New Geometric Methods for Computer Vision: an application to structure and motion estimation

J. Lasenby*, A.N. Lasenby†, C.J.L. Doran† and W.J. Fitzgerald*
* Department of Engineering, Trumpington Street, Cambridge CB2 1PZ, U.K.
† MRAO, Cavendish Laboratory, Madingley Road, Cambridge CB3 OHE, U.K.

July 16, 1996

Abstract

We discuss a coordinate-free approach to the geometry of computer vision problems. The technique we use to analyse the 3-dimensional transformations involved will be that of **geometric algebra**: a framework based on the algebras of Clifford and Grassmann. This is not a system designed specifically for the task in hand, but rather a framework for all mathematical physics. Central to the power of this approach is the way in which the formalism deals with rotations; for example, if we have two arbitrary sets of vectors, known to be related via a 3-D rotation, the rotation is easily recoverable if the vectors are given. Extracting the rotation by conventional means is not as straightforward. The calculus associated with geometric algebra is particularly powerful, enabling one, in a very natural way, to take derivatives with respect to any multivector (general element of the algebra). What this means in practice is that we can minimize with respect to rotors representing rotations, vectors representing translations, or any other relevant geometric quantity. This has important implications for many of the least-squares problems in computer vision where one attempts to find optimal rotations, translations etc., given observed vector quantities. We will illustrate this by analysing the problem of estimating motion from a pair of images, looking particularly at the more difficult case in which we have available only 2D information and no information on range. While this problem has already been much discussed in the literature, we believe the present formulation to be the only one in which least-squares estimates of the motion and structure are derived simultaneously using analytic derivatives.

Categories: Physics-based vision; structure and motion estimation; geometry.

1 Introduction

Geometric algebra is a coordinate-free approach to geometry which, unlike conventional vector algebra, has available a fully invertible, associative product between vectors. This is achieved via a *Clifford product* and entails the adding together of objects of different type, such as scalars and vectors, in much the same way as a complex number is built out of real and imaginary parts. The new product gives the algebra tremendous power, and its applications spread across many areas of physics, mathematics and engineering. In the following sections we describe several geometric techniques particularly suited to computer vision problems and illustrate their use for estimating object or camera motion and scene reconstruction.

We will begin by illustrating a method by which the 3-D rotation which takes a set of three known vectors to another given set of three vectors can be easily found. In a conventional approach, one method would be to solve for the rotation matrix, $\mathcal{R}$, from a set of simultaneous equations. In the geometric algebra approach one can construct the rotor R (the entity which performs the rotation) in a coordinate-free manner using the known vectors. This will serve as an example of how geometric manipulations can be simplified and extended to higher dimensions with little extra effort. While it is often useful to solve directly for a rotor R, it is usually the case that observations are corrupted by noise. In such cases minimization of some least-squares expression is generally the preferred technique. We will show that we are able to minimize with respect to certain multivector quantities – in particular the rotor R – to simplify the optimization procedure.

In order to show how these techniques can be applied we will look at the area of motion analysis. One approach to deriving motion and scene structure is to invoke the idea of 'optical flow'. This method attempts to set up a system of differential equations relating position and spatial velocities in the scene to position and velocity of intensity structures in the image. Another approach is to estimate camera or object motion and scene parameters from token matching – i.e. using the exact geometrical relationships between the same points and lines in a series of scene projections. It is the second approach which the examples will address. Throughout, we will assume that the matching of points or lines between scenes, which may often be the hardest part of the procedure, has been achieved and will therefore only concern ourselves with the estimation problem given these matches. The problems of estimating camera motion or object motion from two images are easily shown to be equivalent. Two recent reviews, [Sabata and Aggarwal 1991], [Huang and Netravali 1994] provide comprehensive overviews of the techniques currently employed to recover the motion of an object and the scene structure when two images are available. In this paper we firstly discuss the simplest case in which the range data is known in both images. The harder case, where no range data and only 2D projections are available, is also discussed. We will illustrate the geometric algebra solutions to both cases. We believe that our approach to estimating structure and motion from 2D data via point correspondences is the first algorithm which obtains a least-squares estimate of all quantities simultaneously using analytic derivatives. As pointed out in [Huang and Netravali 1994], all current algorithms can be essentially split into two categories, *structure first* or *motion first* techniques. The errors in estimating structure will affect the subsequent estimation of the motion (and vice versa) in an unpredictable way. The motivation for our approach is the desire to estimate all quantities simultaneously in a straightforward least-squares sense and so avoid the inevitable problems of obtaining different results according to whether structure or motion is estimated first. A comparison of our results with another commonly used *motion first* algorithm shows that the improvements in parameter estimation are considerable.

The so-called *eight-point algorithm* [Longuet-Higgins 1981] is a technique whereby the relative orientation of two projections and the structure of the scene can be computed if there are at least eight point matches in the two projections. Subsequent work [Faugeras *et al.* 1987] extended this method to handle a situation in which errors or some form of uncertainty is present. An analysis of these techniques in the geometric algebra framework has been presented elsewhere [Bayro and Lasenby 1995]. There it was shown that many of the quantities introduced in establishing the algorithms have direct geometrical relevance. The unwrapping of the rotational element of the general displacement is also made easier by this approach. In the presence of errors, [Faugeras *et al.* 1987] adopt a least-squares solution which can be solved via quaternionic methods. Section 3 of this paper will illustrate how the geometric algebra formalism for rotations can be used in such least-squares problems.

One notion we would like to stress is that the framework presented here to analyse a particular computer vision problem is precisely the same framework that one could use for relativity, quantum mechanics and other problems over a large range of mathematical physics. We believe that the reformulation of the problem in purely geometric terms will give greater intuitive insight and will perhaps enable more complicated problems to be successfully addressed.

As we will use a mathematical system which is not familiar to most people, a few preliminaries are necessary. There follows a brief introduction to the history and the basic elements of geometric algebra. More detailed accounts are given in [Gull *et al.* 1993],[Hestenes 1986a], [Hestenes and Sobczyk 1984]. Throughout the paper we adopt the convention that repeated indices are summed over unless stated otherwise.

2 An Introduction to Geometric Algebra

The late 19^{th} century saw the creation of many new algebraic systems. Amongst these were the algebras of Clifford and Grassmann [Clifford 1878], [Grassmann 1877], and although there was much interest at the time, both were soon abandoned by physicists in their search for an algebra to deal with real problems in space and time. Gibbs' vector algebra, which used a system based on scalar and vector products, had many deficiencies but appeared to be successful in dealing with some areas of physics (particularly electromagnetism). This system was widely adopted and, indeed, is still the one that is most commonly used today. Over the years physicists and mathematicians have largely abandoned the search for a new unifying algebra and have instead created a series of novel algebraic systems as and when they were needed to cope with new problems (e.g. spinor algebra, tensor algebra etc.).

Since the late 19^{th} century, Clifford algebras have been 'rediscovered' and reapplied many times [Hestenes 1986b]. However, the approach which is used here was pioneered in the 1960s by David Hestenes [Hestenes 1966], and will be referred to as 'geometric algebra'. For 30 years Hestenes has worked on developing Clifford algebra into his version of geometric algebra with the aim of providing a complete unifying language for physics and mathematics. Recent developments in various areas of mathematical physics are proving that genuinely new and exciting results are emerging from the reformulation of problems in geometric algebra terms [Doran *et al.* 1996], [Lasenby *et al.* 1995], [Lasenby *et al.* 1993b].

2.1 Multiplying Vectors

This section introduces the basic elements of Hestenes' approach to geometric algebra. In conventional vector algebra there are 2 standard products available:

The **scalar** product: $\boldsymbol{a}\cdot\boldsymbol{b}$: produces a scalar

The **cross** or **vector** product: $\boldsymbol{a}\times\boldsymbol{b}$: produces another vector

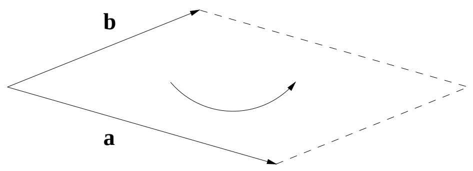

Figure 1: The directed area, or bivector, $\boldsymbol{a}\wedge\boldsymbol{b}$.

The scalar product (also known as the *inner* or *dot* product) of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is a scalar with magnitude $|\boldsymbol{a}||\boldsymbol{b}|\cos\theta$. Here, $|\boldsymbol{a}|$ and $|\boldsymbol{b}|$ are the lengths of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and θ is the angle between them. The cross product, $\boldsymbol{a}\times\boldsymbol{b}$ of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is a third vector with magnitude $|\boldsymbol{a}||\boldsymbol{b}|\sin\theta$ in the direction perpendicular to the plane containing $\boldsymbol{a}$ and $\boldsymbol{b}$. The scalar product tells us something about the *relative* direction of any two vectors while the cross product recovers some of the absolute directional information lost in taking the scalar product.

Although the cross product is fundamental to standard teaching of mathematical physics, it is particularly deficient. For example, in 2 dimensions, a direction perpendicular to $\boldsymbol{a}$ and $\boldsymbol{b}$ has no meaning and in 4 or more dimensions, what is meant by such a perpendicular direction is ambiguous. It is apparent that there is a need for a more general product which conveys directional information and is generalizable to all dimensions. Thus we shall cease to use the vector cross-product and instead introduce a new product: the **outer** (also called *wedge* or *exterior*) product.

The **outer product** of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is written

$$\boldsymbol{a}\wedge\boldsymbol{b} \tag{1}$$

and has magnitude $|\boldsymbol{a}||\boldsymbol{b}|\sin\theta$. But what is $\boldsymbol{a}\wedge\boldsymbol{b}$? It is not a scalar or a vector; we think of $\boldsymbol{a}\wedge\boldsymbol{b}$ as a directed *area* in the plane segment containing $\boldsymbol{a}$ and $\boldsymbol{b}$, and we call this quantity a **bivector**. More precisely, $\boldsymbol{a}\wedge\boldsymbol{b}$ is the directed area swept out by displacing vector $\boldsymbol{a}$ along vector $\boldsymbol{b}$ as illustrated in Figure 1.

Similarly, $\boldsymbol{b}\wedge\boldsymbol{a}$ is the directed area swept out by displacing vector $\boldsymbol{b}$ along vector $\boldsymbol{a}$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are as shown in Figure 1, then we define the orientation of $\boldsymbol{a}\wedge\boldsymbol{b}$ to be anticlockwise and that of $\boldsymbol{b}\wedge\boldsymbol{a}$ to be of the opposite orientation, i.e. clockwise. Thus, the outer product is anticommutative,

$$\boldsymbol{a}\wedge\boldsymbol{b} = -\boldsymbol{b}\wedge\boldsymbol{a}. \tag{2}$$

So, while the outer product has the same magnitude as the vector product and shares the same anticommutative property, we must stress that it is a fundamentally different quantity, a **bivector** rather than a vector. The attraction of this geometric interpretation of the outer product is that it is immediately generalizable to higher dimensions. For example, according to the previous definitions, we interpret $(\boldsymbol{a}\wedge\boldsymbol{b})\wedge\boldsymbol{c}$ as the oriented 3-dimensional volume obtained by sweeping the bivector $\boldsymbol{a}\wedge\boldsymbol{b}$

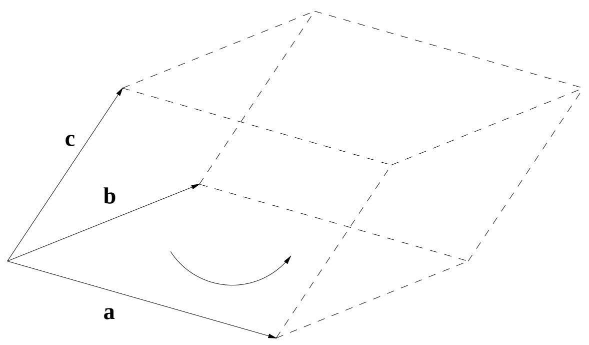

Figure 2: The oriented volume, or trivector, $\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}$.

(or directed area) along the vector $\boldsymbol{c}$: we call this oriented volume a **trivector**. This is illustrated in Figure 2.

From this approach it is intuitively obvious that the outer product is **associative**:

$$(\boldsymbol{a}\wedge\boldsymbol{b})\wedge\boldsymbol{c} = \boldsymbol{a}\wedge(\boldsymbol{b}\wedge\boldsymbol{c}) \equiv \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}, \tag{3}$$

since we obtain the same volume if we sweep the bivector $\boldsymbol{a}\wedge\boldsymbol{b}$ along the vector $\boldsymbol{c}$ or sweep the vector $\boldsymbol{a}$ along the bivector $\boldsymbol{b}\wedge\boldsymbol{c}$. In a rigorous approach to geometric algebra [Hestenes and Sobczyk 1984] the associativity of the outer product follows directly from the structure of the algebra imposed by the axioms (see next section). Thus, there is no ambiguity in writing $\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}$. In 3-dimensional space we are unable to sweep our trivector along any 4th dimension and therefore we cannot go beyond a trivector, so that the outer product of any four 3-D vectors must be zero. We can see this mathematically by noting that if $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are not coplanar then any other vector $\boldsymbol{d}$ can be written as a linear combination of them. Wedging these 4 vectors together will therefore result in zero since every term will contain the wedge product of two identical vectors – which is zero;

$$\begin{aligned}
\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge\boldsymbol{d} &= \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge(\lambda_a\boldsymbol{a} + \lambda_b\boldsymbol{b} + \lambda_c\boldsymbol{c}) \\
&= \lambda_a\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge\boldsymbol{a} + \lambda_b\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge\boldsymbol{b} + \lambda_c\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge\boldsymbol{c} \qquad (4) \\
&= 0. \qquad (5)
\end{aligned}$$

In general, the principles outlined here can be extended to however many dimensions, n, there are in the space – for example, $\boldsymbol{a_1}\wedge\boldsymbol{a_2}\wedge\ldots\wedge\boldsymbol{a_m}$ $(n \geq m)$ would be an oriented m-volume or an m-vector.

In what follows we will sometimes refer to the dimensionality of an m-vector as its **grade**: for example, a scalar has grade 0, a vector grade 1, a bivector grade 2, etc. As we will see shortly, it is possible to manipulate expressions containing quantities of any grade, for instance we can add a scalar to a bivector in a perfectly well defined way. A **multivector** will mean a sum of quantities of any grade and a multivector is said to be *homogeneous* if it contains terms of only a single grade. We will adopt the convention of using bold-face type for vectors (with a certain straightforward exception discussed below) and ordinary type for all other multivectors.

Now that we are familiar with the inner (scalar) and outer products, the next important step is the introduction of a new product.

2.2 The Geometric Product

The inner and outer products of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, together give us information about the magnitudes and directions of the vectors. It would thus be desirable to combine this information into a single product. This is precisely what is done in forming the **geometric** (or Clifford) product of $\boldsymbol{a}$ and $\boldsymbol{b}$ which is written $\boldsymbol{ab}$ and defined as

$$\boldsymbol{ab} = \boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\wedge\boldsymbol{b}. \tag{6}$$

The geometric product $\boldsymbol{ab}$ is the sum of a scalar, $\boldsymbol{a}\cdot\boldsymbol{b}$, and a bivector, $\boldsymbol{a}\wedge\boldsymbol{b}$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are parallel, then $\boldsymbol{a}\wedge\boldsymbol{b} = 0$ and $\boldsymbol{ab}$ is a scalar containing only magnitude information. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are perpendicular then $\boldsymbol{a}\cdot\boldsymbol{b} = 0$ and $\boldsymbol{ab}$ is then a directed area. For $\boldsymbol{a}$ and $\boldsymbol{b}$ neither parallel or perpendicular it is therefore natural to expect their product to contain both scalar and bivector parts.

When we take the inner product of a vector $\boldsymbol{b}$ with a vector $\boldsymbol{a}$, $\boldsymbol{a}\cdot\boldsymbol{b}$, we are lowering the grade of $\boldsymbol{b}$ by 1, i.e. producing a scalar (grade 0) from a vector (grade 1). Similarly, taking the outer product of a vector $\boldsymbol{b}$ with a vector $\boldsymbol{a}$, $\boldsymbol{a}\wedge\boldsymbol{b}$, raises the grade by 1 – i.e. produces a bivector (grade 2) from a vector (grade 1). This is an example of what we shall see later is an important principle, namely that inner and outer products with a vector respectively lower and raise the grade of an m-vector by 1.

The inner and outer products have opposite commutation properties:

$$\boldsymbol{a}\cdot\boldsymbol{b} = \boldsymbol{b}\cdot\boldsymbol{a}$$

$$\boldsymbol{a}\wedge\boldsymbol{b} = -\boldsymbol{b}\wedge\boldsymbol{a}.$$

Thus, suppose we consider the sum and difference of $\boldsymbol{ab}$ and $\boldsymbol{ba}$:

$$\boldsymbol{ab} + \boldsymbol{ba} = \boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\wedge\boldsymbol{b} + \boldsymbol{b}\cdot\boldsymbol{a} + \boldsymbol{b}\wedge\boldsymbol{a} = 2\boldsymbol{a}\cdot\boldsymbol{b} \tag{7}$$

$$\boldsymbol{ab} - \boldsymbol{ba} = \boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\wedge\boldsymbol{b} - \boldsymbol{b}\cdot\boldsymbol{a} - \boldsymbol{b}\wedge\boldsymbol{a} = 2\boldsymbol{a}\wedge\boldsymbol{b}. \tag{8}$$

Equations (7), (8) make it possible for us to define the inner and outer products as the symmetric and antisymmetric parts of the geometric product:

$$\boldsymbol{a}\cdot\boldsymbol{b} = \tfrac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba}) \tag{9}$$

$$\boldsymbol{a}\wedge\boldsymbol{b} = \tfrac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba}). \tag{10}$$

In a rigourous mathematical approach [Hestenes and Sobczyk 1984] one defines a geometric algebra in the following way. Let $\mathcal{V}_n$ be an n-dimensional space defined over the real numbers. We can then generate a *geometric algebra* $\mathcal{G}_n$ by defining a **geometric product** between two vectors, $\boldsymbol{ab}$, such that for all vectors in $\mathcal{V}_n$, the geometric product is associative and distributive over addition, multiplication by a scalar is well-defined and every vector squares to give a scalar. These properties are summarized below

$$\boldsymbol{a}(\boldsymbol{bc}) = (\boldsymbol{ab})\boldsymbol{c} \tag{11}$$

$$\boldsymbol{a}(\boldsymbol{b}+\boldsymbol{c}) = \boldsymbol{ab} + \boldsymbol{ac} \tag{12}$$

$$(\boldsymbol{b}+\boldsymbol{c})\boldsymbol{a} = \boldsymbol{ba} + \boldsymbol{ca} \tag{13}$$

$$\boldsymbol{a}\lambda = \lambda\boldsymbol{a} \tag{14}$$

$$\boldsymbol{a}^2 = \pm|\boldsymbol{a}|^2 \tag{15}$$

where λ is a scalar and $|\boldsymbol{a}|$ is the scalar modulus of the vector $\boldsymbol{a}$. Given these definitions one can then introduce the inner and outer products through equations (9), (10). We note here that expressions in the geometric algebra will be interpreted so that the inner and outer products take precedence over the geometric product; for example, $\boldsymbol{a}\boldsymbol{b}\wedge\boldsymbol{c}$ is understood to mean $\boldsymbol{a}(\boldsymbol{b}\wedge\boldsymbol{c})$.

In a space of dimension n we can have homogeneous multivectors of grade 0 (scalars), grade 1 (vectors), grade 2 (bivectors), etc... up to grade n. Multivectors containing only even-grade elements are termed *even* and those containing only odd-grade elements are termed *odd*. The geometric product can, of course, be defined for any two multivectors. Considering two homogeneous multivectors A_r and B_s of grades r and s respectively, it is clear that the geometric product of such multivectors will contain parts of various grades. It can be shown [Hestenes and Sobczyk 1984] that the geometric product of A_r and B_s can be written as

$$A_r B_s = \langle AB\rangle_{r+s} + \langle AB\rangle_{r+s-2} + \ldots + \langle AB\rangle_{|r-s|}, \tag{16}$$

where $\langle M\rangle_t$ is used to denote the t-grade part of multivector M. For non-homogeneous multivectors we again simply apply the distributive rule over their homogeneous parts. Using the above we can now generalize the definitions of inner and outer products given in equations (7),(8). For two homogeneous multivectors A_r and B_s, we define the inner and outer products as

$$A_r \wedge B_s = \langle A_r B_s\rangle_{|r-s|} \tag{17}$$

$$A_r \cdot B_s = \langle A_r B_s\rangle_{r+s}. \tag{18}$$

Thus, the inner product produces an $|r-s|$-vector – which means it effectively reduces the grade of B_s by r; and the outer product gives an $r+s$-vector, therefore increasing the grade of B_s by r. This is an extension of the general principle that dotting with a vector lowers the grade of a multivector by 1 and wedging with a vector raises the grade of a multivector by 1. The symbol $\langle AB\rangle$ is used to denote the **scalar part** of the product AB. Taking scalar parts of a product of multivectors satisfies the cyclic-reordering property

$$\langle A....BC\rangle = \langle CA....B\rangle. \tag{19}$$

The operation of **reversion** reverses the order of vectors in any multivector. The reverse of A is written as $\tilde{A}$, and for a product we have

$$(AB)\tilde{} = \tilde{B}\tilde{A}. \tag{20}$$

In manipulations with multivectors we are simply keeping track of different-grade objects – precisely as we do when we manipulate complex numbers, keeping track of real and imaginary parts.

For an n-dimensional Euclidean space we can introduce an orthonormal frame of vectors $\{\sigma_i\}$ $i = 1,.,n$, such that $\sigma_i\cdot\sigma_j = \delta_{ij}$. This leads to a basis for the entire algebra; i.e. any multivector of the algebra can be expressed as a linear combination of the following basis elements:

$$1 \quad \{\sigma_i\} \quad \{\sigma_i\wedge\sigma_j\} \quad \{\sigma_i\wedge\sigma_j\wedge\sigma_k\} \quad \ldots \quad \sigma_1\wedge\sigma_2\wedge\ldots\wedge\sigma_n. \tag{21}$$

Note that we shall not use bold-type for these basis vectors. The highest grade element is called the **pseudoscalar** and is given the symbol I, or i in 3 or 4 dimensions – this use of 'i' for the pseudoscalar will be addressed further in the following section. We can express any multivector in terms of this basis. While such an expansion in terms of the basis is often essential, we stress that one of the main strengths of geometric algebra is the ability to do many manipulations in a basis-free manner.

The conventional vector product takes two vectors to produce a third and is not easily generalizable to more than 3 dimensions. In 3 dimensions we can relate the vector product to the outer product using a duality operation;

$$\boldsymbol{a}\times\boldsymbol{b} \equiv -i\boldsymbol{a}\wedge\boldsymbol{b}. \tag{22}$$

Thus, multiplication by the pseudoscalar i interchanges lines (vectors) and planes (bivectors) in 3-space. This concept of forming duals via multiplication by the pseudoscalar is extendable to any dimension and is important in the formulation of projective geometry in geometric algebra [Hestenes and Ziegler 1991].

2.3 Rotations and the geometric algebra of 3-D space

For 3-D space our basis has $2^3 = 8$ elements given by:

$$1, \quad \{\sigma_1, \sigma_2, \sigma_3\}, \quad \{\sigma_1\sigma_2,\, \sigma_2\sigma_3,\, \sigma_3\sigma_1\}, \quad \sigma_1\sigma_2\sigma_3 \equiv i. \tag{23}$$

Note that $\sigma_i\wedge\sigma_j$ (for $i \neq j$) can be written as $\sigma_i\sigma_j$ since the vectors are orthonormal so that $\sigma_i\cdot\sigma_j = \delta_{ij}$. The trivector or pseudoscalar $\sigma_1\sigma_2\sigma_3$ squares to -1 and commutes with all multivectors in the space, it is therefore given the symbol i;

$$\sigma_1\sigma_2\sigma_3 = i.$$

Note that this i is a pseudoscalar and should not be confused with the commutative scalar imaginary used in quantum mechanics and engineering. By straightforward multiplication we can see that the 3 bivectors can also be written as

$$\sigma_1\sigma_2 = i\sigma_3, \qquad \sigma_2\sigma_3 = i\sigma_1, \qquad \sigma_3\sigma_1 = i\sigma_2. \tag{24}$$

These simple bivectors rotate vectors in their own plane by 90°, e.g. $(\sigma_1\sigma_2)\sigma_2 = \sigma_1$, $(\sigma_1\sigma_2)\sigma_1 = -\sigma_2$ etc. In order to find out more about rotations in the geometric algebra we note that any rotation can be represented by a pair of reflections. Consider a vector $\boldsymbol{a}$ reflected in the plane perpendicular to a unit vector $\boldsymbol{n}$. Let this reflected vector be $\boldsymbol{a}'$. If we write $\boldsymbol{a} = \boldsymbol{a}_\perp + \boldsymbol{a}_{||}$ where $\boldsymbol{a}_\perp$ and $\boldsymbol{a}_{||}$ respectively denote the parts of $\boldsymbol{a}$ perpendicular and parallel to $\boldsymbol{n}$, then we clearly have $\boldsymbol{a}' = \boldsymbol{a}_\perp - \boldsymbol{a}_{||}$. To see how we express $\boldsymbol{a}'$ in terms of $\boldsymbol{a}$ and $\boldsymbol{n}$ we write

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{n}^2\boldsymbol{a} \\ &= \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{a} + \boldsymbol{n}\wedge\boldsymbol{a}) \\ &= (\boldsymbol{n}\cdot\boldsymbol{a})\boldsymbol{n} + \boldsymbol{n}(\boldsymbol{n}\wedge\boldsymbol{a}). \end{aligned} \tag{25}$$

Thus, $\boldsymbol{a}_{||} = (\boldsymbol{n}\cdot\boldsymbol{a})\boldsymbol{n}$ and $\boldsymbol{a}_\perp = \boldsymbol{n}(\boldsymbol{n}\wedge\boldsymbol{a})$. This now enables us to write

$$\begin{aligned} \boldsymbol{a}' &= \boldsymbol{a}_\perp - \boldsymbol{a}_{||} \\ &= \boldsymbol{n}(\boldsymbol{n}\wedge\boldsymbol{a}) - (\boldsymbol{n}\cdot\boldsymbol{a})\boldsymbol{n} \\ &= -(\boldsymbol{n}\cdot\boldsymbol{a})\boldsymbol{n} - (\boldsymbol{n}\wedge\boldsymbol{a})\boldsymbol{n} = -\boldsymbol{n}\boldsymbol{a}\boldsymbol{n}. \end{aligned} \tag{26}$$

The above uses the fact that $\boldsymbol{n}(\boldsymbol{n}\wedge\boldsymbol{a}) = -(\boldsymbol{n}\wedge\boldsymbol{a})\boldsymbol{n}$. The result of reflecting $\boldsymbol{a}$ in the plane perpendicular to $\boldsymbol{n}$ is therefore $-\boldsymbol{nan}$. A reflection of $\boldsymbol{a}$ in the plane perpendicular to $\boldsymbol{n}$, followed by a reflection in the plane perpendicular to $\boldsymbol{m}$ results in a new vector

$$-\boldsymbol{m}(-\boldsymbol{nan})\boldsymbol{m} = (\boldsymbol{mn})\boldsymbol{a}(\boldsymbol{nm}) = R\boldsymbol{a}\tilde{R}.$$

The multivector $R = \boldsymbol{mn}$ is called a **rotor**, contains only even-grade elements and satisfies $R\tilde{R} = 1$. We note here that general even elements of the space which transform one-sidedly under rotations are called **spinors**, i.e. under a rotation R, a spinor ψ becomes $R\psi$. The transformation $\boldsymbol{a} \mapsto R\boldsymbol{a}\tilde{R}$ is a very general way of handling rotations and works for spaces of *any* dimension. In addition, it works for objects of any grade, not only vectors. In 3-D we use the term 'rotor' for those even elements of the space that represent rotations.

Let us consider the problem of rotating a unit vector $\boldsymbol{n}_1$ into another unit vector $\boldsymbol{n}_2$ in 3D space, where the angle between these two vectors in θ. What is the rotor R which performs such a rotation? To answer this question let us consider the scalar plus bivector (spinor) quantity $(1 + \boldsymbol{n}_2\boldsymbol{n}_1)$. If R is the rotor we require then it must satisfy $\boldsymbol{n}_2 = R\boldsymbol{n}_1\tilde{R}$, which under multiplication on the right by R gives

$$\boldsymbol{n}_2 R = R\boldsymbol{n}_1. \tag{27}$$

Since ${\boldsymbol{n}_1}^2 = {\boldsymbol{n}_2}^2 = 1$ we see that we have

$$\begin{aligned} \boldsymbol{n}_2(1 + \boldsymbol{n}_2\boldsymbol{n}_1) &= \boldsymbol{n}_2 + \boldsymbol{n}_1 & (28)\\ (1 + \boldsymbol{n}_2\boldsymbol{n}_1)\boldsymbol{n}_1 &= \boldsymbol{n}_1 + \boldsymbol{n}_2 & (29) \end{aligned}$$

and therefore that equation (27) is satisfied if $R \propto (1 + \boldsymbol{n}_2\boldsymbol{n}_1)$. It remains simply to normalize R so that it satisfies $R\tilde{R} = 1$. If $R = \alpha(1 + \boldsymbol{n}_2\boldsymbol{n}_1)$ we obtain

$$\begin{aligned} R\tilde{R} &= \alpha^2(1 + \boldsymbol{n}_2\boldsymbol{n}_1)(1 + \boldsymbol{n}_1\boldsymbol{n}_2)\\ &= 2\alpha^2(1 + \boldsymbol{n}_2\cdot\boldsymbol{n}_1), & (30) \end{aligned}$$

which gives us the following formula for R

$$R = \frac{1 + \boldsymbol{n}_2\boldsymbol{n}_1}{\sqrt{2(1 + \boldsymbol{n}_2\cdot\boldsymbol{n}_1)}}. \tag{31}$$

Noting that $\sqrt{2(1 + \boldsymbol{n}_2\cdot\boldsymbol{n}_1)} = 2\cos(\frac{\theta}{2})$, the rotor R can be written as

$$\begin{aligned} R &= \cos(\frac{\theta}{2}) + \frac{\boldsymbol{n}_2\wedge\boldsymbol{n}_1}{|\boldsymbol{n}_2\wedge\boldsymbol{n}_1|}\sin(\frac{\theta}{2})\\ &= \exp(\frac{\theta}{2}\frac{\boldsymbol{n}_2\wedge\boldsymbol{n}_1}{|\boldsymbol{n}_2\wedge\boldsymbol{n}_1|}) & (32) \end{aligned}$$

Indeed it can be shown that for a Euclidean space of any dimension all rotors can be written in the form

$$R = e^{B/2}, \tag{33}$$

where B is a bivector representing the plane in which the rotation occurs. In particular, in 3-D we can write equation (32) as

$$R = \exp(-i\frac{\theta}{2}\boldsymbol{n}) = \cos\frac{\theta}{2} - i\boldsymbol{n}\sin\frac{\theta}{2}, \tag{34}$$

which represents a rotation of θ radians about an axis parallel to the unit vector $\boldsymbol{n}$ in a right-handed screw sense. This is precisely how 3-D rotations are represented in the quaternion algebra. If the rotor R_1 takes the vector $\boldsymbol{a}$ to the vector $\boldsymbol{b}$ then

$$\boldsymbol{b} = R_1\boldsymbol{a}\tilde{R}_1.$$

If the vector $\boldsymbol{b}$ is rotated by R_2 to the vector $\boldsymbol{c}$, then

$$\boldsymbol{c} = R_2\boldsymbol{b}\tilde{R}_2,$$

and therefore

$$\boldsymbol{c} = (R_2R_1)\boldsymbol{a}(R_2R_1)\tilde{},$$

which implies that rotors combine in a straightforward manner, i.e. $\boldsymbol{c} = R\boldsymbol{a}\tilde{R}$ where $R = R_2R_1$. This also tells us that rotors form a subclass of spinors.

Now, using nothing other than simple bivectors we can illustrate how the quaternion algebra of Hamilton is simply a subset of the geometric algebra of space. We explicitly identify the $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ of quaternions with

$$\boldsymbol{i} = i\sigma_1, \quad \boldsymbol{j} = -i\sigma_2, \quad \boldsymbol{k} = i\sigma_3. \tag{35}$$

Since $(i\sigma_1)^2 = -1$, $(-i\sigma_2)^2 = -1$, $(i\sigma_3)^2 = -1$ and $(i\sigma_1)(-i\sigma_2)(i\sigma_3) = i\sigma_1\sigma_2\sigma_3 = -1$, the famous Hamilton relations

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1, \tag{36}$$

are easily recovered. Interpreting the $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ as bivectors, we see that they do indeed represent 90° rotations in orthogonal directions and will therefore provide a system particularly suited for the representation of 3-D rotations. It is worth noting here that in the geometric algebra our bivectors $i\sigma_1$,$i\sigma_2$,$i\sigma_3$, are derived from a proper right-handed set of basis vectors $\sigma_1, \sigma_2, \sigma_3$ and represent 90° rotations in the same sense about each of the basis vectors. However, the quantities $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ in the quaternion algebra are not derived from any basis and there is therefore no control on the handedness of the rotations – this explains the appearance of the minus sign in the middle relation of equation (35). Even now, in cases where quaternions are used in research applications, authors often incorrectly regard the $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ as vectors [Weng *et al.* 1989], indicating that there is some degree of confusion in their use.

If a quaternion $\mathcal{A}$ is represented by $[a_0, a_1, a_2, a_3]$, then there exists a one-to-one mapping between quaternions and rotors given by

$$\mathcal{A} = [a_0, a_1, a_2, a_3] \leftrightarrow a_0 + a_1(i\sigma_1) - a_2(i\sigma_2) + a_3(i\sigma_3). \tag{37}$$

Another concept which will be used in subsequent sections is that of the *reciprocal frame.* Given a set of linearly independent vectors $\{\boldsymbol{e}_i\}$ (where no assumption of orthonormality is made), we can form a reciprocal frame, $\{\boldsymbol{e}^i\}$, which is such that

$$\boldsymbol{e}^i \cdot \boldsymbol{e}_j = \delta^i_j. \tag{38}$$

For details of the explicit construction of such a reciprocal frame in n-dimensions see
[Hestenes and Sobczyk 1984]. In three dimensions this is a very simple operation and the reciprocal

frame vectors for a linearly independent set of vectors $\{\boldsymbol{e}_i\}$ are as follows

$$\begin{aligned}
\boldsymbol{e}^1 &= \frac{1}{\alpha} i\boldsymbol{e}_2 \wedge \boldsymbol{e}_3 \\
\boldsymbol{e}^2 &= \frac{1}{\alpha} i\boldsymbol{e}_3 \wedge \boldsymbol{e}_1 \\
\boldsymbol{e}^3 &= \frac{1}{\alpha} i\boldsymbol{e}_1 \wedge \boldsymbol{e}_2,
\end{aligned} \tag{39}$$

where $i\alpha = \boldsymbol{e}_3 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_1$.

There are many other interesting results regarding the geometric algebra of 3-D space but those given above are sufficient for understanding the discussions to follow in subsequent sections.

3 Geometric techniques for computer vision applications

In Section 2 we claimed that rotors are a very natural way of expressing rotations in any dimension. In three dimensions any quaternionic means of dealing with rotations can be replaced by a simpler method using rotors. In this section we will look at some general techniques and methods involving rotor manipulations which can be used in many computer vision problems.

3.1 Differentiation with respect to multivector quantities

Here we will give a brief discussion of the process of differentiating with respect to any multivector – we will omit a number of the proofs of the relations we write down but refer the interested reader to [Hestenes and Sobczyk 1984]. Having available such differentiation means that, in practice, it is easy to take derivatives with respect to rotors and vectors and this makes many least-squares minimization problems much simpler to deal with. In computer vision and motion analysis one tends to draw frequently on an approach which minimizes some expression in order to find the relevant rotations and translations – this is a standard technique for any estimation problem in the presence of uncertainty.

If X is a mixed-grade multivector, $X = \sum_r X_r$, and $F(X)$ is a general multivector-valued function of X, then the derivative of F in the A 'direction' (where A has the same grades as X) is written as $A * \partial_X F(X)$ (here we use $*$ to denote the scalar part of the product of two multivectors, i.e. $A * B \equiv \langle AB \rangle$), and is defined as

$$A * \partial_X F(X) \equiv \lim_{\tau \to 0} \frac{F(X + \tau A) - F(X)}{\tau}. \tag{40}$$

We impose the constraint that A must have the same grades as X so that the differentiation in terms of the limit makes some physical sense. If X contains no terms of grade-r and A_r is a homogeneous multivector, then we define $A_r * \partial_X = 0$. This definition of the derivative also ensures that the operator $A * \partial_X$ is a scalar operator and satisfies all of the usual partial derivative properties. We can now use the above definition of the directional derivative to formulate a general expression for the multivector derivative ∂_X without reference to one particular direction. This is accomplished by introducing an arbitrary frame $\{\boldsymbol{e}_j\}$ and extending this to a basis (vectors, bivectors, etc..) for the entire algebra, $\{\boldsymbol{e}_J\}$. Then ∂_X is defined as

$$\partial_X \equiv \sum_J \boldsymbol{e}^J \boldsymbol{e}_J * \partial_X, \tag{41}$$

where $\{\boldsymbol{e}^J\}$ is an extended basis built out of the reciprocal frame. The directional derivative, $\boldsymbol{e}_J * \partial_X$, is only non-zero when $\boldsymbol{e}_J$ is one of the grades contained in X (as previously discussed) so that ∂_X inherits the multivector properties of its argument X. Although we have here defined the multivector derivative using an extended basis, it should be noted that the sum over all the basis ensures that ∂_X is independent of the choice of $\{e_j\}$ and so all of the properties of ∂_X can be formulated in a frame-free manner. One of the most useful results concerning multivector derivatives is

$$\partial_X \langle XB \rangle = \mathrm{P}_X(B), \tag{42}$$

where $\mathrm{P}_X(B)$ is the projection of the multivector B onto the grades contained in X. We can see this as follows. Since

$$\partial_X \langle XB \rangle = \sum_J \boldsymbol{e}^J \boldsymbol{e}_J * \partial_X \langle XB \rangle \tag{43}$$

and $\boldsymbol{e}_J * \partial_X = 0$ if $\boldsymbol{e}_J$ is not a grade of X, we see that

$$\begin{aligned} \partial_X \langle XB \rangle &= \sum_{J'} \boldsymbol{e}^{J'} \lim_{\tau \to 0} \frac{\langle (X + \tau \boldsymbol{e}'_J) B \rangle - \langle XB \rangle}{\tau} \\ &= \sum_{J'} \boldsymbol{e}^{J'} \langle \boldsymbol{e}_{J'} B \rangle \\ &= P_X(B) \end{aligned} \tag{44}$$

where the sum over J' is over those grades contained in X. Using this result the following relations can be shown to hold

$$\partial_X \langle \tilde{X} B \rangle = \mathrm{P}_X(\tilde{B}) \tag{45}$$

$$\partial_{\tilde{X}} \langle \tilde{X} B \rangle = \mathrm{P}_{\tilde{X}}(B) = \mathrm{P}_X(B) \tag{46}$$

$$\partial_\psi \langle M \psi^{-1} \rangle = -\psi^{-1} \mathrm{P}_\psi(M) \psi^{-1} \qquad \text{for general multivectors } \psi \text{ and } M. \tag{47}$$

It is often convenient to indicate, via an overdot, the quantity on which ∂_X operates. For example, $\dot{\partial}_X A \dot{B}$ means that the derivative part of ∂_X act on B. A complete discussion of the geometric calculus is given in [Hestenes and Sobczyk 1984], where the results in equations (45) and (46) are proved. The result in equation (47) is discussed in [Doran 1994].

One often wants to take the derivative, ∂_a, with respect to a vector quantity $\boldsymbol{a}$. If we replace A by $\boldsymbol{a}$, $\boldsymbol{e}^J$ by $\boldsymbol{e}^j$ and $\boldsymbol{e}_J$ by $\boldsymbol{e}_j$ in equation (41) and use the definition in equation (40) we see that the differential operator ∂_a can be written as

$$\partial_a = \boldsymbol{e}^i \frac{\partial}{\partial a^i} \qquad \text{where} \qquad \boldsymbol{a} = a^i \boldsymbol{e}_i. \tag{48}$$

This will be used several times in the following sections. Note that we will not write vectors in bold when they appear as subscripts in the vector derivative. At this point it is appropriate to give some explanation of the expansion of $\boldsymbol{a}$ used above, namely $\boldsymbol{a} = a^i \boldsymbol{e}_i$. Recall that our basis $\{\boldsymbol{e}_i\}$ was not necessarily an orthonormal basis and that we were able to define a reciprocal basis $\{\boldsymbol{e}^i\}$, such that $\boldsymbol{e}^i\cdot\boldsymbol{e}_j = \delta_{ij}$. Consider the component of the vector $\boldsymbol{a}$ in the $\boldsymbol{e}^j$ direction – this is given by $\boldsymbol{a}\cdot\boldsymbol{e}^j$ and we will call this component a^j, where the upstairs index tells us that it is the component of $\boldsymbol{a}$ in the direction of the jth basis vector of the *reciprocal* frame. Similarly, the component of $\boldsymbol{a}$ in the $\boldsymbol{e}_j$ direction is given by $\boldsymbol{a}\cdot\boldsymbol{e}_j$ which we call a_j. We note the useful identities

$$\boldsymbol{a} = (\boldsymbol{a}\cdot\boldsymbol{e}_j)\boldsymbol{e}^j \qquad \text{and} \qquad (\boldsymbol{a}\cdot\boldsymbol{e}^j)\boldsymbol{e}_j. \tag{49}$$

Of course, if the basis is an orthonormal basis, say $\{\sigma_i\}$, then $\sigma^i = \sigma_i$ and we would, without ambiguity write $\boldsymbol{a} = a_i\sigma_i$.

As an example of multivector differentiation we will consider the problem of finding the rotor R which 'most closely' rotates the vectors $\{\boldsymbol{u}_i\}$ onto the vectors $\{\boldsymbol{v}_i\}$, $i = 1, ..., n$. More precisely, we wish to find the rotor R which minimizes

$$\phi = \sum_{i=1}^{n}(\boldsymbol{v}_i - R\boldsymbol{u}_i\tilde{R})^2. \tag{50}$$

Expanding ϕ gives

$$\begin{aligned}\phi &= \sum_{i=1}^{n}(\boldsymbol{v}_i{}^2 - \boldsymbol{v}_iR\boldsymbol{u}_i\tilde{R} - R\boldsymbol{u}_i\tilde{R}\boldsymbol{v}_i + R(\boldsymbol{u}_i{}^2)\tilde{R}) \\ &= \sum_{i=1}^{n}\{(\boldsymbol{v}_i{}^2 + \boldsymbol{u}_i{}^2) - 2\langle\boldsymbol{v}_iR\boldsymbol{u}_i\tilde{R}\rangle\}.\end{aligned} \tag{51}$$

To minimize ϕ we choose not to differentiate directly with respect to R since the definition of R involves the constraint $R\tilde{R} = 1$, and this would have to be included via a Lagrange multiplier. Instead we use equation (47) to take the multivector derivative of ϕ with respect to ψ, where we replace $R\boldsymbol{u}_i\tilde{R}$ with $\psi\boldsymbol{u}_i\psi^{-1}$.

$$\begin{aligned}\partial_\psi\phi(\psi) &= -2\sum_{i=1}^{n}\partial_\psi\langle\boldsymbol{v}_i\psi\boldsymbol{u}_i\psi^{-1}\rangle \\ &= -2\sum_{i=1}^{n}\{\dot{\partial}_\psi\langle\dot{\psi}A_i\rangle + \dot{\partial}_\psi\langle B_i\dot{\psi}^{-1}\rangle\},\end{aligned} \tag{52}$$

where $A_i = \boldsymbol{u}_i\psi^{-1}\boldsymbol{v}_i$ and $B_i = \boldsymbol{v}_i\psi\boldsymbol{u}_i$ (using the cyclic reordering property). The first term is easily evaluated to give A_i. To evaluate the second term we can use equation (47). One can then substitute $\psi = R$ and note that $R^{-1} = \tilde{R}$ as $R\tilde{R} = 1$. We then have

$$\begin{aligned}\partial_\psi\phi(\psi) &= -2\sum_{i=1}^{n}\{\boldsymbol{u}_i\psi^{-1}\boldsymbol{v}_i - \psi^{-1}(\boldsymbol{v}_i\psi\boldsymbol{u}_i)\psi^{-1}\} \\ &= -2\psi^{-1}\sum_{i=1}^{n}\{(\psi\boldsymbol{u}_i\psi^{-1})\boldsymbol{v}_i - \boldsymbol{v}_i(\psi\boldsymbol{u}_i\psi^{-1})\} \\ &= 4\tilde{R}\sum_{i=1}^{n}\boldsymbol{v}_i\wedge(R\boldsymbol{u}_i\tilde{R}).\end{aligned} \tag{53}$$

Thus the rotor which minimizes the least-squares expression $\phi(R) = \sum_{i=1}^{n}(\boldsymbol{v}_i - R\boldsymbol{u}_i\tilde{R})^2$ must satisfy

$$\sum_{i=1}^{n}\boldsymbol{v}_i\wedge(R\boldsymbol{u}_i\tilde{R}) = 0. \tag{54}$$

This is intuitively obvious – we want the R which makes $\boldsymbol{u}_i$ 'most parallel' to $\boldsymbol{v}_i$ in the average sense. The solution of equation (54) for R will utilize the linear algebra framework of geometric algebra and will be described in Section 3.3.

3.2 Some formulae for general rotors

In Section 2.3 we saw that the rotor R which rotates a unit vector $\boldsymbol{a}$ into a unit vector $\boldsymbol{b}$ while leaving all vectors perpendicular to the plane containing $\boldsymbol{a}$ and $\boldsymbol{b}$ unchanged is given by

$$R = \frac{1+\boldsymbol{ba}}{\sqrt{2(1+\boldsymbol{b}\cdot\boldsymbol{a})}}, \tag{55}$$

so that $\boldsymbol{b} = R\boldsymbol{a}\tilde{R}$. We can see this via the procedure outlined in Section 2.3 or by decomposing the rotation into 2 reflections, the first in the plane perpendicular to $\boldsymbol{n} = \frac{(\boldsymbol{a}+\boldsymbol{b})}{|\boldsymbol{a}+\boldsymbol{b}|}$ and the second in the plane perpendicular to $\boldsymbol{b}$.

In 3 dimensions, equation (55) gives us this rotor in a very concise manner. We now look at a related problem: given a known set of three vectors which are rotated to give another known set of three vectors, how do we obtain the rotor R which performs this rotation?

Suppose we have two sets of vectors $\{\boldsymbol{e}_i\}$ and $\{\boldsymbol{f}_i\}$, $i = 1,2,3$, which are related via

$$\boldsymbol{f}_i = R\boldsymbol{e}_i\tilde{R} \qquad\qquad i = (1,2,3), \tag{56}$$

where R is some rotor and no assumption of orthonormality is made. As R is a rotor, it is possible to write it as $R = R_0 + R_2$, where R_0 is a scalar and R_2 is a bivector (see Section 2.3). We have seen that if the vectors $\{\boldsymbol{e}_i\}$ are linearly independent, we can form a reciprocal frame, $\{\boldsymbol{e}^j\}$. Now consider the product $\boldsymbol{e}^i\boldsymbol{f}_i$ (summation implied). From equation (56) this product can be written as

$$\boldsymbol{e}^i\boldsymbol{f}_i = \boldsymbol{e}^iR\boldsymbol{e}_i\tilde{R} = (\boldsymbol{e}^iR_0\boldsymbol{e}_i + \boldsymbol{e}^iR_2\boldsymbol{e}_i)\tilde{R}. \tag{57}$$

R_0 is a scalar, and therefore $\boldsymbol{e}^iR_0\boldsymbol{e}_i = 3R_0$. To simplify the term $\boldsymbol{e}^iR_2\boldsymbol{e}_i$ we note that it can be written as $\partial_aR_2\boldsymbol{a}$, where the differential operator ∂_a is as given in equation (48). We can see this equivalence as follows:

$$\begin{aligned} \partial_aR_2\boldsymbol{a} &= \boldsymbol{e}^i\frac{\partial}{\partial a^i}(R_2a^j\boldsymbol{e}_j) \\ &= \boldsymbol{e}^iR_2\delta_i^j\boldsymbol{e}_j \\ &= \boldsymbol{e}^iR_2\boldsymbol{e}_i. \end{aligned} \tag{58}$$

Since $\partial_aR_2\boldsymbol{a}$ is independent of the basis we choose, we can rewrite $\boldsymbol{e}^iR_2\boldsymbol{e}_i$ as $\sigma^iR_2\sigma_i$. In this standard orthogonal basis we can write R_2 (a bivector) as $R_2 = R_2^1i\sigma_1 + R_2^2i\sigma_2 + R_2^3i\sigma_3$; $\sigma^iR_2\sigma_i$ can then be evaluated to give

$$\sigma^iR_2\sigma_i = \sigma^i(R_2^1i\sigma_1 + R_2^2i\sigma_2 + R_2^3i\sigma_3)\sigma_i = -R_2. \tag{59}$$

Using equation (59) we can now write

$$\begin{aligned} \boldsymbol{e}^i\boldsymbol{f}_i &= (3R_0 - R_2)\tilde{R} \\ &= (3R_0 - R_2)(R_0 - R_2) \\ &= (4R_0^2 - 1) - 4R_0R_2. \end{aligned} \tag{60}$$

The final step in equation (60) uses $R\tilde{R} = R_0^2 - R_2^2 = 1$. If we now use the fact that $R_0\tilde{R} = R_0^2 - R_0R_2$, we can obtain an expression for R in terms of only $\boldsymbol{e}^i$ and $\boldsymbol{f}_i$;

$$\begin{aligned} 4R_0\tilde{R} &= 1 + \boldsymbol{e}^i\boldsymbol{f}_i \\ \Rightarrow R &\propto 1 + \boldsymbol{f}_i\boldsymbol{e}^i. \end{aligned} \tag{61}$$

See also [Hestenes 1986a] for an alternative derivation of this result. If our set of three vectors $\{\boldsymbol{e}_i\}$ are not linearly independent, but are coplanar, then we are still able to use equation (61) to recover the rotor R. More explicitly, we rename the sets of coplanar vectors $(\boldsymbol{e}_1, \boldsymbol{e}_2, \hat{\boldsymbol{e}}_3)$ and $(\boldsymbol{f}_1, \boldsymbol{f}_2, \hat{\boldsymbol{f}}_3)$, and then define two additional vectors $\boldsymbol{e}_3 = \frac{i\boldsymbol{e}_1\wedge\boldsymbol{e}_2}{|i\boldsymbol{e}_1\wedge\boldsymbol{e}_2|}$ and $\boldsymbol{f}_3 = \frac{i\boldsymbol{f}_1\wedge\boldsymbol{f}_2}{|i\boldsymbol{f}_1\wedge\boldsymbol{f}_2|}$. It is easy to show that we then have $\boldsymbol{f}_3 = R\boldsymbol{e}_3\tilde{R}$. We can then form the reciprocal frame from $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ as before and the rotor R is again given by equation (61). Of course, in this coplanar case, we can work with any two of the original set of three vectors, since the rotor which rotates any two $\boldsymbol{e}$-vectors to their corresponding $\boldsymbol{f}$-vectors must also rotate the third $\boldsymbol{e}$-vector to its corresponding $\boldsymbol{f}$-vector. Thus, as a by-product of the case of rotating three vectors into another three vectors, we can easily see how we should handle the case of rotating a set of two vectors into another set of two vectors.

In finding an explicit form for the rotor we have at our disposal all the power of the geometric algebra approach to rotations, and it is perhaps remarkable that such a simple form exists for this case.

3.3 Linear algebra

Geometric algebra is a very natural framework for the study of linear functions and non-orthonormal frames; this approach is discussed at length in [Hestenes and Sobczyk 1984]. Here we will give a brief account of how geometric algebra deals with linear algebra; we do this since many computer vision problems can be formulated as problems in linear algebra.

If we take a linear function $f(\boldsymbol{a})$ which maps vectors to vectors in the same space then it is possible to extend f to act linearly on multivectors. This extension of f is written as $\underline{f}$ and is given by

$$\underline{f}(\boldsymbol{a}_1\wedge\boldsymbol{a}_2\wedge\ldots\wedge\boldsymbol{a}_r) \equiv f(\boldsymbol{a}_1)\wedge f(\boldsymbol{a}_2)\wedge\ldots\wedge f(\boldsymbol{a}_r). \tag{62}$$

Thus, f maps an r-grade multivector to another r-grade multivector. The **adjoint** to $\underline{f}$ is written as $\overline{f}$ and defined by

$$\overline{f}(\boldsymbol{a}) \equiv \boldsymbol{e}^i\langle\underline{f}(\boldsymbol{e}_i)\boldsymbol{a}\rangle, \tag{63}$$

where, as before, $\{\boldsymbol{e}_i\}$ is an arbitrary frame and $\{\boldsymbol{e}^i\}$ is its reciprocal frame. One can then use equation (48) to show that a frame independent definition of the adjoint is

$$\overline{f}(\boldsymbol{a}) \equiv \partial_b\langle\boldsymbol{a}\underline{f}(\boldsymbol{b})\rangle. \tag{64}$$

A consequence of these definitions is that, for vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, we have $\underline{f}(\boldsymbol{a})\cdot\boldsymbol{b} = \overline{f}(\boldsymbol{b})\cdot\boldsymbol{a}$; the adjoint therefore represents the function corresponding to the transpose of the matrix which is represented by $\underline{f}$. If $\underline{f} = \overline{f}$, i.e. a function is self-adjoint, then it is a **symmetric** function. This will mean that the matrix representation of $\underline{f}$, say F, is a symmetric matrix. Similarly if $\underline{f} = -\overline{f}$ then the function is antisymmetric. It is shown in [Hestenes and Sobczyk 1984] that if $\underline{f}$ satisfies

$$\partial_a \wedge \underline{f}(\boldsymbol{a}) = 0, \tag{65}$$

then the function $\underline{f}$ is symmetric (the left hand side of the above expression is understood to mean $\langle \partial_a \underline{f}(\boldsymbol{a}) \rangle_2$ or equivalently $\boldsymbol{e}^i \wedge \frac{\partial}{\partial a^i} \underline{f}(\boldsymbol{a})$). The proof of this result relies on the fact that the vector derivative ∂_a has the same algebraic properties as vectors, which means that one is able to use standard vector identities with the vector replaced with ∂_a.

As an illustration of the use of linear algebra techniques, we will discuss the solution of equation (54). Using the definition of the vector derivative given in equation (48), it is possible to rewrite equation (54) as

$$\partial_a \wedge \left[\sum_{i=1}^{n} R[(\boldsymbol{a}\cdot\boldsymbol{v}_i)\boldsymbol{u}_i]\tilde{R} \right] = 0. \tag{66}$$

We now introduce a function $\underline{f}$ defined by

$$\underline{f}(\boldsymbol{a}) = \sum_{i=1}^{n} (\boldsymbol{a}\cdot\boldsymbol{v}_i)\boldsymbol{u}_i. \tag{67}$$

Equation (66) can then be written as

$$\partial_a \wedge R\underline{f}(\boldsymbol{a})\tilde{R} = 0. \tag{68}$$

Let us now define another function $\underline{R}$ mapping vectors onto vectors such that $\underline{R}\boldsymbol{a} = R\boldsymbol{a}\tilde{R}$. With these definitions equation (68) takes the form

$$\partial_a \wedge \underline{R}\underline{f}(\boldsymbol{a}) = 0, \tag{69}$$

for any vector $\boldsymbol{a}$. This then tells us that if $\underline{G} = \underline{R}\underline{f}$ then $\underline{G}$ is symmetric. Since R is a rotor and is orthogonal, we have that $R^{-1} = \tilde{R}$ from which it follows that $\underline{R}^{-1} = \overline{R}$. This enables us to write $\underline{f}$ as

$$\underline{f}(\boldsymbol{a}) = \overline{R}\underline{G}(\boldsymbol{a}). \tag{70}$$

If we now perform a singular-value decomposition (SVD) on $\underline{f}$, so that $\underline{f} = \underline{U}\ \underline{S}\ \underline{V}$, where $\underline{U}$ and $\underline{V}$ are orthogonal and $\underline{S}$ is diagonal (this is equivalent to saying that the matrix representations of these functions are respectively orthogonal and diagonal), we have

$$\underline{f} = \underline{U}\ \underline{V}(\overline{V}\underline{S}\ \underline{V}) \equiv \overline{R}\underline{G}. \tag{71}$$

$\overline{V}\underline{S}\ \underline{V}$ is obviously symmetric, as is $\underline{G}$, which tells us that the rotation $\underline{R}$ must be given by

$$\underline{R} = \overline{V}\ \overline{U}. \tag{72}$$

The rotation $\underline{R}$ is therefore found in terms of the SVD of the function $\underline{f}$.

It is useful to summarize the procedure, as we have necessarily had to include a considerable amount of detail in order to establish the ultimately simple solution. The computational method would be to take the two sets of vectors $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{v}_i\}$ and form F, the matrix representation of $\underline{f}$ by

$$\begin{aligned} F_{\alpha\beta} &\equiv \sigma_\alpha \cdot \underline{f}(\sigma_\beta) \\ &= \sum_{i=1}^{n} (\sigma_\alpha \cdot \boldsymbol{u}_i)(\sigma_\beta \cdot \boldsymbol{v}_i). \end{aligned} \tag{73}$$

An SVD of F gives $F = USV^T$ and the rotation matrix $\mathcal{R}$ is then simply given by the product $\mathcal{R} = VU^T$. At this point it should be noted that it is straightforward to recover the rotor R from the rotation matrix $\mathcal{R}$, but that there is a sign ambiguity in this process. Since the action of a rotor is a two-sided operation, it is easy to see that $-R$ will have the same effect as R,

$$R\boldsymbol{a}\tilde{R} = (-R)\boldsymbol{a}(-\tilde{R}). \tag{74}$$

In the algorithm to estimate structure and motion which will be described in subsequent sections the rotor always acts two-sidedly (i.e. we are always simply rotating a vector) and so the sign of R makes no difference. We note at this point that although the final solution is in matrix terms, all the previous analytic manipulations were carried out using geometric algebra.

4 Applications to the structure and motion estimation problem

In this section we will consider two specific applications of the geometric methods discussed in the previous sections. The first looks at the reconstruction of camera motion from point correspondences in different views when the range data is known, a situation often referred to as *3D-to-3D correspondences*. The second addresses the case in which the range data is unavailable and we must estimate not only the camera motion but also the range coordinate of each of the points considered, the so-called *2D-to-2D correspondence* case. Both techniques assume a number of point correspondences in the images and allow for errors in the measured data; an optimal solution is found by minimizing a least squares expression. Obviously, the scenario in which the camera is stationary and the object moves can be analysed in precisely the same way. In the case of 3D-to-3D correspondences the method presented here is effectively equivalent to many existing algorithms. However, in the case of 2D-to-2D correspondences the algorithm we present is, to the best of our knowledge, the first to solve the structure and motion problem simultaneously using analytic derivatives – most previous solutions to this problem can be split into two types, those that estimate motion first and then structure and those that estimate structure first and motion second [Huang and Netravali 1994]. It is clear that, in any problem where one wants to estimate parameters given noisy data, it is optimal to estimate all parameters simultaneously if this is possible. A process which obtains a solution in stages leads to errors which can build up in an unpredictable fashion. For example, errors in estimating motion will affect the subsequent estimation of the structure in a much more serious way than estimating all unknowns in a global minimization scheme. We illustrate the performance of this algorithm by comparing it with a *motion-first* algorithm and show that it can result in significant improvements.

4.1 Camera motion from two scene projections: range data known

There have been many methods proposed for the solution of this problem. Indeed, many of these methods are almost exactly equivalent but have used different formalisms for their construction. Sabata and Aggarwal [Sabata and Aggarwal 1991] review most of the current techniques and make the very important point that the best techniques are those which take into account the fact that the transformation we are looking for is a *motion transformation* and not a general affine

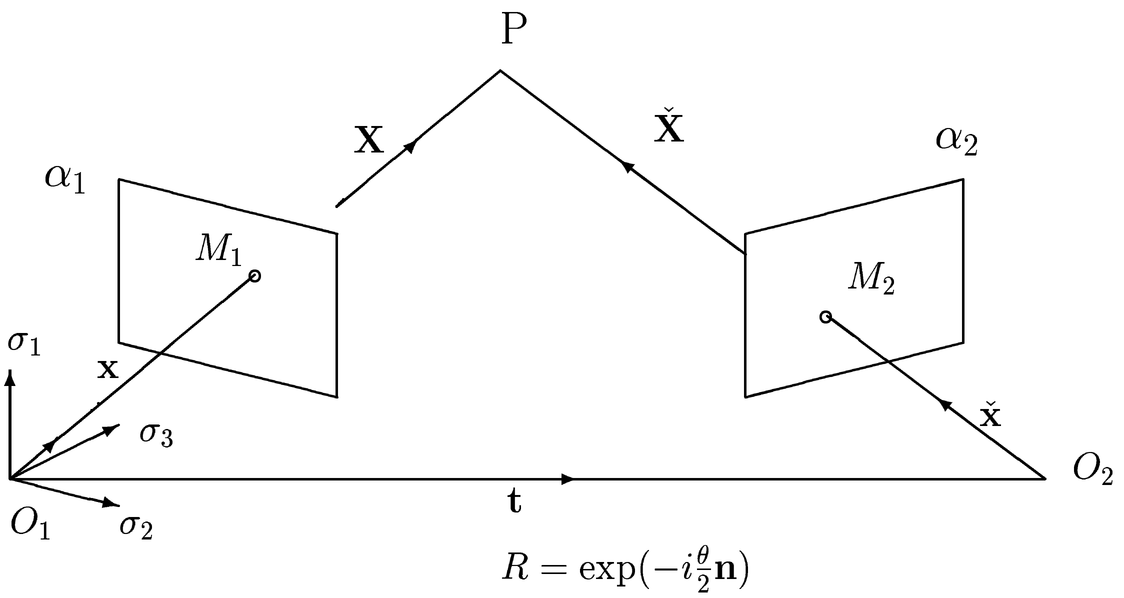

Figure 3: Point in object viewed in two camera positions

transformation (which may not be feasible for a rigid body). Of the techniques which employ the constraints relevant to rigid body motion, those which involve a least-squares approach are generally preferred. Similarly, the more recent paper of [Huang and Netravali 1994] presents a very thorough review of algorithms for determining structure and motion and deals with the case of 3D-to-3D correspondences for both point and line matching. Lin *et al.* 1986 solve the least-squares expression recursively, while Arun *et al.* 1987 provide a closed form solution using SVD techniques. Horn 1987 and Horn *et al.* 1988 similarly obtain closed form solutions via orthonormal matrices and quaternionic methods respectively. Walker *et al.* 1991 use dual quaternions to obtain their closed form solution of the problem. In what follows it will become apparent that our geometric algebra approach results in a technique which is very similar to the methods used in [Arun *et al.* 1987, Horn 1987, Horn *et al.* 1988, Walker *et al.* 1991]. Since each of these techniques correctly solve the same least squares problem it follows that they are essentially equivalent, although the use of different calculational procedures may give rise to slightly different numerical results. This would appear to contradict claims in [Walker *et al.* 1991] that the dual quaternion method is superior to the SVD approach. The attractiveness of the geometric algebra approach is that we can apply exactly the same procedures to the case where the range data is unknown as we shall see in Section 4.2.

We will assume in what follows that the camera calibration is known. Figure 3 shows the camera at two spatial positions 1 and 2. At each position the camera observes some point P in the scene. The image planes of the camera are α_1 and α_2. Let O_1 be the optical centre of the camera at position 1 and consider a frame with origin O_1 and axes $\{\sigma_1, \sigma_2, \sigma_3\}$, where σ_3 is perpendicular to the plane α_1. Let $\mathbf{X} = \overrightarrow{O_1P}$ and $\boldsymbol{x} = \overrightarrow{O_1M_1}$ be the position vectors of P and M_1 relative to the axes at O_1 respectively, where M_1 is the projection of the point P onto the plane α_1.

Now suppose the camera undergoes a displacement taking it to position 2. Such a general displacement will consist of a translation plus a rotation. Let the translation be given by the vector $\boldsymbol{t}$ such that $\overrightarrow{O_1O_2} = \boldsymbol{t}$, where O_2 is the translation of O_1. A general rotation of θ about some axis $\boldsymbol{n}$ will be described by the rotor R, where $R = \exp(-i\frac{\theta}{2}\boldsymbol{n})$. Thus, the frame $\{\sigma_1, \sigma_2, \sigma_3\}$ is rotated to a frame $\{\sigma'_1, \sigma'_2, \sigma'_3\}$ at O_2 where $\sigma'_i = R\sigma_i\tilde{R}$ for $i = 1, 2, 3$. At position 2 we have a new image plane α_2, and we let M_2 be the projection of P onto α_2. If $\check{\mathbf{X}} = \overrightarrow{O_2P}$ then it is clear that

$$\check{\mathbf{X}} = \mathbf{X} - \boldsymbol{t}. \tag{75}$$

Now, our observables in the $\{\sigma_i\}$ frame are $X_i = \mathbf{X}\cdot\sigma_i$. In the $\{\sigma'_i\}$ frame they are $X'_i = \check{\mathbf{X}}\cdot\sigma'_i$. If we define the vector $\mathbf{X}'$ which lives in the $\{\sigma_i\}$ frame as $\mathbf{X}' = X'_i\sigma_i$ then we can write this vector as

$$\begin{aligned} \mathbf{X}' &= X'_i(\tilde{R}\sigma'_iR) = \tilde{R}(X'_i\sigma'_i)R \\ &= \tilde{R}\check{\mathbf{X}}R \\ &= \tilde{R}(\mathbf{X} - \boldsymbol{t})R \end{aligned} \tag{76}$$

Thus, $\mathbf{X}$ and $\mathbf{X}'$ are related by

$$\mathbf{X}' = \tilde{R}(\mathbf{X} - \boldsymbol{t})R \quad \text{or} \quad \mathbf{X} = R\mathbf{X}'\tilde{R} + \boldsymbol{t}. \tag{77}$$

In what follows we shall take the camera displacement as the unit of distance so that $\boldsymbol{t}^2 = 1$.

Suppose we have n point correspondences in the two views and that the coordinates $\{\mathbf{X}_i\}$ and $\{\mathbf{X}'_i\}$ are known in each of these views. In any realistic case we would expect measurement errors on both sets of coordinates; the camera motion will then be best recovered using a least-squares approach.

For n matched points in the two frames, we want to find the R and $\boldsymbol{t}$ which minimize

$$S = \sum_{i=1}^{n} \left[\boldsymbol{X'}_i - \tilde{R}(\boldsymbol{X}_i - \boldsymbol{t})R\right]^2. \tag{78}$$

If the $\{\mathbf{X}_i\}$ and $\{\mathbf{X'}_i\}$ are our observed quantities, then we must minimize with respect to R and $\boldsymbol{t}$. The differentiation with respect to $\boldsymbol{t}$ is straightforward;

$$\partial_t S = 2\sum_{i=1}^{n} \left[\boldsymbol{X'}_i - \tilde{R}(\boldsymbol{X}_i - \boldsymbol{t})R\right] \partial_t(\tilde{R}\boldsymbol{t}R). \tag{79}$$

At the minimum, $\partial_t S = 0$ and therefore

$$\sum_{i=1}^{n} \left[\boldsymbol{X'}_i - \tilde{R}(\boldsymbol{X}_i - \boldsymbol{t})R\right] = 0. \tag{80}$$

Solving this gives us an expression for $\boldsymbol{t}$ in terms of R and the data;

$$\check{\boldsymbol{t}} = \frac{1}{n}\sum_{i=1}^{n} \left[\tilde{R}\boldsymbol{X}_iR - \boldsymbol{X'}_i\right] \tag{81}$$

or

$$\boldsymbol{t} = \frac{1}{n}\sum_{i=1}^{n} \left[\boldsymbol{X}_i - R\boldsymbol{X'}_i\tilde{R}\right], \tag{82}$$

where $\check{\boldsymbol{t}} = \tilde{R}\boldsymbol{t}R$. This is obviously equivalent to saying that $\boldsymbol{t} = \bar{\boldsymbol{X}} - R\bar{\boldsymbol{X}}'\tilde{R}$ where $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{X}}'$ are the centroids of the data points in the two views. It is indeed a well known result that the value of

$\boldsymbol{t}$ which minimizes the least squares expression is the difference between the centroid of one set of points and the rotated centroid of the other set of points [Faugeras and Hebert 1983, Huang 1986, Horn *et al.* 1988] – although in the past this result has been arrived at by methods other than direct differentiation.

We now turn to the differentiation of equation (78) with respect to R. As the expression for S is very similar to that given in equation (50), with $\boldsymbol{v}_i = \boldsymbol{X}'_i$ and $\boldsymbol{u}_i = \boldsymbol{X}_i - \boldsymbol{t}$, the solution is as outlined in Sections 3.1 and 3.3 and can be directly written down as

$$\sum_{i=1}^{n} \boldsymbol{X}'_i \wedge \tilde{R}(\boldsymbol{X}_i - \boldsymbol{t})R = 0. \tag{83}$$

If we substitute our optimal value of $\boldsymbol{t}$ in this equation we have

$$\sum_{i=1}^{n} \boldsymbol{X}'_i \wedge \tilde{R}(\boldsymbol{X}_i - \bar{\boldsymbol{X}} - R\bar{\boldsymbol{X}}'\tilde{R})R = 0. \tag{84}$$

Noting that the $\sum_{i=1}^{n} \boldsymbol{X}'_i \wedge \bar{\boldsymbol{X}}'$ term vanishes, this reduces to $\sum_{i=1}^{n} \boldsymbol{v}_i \wedge \tilde{R}\boldsymbol{u}_i R = 0$ with

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{X}_i - \overline{\boldsymbol{X}} \\ \boldsymbol{v}_i &= \boldsymbol{X}'_i. \end{aligned} \tag{85}$$

The rotor $\tilde{R}$ is then found from an SVD of the matrix F where F is defined in terms of the $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ as given in equation (73). A comparison of this closed form solution to the SVD technique of [Arun *et al.* 1987], the orthonormal matrix approach of [Horn *et al.* 1988] and the dual quaternion approach of [Walker *et al.* 1991], reveals that each method should effectively produce the same results.

4.2 Camera motion and structure from two scene projections: range data unknown

We now look at the more difficult problem of extracting the camera motion when only 2D projections of the true 3D coordinate sets are available. Sabata and Aggarwal 1991 stress that in this case the errors in the estimation of the range coordinates will tend to adversely effect the estimation of the motion parameters or vice versa. In [Huang and Netravali 1994] this 2D-to-2D correspondence problem is discussed in some detail for a variety of features. For points, **all** algorithms considered are effectively *two stage* algorithms. In [Mitchie and Aggarwal 1986] the structure is estimated first by the application of rigidity constraints; these can then be used to estimate the motion. It is generally acknowledged that the errors incurred in estimating structure first are greater than when one estimates motion first. Consequently there are many more algorithms which attempt to solve the 2D-to-2D correspondence problem using a *motion first* approach. One such method is described in [Weng *et al.* 1989]; this is a linear algorithm based on point correspondences which gives a closed-form solution for estimating first motion and then structure in the presence of noise. Since this technique has become one of the established tools in structure and motion estimation the results section will compare the geometric algebra (GA) algorithm with the Weng *et al.* method.

Given the drawbacks with two-stage algorithms it is obviously wise to attempt an optimization of the appropriate least squares expression simultaneously over *all* of the unknowns in the problem.

We can write $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$, the vectors to the image points in the two views, as

$$\boldsymbol{x}_i = \frac{f}{\mathbf{X}_i\cdot\sigma_3}\mathbf{X}_i, \qquad \boldsymbol{x}'_i = \frac{f}{\mathbf{X}'_i\cdot\sigma_3}\mathbf{X}'_i, \tag{86}$$

where f is the focal length. Taking f as 1, and writing $X_{i3} = \mathbf{X}_i\cdot\sigma_3$, $X'_{i3} = \mathbf{X}'_i\cdot\sigma_3$, the least squares expression in equation (78) can be rewritten as

$$S = \sum_{i=1}^{n} \Big[X'_{i3}\boldsymbol{x}'_i - \tilde{R}(X_{i3}\boldsymbol{x}_i - \boldsymbol{t})R\Big]^2. \tag{87}$$

The $\{\boldsymbol{x}_i\}$ and the $\{\boldsymbol{x}'_i\}$ are now our observed quantities (which will suffer from possible measurement errors) and the set of unknowns in the problem increases considerably to $\{R, \boldsymbol{t}, (X_{i3}, X'_{i3}, i = 1, n)\}$. In order to minimize equation (87) we adopt the most general approach we can and differentiate S with respect to R, $\boldsymbol{t}$ and the range coordinates $\{X_{i3}, X'_{i3}\}$.

From the previous sections we know that differentiating with respect to $\boldsymbol{t}$ and R give the following equations;

$$\boldsymbol{t} = \frac{1}{n}\sum_{i=1}^{n}\Big[X_{i3}\boldsymbol{x}_i - X'_{i3}R\boldsymbol{x}'_i\tilde{R}\Big] \tag{88}$$

$$\sum_{i=1}^{n} X'_{i3}\boldsymbol{x}'_i\wedge\tilde{R}\left(X_{i3}\boldsymbol{x}_i - \frac{1}{n}\sum_{j=1}^{n}X_{j3}\boldsymbol{x}_j\right)R = 0. \tag{89}$$

Note now that we cannot sensibly find the 'centroid' of the data sets. Differentiation of S with respect to each of the X_{i3} and X'_{i3} is simply scalar differentiation, and solutions to the equations $\frac{\partial S}{\partial X_{i3}} = 0$ and $\frac{\partial S}{\partial X'_{i3}} = 0$ are as follows (no summation implied):

$$X_{i3} = \frac{-(\boldsymbol{x}'_i\wedge\boldsymbol{s})\cdot(\boldsymbol{q}_i\wedge\boldsymbol{x}'_i)}{(\boldsymbol{q}_i\wedge\boldsymbol{x}'_i)\cdot(\boldsymbol{q}_i\wedge\boldsymbol{x}'_i)} \tag{90}$$

$$X'_{i3} = \frac{-(\boldsymbol{q}_i\wedge\boldsymbol{s})\cdot(\boldsymbol{q}_i\wedge\boldsymbol{x}'_i)}{(\boldsymbol{q}_i\wedge\boldsymbol{x}'_i)\cdot(\boldsymbol{q}_i\wedge\boldsymbol{x}'_i)}, \tag{91}$$

for $i = 1, ..., n$, where $\boldsymbol{q}_i = \tilde{R}\boldsymbol{x}_i R$ and $\boldsymbol{s} = \tilde{R}\boldsymbol{t}R$. These equations can be written more concisely as $X_{i3} = -(\boldsymbol{x}'_i\wedge\boldsymbol{s})\cdot B_i$ and $X'_{i3} = -(\boldsymbol{q}_i\wedge\boldsymbol{s})\cdot B_i$ where B_i is the bivector $B_i = \beta_i/(\beta_i\cdot\beta_i)$ with $\beta_i = \boldsymbol{q}_i\wedge\boldsymbol{x}'_i$. Our task is now to find a way of simultaneously solving the equations (88), (89), (90), (91). To accomplish this solution we will develop methods of (a) finding $\boldsymbol{t}$ given R and the data, (b) finding $\{X_{i3}\}$ and $\{X'_{i3}\}$ given R, $\boldsymbol{t}$ and the data and (c) finding R given $\{X_{i3}\}$ and $\{X'_{i3}\}$ and the data.

From equations (88), (90), (91) it is clear that $\boldsymbol{t}$ and $\{\mathbf{X}_i, \mathbf{X}'_i\}$ are only determined up to a scale factor. This factor can be fixed by requiring that $\boldsymbol{t}^2 = 1$. Substituting equations (90), (91) into the expression for $\boldsymbol{t}$ given in equation (88), gives an equation in only R, $\boldsymbol{t}$ and the data. After some simplification this can be written as

$$\boldsymbol{t} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta_i\cdot\beta_i}\left[\boldsymbol{A}_i(\boldsymbol{s}\cdot\boldsymbol{x}'_i) - \boldsymbol{B}_i(\boldsymbol{t}\cdot\boldsymbol{x}_i)\right], \tag{92}$$

where $\boldsymbol{A}_i = (\boldsymbol{x}'_i\cdot\boldsymbol{q}_i)\boldsymbol{x}_i - (\boldsymbol{x}_i)^2R\boldsymbol{x}'_i\tilde{R}$ and $\boldsymbol{B}_i = (\boldsymbol{x}'_i)^2\boldsymbol{x}_i - (\boldsymbol{x}'_i\cdot\boldsymbol{q}_i)R\boldsymbol{x}'_i\tilde{R}$. Taking the scalar product of equation (92) with $\boldsymbol{x}_j$ enables us to rearrange the above equation to give

$$\boldsymbol{t}\cdot\boldsymbol{x}_j = \boldsymbol{t}\cdot\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta_i\cdot\beta_i}\left[(\boldsymbol{A}_i\cdot\boldsymbol{x}_j)\boldsymbol{u}'_i - (\boldsymbol{B}_i\cdot\boldsymbol{x}_j)\boldsymbol{x}_i)\right], \tag{93}$$

where $\boldsymbol{u}'_i = R\boldsymbol{x}'_i\tilde{R}$. Thus for any $j = 1, 2, .., n$ we have

$$\boldsymbol{t}\cdot(\boldsymbol{x}_j - \boldsymbol{P}_j) = 0, \tag{94}$$

where $\boldsymbol{P}_j = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta_i\beta_i}\left[(\boldsymbol{A}_i\cdot\boldsymbol{x}_j)\boldsymbol{u}'_i - (\boldsymbol{B}_i\cdot\boldsymbol{x}_j)\boldsymbol{x}_i)\right]$. If $\boldsymbol{w}_j = \boldsymbol{x}_j - \boldsymbol{P}_j$, then, since $\boldsymbol{t}$ is perpendicular to each of the $\boldsymbol{w}_j$'s we can reconstruct $\boldsymbol{t}$ by taking the cross-product of any two distinct $\boldsymbol{w}_j$'s:

$$\boldsymbol{t} = \frac{i\boldsymbol{w}_n\wedge\boldsymbol{w}_m}{|i\boldsymbol{w}_n\wedge\boldsymbol{w}_m|}, \tag{95}$$

for $n \neq m$. Thus, given our estimate of R and the data, we can construct an estimate for $\boldsymbol{t}$ via equation (95) without any estimate of the range coordinates. Given estimates of R, $\boldsymbol{t}$ and the data, $\{X_{i3}\}$ and $\{X'_{i3}\}$ can be found simply from equations (90), (91). Note here that choosing $f = 1$ is independent of this normalization of $\boldsymbol{t}$ and that this can be confirmed by examination of equation (92).

Our last requirement is to be able to estimate R given the range coordinates and the data. We do this by solving equation (89) via the approach described in Sections 3.1 and 3.3. The particular form of the iterative scheme is summarized below. It should be noted that there are a number of ways in which the ordering of the scheme could have proceeded; the ordering we have chosen seems to be fairly robust as we will show via simulations. At present, we are unable to *prove* that the iterative scheme converges to the best solution, but we will show via the simulations that its performance, even in the presence of considerable measurement noise, is very good. The following section presents an iterative technique for the solution of equations (88) to (91).

4.3 Simulations and iterative scheme for solving the unknown-range problem

The method is very simple and will be summarized by a series of steps:

1. Make an initial guess for the rotor. The identity rotor (i.e. no rotation) is a suitable choice.
2. Given the estimate of R and the two data sets (each containing n points), make an estimate of $\boldsymbol{t}$. Normalize so that $\boldsymbol{t}^2 = 1$.
3. Given the data and the estimates of R and $\boldsymbol{t}$ we then produce estimates for the X_{i3} and X'_{i3}.
4. Form a new value for R given the data and the estimates for the X_{i3} and X'_{i3} from the previous step.
5. Check for convergence; stop if criteria are satisfied. Otherwise return to step 2.

A number of convergence criteria may be used. We choose to say that the process has converged if the differences in the values of $\boldsymbol{t}$ and $\boldsymbol{n}\theta$ over successive iterations are less than some values set by the user. In the simulations this proved adequate. From the form of the rotor R it is clear

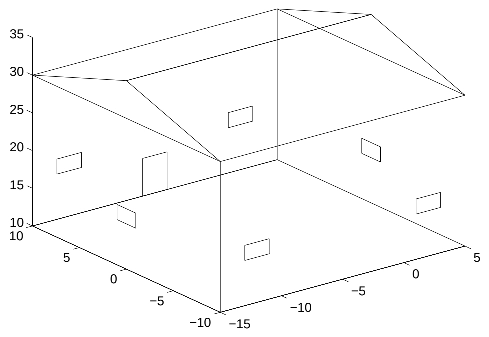

Figure 4: Framework of the 'house' used in the simulations, showing the 38 vertices.

that $(\boldsymbol{n}, \theta)$ and $(-\boldsymbol{n}, -\theta)$ will produce the same rotation. It therefore seems sensible to check for convergence in the sign-invariant quantity $\boldsymbol{n}\theta$. The use of the quantity $\boldsymbol{n}\theta$ will be justified again in the formation of average values from a number of samples as discussed in Section 4.3.2.

It is interesting to note that step 1 suggests that the use of the identity rotor is an adequate starting point. The fact that we are then giving the algorithm a starting point which is usually a long way from the true solution does not affect the eventual convergence. However, the method can be speeded up considerably by making some alternative initial estimate of the rotor – this is also discussed further in Section 4.3.2.

4.3.1 Simulations: I

In order to test the above algorithms we have carried out some simulations. In accordance with the analysis described here, we will restrict ourselves to point correspondences. We simulated the framework of a 'house', the 38 vertices of which provided our points – this is shown in Figure 4 and the coordinates of these vertices are listed in Table 1. The axes in Figure 4 show how the house is situated relative to the origin at $(0, 0, 0)$, which will be the centre of projection. The image plane will be the plane $z = 1$ if we use $f = 1$. The simulations in this section will be a straightforward investigation of the estimation process in the presence of increasing noise on the image points. The following section will illustrate how the algorithm compares to one of the established techniques for motion and structure estimation for a variety of baseline directions and noise scenarios.

The house was subjected to a rotation of 36° about an axis through the origin in the direction of the vector $(3, 4, 6)$ and then translated by $(7, 8, 13)$ – these values were chosen to coincide with the values selected by [Walker *et al.* 1991]. We therefore have two sets of coordinates, one in the original position and one in the translated and rotated position. From each set of coordinates we also produce the sets of projected coordinates (assuming a focal length, f, of unity). Note here that we are now considering the case in which the camera is stationary and the object is moving; the analysis given in the above sections is directly applicable to this case if we simply interchange the $\mathbf{X}_i$s and $\mathbf{X}'_i$s. If $\mathbf{X}_i$ undergoes a rotation R about an axis through the centre of projection (origin in this case) followed by a translation, the new vector $\mathbf{X}'_i$ is given by

$$\mathbf{X}'_i = R\mathbf{X}_i\tilde{R} + \boldsymbol{t}. \tag{96}$$

Vertex	(x,y,z)	Vertex	(x,y,z)	Vertex	(x,y,z)
1	(5,10,10)	14	(-6,10,15)	27	(-11,10,16)
2	(-15,10,10)	15	(5,-1,16)	28	(-13,10,16)
3	(-15,-10,10)	16	(5,1,16)	29	(-13,10,18)
4	(5,-10,10)	17	(5,1,18)	30	(-11,10,18)
5	(5,10,30)	18	(5,-1,18)	31	(3,-10,16)
6	(-15,10,30)	19	(-15,-1,16)	32	(1,-10,16)
7	(-15,-10,30)	20	(-15,1,16)	33	(1,-10,18)
8	(5,-10,30)	21	(-15,1,18)	34	(3,-10,18)
9	(5,0,35)	22	(-15,-1,18)	35	(-11,-10,16)
10	(-15,0,35)	23	(3,10,16)	36	(-13,-10,16)
11	(-4,10,10)	24	(1,10,16)	37	(-13,-16,18)
12	(-6,10,10)	25	(1,10,18)	38	(-11,-10,18)
13	(-4,10,15)	26	(3,10,18)		

Table 1: (x, y, z) coordinates of the 'house' vertices.

We can obviously rearrange this to put it into the form of equation (77)

$$\mathbf{X}_i = \tilde{R}(\mathbf{X}_i' - \boldsymbol{t})R. \tag{97}$$

Range data known
In accordance with [Walker *et al.* 1991], we added random gaussian noise with zero-mean and standard deviation of 0.5 to each of the data sets of 3-D coordinates. In addition to testing the geometric algebra algorithm described above we also programmed the dual quaternion method of Walker *et al.* 1991 and the SVD approach of Arun *et al.* 1987 so that the three algorithms were available for comparison. Each of these algorithms solves the **same** least-squares problem, and each finds the **optimal** values for the rotation and translation, albeit by different means. Given this fact, we should surely expect the three algorithms to give precisely the same answers, modulo numerical effects caused by different intermediate procedures. This is indeed precisely what we found in our simulations. All three methods produce the same results, and the only differences come in running the simulations many times and letting numerical effects accumulate. Contrary to the findings in [Walker *et al.* 1991] we found that the dual quaternion method was considerably slower than the geometric algebra and SVD approaches (which are effectively the same) and produced higher accumulated numerical errors, due to the greater number of operations which the dual quaternion method requires.

Range data unknown
For these simulations we added zero mean random gaussian noise to the two sets of projected coordinates (as these would be what we would measure). The algorithm described above was then run for various noise levels. Table 2 lists the results for values of σ, the standard deviation of the gaussian distribution, ranging from 0.005 to 0.08. The values of the rotation angle and rotation axis were derived from the optimal rotor found from the algorithm – the values of θ, $\boldsymbol{n}$ and $\boldsymbol{t}$ given are averages over a sequence of 20 runs, using a different realization of the noise each time. For

σ	$\theta(°)$	n_1	n_2	n_3	t_1	t_2	t_3
true values	36	0.3841	0.5121	0.7682	0.4168	0.4764	0.7741
0.005	35.9568	0.3819	0.5132	0.7686	0.4177	0.4749	0.7746
0.01	35.8898	0.3828	0.5116	0.7692	0.4198	0.4742	0.7737
0.02	35.3885	0.3685	0.5203	0.7704	0.4297	0.4558	0.7792
0.03	35.0082	0.3546	0.5326	0.7685	0.4365	0.4351	0.7866
0.04	32.8348	0.3408	0.5292	0.7771	0.4820	0.3903	0.7797
0.08	26.3665	0.1633	0.6669	0.7271	0.5593	0.1491	0.8135

Table 2: Average values for rotation and translation estimates for various noise levels.

each run, a maximum of 500 iterations were allowed and convergence was deemed to have been achieved when the 'distance' between successive estimated values of $\boldsymbol{t}$ and $\boldsymbol{n}\theta$ differed by less than 10^{-5} and 10^{-4} respectively, i.e. $|(\boldsymbol{t}_{i+1} - \boldsymbol{t}_i)| < 10^{-5}$ and $|((\boldsymbol{n}\theta)_{i+1} - (\boldsymbol{n}\theta)_i| < 10^{-4}$. In each case the values of the range data $\{X_{i3}\}$ and $\{X'_{i3}\}$ were also estimated but only the values for three vertices, 3, 5 and 38, are listed in Table 3 in the interests of simplicity. A total of 38 points (each vertex in the 'house') was used and the true rotation and translation were as given above, $\theta = 36°$, $\boldsymbol{n} = \frac{1}{\sqrt{61}}(3,4,6) = (0.3841, 0.5121, 0.7682)$ and $\boldsymbol{t} = (7,8,13) = \sqrt{282}(0.4168, 0.4764, 0.7741)$. Since the algorithm normalizes $\boldsymbol{t}$ so that $\boldsymbol{t}^2 = 1$, we should compare the values in Table 2 with the true unit vector in the $\boldsymbol{t}$ direction $(0.4168, 0.4764, 07741)$. As mentioned in previous sections, the values of the depth coordinates are also only determined up to a scale factor. We have multiplied the depth coordinates in Table 3 by the factor $\sqrt{282} = 16.79$ which relates the unit vector $\boldsymbol{t}$ to the magnitude of the translation used and they can therefore be directly compared with the depth coordinates listed in Table 1.

From the coordinate values in Table 1 we can obtain the projected coordinates by dividing each vector by its z-component. These projected coordinates will therefore range in magnitude from 0 to 1.5. Thus, the highest noise levels imposed (0.04 and 0.08) constitute a fairly low signal to noise level. We can see from Table 3 that the estimated values of $\{X_{i3}\}$ and $\{X'_{i3}\}$ decrease in

	Vertex 3:X_{i3}, X'_{i3}	Vertex 5:X_{i3}, X'_{i3}	Vertex 38:X_{i3}, X'_{i3}
(true values)	(10, 22.88)	(30, 42.44)	(18, 29.27)
$\sigma = 0.005$	(10.01, 22.88)	(30.37, 42.79)	(17.99, 29.25)
$\sigma = 0.01$	(10.01, 22.89)	(30.69, 43.06)	(17.98, 29.26)
$\sigma = 0.02$	(10.05, 23.24)	(30.83, 43.33)	(18.00, 29.63)
$\sigma = 0.03$	(10.07, 23.67)	(40.00, 51.80)	(17.95, 29.90)
$\sigma = 0.04$	(10.10, 24.47)	(59.35, 70.43)	(18.03, 30.80)
$\sigma = 0.08$	(9.34, 25.49)	$(7.2,\ 6.7)\times 10^{-3}$	(15.69, 30.34)

Table 3: Average values for depth coordinates in the two views for various noise levels (scaled values shown).

accuracy as the noise is increased, but remain adequate estimates. This is indeed the case for all vertices except vertex 5 (at (5,10,30)), the results for which are included in Table 3. We see that as σ increases above 0.03, the estimates for the depth coordinates of vertex 5 become increasingly unreliable until they break down at $\sigma = 0.08$. To see why this is happening we can return to equations (90),(91) and note that in determining X_{i3} and X'_{i3} the factor $(\boldsymbol{q}_i \wedge \boldsymbol{x}'_i)$ occurs in the denominator. Therefore, if $\boldsymbol{q}_i$ is approximately parallel to $\boldsymbol{x}'_i$ the above factor will be close to zero and produce artificially large values for the depth estimates. In physical terms, this corresponds to the situation where the rotated vector from position 1, $R\boldsymbol{x}_i\tilde{R}$, is almost parallel to the translation vector $\boldsymbol{t}$. For larger values of the noise such occurrences become more probable. We can easily avoid this by omitting any point from the dataset for which this problem occurs. The tables show that for $\sigma < 0.08$ the estimates of the rotation, translation and depth coordinates produced by the algorithm are reasonably accurate.

4.3.2 Simulations: II

In this section we will compare the results of the geometric algebra algorithm with the closed-form solution described in [Weng *et al.* 1989]. We will choose parameters for the motion and structure as described in [Weng *et al.* 1989] so that the results here may also be compared to their results for the closed-form algorithm against other linear algorithms. The advantages of such algorithms is their speed – the geometric algebra algorithm is iterative and is therefore slower. However, we hope to show that the results obtained with the GA algorithm are significantly more accurate. Indeed, for real noisy data and approximate calibration the geometric algebra is able to obtain good solutions for cases in which the closed-form algorithms fail completely.

In the following simulations 12 object points are used and these are generated randomly from a uniform distribution in a cube of side 10 units whose centre is 11 units from the centre of projection of the camera. The focal length will be taken as 1 and the image size as 2 units. Unlike the simulations in the previous section, the noise is now solely due to quantization effects in the image plane and we consider two noise scenarios, one given by an image resolution of 256×256 and the other by a resolution of 128×128. Such a decrease in resolution will considerably increase the quantization noise. The rotation of the camera is 8° about an axis in the $(-0.2, 1, 0.2)$ direction and 21 different translations, $\boldsymbol{t}$, are used with these given by

$$\boldsymbol{t} = 3\cos(i-1)\Delta\phi\,\sigma_1 + \sigma_2 - 3\sin(i-1)\Delta\phi\,\sigma_3 \tag{98}$$

for $i = 1$ to 21, and $\Delta\phi = 90/20 = 4.5^\circ$. Thus, when these translation vectors are projected onto the $x - z$ plane they describe a sequence of equally spaced vectors between the x axis and the $-z$ axis. This situation is designed to be similar to that described in [Weng *et al.* 1989]. The translation vectors were chosen in this way so as to illustrate the greater potential for error in the motion estimation when the translation is parallel to the image plane ([Weng *et al.* 1989] explain why this is via explicit reference to their algorithm). We would not expect the performance of the GA algorithm to be significantly affected by the translation direction.

For each of the 21 values of the translation vector the estimated values of $\boldsymbol{t}$, $\boldsymbol{n}$ and θ are found by averaging over 100 trials (in each trial the 12 points are generated at random within the allowed region). The *relative error* is calculated by taking the norm of the difference between the mean value and true value divided by the norm of the true value. In the case of $\boldsymbol{t}$ the trial values are normalized before forming the mean. To form the mean values of $\boldsymbol{n}$ and θ, we form the product $\boldsymbol{n}\theta$ for each trial (having first normalized $\boldsymbol{n}$) and then form the average of $\boldsymbol{n}\theta$ over the total number of trials. The mean values of $\boldsymbol{n}$ and θ separately are then extracted from this mean value of $\boldsymbol{n}\theta$. Recall that above we identified $\boldsymbol{n}\theta$ as being a sign-invariant quantity; it also has 3 degrees of freedom (dof). A general rotation has 3 dof and in forming an 'average' rotation it therefore seems sensible to find the mean value of a quantity with a sufficient set of independent quantities. This would therefore suggest that one should average over the quantity $\boldsymbol{n}\theta$ rather than over θ and $\boldsymbol{n}$ individually or over the components of the rotation matrices from each trial.

Figures 5 and 6 show the relative error for $\boldsymbol{t}$, $\boldsymbol{n}$ and θ for the two different resolutions. In addition, for comparison with [Weng *et al.* 1989], we also plot the relative error in the estimated rotation matrix $\mathcal{R}$; in this case the 'mean' value of $\mathcal{R}$ is that rotation matrix formed from the mean value of $\boldsymbol{n}\theta$ and the relative error is formed from the norm of the difference of the estimated $\mathcal{R}$ and the true $\mathcal{R}$ divided by the norm of the true $\mathcal{R}$ – where the norm of a matrix A is given by

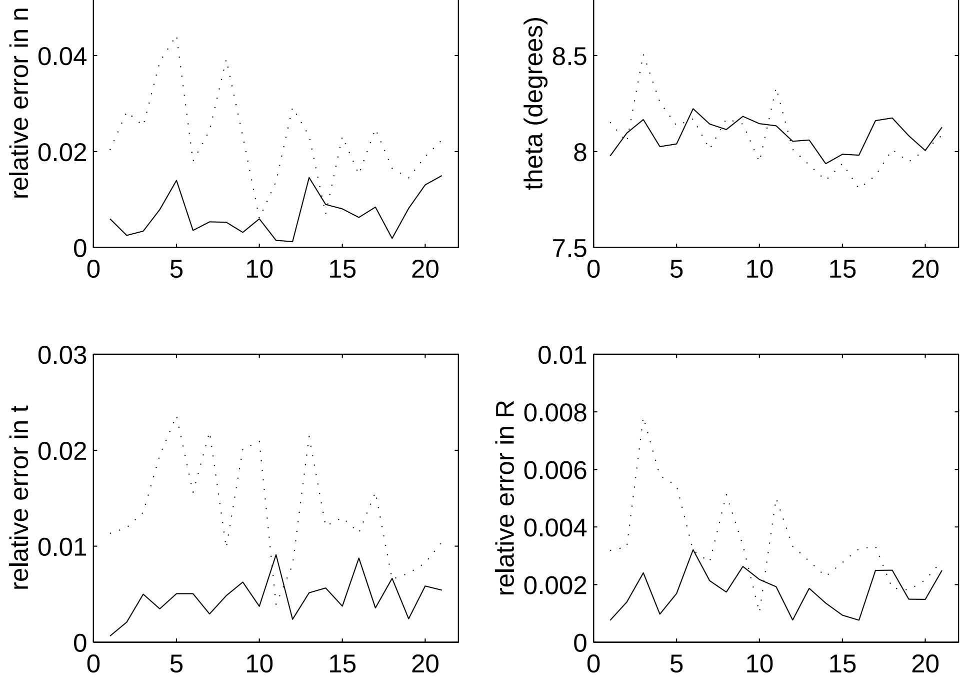

Figure 5: Relative errors in a) the rotation axis b) the rotation angle c) the translation direction and d) the rotation matrix, plotted against the 21 different translation vectors used. The results are the average of 100 samples at each point and use a resolution of 256×256. The solid and dashed lines respectively show the results of the GA algorithm and the linear algorithm of Weng *et al.*

$||A|| = \sqrt{\sum_{ij} a_{ij}{}^2}$. In each figure the relative errors are plotted against the 21 different translation vectors used (equation 98) and the results of both the GA algorithm and the algorithm of Weng *et al.* are given. It is worth noting at this point that the relative errors in $\mathcal{R}$ formed by the method just described will *not* be equal to the relative errors formed by averaging the individual rotation matrices which have 9 dof. We believe that one should therefore form an average $\mathcal{R}$ from an average $\boldsymbol{n}\theta$. It is not clear which method of averaging is used in [Weng *et al.* 1989].

We can see from Figures 5 and 6 that the errors in the linear algorithm of Weng *et al.* are greater for translation directions parallel to the image plane (corresponding to small values on the x axis), whereas the GA algorithm shows no such tendencies, giving relative errors which are roughly of equal magnitude over the range of directions used; this is in agreement with our expectations. Overall we see that the GA algorithm gives more accurate predictions than the linear algorithm for all parameters but more particularly for the vectors $\boldsymbol{n}$ and $\boldsymbol{t}$ – in many cases there is almost a factor 10 improvement on average. The obvious advantage of the linear algorithm is its simplicity and its speed. If we start the GA algorithm with an initial guess for the rotor as the identity (i.e. no rotation) then the algorithm is robust, in the sense that it will converge to the given solutions but will take perhaps 100 iterations to do so. However, in generating the results shown in Figures 5 and 6 the procedure was to evaluate $\mathcal{R}$ for the linear algorithm and to use this as the initial estimate of the rotation in the GA algorithm. This had the effect of decreasing the number of iterations

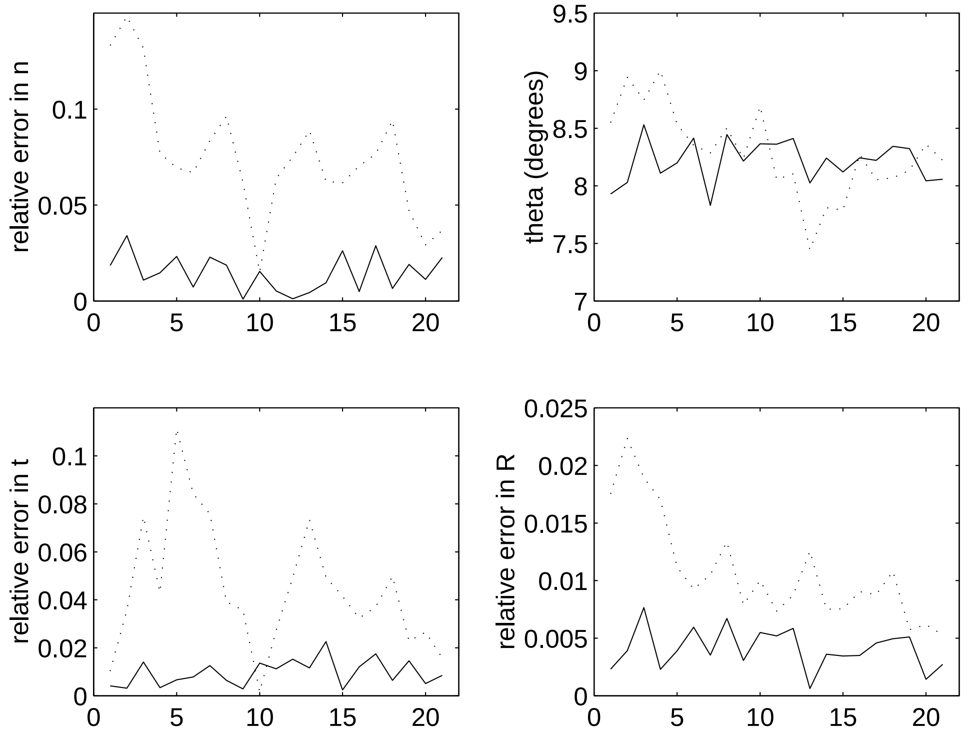

Figure 6: Relative errors in a) the rotation axis b) the rotation angle c) the translation direction and d) the rotation matrix, plotted against the 21 different translation vectors used. The results are the average of 100 samples at each point and use a resolution of 128 × 128. The solid and dashed lines respectively show the results of the GA algorithm and the linear algorithm of Weng *et al.*
.

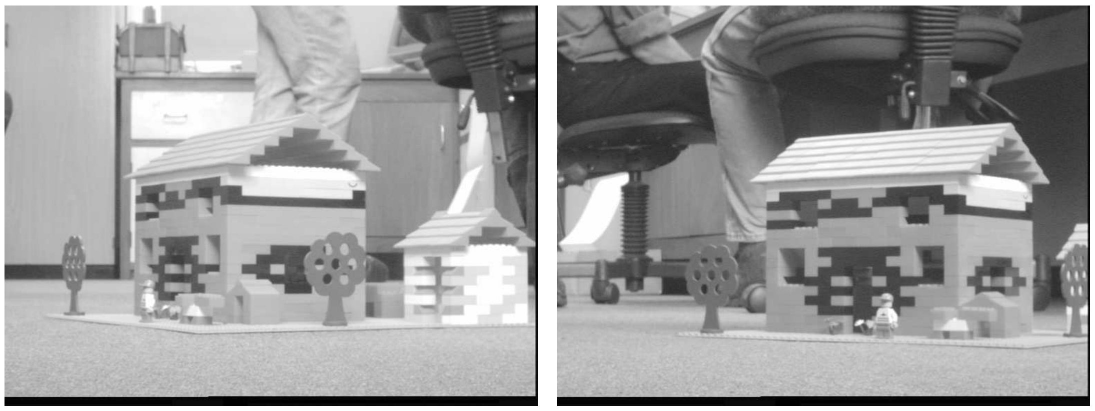

Figure 7: Two views of Lego house from a single camera undergoing translation and rotation.

dramatically. Indeed, the current algorithm can be seen as an addition to the linear algorithm of Weng *et al.* to refine the estimates if greater accuracy is required. We note also from the figures that the improvements in accuracy given by the GA algorithm increase as the resolution decreases, i.e. as the noise increases. The algorithm presented here therefore seems to be a better tool if there is a high likelihood that the data is very noisy.

4.4 Real Data

In this section we will give some results of applying the algorithm to a pair of real images. The camera was approximately calibrated before the experiment and was then used to observe a 'LEGO' house from two different viewpoints. The two views are shown in figure 7. The motion of the camera and the distances to various points on the house were measured. From the two images a set of 12 corresponding features (mostly corners) were found and these were used as input to both the algorithm of Weng *et al.* and to the GA algorithm with an identity rotor as starting point. The axes in both camera positions were such that the z-axis passed through the optical centre perpendicular to the image plane and towards the scene, the x-axis was vertically upwards and the y-axis was then such that the directions formed a right-handed orthogonal set. Relative to the frame in view 1 the true values of the rotation and translation between the views was:

$$\boldsymbol{n} = [1,0,0]$$

$$\begin{aligned}\theta &= -27^\circ \\ \boldsymbol{t} &= [0, -72.6, 25.1]. \end{aligned} \tag{99}$$

The translation values are given in centimetres; writing $\boldsymbol{t}$ as a unit vector gives $\boldsymbol{t} = [0, -0.945, 0.327]$.

Giving the 12 point matches to the algorithm of Weng *et al.* produced the following results:

$$\begin{aligned}\boldsymbol{n} &= [0.456, -0.290, 0.842] \\ \theta &= 94.7^\circ \\ \boldsymbol{t} &= [-0.801, 0.598, -0.037]. \end{aligned} \tag{100}$$

In this particular case it is clear that the closed form solution above gives something very far from the truth. Now, if we give the same 12 points to the GA algorithm with an initial identity rotor (i.e. no rotation), after 4500 iterations the results are;

$$\begin{aligned}\boldsymbol{n} &= [-0.999, 0.006, -0.039] \\ \theta &= 28.33^\circ \\ \boldsymbol{t} &= [-0.004, -0.941, 0.339]. \end{aligned} \tag{101}$$

It is clear that we are able to obtain a very accurate solution given a fairly large number of iterations – a less accurate solution is obtained with relatively few iterations, but this answer is still much better than that given in equation (100). In addition, the depths obtained from the algorithm are also in good agreement with the true values. A more detailed study of the performance of the algorithm on real data (with details of calibration etc.) and scene reconstruction will be given elsewhere.

While the performance of closed-form algorithms such as that of Weng *et al.* on simulated data is often good, the inherent problems arising from the fact that quantities are not estimated simultaneously can lead to serious inaccuracies when applied to real noisy data. We have shown here that if one is willing to adopt an iterative approach which inevitably means more computation time, the GA algorithm is robust and accurate.

5 Other applications of geometric algebra in computer vision

The major focus of this paper has been in applying the GA techniques described in Section 3 to the structure and motion estimation problem. However, we would like to outline briefly other areas of computer vision in which GA has been applied and discuss how this work relates to standard projective geometry techniques and recent applications of Grassmann algebras and Lie group theory in computer vision.

Since about the mid 1980's most of the computer vision literature discussing geometry and invariants has used the language of projective geometry. The usefulness of the projective description of space is often only realised via the introduction of homogeneous coordinates. In the geometric algebra description of projective space, we therefore dispense with much of the machinery of classical projective geometry and postulate a 4D space with a well-defined means of moving between this space and normal Euclidean 3-space; this process is known as the *projective split*.

Since the formalism of projective geometry has already been translated into geometric algebra [Hestenes and Ziegler 1991], it is fairly simple to find concise expressions for the algebra of incidence in projective space, i.e. intersections of planes and planes, planes and lines etc. All of this has been implemented in the computer algebra package *Maple.* There has been much recent interest in the use of the Grassmann-Cayley or double algebra as an elegant means of formulating the algebra of incidence [Carlsson 1994, Csurka and Faugeras 1995, Faugeras and Mourrain 1995] and this framework has been applied to problems in computer vision mainly in the study of invariants and of multiple view constraints. The geometric algebra approach to the methodology of forming invariants and applications of this to multiple view constraints is discussed in [Lasenby *et al.* 1996, Bayro *et al.* 1996]. The Grassmann-Cayley algebra is indeed well suited for the areas to which it has been applied but we would argue that the fact that it has only an exterior product and no interior product means that it is not a framework one could use for a wide class of problems; for instance, there is no obvious way to represent a rotation efficiently in this algebra. In [Lasenby *et al.* 1996] we have shown that the GA formalism provides a neat means of dealing with the algebra of incidence and the methodology of forming invariants.

The use of Lie groups in the modelling of rigid motions has received significant attention [Chevalier 1991]. GA has been used effectively for modelling general displacements and the motion of linked rigid bodies [Lasenby 1996, Hestenes 1994]. In addition, in the discussion of Lie groups in a GA context in [Doran *et al.* 1993] the authors present many techniques which may have interesting applications in the computer vision field – this remains an area which we would like to explore.

6 Conclusions

In the first part of the paper we have presented a variety of techniques based on the geometric algebra which we feel may be particularly useful for applications in computer vision. Specifically, the ability to deal efficiently with rotations and the ability to differentiate with respect to multivector quantities are areas which are considered in some detail. The latter half of the paper deals with the problem of reconstructing camera or object motion and scene structure from two images and is discussed in the framework of geometric algebra. In the case where the range data is known we have shown that the geometric algebra solution is essentially the same as the many other closed form solutions in the literature and that, contrary to claims in [Walker *et al.* 1991], the use of dual quaternions offers no real advantage over the more straightforward SVD approach. However, in the case of unknown range data, we have presented an iterative algorithm for computing both the motion and the depth coordinates of image points in the presence of measurement uncertainty. The algorithm incorporates all of the constraints of general rigid body displacements. As far as we are aware, this is the first such algorithm to solve directly the relevant least-squares equation simultaneously using analytic derivatives over the whole set of unknowns in the problem. We are able to do this because of the powerful tools made available to us in the geometric algebra – for instance, being able to minimize directly with respect to the rotors underpins the main part of the algorithm, a facility which is not readily available in other frameworks. We have compared the algorithm to one of the basic linear algorithms for estimating structure and motion showing that it can improve the accuracy of the results by, in some cases, large factors. One of the main uses of the algorithm may be as an additional stage in other quicker algorithms when more refined estimates are required.

References

[Arun *et al.* 1987] Arun, K., Huang, T.S. and Blostein, S.D. 1987. Least squares fitting of two 3-D point sets. *IEEE Trans.Pattern Anal.Mach.Intelligence* **PAMI-9**, 698–700.

[Bayro and Lasenby 1995] Bayro-Corrochano, E. and Lasenby, J. 1995. Object modelling and motion analysis using Clifford algebra. Proceedings of Europe-China Workshop on*Geometric Modeling and Invariants for Computer Vision*, Ed. Roger Mohr and Wu Chengke, Xi'an, China, April 1995.

[Bayro *et al.* 1996] Bayro-Corrochano, E., Lasenby, J. and Sommer, G. 1996. Geometric Algebra: a framework for computing point and line correspondences and projective structure using n-uncalibrated cameras. Proceedings of ICPR'96, Vienna.

[Carlsson 1994] Carlsson, S. 1994. The Double Algebra: and effective tool for computing invariants in computer vision. *Applications of Invariance in Computer Vision*, Lecture Notes in Computer Science 825; Proceedings of 2nd-joint Europe-US workshop, Azores, October 1993. Eds. Mundy, Zisserman and Forsyth. Springer-Verlag.

[Chevalier 1991] Chevalier, D.P. 1991. Lie Algebras, Modules, Dual Quaternions and Algebraic Methods in Kinematics. *Mech. Mach. Theory* 26: 350–358.

[Clifford 1878] Clifford, W.K. 1878. Applications of Grassmann's extensive algebra. *Am. J. Math.* 26(6)=: 613–627.

[Csurka and Faugeras 1995] Csurka, G. and Faugeras, O. 1995. Computing three-dimensional projective invariants from a pair of images using the Grassmann-Cayley algebra. Proceedings of Europe-China Workshop on *Geometric Modeling and Invariants for Computer Vision*, Ed. Roger Mohr and Wu Chengke, Xi'an, China, April 1995.

[Doran 1994] Doran, C.J.L. 1994. Geometric Algebra and its Applications to Mathematical Physics. Ph.D. Thesis, University of Cambridge.

[Doran *et al.* 1993] Doran,C.J.L., Hestenes, D., Sommen, F. and van Acker, N. 1993. Lie groups as spin groups. *J. Math. Phys.*, 34(8): 3642.

[Doran *et al.* 1993] Doran, C.J.L., Lasenby, A.N. and Gull, S.F. 1993. Gravity as a gauge theory in the spacetime algebra. In F. Brackx and R. Delanghe., editors, *Third International Conference on Clifford Algebras and their Applications in Mathematical Physics.* Kluwer, Dordrecht.

[Doran *et al.* 1996] Doran, C.J.L., Lasenby,A.N., Gull, S.F., Somaroo, S. and Challinor, A. 1996. Spacetime Algebra and Electron Physics. to appear in *Advances in Electronics and Electron Physics.*

[Faugeras and Hebert 1983] Faugeras, O.D. and Hebert, M. 1983. A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. *Proceedings International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, 996–1002.

[Faugeras *et al.* 1987] Faugeras, O.D., Lustman, F. and Toscani, G. 1987. Motion and Structure from Motion. *Proceedings ICCV*, 25–34.

[Faugeras and Mourrain 1995] Faugeras, O. and Mourrain, B. 1995. On the geometry and algebra of the point and line correspondences between N images. Proceedings of Europe-China Workshop on *Geometric Modeling and Invariants for Computer Vision*, Ed. Roger Mohr and Wu Chengke, Xi'an, China, April 1995.

[Grassmann 1877] Grassmann, H. 1877. Der ort der Hamilton'schen quaternionen in der ausdehnungslehre. *Math. Ann.*, 12: 375.

[Gull *et al.* 1993] Gull, S.F., Lasenby, A.N. and Doran, C.J.L. 1993. Imaginary numbers are not real — the geometric algebra of spacetime. *Found. Phys.*, 23(9): 1175.

[Hartley 1995] Hartley, R.S. 1995. In defence of the eight-point algorithm

[Hestenes 1966] Hestenes, D 1966. Space-Time Algebra. *Gordon and Breach.*

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

[Hestenes 1986a] Hestenes, D. 1986. New Foundations for Classical Mechanics *D. Reidel*, Dordrecht.

[Hestenes 1986b] Hestenes, D. 1986. A unified language for mathematics and physics. *Clifford algebras and their applications in mathematical physics.* Eds. J.S.R. Chisolm and A.K. Common, D.Reidel, Dordrecht, p1.

[Hestenes and Ziegler 1991] Hestenes, D. and Ziegler, R. 1991. Projective Geometry with Clifford Algebra. *Acta Applicandae Mathematicae*, 23: 25–63.

[Hestenes 1994] Hestenes, D. 1994. Invariant Body Kinematics: II. Reaching and Neurogeometry. *Neural Networks*, 7(1): 79–88.

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

[Horn *et al.* 1988] Horn, B.K.P., Hilden, H.M. and Negahdaripour, S. 1988. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am.*, 5: 1127–1135.

[Huang 1986] Huang, T.S. 1986. Determining three-dimensional motion and structure from two perspective views. *Handbook of pattern recognition and image processing*, Chapter 14, Academic Press.

[Huang and Netravali 1994] Huang, T.S. and Netravali, A.N. 1994. Motion and Structure from Feature Correspondences: A Review. *Proceedings of the IEEE*, 82(2): 252–268.

[Lasenby *et al.* 1993b] Lasenby, A.N., Doran, C.J.L. and Gull, S.F. 1993. Grassmann calculus, pseudoclassical mechanics and geometric algebra. *J. Math. Phys.*, 34(8): 3683.

[Lasenby *et al.* 1995] Lasenby, A.N., Doran, C.J.L., and Gull, S.F. 1994. Astrophysical and cosmological consequences of a gauge theory of gravity. Advances in astrofundamental physics, Proceedings of 1994 International School of Astrophysics, Erice: 359. Eds. N. Sanchez and A. Zichichi, World Scientific, Singapore,

[Lasenby 1996] Lasenby, J. 1996. Geometric Algebra: Applications in Engineering. In W.E. Baylis, editor, *Geometric (Clifford) Algebras in Physics.* Birkhauser, Boston, 1996.

[Lasenby *et al.* 1996] Lasenby, J., Bayro-Corrochano, E., Lasenby, A.N. and Sommer, G. 1996. A new methodology for computing invariants in computer vision. Proceedings of ICPR'96, Vienna.

[Lin *et al.* 1986] Lin, Z., Huang, T.S., Blostein, S.D., Lee, H. and Margerum, E.A. 1986. Motion estimation from 3-D point sets with and without correspondences. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, 194–201.

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

[Mitchie and Aggarwal 1986] Mitchie, A. and Aggarwal, J.K. 1986. A computational analysis of time-varying images. *Handbook of Pattern Recognition and Image Processing*, Eds. Young, T.Y. and Fu, K.S., New York, Academic Press.

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

[Walker *et al.* 1991] Walker, M.W., Shao, L. and Volz, R.A. 1991. Estimating 3-D location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54(3): 358–367.

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