

Combining orientation estimates

BY LEO DORST

*Informatics Institute, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
leo@science.uva.nl*

Orientation measurements estimate relative rotations of objects. Such estimates may need to be averaged according to their covariances, e.g. for a Kalman filter. The non-commutative algebra of rotations makes transference of techniques inspired by the usual vector-based approaches for translations non-trivial. This paper shows in tutorial fashion how the rotor representation of rotations (which is an embedded form of the quaternion representation) permits straightforward computations with rotation estimates, from averaging and interpolation to filtering. We characterise rotational noise and present a way to combine estimates of orientations which minimises error covariance.

Keywords: orientation estimation, Kalman filtering, quaternions, geometric algebra

1. Processing orientations and rotations

In many applications of computer vision to 3D scenes, we encounter the problem of pose estimation. We may need to know where an object is in space relative to the camera, either to measure that object or, as in robotics, to calibrate our own stance in space. Typically, the scene or camera is moving, and we obtain a sequence of orientation measurements. We would like to combine the various measurements in a consistent way to obtain the best current pose estimate, using some kind of Kalman filter. For the translational (i.e. positional) part of the stance, this is not too hard; but what to do with the orientations? Typical problems are:

- Combining multiple estimates of the same pose
- Interpolation or extrapolation of orientations
- Filtering of orientation sequences
- Minimum variance estimation of orientations

We describe orientations using rotations, since the *orientation* of an object is naturally characterised by referring to a standard object and a *rotation*: namely the (relative) rotation one has to give the reference copy of the object to reach the actual orientation.

We cannot immediately use familiar classical techniques to solve these orientation problems, since the parameter space of rotations is not just a vector space to which linear techniques apply. Rotations form a group with a non-commutative

multiplication (composition of rotations), and ‘live’ on a somewhat unusual manifold. Yet we can still use the intuition of the processing of translations to develop very similar techniques for orientations, but now carefully taking into account the proper algebra.

We shall find that many results look very similar to the translational techniques, but they only take this familiar form when we represent rotations by *quaternions* (rather than by rotation matrices with Euler angles). In this paper, we prefer to use *rotors*, which are algebraically similar to quaternions, but now introduced as elements of a real geometric algebra rather than of a complex number system. It will give us more natural ways of transferring the classical techniques of translational error estimation to rotational estimates.

We build up our understanding of this representation in a tutorial manner, first introducing geometric algebra and its rotors in section 2. We then show how to average estimates of poses in section 3. This leads to interpolation and filtering (section 4). All those techniques are known, but in section 5 we present new results, when we investigate how to characterise noise in orientation, and how to combine various estimates in an optimal manner by minimising the total error covariance, as required in Kalman filtering.

2. Rotation representation by rotors (or quaternions)

(a) Geometric algebra

We give a brief summary of geometric algebra, focusing on the concepts and operations we need. More complete tutorial introductions may be found elsewhere (Doran & Lasenby 2001, Dorst & Mann 2002, Mann & Dorst 2002, Lasenby et al. 2000). This section may be a bit abstract if you have not seen things like this before. If you get stranded, you could continue this paper by reading Section c first, which you will be able to follow approximately; and then later return to the present section, now fully motivated to read it. (For mathematicians: geometric algebra is like Clifford algebra, but limiting its operations to multiplicative constructions which have a clear geometric semantics.)

Any vector space with an inner product based on a bilinear form has a geometric algebra. In particular the n -dimensional Euclidean spaces have one, and so it makes sense to use it. The basis for the algebra is the *geometric product*, which is linear, associative, and for a vector \mathbf{x} with itself equals a scalar (equal to the inner product $\mathbf{x} \cdot \mathbf{x}$). Because of its fundamental nature, we will denote the geometric product of two quantities \mathbf{A} and \mathbf{B} simply as $\mathbf{A}\mathbf{B}$. (This gives no confusion with the scalar product in the vector space, since that is just a special case of the geometric product, as is the product of two scalars.)

The geometric product is not commutative, and we will find two derived products convenient, the inner and outer product. For vectors \mathbf{a} and \mathbf{b} , these are defined as

$$\begin{aligned} \text{inner product :} & \quad \mathbf{a} \cdot \mathbf{b} \equiv \frac{1}{2}(\mathbf{a}\mathbf{b} + \mathbf{b}\mathbf{a}) \\ \text{outer product :} & \quad \mathbf{a} \wedge \mathbf{b} \equiv \frac{1}{2}(\mathbf{a}\mathbf{b} - \mathbf{b}\mathbf{a}). \end{aligned}$$

The former is symmetrical and scalar-valued, and for vectors it coincides with the familiar inner product. It is of course associated with the geometrical intuition

of projection. The latter is anti-symmetric and is ‘bivector-valued’. Bivectors are interpretable as directed area elements (similar to the way that vectors are directed line elements). The outer product thus makes the geometrical concept of a ‘span’ of vectors a computable element of the algebra. In 3-dimensional space it is related to the cross product from classical vector algebra.

The outer product can be extended by associativity, and generates elements representing subspaces of various dimensionalities. These subspaces are called *blades* and their dimensions *grades*. Due to the anti-symmetry, the k -blades form an $\binom{n}{k}$ -dimensional linear space. The highest non-zero blade in an n -dimensional space is an n -blade; this is called a *pseudoscalar* of that space, it represents an n -dimensional oriented hypervolume.

Geometric algebra focuses on blades and their products. The geometric product of an element grade k and an element of grade ℓ in general contains elements of grades $k+\ell, k+\ell-2, \dots, |k-\ell|$. The part of grade i of an element A is selected by the *grade operator* denoted $\langle A \rangle_i$. So for example for two vectors \mathbf{a} and \mathbf{b} we have $\mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}\mathbf{b} \rangle_0$ and $\mathbf{a} \wedge \mathbf{b} = \langle \mathbf{a}\mathbf{b} \rangle_2$.

The *norm squared* $\|A\|^2$ of a quantity A is defined as

$$\|A\|^2 = \langle \tilde{A}A \rangle_0$$

where \tilde{A} (alternatively denoted A^\sim) is the *reverse*, obtained by writing all multiplicative factors in A in reverse order. For instance for vectors \mathbf{a} and \mathbf{b} we have $\tilde{\mathbf{a}} = \mathbf{a}$ and $(\mathbf{a} \wedge \mathbf{b})^\sim = \mathbf{b} \wedge \mathbf{a} = -\mathbf{a} \wedge \mathbf{b}$.

A very useful property of the geometric product is its *invertibility*: we can divide by vectors, bivectors etcetera. Dividing is multiplication by the inverse of an element, and in the geometric algebra of 3D Euclidean space, each element has a unique inverse:

$$A^{-1} \equiv \frac{\tilde{A}}{\tilde{A}A}$$

The inverse of a vector is therefore

$$\mathbf{a}^{-1} = \frac{\mathbf{a}}{\mathbf{a}\mathbf{a}} = \frac{\mathbf{a}}{\mathbf{a} \cdot \mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|^2},$$

and indeed it is easily verified that $\mathbf{a}^{-1}\mathbf{a} = 1$. A unit vector is its own reverse, a unit bivector \mathbf{B} has its reverse ($-\mathbf{B}$) as inverse.

(b) The geometric algebra of 3-dimensional Euclidean space

Let us list the elements of the geometric algebra of 3-dimensional Euclidean space, our only concern in this paper. Linearity of the construction means that we are free to choose our basis. For convenience, we use an orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ for the vectors. The ‘basis’ for the scalars is $\{1\}$. Coefficients of vectors or scalars on this basis are all real. For an orthonormal basis, $\mathbf{e}_i \mathbf{e}_i = 1$, and for $i \neq j$ we have $\mathbf{e}_i \mathbf{e}_j = -\mathbf{e}_j \mathbf{e}_i (= \mathbf{e}_i \wedge \mathbf{e}_j)$.

With this we can develop the geometric product of two general vectors \mathbf{a} and \mathbf{b} . Choose the basis to have $\mathbf{a} = a\mathbf{e}_1$, and $\mathbf{b} = b(\cos \phi \mathbf{e}_1 + \sin \phi \mathbf{e}_2)$. This implies that $\mathbf{I} \equiv \mathbf{e}_1 \wedge \mathbf{e}_2$ can be interpreted as the unit area element of the (\mathbf{a}, \mathbf{b}) -plane, and ϕ is the angle *from* \mathbf{a} *to* \mathbf{b} in that plane \mathbf{I} . That gives:

$$\mathbf{a}\mathbf{b} = ab(\cos \phi + \mathbf{I} \sin \phi)$$

Since this result is independent of the coordinate system, the geometric product always contains this complete geometric information about the relationship of the vectors: relative angle and common plane, and relative size (the latter is slightly better conveyed by the geometric division $\mathbf{a}/\mathbf{b} = \mathbf{a}\mathbf{b}^{-1} = (a/b)(\cos\phi + \mathbf{I}\sin\phi)$ which gives a/b as the magnitude). The fact that the full relative information of the vectors is contained in a retrievable manner is the geometric intuition behind the invertibility of the geometric product.

Actually computing the geometric product for two general vectors on an arbitrary orthonormal basis shows how familiar the coefficients are:

$$\begin{aligned} (a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3)(b_1\mathbf{e}_1 + b_2\mathbf{e}_2 + b_3\mathbf{e}_3) \\ = (a_1b_1 + a_2b_2 + a_3b_3) + (a_2b_3 - a_3b_2)\mathbf{e}_2\mathbf{e}_3 + (a_3b_1 - a_1b_3)\mathbf{e}_3\mathbf{e}_1 + (a_1b_2 - a_2b_1)\mathbf{e}_1\mathbf{e}_2 \end{aligned} \quad (2.1)$$

This shows the scalar part equal to the inner product, and the part of grade 2 of which the coefficients are similar to that of a cross product but now on a *bivector basis* $\{\mathbf{e}_2 \wedge \mathbf{e}_3, \mathbf{e}_3 \wedge \mathbf{e}_1, \mathbf{e}_1 \wedge \mathbf{e}_2\} = \{\mathbf{e}_2\mathbf{e}_3, \mathbf{e}_3\mathbf{e}_1, \mathbf{e}_1\mathbf{e}_2\}$.

As you compute on, taking the geometric product of three vectors $\mathbf{a}\mathbf{b}\mathbf{c}$, you find that the product of 3 vectors contains a 1-vector part and a 3-vector part. That 3-vector part is a multiple of a *trivector basis* $\{\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3\}$ (and by a scalar factor equal to the determinant of the matrix $(\mathbf{a}\ \mathbf{b}\ \mathbf{c})$). We will often denote this element $\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3$, which is a unit pseudoscalar for three dimensional space, by the special symbol \mathcal{I} . You can think of it as the oriented unit volume element.

As a basis to denote elements of the geometric algebra of 3-dimensional real Euclidean space, we thus obtain:

$$\left\{ \underbrace{1}_{\text{scalars}}, \underbrace{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3}_{\text{vector space}}, \underbrace{\mathbf{e}_2 \wedge \mathbf{e}_3, \mathbf{e}_3 \wedge \mathbf{e}_1, \mathbf{e}_1 \wedge \mathbf{e}_2}_{\text{bivector space}}, \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3}_{\text{trivector space}} \right\}$$

or using the symbol \mathcal{I} and orthogonality of the basis vectors we may rewrite this as:

$$\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathcal{I}\mathbf{e}_1, \mathcal{I}\mathbf{e}_2, \mathcal{I}\mathbf{e}_3, \mathcal{I}\}.$$

The pseudoscalar \mathcal{I} squares to -1 (as you may verify), and so do each of the bivector basis elements $\mathcal{I}\mathbf{e}_i$. Yet these are not complex numbers: for instance, $\mathcal{I}\mathbf{e}_1$ does not commute with \mathbf{e}_2 , whereas a complex scalar would have done so. You should think of the basis elements of geometric algebra as real representations of spanned subspaces of dimension 0, 1, 2 and 3 within a 3-dimensional space.

The linearity and associativity of the geometric product implies that it can simply be implemented on this basis as a matrix multiplication, see (Dorst et al. 1999). For the geometric algebra of n -dimensional space, the basis has 2^n elements, so you soon need a more efficient implementation such as (Fontijne et al. 2002).

(c) Reflections and rotations

The reflection of a vector \mathbf{x} into a line characterised by the *unit* vector \mathbf{e} is given by the elementary formula (see Figure 1):

$$\mathbf{x} \mapsto 2(\mathbf{e} \cdot \mathbf{x})\mathbf{e} - \mathbf{x}.$$

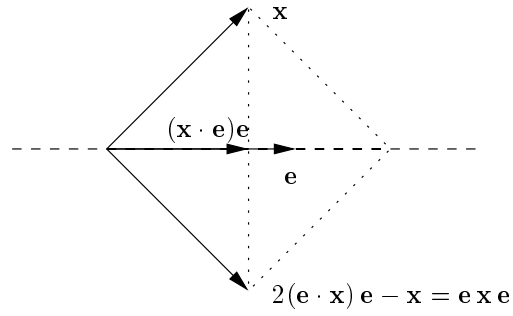


Figure 1. Reflection of a vector \mathbf{x} in a unit vector \mathbf{e} .

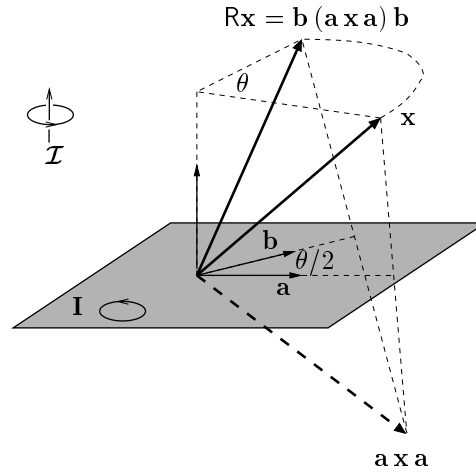


Figure 2. One rotation in a plane parallel to \mathbf{I} is two reflections in vectors in that plane, separated by half the rotation angle.

Using the geometric product definition of the inner product as $\mathbf{e} \cdot \mathbf{x} \equiv \frac{1}{2}(\mathbf{e}\mathbf{x} + \mathbf{x}\mathbf{e})$, we can rewrite this to the form

$$\mathbf{x} \mapsto \mathbf{e}\mathbf{x}\mathbf{e}.$$

Two reflections make a rotation, even in space, see Figure 2. First reflecting in \mathbf{a} , then in \mathbf{b} , gives a rotation over an axis perpendicular to the $\mathbf{a} \wedge \mathbf{b}$ -plane, over an angle that is *twice* the angle from \mathbf{a} to \mathbf{b} (and this angle also gives the sense of rotation).

Therefore if we have two unit vectors \mathbf{a} and \mathbf{b} , the operation

$$\mathbf{x} \mapsto \mathbf{b}(\mathbf{a}\mathbf{x}\mathbf{a})\mathbf{b} = (\mathbf{b}\mathbf{a})\mathbf{x}(\mathbf{b}\mathbf{a})^\sim.$$

is that rotation in the $\mathbf{a} \wedge \mathbf{b}$ plane. We observe that this rotation is generated by an element $R = \mathbf{b}\mathbf{a}$, as applied to a vector by the recipe:

$$\mathbf{x} \mapsto R\mathbf{x}\tilde{R}.$$

This object R is called a (*unit*) rotor. It is the product of two unit vectors and satisfies

$$R\tilde{R} = 1,$$

as is easily verified: $R\tilde{R} = \mathbf{baab} = 1$. We will use these rotors as the representation of rotations. Writing out the geometric product, we find for the unit vectors \mathbf{a} and \mathbf{b} :

$$\mathbf{ba} = \mathbf{b} \cdot \mathbf{a} + \mathbf{b} \wedge \mathbf{a} = \cos(\theta/2) - \mathbf{I} \sin(\theta/2) = e^{-\mathbf{I}\theta/2},$$

where $\theta/2$ is the angle from \mathbf{a} to \mathbf{b} , and \mathbf{I} is the unit 2-blade for the $(\mathbf{a} \wedge \mathbf{b})$ -plane. The exponential notation is based on the algebraic property that $\mathbf{I}\mathbf{I} = -1$, just as for complex numbers, and is often a convenient shorthand. It shows that for a rotation, the *bivector angle* $\mathbf{I}\theta$ contains all information: both the angle and the plane in which it should be measured. From this bivector angle, one can immediately construct the rotor performing the corresponding rotation.

In this representation of rotations, half angles occur a lot, and so we will introduce the shorthand notation $\theta' \equiv \theta/2$. A rotor is then written as:

$$R = e^{-\mathbf{I}\theta'} = \cos \theta' - \mathbf{I} \sin \theta'.$$

So the primed angles are ‘rotor angles’ to be used in this two-sided representation of rotations; the non-primed angles are actual angles of rotation when one starts applying the rotors using $R\mathbf{x}\tilde{R}$. This notation is non-standard, but we find it convenient for our purposes in this paper, since it unclutters many formulas.

A small remark about necessary normalisation: one obtains slightly more general formulas by not stipulating that the reflection be done in a *unit* vector; then $\mathbf{x} \mapsto \mathbf{e}\mathbf{x}\mathbf{e}^{-1}$ performs the reflection since it automatically performs the normalisation by \mathbf{e}^2 ; similarly for the rotation $\mathbf{x} \mapsto (\mathbf{ba})\mathbf{x}(\mathbf{ba})^{-1}$. These non-unit rotors correspond to non-unit quaternions. In this paper, we have decided to normalise all operators to unity.

Using the transformation formula $\mathbf{x} \mapsto R\mathbf{x}\tilde{R}$, we see that a rotor R and ‘minus that rotor’ $-R$ give the same resulting rotation. This does not necessarily mean that the representation of rotations by rotors is two-valued: these rotors can be distinguished when doing relative rotations of connected objects, as we will see in section a; they correspond to the two ways of achieving the same rotation by going clockwise or counterclockwise. For some applications, that is very useful to have, and it comes for free with the rotor representation.

of a

(d) Quaternions

Rotors are closely related to quaternions: quaternions are simply rotors separated from their natural context in geometric algebra, and because of that to many people unfortunately more mysterious than they need to be. Rotors are real operators in a real vector space, with a scalar part (related to the cosine of the angle) and a bivector part (containing sine and rotation plane). (Yes, the bivectors have a negative square, but that does not make them any less ‘real’.) In quaternion literature, the non-scalar part of a quaternion is often seen as a vector that denotes the rotation axis, but expressed on a strange basis of complex vector quantities i ,

j , k that square to -1 and do not commute. This makes them harder to imagine than is necessary. For us, these basis elements are not vectors but bivectors:

$$i = \mathcal{I}\mathbf{e}_1 = \mathbf{e}_2\mathbf{e}_3, \quad j = \mathcal{I}\mathbf{e}_2 = \mathbf{e}_3\mathbf{e}_1, \quad k = \mathcal{I}\mathbf{e}_3 = \mathbf{e}_1\mathbf{e}_2,$$

Note that $ji = k$ and cyclic, and $ijk = 1$. (In much quaternion literature, one uses $ij = k$ and $ijk = -1$. This is a trivial algebraic isomorphism which can be achieved by putting $j \rightarrow -j$. We use the above since it makes the correspondence between rotors and quaternions more symmetrical, avoiding extraneous minus signs.) This bivector basis represents not the rotation *axis* \mathbf{e} , but the rotation *plane* \mathbf{I} . The two are related simply by geometric duality (i.e. quantitative orthogonal complement) as

$$\text{axis } \mathbf{e} \text{ to bivector } \mathbf{I}: \quad \mathcal{I}\mathbf{e} = \mathbf{I},$$

and their coefficients are similar, though on totally different bases (one for vectors, the other for bivectors), see eq.(2.1).

The standard notation for a unit quaternion $q = q_0 + \mathbf{q}$ separates it into a scalar part and a supposedly ‘complex vector’ part \mathbf{q} denoting the axis. This naturally corresponds to a rotor $R = q_0 - \mathcal{I}\mathbf{q}$ having a scalar part and a bivector part:

$$\text{quaternion } q_0 + \mathbf{q} \quad \leftrightarrow \quad \text{rotor } q_0 - \mathcal{I}\mathbf{q}. \quad (2.2)$$

In the latter \mathbf{q} is now a *real* vector denoting a rotation axis. When combining these quantities, the common geometric product naturally takes over the role of the rather unexpected quaternion product (with the same result, as we will see), and we re-emphasise that all quantities are now real, in the sense of being directly interpretable as tangible objects in a Euclidean space (although some of them square to -1).

Little of what we are going to do could not be formulated in terms of quaternions, but the geometric algebra method gives us a more natural context to use – and to tie into other operations we want to do later (not in this paper), such as the combination with translations or conformal mappings, and using geometric calculus.

(e) Composition of rotations

Let us look at the product $R_t = R_2R_1$ of two rotors: R_1 followed by R_2 , expressed in their rotor angles. The geometric product gives the answer (we use the shorthand $c'_i = \cos \theta'_i = \cos(\theta_i/2)$ and $s'_i = \sin \theta'_i = \sin(\theta_i/2)$):

$$\begin{aligned} c'_t - \mathbf{I}_t s'_t &= (c'_2 - \mathbf{I}_2 s'_2)(c'_1 - \mathbf{I}_1 s'_1) \\ &= c'_1 c'_2 + s'_1 s'_2 \langle \mathbf{I}_2 \mathbf{I}_1 \rangle_0 - c'_2 s'_1 \mathbf{I}_1 - c'_1 s'_2 \mathbf{I}_2 + s'_1 s'_2 \langle \mathbf{I}_2 \mathbf{I}_1 \rangle_2 \end{aligned}$$

We have split the result in scalar (i.e. 0-blade) and 2-blade parts. Note that the two objects in grade brackets are components of the products of \mathbf{I}_1 and \mathbf{I}_2 . These are rotations with rotor angles of $\pi/2$ in the planes of the rotations we want to compose (they correspond to 180 degree rotations). The scalar $\langle \mathbf{I}_2 \mathbf{I}_1 \rangle_0$ turns out to be the cosine c_\perp of the angle θ_\perp between those planes, $\langle \mathbf{I}_2 \mathbf{I}_1 \rangle_2$ is the plane \mathbf{I}_\perp perpendicular to both, weighted by the sine s_\perp of the angle θ_\perp from \mathbf{I}_1 to \mathbf{I}_2 . Substituting this, we get:

$$c'_t - \mathbf{I}_t s'_t = c'_1 c'_2 + s'_1 s'_2 c_\perp - c'_2 s'_1 \mathbf{I}_1 - c'_1 s'_2 \mathbf{I}_2 + s'_1 s'_2 s_\perp \mathbf{I}_\perp$$

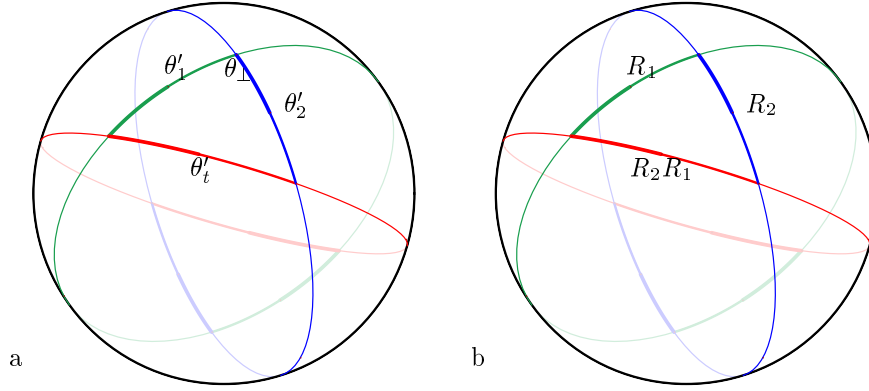


Figure 3. (a) A spherical triangle. (b) Composition of rotations through concatenation of rotor arcs. R_2R_1 is the composite rotor of doing first R_1 , then R_2 , and is the arc completing the spherical triangle. In this figure, the arcs are oriented, running from thick to thin (unfortunately Cinderella 1.2 (Richter-Gebert & Kortenkamp 2000) which was used to generate these pictures has no arrows in its spherical view).

Now consider the equation given by the scalar part, and write it out in full detail:

$$\cos \theta'_t = \cos \theta'_1 \cos \theta'_2 + \sin \theta'_1 \sin \theta'_2 \cos \theta_\perp.$$

This is precisely the ‘cosine rule for sides’ from spherical trigonometry, depicted in Figure 3a. It suggests thinking of the composition of rotors as the addition of spherical arcs, as in Figure 3b, and this is indeed correct (see also (Kuipers 1999)). Note that it is essential that the rotations are represented by the *rotor angles* (the primed angles in the figure which are half the actual rotation angles), whereas the angle θ_\perp between the rotation planes occurs straight, not halved. As we saw, this follows immediately from the mathematics of their multiplication.

For example, let us take a rotation over in the $\mathbf{e}_1\mathbf{e}_3$ -plane over $\pi/2$ followed by a rotation in the $\mathbf{e}_3\mathbf{e}_2$ -plane over $\pi/2$. As rotors, these are $(1 - \mathbf{e}_1\mathbf{e}_3)/\sqrt{2}$ and $(1 - \mathbf{e}_3\mathbf{e}_2)/\sqrt{2}$. We draw two great circles, with poses corresponding to the rotation planes $\mathbf{e}_1\mathbf{e}_3$ and $\mathbf{e}_3\mathbf{e}_2$. On these great circles, the rotations over $\pi/2$ are represented as oriented arcs of length $\pi/4$ (the corresponding rotor angle). These arcs are freely movable along their great circles. To compose the rotations, we need to make them meet, so that we can do R_1 , then R_2 . This is depicted in Figure 3. The arc completing the spherical triangle is in a skew plane, with a length that looks like it might be $\pi/3$. Actual computation confirms this value for the rotor angle, and a plane of $(-\mathbf{e}_3\mathbf{e}_2 - \mathbf{e}_1\mathbf{e}_3 + \mathbf{e}_2\mathbf{e}_1)/\sqrt{3}$. Therefore the resulting rotation is over $2\pi/3$ in this plane. Rewriting to $-\mathcal{I}(-\mathbf{e}_1 - \mathbf{e}_2 + \mathbf{e}_3)/\sqrt{3}$ using eq.(2.2) shows that the rotation axis is $(-\mathbf{e}_1 - \mathbf{e}_2 + \mathbf{e}_3)$.

The rotor composition assumes the perhaps more familiar form of a quaternion product if we use eq.(2.2) to express it as a quaternion multiplication. We embed

quaternions as rotors, perform the multiplication, and transfer back:

$$\begin{aligned}
qp &= (q_0 + \mathbf{q})(p_0 + \mathbf{p}) \\
&\leftrightarrow (q_0 - \mathcal{I}\mathbf{q})(p_0 - \mathcal{I}\mathbf{p}) \\
&= q_0p_0 - \langle \mathcal{I}\mathbf{q}\mathcal{I}\mathbf{p} \rangle_0 + \mathcal{I}(\mathbf{q}p_0 + \mathbf{p}q_0 - \mathcal{I}^{-1}\langle \mathcal{I}\mathbf{q}\mathcal{I}\mathbf{p} \rangle_2) \\
&= q_0p_0 - \langle \mathbf{q}\mathbf{p} \rangle_0 - \mathcal{I}(\mathbf{q}p_0 + \mathbf{p}q_0 + \mathcal{I}^{-1}\langle \mathbf{q}\mathbf{p} \rangle_2) \\
&= p_0q_0 - \mathbf{p} \cdot \mathbf{q} - \mathcal{I}(p_0\mathbf{q} + q_0\mathbf{p} - \mathbf{p} \times \mathbf{q}) \\
&\leftrightarrow (p_0q_0 - \mathbf{p} \cdot \mathbf{q}) + (p_0\mathbf{q} + q_0\mathbf{p} - \mathbf{p} \times \mathbf{q})
\end{aligned}$$

(where we used the definition of the cross product in 3D geometric algebra, which is $\mathbf{p} \times \mathbf{q} = (\mathbf{q} \wedge \mathbf{p})\mathcal{I}$), retrieving the usual multiplication formula from quaternion literature. This shows that the quaternion product is just the geometric product on rotors.

As an alternative way of looking at the oriented arc representation of rotations, consider again that a rotation is a double reflection in a ‘vee’ formed by two unit vectors in the rotation plane through the origin, separated by half the rotation angle as in Figure 2. The actual absolute orientation of these vectors in the plane is immaterial (as you may check, it must be rotationally invariant since any vector out of the plane can be rotated by the construction!). Now, composing two rotations (in possibly different planes) is identical to composing two double reflections; it is natural to rotate the two vees of vectors so that the first and last of both vees coincide. Then it is obvious that those two reflections cancel each other in the composition, while the other two remain to give the vee for the resulting rotation. Now surround these unit vectors by a sphere, and you see the characteristics of the rotation sphere representation: each vee of vectors determines an arc of half the rotation angle; the rotation in their plane is the permissible free sliding of this arc along its great circle; and the composition is the completion of a spherical triangle.

3. Averaging orientations

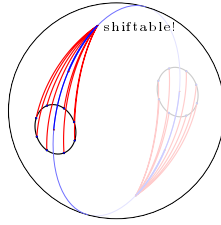
(a) A metric for orientations and rotations

If one had to design a metric for orientations, to give a measure for their difference, some function of the angle between them would seem appropriate. When considering rotations, which differ in both angle and in rotation planes, it seems less obvious how to mix those into a sensible measure.

In the embedding ‘rotor space’, a natural distance measure for two rotations characterised by rotors R_1 and R_2 is arguably the ‘(squared) norm of the difference’:

$$d^2(R_1, R_2) = \|R_2 - R_1\|^2 = \langle (R_2 - R_1)(R_2 - R_1)^\sim \rangle_0.$$

When we measure that for unit rotors, we should make use of the fact that they have norm 1. This is not quite the same as measuring distances ‘along the unit rotor manifold in rotor space’, for we are measuring a chord rather than an arc.

Figure 4. *A metric on rotations.*

Working this out, we find:

$$\begin{aligned}
 d^2(R_1, R_2) &= \langle (R_2 - R_1)(\tilde{R}_2 - \tilde{R}_1) \rangle_0 \\
 &= \langle R_2 \tilde{R}_2 + R_1 \tilde{R}_1 - R_1 \tilde{R}_2 - R_2 \tilde{R}_1 \rangle_0 \\
 &= 2 - \langle R_2 \tilde{R}_1 + R_1 \tilde{R}_2 \rangle_0 \\
 &= 2(1 - \langle R_2 \tilde{R}_1 \rangle_0)
 \end{aligned}$$

The ‘ratio’ $R_2 \tilde{R}_1$ of two rotors is again a rotor; the scalar part of a rotor is the cosine of its angle $\theta/2$ (which is half the rotation angle of the corresponding rotation θ). Therefore we find for *the (squared) distance between two rotations*:

$$d(R_1, R_2) = \sqrt{2(1 - \cos \theta')} = 2|\sin \frac{\theta'}{2}| = 2|\sin \frac{\theta}{4}| \quad (3.1)$$

where $\theta = 2\theta'$ is the angle of the ‘difference’ in rotations characterising the rotor $R_2 \tilde{R}_1$, which turns R_1 into R_2 . Orientations are characterised as relative rotations, and so the same distance measure can be applied to them.

The distance measure has two important properties: first, only the *relative rotation* matters (so that the distance measure is uniform over the whole unit rotor manifold); second, of the relative rotation only the *angle* contributes to the distance measure, not its plane. Figure 4 shows a rotor (as an arc on the rotation sphere), and some rotors that are all equally different from it. Their endpoints all lie on a circle around the endpoint of the arc. You can see that some differ mainly in angle, others mainly in rotation plane. It is comforting that the distance measure derived on the unit rotor manifold has such a clear interpretation for the rotational arcs. You should realise, though, that the figure is slightly too specific: since the arc representation can be freely slid along the great circle, we only get *all* rotations with the same distance to the give rotation of we perform this construction all along the great circle indicated.

Note that for small rotational differences, the distance of eq. (3.1) is proportional to $|\theta/2|$, and this is perhaps half of what one would have expected. For the finite rotation over angle π , the distance is $\sqrt{2}$, for a rotation over 2π the measure is 2, which is its maximum, and only for a rotation of 4π do we get a rotor of distance zero to the original rotation. This is strange, but strangely correct. It is related to the property of rotors to describe relative rotations rather than absolute rotations (what are those anyway?). You can see this in a ‘Balinese dance experiment’. Stand up straight, with your hand immediately in front of your left shoulder, flat, palm up; now make the motions necessary to make your hand turn 2π in its own plane,

around its center (you may have to move your torso a little); you find that your elbow sticks up in the air; continue turning the hand in the same direction; when you have reached 4π , you are (surprise!) back in your original pose and ready to do this once more. The moral: the periodicity of the relative rotation of the hand was 4π , not 2π ; the maximally different orientation was halfway the experiment, at the 2π rotation, after that the rotational distance decreased again. This is what our distance measure gives as well.

This experiment also illuminates our earlier remark on the physical significance of the apparent sign multiplicity of the rotor representation. Merely observing the hand, its orientation might just as well be characterised by $R = e^{-\mathbf{I}\theta/2}$ as by $R' = -R = -e^{-\mathbf{I}\theta/2}$. However, the latter equals $R' = e^{-\mathbf{I}\pi} e^{-\mathbf{I}\theta/2} = e^{-\mathbf{I}(2\pi+\theta)/2}$, so it is more properly interpreted as the rotation over $(2\pi + \theta)$. For the hand there is no difference, but when you consider the whole arm the elbow demonstrates the non-equivalence.

(b) *The best average*

Now that we have a distance measure, we can treat our first problem: how to average rotations. Suppose we have a set of measured rotors $\{R_i\}_{i=1}^n$, all with the same accuracy (we will refine this in section c to ‘independently identically distributed’), and that we are looking for the rotor that has the minimum total squared distance to all of them. This is the rotor \hat{R} that minimises

$$\sum_i \langle (R_i - \hat{R}) \sim (R_i - \hat{R}) \rangle_0 = 2 \sum_i \langle 1 - \tilde{R}_i \hat{R} \rangle_0 = 2(n - \langle \sum_i \tilde{R}_i \hat{R} \rangle_0) = 2(n - \langle \tilde{S} \hat{R} \rangle_0), \quad (3.2)$$

by a rewriting similar to section a, and linearity of the scalar product. The temporary quantity $S \equiv \sum_i R_i$, (obviously consisting of a scalar and bivector part) is not a unit rotor, so let us normalise it:

$$\bar{R} \equiv \frac{S}{\|S\|} = \frac{\sum_i R_i}{\|\sum_i R_i\|}$$

Now we can rewrite the quantity to be minimised to: $2(n - \|S\| \langle \bar{R} \hat{R} \rangle_0)$. The first term is a given constant, the second is maximal when $\hat{R} = \bar{R}$, so that is the solution. Therefore:

The least-square-error estimate of a set of independent identically distributed rotors is their normalised sum \bar{R} .

This is the proper formula for the ‘mean rotor’ of an ensemble.

It is not that different from translations, where the best linear estimate of a set of measurements $\{\mathbf{x}_i\}_{i=1}^n$ of the same position would be the mean $\hat{\mathbf{x}} = (\sum_i \mathbf{x}_i)/n$.

Some care is required when averaging these rotors, due to the double representation of orientations, in which $-R$ may represent the same rotation as R (if seen in isolation). If you have been sloppy in casting your measurement data into rotors, you may have both forms and should not average over them. So then the procedure is: pick one rotor, and change the signs of all the other such that the scalar part of their products is positive, then perform the averaging.

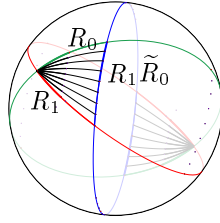


Figure 5. *Illustration of the interpolation of rotors R_0 and R_1 .*

4. Interpolating and filtering rotations

(a) *Interpolation: the slerp*

If we would have two position vectors \mathbf{x}_0 and \mathbf{x}_1 , we would linearly interpolate between them using the familiar

$$\text{lerp}(\mathbf{x}_0, \mathbf{x}_1; \lambda) = (1 - \lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1, \quad (4.1)$$

with λ running from 0 to 1. For interpolation between two rotors R_0 and R_1 , the analogous formula (i.e. $(1 - \lambda)R_0 + \lambda R_1$) would not work since the result would not be a rotor; and renormalisation would make the interpolation non-uniform.

We would like to have a rotation that interpolates rotation angles linearly, so that divides the total rotation angle θ between R_0 and R_1 in a linear manner. Visualising this on the sphere, the intermediate rotations have rotors which are pictured as arcs to evenly spaced intermediate points on the completing side of the spherical triangle, as in Figure 5. In summary, we need to find the relative rotation and split it up into equal parts. We write the relative rotation as:

$$R_1 \tilde{R}_0 \equiv e^{-\mathbf{I}\theta/2} = e^{-\mathbf{I}\theta'} = \cos \theta' - \mathbf{I} \sin \theta',$$

so that \mathbf{I} denotes the relative rotation plane, θ is the relative rotation angle, and $\theta' \equiv \theta/2$ is the relative rotor angle. This gives $\mathbf{I}R_0 = (R_1 - R_0 \cos \theta') / \sin \theta'$. Then the linear interpolation is achieved by rotation over a fraction $\lambda\theta'$ of the angle, from R_0 towards R_1 . This is the rotor $e^{-\lambda\mathbf{I}\theta'} R_0$, which may be expressed as:

$$\begin{aligned} R_\lambda &= (\cos(\lambda\theta') - \mathbf{I} \sin(\lambda\theta')) R_0 \\ &= \frac{R_0 \sin(\theta') \cos(\lambda\theta') - R_0 \cos(\theta') \sin(\lambda\theta') + R_1 \sin(\lambda\theta')}{\sin \theta'} \\ &= \frac{\sin(1 - \lambda)\theta'}{\sin \theta'} R_0 + \frac{\sin \lambda\theta'}{\sin \theta'} R_1 \equiv \text{slerp}(R_0, R_1; \lambda) \end{aligned} \quad (4.2)$$

This is the linear interpolation formula for rotations, which we abbreviate as $\text{slerp}[R_0, R_1; \lambda]$ (following (Shoemake 1985)). It is slightly more involved than linear interpolation of vectors since the linear interpolation is now ‘under the sine’. Please note that the angles involved in the interpolation are the rotor angles, which are *half* the rotation angles. When R_0 and R_1 are not very different, θ' is small, and eq.(4.2) becomes a linear interpolation of rotors: $\text{slerp}(R_0, R_1; \lambda) \approx \text{lerp}(R_0, R_1; \lambda)$.

In computer graphics, more general interpolations (such as Bezier) between a set of key rotations are done by extending this principle. The fact that this

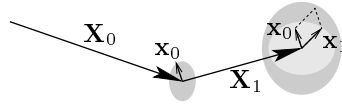


Figure 6. *Propagation of errors in translations.*

was demonstrated to be possible in (Shoemaker 1985), was one of the motivations for that field to begin using quaternions rather than rotation matrices for their representation of rotations and orientations.

(b) *Filtering orientations*

With the rotor/quaternion representation of rotations, one has ‘linearised’ the rotational issues. In a recent paper (Lee & Shin 2002), Lee and Shin capitalise on this by lifting the intuition and techniques from linear filtering to the exponential argument of the quaternion. This is a powerful way to, say, ‘smooth’ a sequence of orientations, or to ‘sharpen’ it. They give the explicit computation of the coefficients for such quaternion filters based on applying the classical linear techniques in the tangent space to the unit rotor manifold.

Here we will not go into these techniques since we are more interested in statistical combination of estimates, but we recommend the paper to those readers interested in simply filtering orientations.

5. Statistical processing of orientations

To predict orientations, we look at how an orientation is evolving and give a good guess of what it could be at the next time instant. Our derivation of the interpolation formula shows that this can also be used for extrapolation: just extend the interpolated arc from R_0 to R_1 beyond R_1 , by taking $\lambda > 1$. Once we have decided how far to extend it, a prediction for R_2 has been obtained. If we also have measurement data available about the actual value of R_2 , with some noise, then we would like to combine these two optimally. This is the update step of any discrete time Kalman filter, where the optimal combination is done statistically, to produce a minimum variance unbiased linear estimate (Maybeck 1979). In this section, we produce the optimal weighting procedure for prediction and observation of orientations. To develop this weighting, we obviously first need to understand how to represent and compute confidence measures in orientations.

(a) *Characterisation of noise in orientations*

Let us recall how we usually combine noise in translation measurements. Suppose we have a translation vector \mathbf{X}_0 , and a translation \mathbf{X}_1 , both with some noise. To see what the distribution of the result $\mathbf{X}_0 + \mathbf{X}_1$ is, we need to do some averaging: take an error vector \mathbf{x}_0 (located at the tangent space at the endpoint of \mathbf{X}_0) and an error vector \mathbf{x}_1 (located at the tangent space at the endpoint of \mathbf{X}_1); these produce the error vector $\mathbf{x}_0 + \mathbf{x}_1$ at the endpoint of $\mathbf{X}_0 + \mathbf{X}_1$. Drawing \mathbf{X}_0 and \mathbf{X}_1 head to tail, we have basically ‘transferred’ the error \mathbf{x}_0 to the end of $\mathbf{X}_0 + \mathbf{X}_1$, and added, see Figure 6. This is permissible: translations are commutative, and the tangent

spaces are everywhere isomorphic, and even isomorphic to the Euclidean space itself so that this also works for non-infinitesimal displacements. In particular, we are allowed to write $(\mathbf{x}_1 + \mathbf{X}_1) + (\mathbf{x}_0 + \mathbf{X}_0)$ as

$$\mathbf{x}_t + \mathbf{X}_t = (\mathbf{x}_1 + \mathbf{x}_0) + (\mathbf{X}_1 + \mathbf{X}_0),$$

so that the ‘quantity plus error’ retains its form, and the total error $(\mathbf{x}_1 + \mathbf{x}_0)$ is independent of the data \mathbf{X}_0 and \mathbf{X}_1 . If we now want to average this for distributions of \mathbf{x}_0 and \mathbf{x}_1 , then the total distribution becomes the convolution of the contributing distributions:

$$p_t(\mathbf{x}_t) = \int p_0(\mathbf{x}_0) p_1(\mathbf{x}_t - \mathbf{x}_0) d\mathbf{x}_0 \equiv (p_0 * p_1)(\mathbf{x}_t).$$

The Gaussian family of distributions is particularly nice, since convolution of two Gaussians is again a Gaussian, merely coarser. To be precise, denoting the Gaussian with covariance matrix C by $\mathcal{G}[C]$, the resulting distribution is simply:

$$\mathcal{G}[C_0] * \mathcal{G}[C_1] = \mathcal{G}[C_0 + C_1].$$

In an isotropic situation, the equi-probability lines of the resulting pdf are equidistant lines of the metric, induced by the covariance metric.

Compare this to rotations represented by rotors. The main issue is that rotations are *not* commutative, so we have to choose where to represent the noise. It turns out (in appendix Appendix A) that it is enough to represent it only by a small rotor applied after the main rotor, bearing in mind Figure 4 of ‘similar rotors’. So for rotor R_0 we multiply by a small rotor r_0 performed ‘after’ the main rotation to obtain $r_0 R_0$. (There is an alternative way of characterising rotor noise which we will discuss in section d.) Now apply a second rotor R_1 with similar noise and attempt to rewrite to a form showing the resulting rotor $R_1 R_0$, accompanied by some little noise rotor:

$$(r_1 R_1) (r_0 R_0) = (r_1 R_1 r_0 \tilde{R}_1) (R_1 R_0) = r_t R_t.$$

This shows that the resulting noise is still a small rotor, but that the main rotor R_1 plays a role in how it propagates. It implies that the composition of distributions of rotors is not simply a convolution, but assumes the form:

$$p_t(r_t) = \int p_0(r_0) p_1(r_t R_1 \tilde{r}_0 \tilde{R}_1) dr_0$$

That the pdfs are defined in terms of rotors is less of a problem than the complicated argument of p_1 . This composition is not a convolution, and not closed for the Gaussian family. At the moment, I do not know how to treat this composition in general. In an approximation for small noise, though, the composition becomes tractable.

(b) Composition of small distributions

If the noise rotors are small, we can approximate them and study their ‘infinitesimal interaction’, which is simpler than the general interaction. For a small rotor

r_0 in our metric, the distance between it and the identity should be small, and therefore its rotor angle θ'_0 is small. This implies that we can write, to first order:

$$r_0 = e^{-\mathbf{I}_0 \theta'_0} = \cos \theta'_0 - \mathbf{I}_0 \sin \theta'_0 \approx 1 - \mathbf{I}_0 \theta'_0 \equiv 1 - \mathbf{i}_0,$$

where we defined \mathbf{i}_0 as the small bivector $\mathbf{I}_0 \theta'_0$, and we define \mathbf{i}_1 and \mathbf{i}_t similarly. (This approximation is rather good, for 10 degrees noise in rotation angle the error is about 0.4 percent; 5 percent error is reached for 36 degrees.) Expanding the composition of a total rotor $R_t \equiv R_1 R_0$ and its noise r_t now gives:

$$r_t R_t = (r_1 R_1) (r_0 R_0) \approx (1 - \mathbf{i}_1 - R_1 \mathbf{i}_0 \tilde{R}_1) (R_1 R_0) = (1 - \mathbf{i}_t) R_t.$$

It follows that for small amounts of noise, the propagation is simply:

$$\mathbf{i}_t = \mathbf{i}_1 + R_1 \mathbf{i}_0 \tilde{R}_1 = \mathbf{i}_1 + R_1 \mathbf{i}_0 \quad (5.1)$$

We rewrote this outcome using the linear mapping R_1 (the rotation), to simplify notation for the following derivation. The rotation is defined as a linear mapping on vectors, but through the structure of geometric algebra such transformations on the vector space induce straightforward transformations of all objects through *outermorphism*. For instance, the rotation of a vector \mathbf{x} by $R\mathbf{x}\tilde{R}$ naturally defines the rotation of a bivector $\mathbf{x} \wedge \mathbf{y}$ as $(R\mathbf{x}\tilde{R}) \wedge (R\mathbf{y}\tilde{R})$; and using elementary geometric algebra, you are actually allowed to write this as $R(\mathbf{x} \wedge \mathbf{y})\tilde{R}$. In fact, any object \mathbf{X} in geometric algebra will rotate as $R\mathbf{X}\tilde{R}$.

Note that eq.(5.1) only involves bivectors, elements of grade 2. These define a linear space, and they can all be expressed on the bivector basis of this space. In this space they are just ‘vectors’, the only unusual issue is that their squares are negative. Defining a distribution over such elements is permitted. (If you feel uncomfortable about this, you may use the fact that we are mostly interested in 3-D, and think about the duals of the bivectors instead; these are just vectors which are axes for the small extra noise rotation, and their lengths are the small rotation angles of the noise. Then the small bivector distributions just become distributions of small vectors, a classical and reassuring image.)

If we have a distribution p_0 of \mathbf{i}_0 and an independent distribution p_1 of \mathbf{i}_1 , this induces a distribution p_t on \mathbf{i}_t . Using \mathbf{i}_0 as the parametrisation, this distribution can be integrated as:

$$p_t(\mathbf{i}_t) = \int p_0(\mathbf{i}_0) p_1(\mathbf{i}_t - R_1 \mathbf{i}_0) d\mathbf{i}_0 \equiv (p_0 *_{R_1} p_1)(\mathbf{i}_t). \quad (5.2)$$

The symbol $*_{R_1}$ denotes that this resembles, but is different from, a convolution: we have rotation R_1 in the second argument, so it is a ‘convolution with a twist’.

Although it is not quite a convolution, the Gaussian family of functions is still closed under this composition, and can therefore be used in our thinking about covariance. In fact, denoting the Gaussian with covariance matrix \mathbf{C} by $\mathcal{G}[\mathbf{C}]$, we can show that:

$$\mathcal{G}[\mathbf{C}_0] *_{\mathbf{R}} \mathcal{G}[\mathbf{C}_1] = \mathcal{G}[\mathbf{R}\mathbf{C}_0\bar{\mathbf{R}} + \mathbf{C}_1] \quad (5.3)$$

where $\bar{\mathbf{R}}$ is the adjoint of \mathbf{R} , representable by the transpose of the matrix of \mathbf{R} . This result is immediate from the combination of covariances of distributions (see

e.g. (Maybeck 1979)). since in eq.(5.1), \mathbf{i}_t is a linear combination of \mathbf{i}_0 and \mathbf{i}_1 . The sandwiching between rotation mappings is just how covariance matrices rotate, through the usual linear algebra of bilinear forms.

It is comforting that Gaussians are still natural, since we may expect them to arise in practice whenever we have a reasonable number of rotations with independent errors (since the ensemble will then tend to a Gaussian distribution by the central limit theorem).

(c) *Optimal interpolation of rotors*

Let us recall the procedure for optimally combining translation estimates. The linear combination of two estimates of a translation is a lerp function. The total covariance of the lerp of eq.(4.1) is the ‘qerp’ (quadratic interpolation):

$$\text{qerp}(C_0, C_1; \lambda) \equiv (1 - \lambda)^2 C_0 + \lambda^2 C_1. \quad (5.4)$$

Suppose we desire to do an update in a Kalman filter, optimally combining prediction and measurement by weighting them. One of the equivalent formulations of this step, and particularly convenient for us, is that it is the *minimum variance unbiased linear estimate* (Maybeck 1979). (For Gaussian distributions this linear estimate is optimal.) The updated estimate is then effectively found by minimising the *total error covariance*, defined as the *trace of the error covariance matrix* (in geometric algebra, this is seen as the divergence of the linear mapping which the matrix represents, so we minimise the ‘uncertainty leakage’ in this time step). Since the trace is a linear function, we find that the optimisation involves differentiation of the weighting factors in eq.(5.4). Let us denote the trace of C_i by τ_i , then we find: $\tau_\lambda = \text{qerp}(\tau_0, \tau_1; \lambda)$, and the value of λ that minimises this is

$$\lambda^* = \frac{\tau_0}{\tau_0 + \tau_1}, \quad \text{so} \quad \mathbf{x}^* = \frac{\tau_1 \mathbf{x}_0 + \tau_0 \mathbf{x}_1}{\tau_0 + \tau_1}, \quad (5.5)$$

the well-known classical result for the optimal combination of two translation estimates. We rewrite this into a form that shows how this estimate deviates from the mean, by defining $\tau^* = \tanh(\frac{1}{2} \log(\tau_1/\tau_0))$:

$$\lambda^* = \frac{1}{2} - \frac{1}{2}\tau^*, \quad \text{so} \quad \mathbf{x}^* = \frac{\mathbf{x}_0 + \mathbf{x}_1}{2} + \tau^* \frac{\mathbf{x}_0 - \mathbf{x}_1}{2}. \quad (5.6)$$

Now let us combine two rotations R_0 and R_1 to an intermediate interpolated rotor R_λ , as before, which has the form of the ‘spherical linear interpolation’ $R_\lambda = \text{slerp}(R_0, R_1; \lambda)$. We have established that for small amounts of noise, we can work with covariance matrices. The consequences of this linear combination of R_0 and R_1 for the covariance matrices is easily established, and the covariance for the intermediate rotation R_λ is

$$C_\lambda = \left(\frac{\sin((1 - \lambda)\theta')}{\sin(\theta')} \right)^2 C_0 + \left(\frac{\sin(\lambda\theta')}{\sin(\theta')} \right)^2 C_1 \equiv \text{sqerp}[C_0, C_1; \theta', \lambda] \quad (5.7)$$

(The notation ‘sqerp’ is the ‘spherical quadratic interpolation’, and we need the relative rotor angle θ' as an argument.) We are again interested in the optimal way of combining these distributions, i.e. in a choice of λ which is ‘best’ in a well-defined

sense. And again viewing the Kalman filter update step as the minimum variance unbiased linear estimate helps. For Gaussian distributions this linear estimate is optimal, and as we have seen in section b there are algebraic and statistical reasons for us to focus on Gaussian distributions of small rotation errors.

As in the translation case, the *total error variance* is the *trace of the error covariance matrix*. Denoting the trace of C_i by τ_i , we have that the interpolation for traces is $\tau_\lambda = \text{sqrerp}[\tau_0, \tau_1; \theta', \lambda]$. Differentiating, we find the optimum:

$$\begin{aligned} 0 = \frac{d\tau_\lambda}{d\lambda} &= \frac{d}{d\lambda} \left(\left(\frac{\sin((1-\lambda)\theta')}{\sin(\theta')} \right)^2 \tau_0 + \left(\frac{\sin(\lambda\theta')}{\sin(\theta')} \right)^2 \tau_1 \right) \\ &= \frac{\theta'}{\sin^2 \theta'} \left(-2\tau_0 \sin((1-\lambda)\theta') \cos((1-\lambda)\theta') + 2\tau_1 \sin(\lambda\theta') \cos(\lambda\theta') \right) \\ &= \frac{\theta'}{\sin^2 \theta'} \left(\tau_0 \sin((\lambda-1)\theta) + \tau_1 \sin(\lambda\theta) \right) \\ &= \frac{\theta'}{\sin^2 \theta'} \left(\sin(\lambda\theta)(\tau_0 \cos \theta + \tau_1) - \cos(\lambda\theta)\tau_0 \sin \theta \right) \end{aligned}$$

Therefore the minimum is reached at:

$$\lambda^* = \frac{1}{\theta} \text{atan} \left(\frac{\sin \theta}{\cos \theta + \tau_1/\tau_0} \right) \quad (5.8)$$

Note that in this formula, the actual relative *rotation* angle $\theta = 2\theta'$ occurs, rather than the relative *rotor* angle θ' . We bring this in a more convenient form using $\tau^* = \tanh(\frac{1}{2} \log(T_1/T_0))$:

$$\lambda^* = \frac{1}{2} - \frac{1}{2} \frac{\text{atan}(\tau^* \theta/2)}{\theta/2} \equiv \frac{1}{2} - \frac{1}{2} \rho^* \quad (5.9)$$

defining ρ^* . With this the interpolation formula gives:

$$R_\lambda^* = \frac{\cos(\rho^* \theta/4)}{\cos(\theta/4)} \frac{R_0 + R_1}{2} + \frac{\sin(\rho^* \theta/4)}{\sin(\theta/4)} \frac{R_0 - R_1}{2}. \quad (5.10)$$

Just to verify our basic understanding, the ratio τ_1/τ_0 has the special cases 0 (no noise in R_1), ∞ (no noise in R_0) and 1 (equal noise in both). These give, as they should, $\lambda^* = 1, 0$, and $\frac{1}{2}$, corresponding to the rotors $R_\lambda^* = R_1, R_0$ and $(R_0 + R_1)$ (normalised), respectively. That last case can be generalised to the averaging formula of eq.(3.2), which is now indeed seen to be valid for identically distributed data.

Although different in form, eq.(5.10) is not that different from eq.(5.6) in substance. This is illustrated in the plots of Figure 7. Indeed, when we are using λ^* in a Kalman filter to optimally blend prediction and observation, we can expect that the relative angle θ between them is small. In that case,

$$\lambda^* \approx \frac{1}{1 + \tau_1/\tau_0} = \frac{\tau_0}{\tau_0 + \tau_1}, \quad \text{so } R^* \approx \frac{\tau_1 R_0 + \tau_0 R_1}{\tau_0 + \tau_1}.$$

This coincides in form with the translation case, since the local tangent space to the unit rotor manifold is flat. But unless there is demand for excessive speed, there is no reason to make this linear approximation, the computation of eq.(5.9) and applying the proper interpolation is simple enough.

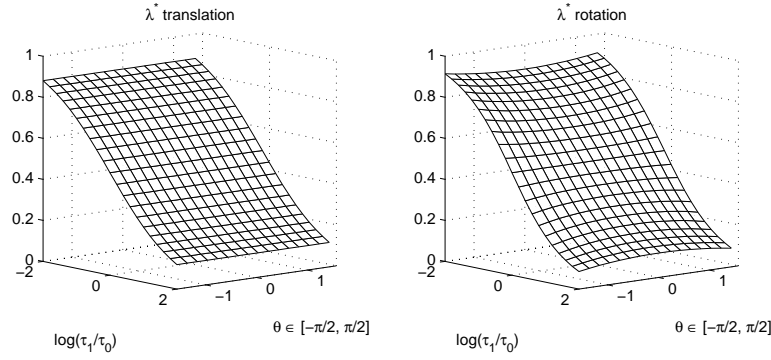


Figure 7. *Minimum total covariance combination of estimates, as a function of the trace ratio τ_1/τ_0 . Left: translation estimates according to eq.(5.6). Right: rotation estimates according to eq.(5.9) (in the small approximation of relatively small noise).*

(d) *Alternative noise characterisation*

In other sources such as (Gamage & Lasenby 2000), the rotor noise is represented by a small additive bivector in the exponent: for $R = e^{\mathbf{B}}$, the noise is modeled as $e^{\mathbf{B}+\mathbf{a}}$, with \mathbf{a} small. We can relate this to our multiplicative rotor noise characterisation of $e^{\mathbf{i}}e^{\mathbf{B}}$. In appendix Appendix A we rederive the result from (McRobie & Lasenby 1999) that (to first order)

$$\mathbf{i} = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp} \text{sinc}(\mathbf{B}) e^{-\mathbf{B}} \equiv \psi_{\mathbf{B}}(\mathbf{a}) \quad (5.11)$$

where \mathbf{a}_{\parallel} and \mathbf{a}_{\perp} are the parts of \mathbf{a} parallel and perpendicular to \mathbf{B} , respectively. This $\psi_{\mathbf{B}}$ is thus a linear function of its argument, parametrised by \mathbf{B} .

Due to the linear relationship between the two characterisations, propagation of the statistics of either characterisation is straightforward: if the multiplicative \mathbf{i} -characterisation has a covariance of \mathbf{C} , then the additive \mathbf{a} -characterisation has a covariance \mathbf{C}' of

$$\mathbf{C}' = \overline{\psi}_{\mathbf{B}} \mathbf{C} \psi_{\mathbf{B}} = \psi_{-\mathbf{B}} \mathbf{C} \psi_{\mathbf{B}}.$$

The Kalman filter update computation of the previous section is now valid for this alternative noise characterisation, if we convert the noise covariances \mathbf{C}'_0 and \mathbf{C}'_1 in the additive bivectors first. This gives

$$\mathbf{C}_0 = \overline{\psi}_0^{-1} \mathbf{C}'_0 \psi_0^{-1}$$

and $\mathbf{C}_1 = \overline{\psi}_1^{-1} \mathbf{C}'_1 \psi_1^{-1}$, where the ψ_i are the conversion functions as in eq.(5.11), different for each rotor. The consequence for the relationship between the traces is

$$\tau_0 = \text{tr} \left(\overline{\psi}_0^{-1} \mathbf{C}'_0 \psi_0^{-1} \right) = \text{tr} \left(\psi_0^{-1} \overline{\psi}_0^{-1} \mathbf{C}'_0 \right) = \text{tr}(\Psi_0 \mathbf{C}'_0) \quad (5.12)$$

with $\Psi_0(\mathbf{a}) \equiv \psi_0^{-1}(\overline{\psi}_0^{-1}(\mathbf{a})) = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}/(\text{sinc}(\mathbf{B}_0))^2$, where \mathbf{a}_{\parallel} and \mathbf{a}_{\perp} are defined relative to \mathbf{B}_0 . The net effect is to increase the contribution of the perpendicular component \mathbf{a}_{\perp} to the trace τ_0 . A similar equation, but involving \mathbf{B}_1 , computes τ_1 . Substituting these in eq.(5.9) gives the optimal λ , which in this additive characterisation therefore involves the absolute orientation bivectors \mathbf{B}_0 and \mathbf{B}_1 of the

interpolants R_0 and R_1 . As a consequence, this noise characterisation is not rotationally invariant. This makes it rather less likely to occur than the multiplicative characterisation, and suggests that the latter is more natural.

(e) *Procrustes orientation estimation*

Applying the procedure of section c of course requires knowledge of the covariance of rotation estimates. For the common Procrustes method, which gives an optimal estimate of the orientation to make two labeled point clouds coincide, we have computed this covariance (Dorst 2003). Combining those results with this paper provides the backbone for a Kalman filter for the Procrustes method. We need this in work on an autonomous vehicle driving through unstructured terrain (van der Mark et al. 2002), the original motivation for this work.

6. Conclusion

We have given a tutorial overview of several known methods of combining orientation estimates based on the convenient rotor/quaternion representation. In the final section, we extended these to a statistically optimal combination of orientation estimates, providing a minimum total covariance estimate, to be used in the update step of a Kalman filter on orientations.

In all these methods, the quaternion/rotor representation permits us to develop the techniques in complete analogy to the classical vector space methods. Yet the different algebra of rotations (notably in its exponential and non-commutative properties) produces subtly different final results. Clear examples of these are the metric eq.(3.1), the ‘linear’ interpolation eq.(4.2), and the optimal (Kalman) combination eq.(5.9).

- C. Doran, J. Lasenby, *Geometric Algebra: New Foundations, New Insights*, In: Course Notes 53: Geometric Algebra, SIGGRAPH 2001.
- L. Dorst, S. Mann, *Geometric algebra: a computational framework for geometrical applications (part I: algebra)*, IEEE Computer Graphics and Applications, vol.22, no.3, May/June 2002, pp.24-31.
- L. Dorst, S. Mann, T. Bouma, *GABLE: a Geometric Algebra Learning Environment*, a Matlab toolkit, 1999, <http://www.wins.uva.nl/~leo/GABLE/>
- L. Dorst, *Error propagation of the Procrustes method for 3-D attitude estimation*, submitted to IEEE TPAMI, 2002.
- D. Fontijne, T. Bouma, L. Dorst, *GAIGEN: a Geometric Algebra Implementation Generator*, C++ software, 2002, <http://carol.wins.uva.nl/~fontijne/gaigen/>
- Gamage, S. and Lasenby, J. Optimal Estimation and Tracking of General Rotations using Geometric Algebra. Proceedings of *Vision Geometry IX, San Diego, July 2000*, SPIE Vol. 4117, 2000.
- J.B. Kuipers, *Quaternions and Rotation Sequences*, Princeton University Press, 1999.
- J. Lasenby, A.N. Lasenby, C.J.L. Doran, *A unified mathematical language for physics and engineering in the 21st century*, Phil. Trans. R. Soc. Lond. A (2000), 358, pp. 21-39.
- J. Lee and S.Y. Shin, *General Construction of Time-Domain Filters for Orientation Data*, IEEE Trans. Vis.and Comp. Graphics, vol.8, no.2, 2002, pp. 119-128.

- W. van der Mark, D. Fontijne, L. Dorst, F.C.A. Groen, *Vehicle Ego-Motion Estimation with Geometric Algebra*, IVS2002, (exact publication data will be supplied)
- S. Mann and L. Dorst, *Geometric algebra: a computational framework for geometrical applications (part II: applications)*, IEEE Computer Graphics and Applications, vol.22 no.4, July/August 2002, pp.58-67.
- P.S. Maybeck, *Stochastic models, estimation and control*, Academic Press, 1979, vol.1.
- F.A. McRobie and J. Lasenby, *Simo-Vu Quoc rods using Clifford Algebra*, Int. J. Num. Methods. Eng, 45,377-398, 1999.
- J. Richter-Gebert, U.H. Kortenkamp, *Cinderella 1.2*, 2000, <http://www.cinderella.de>
- K. Shoemake, *Animating Rotations with Quaternion Curves*, ACM vol. 19, nr. 3, 1985.

Appendix A. Representation of rotor/quaternion noise

In this paper, we have treated the noise as a small rotor applied *after* the main rotation, so that we can write:

$$rR = e^{\mathbf{b}}e^{\mathbf{B}} \approx e^{\mathbf{B}} + \mathbf{b}e^{\mathbf{B}}, \quad (\text{A } 1)$$

where \mathbf{B} and \mathbf{b} are the large and small bivectors characterising these rotors. This is not different in principle from doing the noisy rotor *before* the main rotor. Since $Rr \approx e^{\mathbf{B}} + e^{\mathbf{B}}\mathbf{b} = e^{\mathbf{B}} + (e^{\mathbf{B}}\mathbf{b}e^{-\mathbf{B}})e^{\mathbf{B}}$, that merely differs by a simple linear mapping of the infinitesimal rotor argument, and we know how to deal with the covariance of such noise.

In (Gamage & Lasenby 2000), the authors prefer to model the orientation noise through a small additive term in the 2-blade of the rotor, as $e^{\mathbf{B}+\mathbf{a}}$. This is really different, though we can establish the relationship between this 2-blade noise \mathbf{a} and the rotor noise \mathbf{b} used above. (This is also done in (McRobie & Lasenby 1999) using multivector differentiation.) In the following, assume that the rotor argument \mathbf{B} is a pure 2-blade (which squares to a scalar) and invertible, and that all terms of order 2 and higher in \mathbf{a} (and \mathbf{b}) can be neglected.

$$\begin{aligned} e^{\mathbf{B}+\mathbf{a}} &\approx 1 \\ &+ \frac{1}{1!} (\mathbf{B} + \mathbf{a}) \\ &+ \frac{1}{2!} (\mathbf{B}^2 + \mathbf{B}\mathbf{a} + \mathbf{a}\mathbf{B}) \\ &+ \frac{1}{3!} (\mathbf{B}^3 + 2\mathbf{B}^2\mathbf{a} + \mathbf{B}\mathbf{a}\mathbf{B}) \\ &+ \frac{1}{4!} (\mathbf{B}^4 + 2\mathbf{B}^3\mathbf{a} + 2\mathbf{B}^2\mathbf{a}\mathbf{B}) \\ &+ \dots \\ = 1 + & \frac{1}{2} \mathbf{B}^{-1} (\mathbf{B}^0\mathbf{a} - \mathbf{a}\mathbf{B}^0) \\ &+ \frac{1}{1!} (\mathbf{B}^1 + \frac{1}{2}(\mathbf{B}^0\mathbf{a} + \mathbf{B}^0\mathbf{B}^{-1}\mathbf{a}\mathbf{B})) + \frac{1}{2} \mathbf{B}^{-1} (\mathbf{B}^1\mathbf{a} - \mathbf{a}\mathbf{B}^1) \\ &+ \frac{1}{2!} (\mathbf{B}^2 + \frac{2}{2}(\mathbf{B}^1\mathbf{a} + \mathbf{B}^1\mathbf{B}^{-1}\mathbf{a}\mathbf{B})) + \frac{1}{2} \mathbf{B}^{-1} (\mathbf{B}^2\mathbf{a} - \mathbf{a}\mathbf{B}^2) \\ &+ \frac{1}{3!} (\mathbf{B}^3 + \frac{3}{2}(\mathbf{B}^2\mathbf{a} + \mathbf{B}^2\mathbf{B}^{-1}\mathbf{a}\mathbf{B})) + \frac{1}{2} \mathbf{B}^{-1} (\mathbf{B}^3\mathbf{a} - \mathbf{a}\mathbf{B}^3) \\ &+ \frac{1}{4!} (\mathbf{B}^4 + \frac{4}{2}(\mathbf{B}^3\mathbf{a} + \mathbf{B}^3\mathbf{B}^{-1}\mathbf{a}\mathbf{B})) + \frac{1}{2} \mathbf{B}^{-1} (\mathbf{B}^4\mathbf{a} - \mathbf{a}\mathbf{B}^4) \\ &+ \dots \\ = e^{\mathbf{B}} + \frac{1}{2} & \left(e^{\mathbf{B}}\mathbf{a} + e^{\mathbf{B}}\mathbf{B}^{-1}\mathbf{a}\mathbf{B} + \mathbf{B}^{-1}(e^{\mathbf{B}}\mathbf{a} - \mathbf{a}e^{\mathbf{B}}) \right) \end{aligned} \quad (\text{A } 2)$$

Equating the two expressions eq.(A 1) and eq.(A 2) involving \mathbf{a} and \mathbf{b} gives

$$\mathbf{b} = \frac{1}{2}e^{\mathbf{B}}(\mathbf{a} + \mathbf{B}^{-1}\mathbf{a}\mathbf{B})e^{-\mathbf{B}} + \frac{1}{2}\mathbf{B}^{-1}(e^{\mathbf{B}}\mathbf{a}e^{-\mathbf{B}} - \mathbf{a}) \quad (\text{A } 3)$$

In 3D, the formula cleans up by defining the parts of \mathbf{a} which are in the \mathbf{B} -plane, and perpendicular to it:

$$\mathbf{a}_{\parallel} = \frac{1}{2}\mathbf{B}^{-1}(\mathbf{B}\mathbf{a} + \mathbf{a}\mathbf{B}) \quad \mathbf{a}_{\perp} = \frac{1}{2}\mathbf{B}^{-1}(\mathbf{B}\mathbf{a} - \mathbf{a}\mathbf{B})$$

Then

$$\mathbf{b} = e^{\mathbf{B}}\mathbf{a}_{\parallel}e^{-\mathbf{B}} - \frac{1}{2}\mathbf{a}_{\perp}\mathbf{B}^{-1}(e^{-2\mathbf{B}} - 1) = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}\frac{e^{\mathbf{B}} - e^{-\mathbf{B}}}{2\mathbf{B}}e^{-\mathbf{B}} = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}\text{sinc}(\mathbf{B})e^{-\mathbf{B}},$$

where we introduced the sinc function of bivector arguments by complete analogy to the definition in real analysis (it is easy to show that it equals $\sin(|\mathbf{B}|)/|\mathbf{B}|$). The inverse mapping is:

$$\mathbf{a} = \mathbf{b}_{\parallel} + \mathbf{b}_{\perp}\frac{e^{\mathbf{B}}}{\text{sinc}(\mathbf{B})}.$$

Near $|\mathbf{B}| = \pi$, this grows unbounded and the assumption that the bivector \mathbf{a} is small is no longer valid. This suggests that the ‘post-rotor’ characterisation is better behaved than the ‘additive bivector’ characterisation. However, a rotor angle of π implies a rotation angle of 2π , so in practice usage of the equations in this region is presumably avoided by a proper disambiguation of the rotor representation.

Denoting the relationship between \mathbf{a} and \mathbf{b} above by the parametrised linear mapping $\mathbf{b} = \psi_{\mathbf{B}}(\mathbf{a})$, we also need its adjoint $\overline{\psi}_{\mathbf{B}}$ relative to the scalar product when propagating the covariances. This is simply:

$$\overline{\psi}_{\mathbf{B}} = \psi_{-\mathbf{B}}$$

Derivation: for arbitrary bivectors \mathbf{a} and \mathbf{c} , we have (defining $X * Y \equiv \langle XY \rangle_0$)

$$\begin{aligned} \mathbf{c} * \psi_{\mathbf{B}}(\mathbf{a}) &= \mathbf{c} * (\mathbf{a}_{\parallel} + \mathbf{a}_{\perp}e^{-\mathbf{B}}\text{sinc}(\mathbf{B})) \\ &= \mathbf{c} * \mathbf{a}_{\parallel} + \mathbf{c} * (\mathbf{a}_{\perp}e^{-\mathbf{B}})\text{sinc}(\mathbf{B}) \\ &= \mathbf{a}_{\parallel} * \mathbf{c} + \mathbf{a}_{\perp} * (e^{-\mathbf{B}}\mathbf{c})\text{sinc}(\mathbf{B}) \quad (\text{by cyclic re-ordering of scalar product}) \\ &= \mathbf{a} * \mathbf{c}_{\parallel} + \mathbf{a} * (e^{-\mathbf{B}}\mathbf{c}_{\perp})\text{sinc}(\mathbf{B}) \quad (\text{apply cyclic re-ordering again}) \\ &= \mathbf{a} * (\mathbf{c}_{\parallel} + \mathbf{c}_{\perp}e^{\mathbf{B}}\text{sinc}(-\mathbf{B})) \quad (\text{anti-commutation, and symmetry of sinc}) \\ &= \mathbf{a} * \psi_{-\mathbf{B}}(\mathbf{c}) \end{aligned}$$

As a consequence, the mapping $\overline{\psi}_{\mathbf{B}}^{-1}\psi_{\mathbf{B}}^{-1}$, required to compute the covariance transformation eq.(5.12) is:

$$\overline{\psi}_{\mathbf{B}}^{-1}\psi_{\mathbf{B}}^{-1}(\mathbf{a}) = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}\frac{e^{-\mathbf{B}}}{\text{sinc}(-\mathbf{B})}\frac{e^{\mathbf{B}}}{\text{sinc}(\mathbf{B})} = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}\frac{1}{\text{sinc}^2(\mathbf{B})}$$