}_{ij} = \begin{pmatrix} {a_1}^2 & -a_1 \cdot a_2 \\ -a_1 \cdot a_2 & {a_2}^2 \end{pmatrix}, \tag{6.75}$$

and the vector $b_i$ is

$$b_i = \begin{pmatrix} -a_1 \cdot t \\ a_2 \cdot t \end{pmatrix}. \tag{6.76}$$

It follows that

$$\det \mathsf{T}_{ij} = {a_1}^2 {a_2}^2 - (a_1 \cdot a_2)^2 = -(a_1 \wedge a_2)^2, \tag{6.77}$$

and

$$\mathsf{T}^{-1}_{ij} = -\frac{1}{(a_1 \wedge a_2)^2} \begin{pmatrix} {a_2}^2 & a_1 \cdot a_2 \\ a_1 \cdot a_2 & {a_1}^2 \end{pmatrix}. \tag{6.78}$$

Hence

$$\begin{aligned} b_i b_j \mathsf{T}^{-1}_{ij} &= -\frac{1}{(a_1 \wedge a_2)^2}\Big({a_1}^2 (a_2 \cdot t)^2 + {a_2}^2 (a_1 \cdot t)^2 - 2a_1 \cdot a_2 \, a_1 \cdot t \, a_2 \cdot t\Big) \\ &= -\frac{1}{(a_1 \wedge a_2)^2}(a_1 \cdot t \, a_2 - a_2 \cdot t \, a_1)^2 \\ &= \Big(\frac{t \cdot (a_1 \wedge a_2)}{|a_1 \wedge a_2|}\Big)^2, \end{aligned} \tag{6.79}$$

which assembles into a simple geometric function. Applying these results to the pdf of equation (6.71), and remembering the final $(t_1 - t_2)^2$ term, we arrive at the log-likelihood function

$$S = \sum_{k=1}^{n} \frac{\Big((t_1 - t_2) \wedge \Big((R_1 x_1^k \tilde{R}_1) \wedge (R_2 x_2^k \tilde{R}_2)\Big)\Big)^2}{|(R_1 x_1^k \tilde{R}_1) \wedge (R_2 x_2^k \tilde{R}_2)|^2}. \tag{6.80}$$

This is now a simple function of the vectors $t_\alpha$ and the rotors $R_\alpha$. Again, only the relative translation $(t_1 - t_2)$ enters the problem, and the freedom to choose the $f_i$ reference frame means that one of the rotors is arbitrary.

The function (6.80) has a simple geometric interpretation in terms of the distance between the projective lines for a given point match (see Figure 7). Given

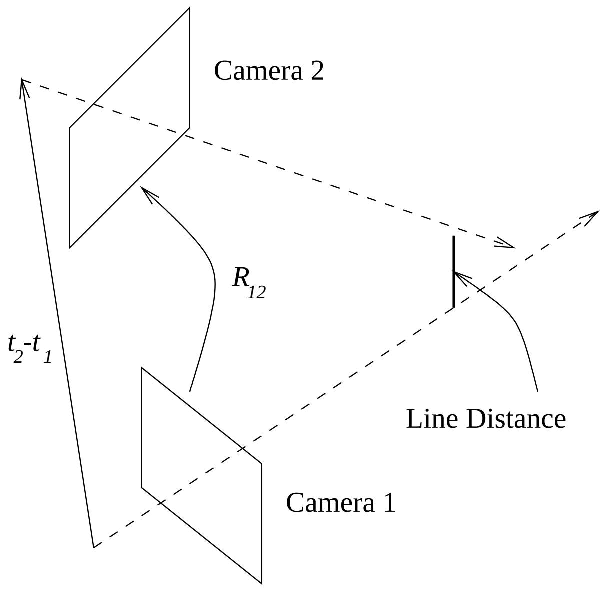

Figure 7: *Line Distance.* Given a point match in the two camera planes, the vectors are extended out to three-dimensional space, and the distance between the lines is found. The sum of the squares of these is minimised to find the best fit translation and rotation.

a point match, the projective lines from the two cameras are extended into space. The function then records the square of the distance between the lines (in units on $|t_1 - t_2|$), and sums these over all point matches. This is certainly a sensible error measure for this problem, and it is instructive to see how it arises from a probabilistic model.

The function (6.80) is scale invariant, since no scale has yet been imposed on the problem. As it stands, therefore, the function is minimised by setting $t_1 - t_2 = 0$. To avoid this we need to impose a scale, which is most simply achieved by setting

$$(t_1 - t_2)^2 = 1. \tag{6.81}$$

This condition is imposed by including a Lagrange multiplier, so the function to minimise becomes

$$S = \sum_{k=1}^{n} \Big((t_1 - t_2)\cdot n^k\Big)^2 - \lambda\Big((t_1 - t_2)^2 - 1\Big), \tag{6.82}$$

where

$$n^k = \frac{I(R_1 x_1^k \tilde{R}_1) \wedge (R_2 x_2^k \tilde{R}_2)}{|(R_1 x_1^k \tilde{R}_1) \wedge (R_2 x_2^k \tilde{R}_2)|}. \tag{6.83}$$

Our final $S$ (6.82) is still quadratic in the relative vector $t = t_1 - t_2$, and minimising gives the simple equation

$$\sum_{k=1}^{n} t \cdot n^k \, n^k = \lambda t. \tag{6.84}$$

We next construct the symmetric, positive definite function

$$\mathsf{F}(a) = \sum_{k=1}^{n} a \cdot n^k \, n^k, \tag{6.85}$$

which is a function of the data and the rotation only. The translation $t$ is an eigenvector of this function, with the eigenvalue

$$\lambda = t \cdot \mathsf{F}(t) = \sum_{k=1}^{n} (t \cdot n^k)^2 = S. \tag{6.86}$$

So to minimise the error function $S$ we need to choose $t$ to be the eigenvector with smallest eigenvalue. All we need do, then, is minimise the lowest eigenvalue of $\mathsf{F}$ with respect to the rotor $R$. This is a fairly simple optimisation problem, as we only need to search in the 3-parameter rotor space. Numerical studies of this function reveal that it contains some local minima, but the global minimum lies in a fairly deep valley and it is not hard to find this numerically.

## 7 Extension to 3 cameras

The Bayesian analysis presented here extends easily to the 3 camera case. A simpler alternative, however, is to take the log-likelihood function of equation (6.80) and sum this function over each of the camera pairs. Incorporating a Lagrange multiplier to impose a suitable constraint, the function we need to minimise is

$$\begin{aligned} S_3 = \sum_{k=1}^{n} \Big( &\big((t_1 - t_2)\cdot n_{12}^k\big)^2 + \big((t_2 - t_3)\cdot n_{23}^k\big)^2 + \big((t_3 - t_1)\cdot n_{31}^k\big)^2 \\ &- \lambda\big((t_1-t_2)^2 + (t_2-t_3)^2 + (t_3-t_1)^2 - 1\big)\Big), \end{aligned} \tag{7.87}$$

where

$$n_{12}^k = \frac{I(R_1 x_1^k \tilde{R}_1)\wedge(R_2 x_2^k \tilde{R}_2)}{|(R_1 x_1^k \tilde{R}_1)\wedge(R_2 x_2^k \tilde{R}_2)|} \quad etc. \tag{7.88}$$

We only get independent equations from minimising with respect to two of the three translation vectors. Taking these to be $t_1$ and $t_2$ the equations we arrive at are

$$\sum_{k=1}^{n} (t_1 - t_2)\cdot n_{12}^k\, n_{12}^k - (t_3 - t_1)\cdot n_{31}^k\, n_{31}^k = \lambda(2t_1 - t_2 - t_3) \tag{7.89}$$

$$\sum_{k=1}^{n} (t_2 - t_3)\cdot n_{23}^k\, n_{23}^k - (t_1 - t_2)\cdot n_{12}^k\, n_{12}^k = \lambda(2t_2 - t_3 - t_1). \tag{7.90}$$

If we now set

$$a = 2t_1 - t_2 - t_3, \quad b = 2t_2 - t_3 - t_1, \tag{7.91}$$

then we recover a $6 \times 6$ eigenvalue problem of the form

$$\begin{pmatrix} \mathsf{F}_{12} + 2\mathsf{F}_{31} & \mathsf{F}_{31} - \mathsf{F}_{12} \\ \mathsf{F}_{23} - \mathsf{F}_{12} & 2\mathsf{F}_{23} + \mathsf{F}_{12} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = 3\lambda \begin{pmatrix} a \\ b \end{pmatrix} \tag{7.92}$$

where

$$\mathsf{F}_{12}(a) = \sum_{k=1}^{n} a\cdot n_{12}^k\, n_{12}^k, \quad etc. \tag{7.93}$$

As in the 2 camera case, the eigenvalue $\lambda$ returns the value of $S_3$ that we are trying to minimise. The minimisation problem therefore reduces to finding a pair of rotors which minimises the lowest eigenvalue of a $6 \times 6$ matrix. Numerical implementation of this algorithm will be presented elsewhere.

## 8 Conclusions

Geometric Algebra is an extremely powerful tool for handling rotations in three dimensions. Vectors and the quantities which act on them are united in a single algebra, which has a number of computational advantages. Relaxing the normalisation condition for rotors provides a simplified calculus for rotations which avoids having to work in the tangent space to the group manifold. As a result, many extremisation problems involving rotations can be studied and solved without ever leaving the geometric algebra of 3-d.

The applications to the camera localization problem given here illustrate the various advantages that geometric algebra can provide. This is particularly so when combined with Bayesian inference techniques. The models considered here are highly simplified, though still quite useful. Much work remains in order to construct robust, accurate algorithms to use with real cameras. The effects of the camera matrix must be included, particularly as the cameras often require recalibrating after they are moved significantly. Similarly, more realistic noise models are required. Discretisation errors, for example, are certainly not well modeled as Gaussian process. In addition, we need to be able to work with arbitrary numbers of cameras, allowing for occlusion effects where point matches may only be shared by a subset of all of the cameras. When tackling each of these problems, however, there seems little doubt that the combination of geometric algebra and Bayesian reasoning advocated here will turn out to be the best way to proceed.

## Acknowledgements

CD gratefully acknowledges the support of the EPSRC.

## References

[1] K. Arun, T.S. Huang, and S.D. Blostein. Least squares fitting of two 3-D point sets. *IEEE Trans. PAMI*, 9:698–700, 1987.

[2] J.F. Cornwell. *Group Theory in Phsics II.* Academic Press Ltd., London, 1984.

[3] C. J. L. Doran. *Geometric Algebra and its Application to Mathematical Physics.* PhD thesis, Cambridge University, 1994.

[4] C.J.L. Doran, A.N. Lasenby, S.F. Gull, and J. Lasenby. Lectures in geometric algebra. In W.E. Baylis, editor, *Clifford (Geometric) Algebras*, pages 65–236. Birkhauser, Boston, 1996.

[5] D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus.* Reidel, Dordrecht, 1984.

[6] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am.*, 4:629–642, 1987.

[7] T.S. Huang and A.N. Netravali. Motion and structure from feature correspondences: A review. *Proc. of the IEEE,*, 82(2):252–268, 1994.

[8] J. Lasenby, W.J. Fitzgerald, A.L. Lasenby, and C.J.L. Doran. New geometric methods for computer vision: An application to structure and motion estimation. *Int. J. Comp. Vision*, 26(3):191, 1998.

[9] J. Lasenby and A. Stevenson. Using geometric algebra in optical motion capture. In E. Bayro and G. Sobczyk, editors, *Geometric algebra: A geometric approach to computer vision, neural and quantum computing, robotics and engineering.* Birkhauser, 2000.

[10] J. Ponce and Y. Genc. Epipolar geometry and linear subspace methods: A new approach to weak calibration. *Int. J. of Comp. Vision*, 28(3):223–243, 1998.

[11] B. Sabata and J.K. Aggarwal. Estimation of motion from a pair of range images: A review. *CVGIP: Image Understanding*, 54(3):309–324, 1991.

[12] D.S. Sivia. *Data Analysis, A Bayesian tutorial.* Oxford University Press, 1996.

[13] 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.

[14] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *Int. J. of Comp. Vision*, 27(2):161–195, 1998.