Tutorial: Structure-Preserving Representation of Euclidean Motions Through Conformal Geometric Algebra

Leo Dorst

Abstract A new and useful set of homogeneous coordinates has been discovered for the treatment of Euclidean geometry. They render Euclidean motions not merely linear (as the classical homogeneous coordinates do), but even turn them into orthogonal transformations, through a clever choice of metric in two (not one) additional dimensions.

To take full advantage of this new possibility, a good representation of orthogonal transformations is required. We find that multiple reflections, while classically giving unwieldy expressions involving the dot product, become practical by introducing the more fundamental geometric product (which has the dot product merely as its symmetric part). We obtain a sandwiching operation between products of vectors as our representation of motions, which is not only easily concatenated, but also incorporates the computational advantages of complex numbers and quaternions in a real manner. The antisymmetric part of the geometric product produces a spanning operation that permits the construction of lines, planes, spheres and tangents from vectors. Since the sandwiching operation distributes over this construction, 'objects' are fully integrated with 'motions', in a structure preserving manner.

Additional techniques such as duality (permitting a universal intersection operation), and the rewriting of operators logarithmically (to obtain quantities that can be interpolated linearly) complete the techniques of what is ultimately a very convenient geometric algebra. It easily incorporates general conformal transformations, and can be implemented to run almost as efficiently as classical homogeneous coordinates. The resulting high-level programming language naturally integrates quantitative computation with the automatic administration of geometric data structures.

Rather than the usual introductions which plow through Clifford algebra before they reach this very useful 'conformal model' (if they do at all), this tutorial does the reverse. It structures the new concepts in a manner that shows how each ad-

L. Dorst (✉)
Intelligent Systems Laboratory, University of Amsterdam, Science Park 107, 1098 XG, Amsterdam, The Netherlands
e-mail: L.Dorst@uva.nl

E. Bayro-Corrochano, G. Scheuermann (eds.), *Geometric Algebra Computing*,
DOI 10.1007/978-1-84996-108-0_2,

ditional sophistication is related to what went before, and how it extends its expressive and computational power. Throughout, the concepts and techniques will be illustrated by interactive visualization software (GAviewer, freely available at www.geometricalgebra.net).

1 Introduction

"Doing geometry" in computer science or engineering requires at least the following ingredients in a practical computational framework:

- *descriptive primitives*: such as points, lines, planes, circles, spheres, tangents
- *basic constructions*: connections, intersections, parametric specification
- *motions*: translation, rotation, reflection, projection
- *properties*: size, location, orientation
- *practical numerics*: approximation, estimation, interpolation, linearization

These ingredients should interweave seamlessly. Notably, the framework should be structure preserving, in the sense that *constructions and properties of primitives should be covariant under motions.* For instance, when moving a circle determined by three points, it should not be necessary to decompose the circle back into the points, move those, and then reconstruct; rather, the circle should be a basic element of computation with an associated motion operator (which should moreover preferably be identical to that for points). Also, all ingredients should be specifiable in a sufficiently high-level programming language, which avoids coordinates as specification level though it may revert to them when executing the operations. The usual linear algebra tools have neither of these desirable properties, not even when using homogeneous coordinates. Yet a practical computational framework exists that can do all of the above. It is called "conformal geometric algebra" (CGA), and this chapter briefly exposes its essential structure. We will explain all elements of Fig. 1, and more.

2 Conformal Geometric Algebra

2.1 Trick 1: Representing Euclidean Points in Minkowski Space

Let us focus on a 3D space in which we want to perform Euclidean motions. We can consider it as a 3D vector space and use a position vector $\mathbf{x}$ to denote a point X relative to an (arbitrary) origin. This is naive practice, and not very convenient, since Euclidean motions are then not even linear transformations. A commonly used improvement is the *homogeneous model*, in which the space is augmented with an extra dimension e_o, and the point X at $\mathbf{x}$ represented as $e_o + \mathbf{x}$. Now Euclidean motions are linear transformations but still not structure preserving. More is required.

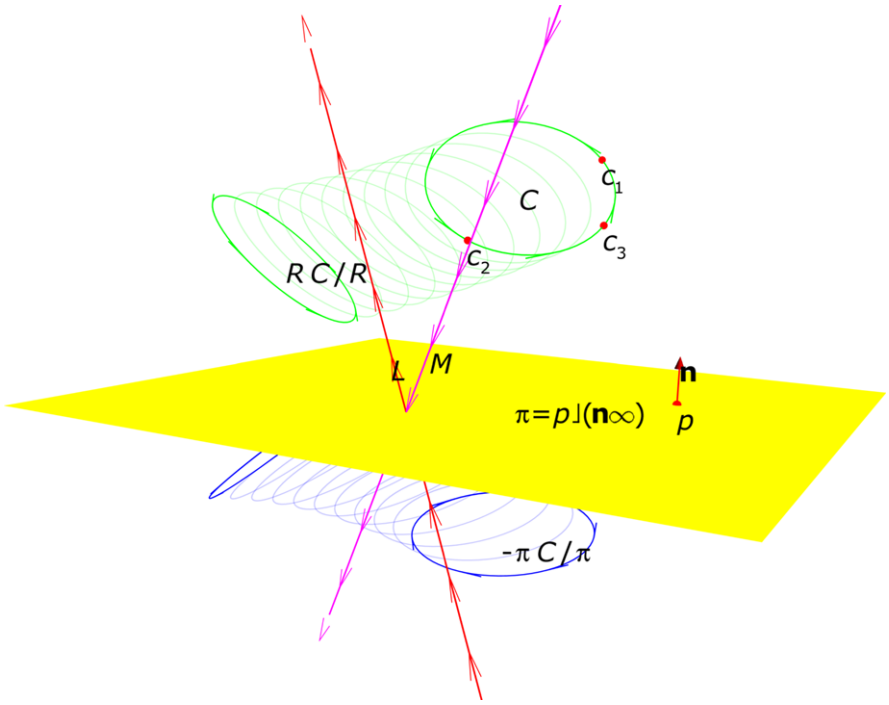


Fig. 1 An example of the ease of CGA (from [2]). A circle C is generated from three points c_1, c_2, c_3 as $C = c_1 \wedge c_2 \wedge c_3$. A line is given as a 3-blade L. The circle C is to be rotated around the line L, producing RCR^{-1}, with R specified as $R = \exp(L^*\phi/2)$. The rotation is interpolated in k steps using $R^{1/k}$. Then the whole scene is reflected in the plane π given by a normal vector $\mathbf{n}$ and a point p on it as $\pi = p \cdot (\mathbf{n} \wedge e_\infty)$; any element X is reflected as $X \mapsto (-1)^{\text{grade}(X)} \pi X \pi^{-1}$. In appropriate software such as [3], these coordinate-free formulas are the literal specification of a computer program producing the scene

In CGA, we introduce *two extra dimensions* for representational purposes, thus constructing a five-dimensional space. We introduce two basis vectors for these extra dimensions, e_o and e_∞, and the specific metric given below. As we will see, the null vectors in this extended space (i.e., the vectors x satisfying $x \cdot x = 0$ in the chosen metric) represent weighted points in the Euclidean space (though one usually employs unit weight points satisfying $x \cdot e_\infty = 0$). Such vectors representing points have algebraic properties to construct other elements in a coordinate-free, invariant manner, as explained in the paper by Hestenes [5] elsewhere in this volume.

In the present introductory paper, we mostly prefer to use an explicit expression for such a vector x representing a point X, relating it to the "classical" Euclidean position vector $\mathbf{x}$ of the point relative to the chosen origin through

$$x = e_o + \mathbf{x} + \frac{1}{2}\|\mathbf{x}\|^2 e_\infty. \tag{1}$$

Table 1

·	e_o	$\mathbf{x}$	e_∞
e_o	0	0	-1
$\mathbf{x}$	0	$\mathbf{x}\cdot\mathbf{x}$	0
e_∞	-1	0	0

If we ignore the component for e_∞, we recognize in $e_o + \mathbf{x}$ just the homogeneous model. In that model, the extra dimension e_o represents *the point at the origin*; and the same interpretation holds in CGA (set $\mathbf{x} = \mathbf{0}$). We see that the term with e_∞ dominates as $\mathbf{x}$ gets large. In fact, e_∞ can be interpreted consistently as *the point at infinity* which is used in mathematics to "compactify" Euclidean space to remove special cases from its algebra.

The Big Trick of CGA is the choice of a specific metric for the 5D representational space. We extend the dot product $\mathbf{x}\cdot\mathbf{x}$ for Euclidean vectors to the new dimensions according to the multiplication table (Table 1), where the bold elements are purely Euclidean and borrow the 3D Euclidean dot product. This table shows that the usual Euclidean metric holds for the bold vectors, but a strange metric applies to the two additional dimensions e_o and e_∞, which are moreover "orthogonal" to the Euclidean part of the representational space since they have dot product zero with Euclidean vectors. (In fact, the full 5D space now has a Minkowski metric, as can be seen by considering the alternative basis vectors $\sigma_+ = e_o - e_\infty/2$ and $\sigma_- = e_o + e_\infty/2$ that have squared norms of $+1$ and -1, respectively. For more on this basis, see [5].)

This metric is introduced to give a sensible real world meaning to the dot product of two point representatives x and y:

$$\begin{aligned} x\cdot y &= \left(e_o + \mathbf{x} + \frac{1}{2}\|\mathbf{x}\|^2 e_\infty\right)\cdot\left(e_o + \mathbf{y} + \frac{1}{2}\|\mathbf{y}\|^2 e_\infty\right) \\ &= \left(0 + 0 - \frac{1}{2}\|\mathbf{y}\|^2\right) + (0 + \mathbf{x}\cdot\mathbf{y} + 0) + \left(-\frac{1}{2}\|\mathbf{x}\|^2 + 0 + 0\right) \\ &= -\frac{1}{2}(\mathbf{x}-\mathbf{y})\cdot(\mathbf{x}-\mathbf{y}) = -\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2. \end{aligned} \tag{2}$$

The dot product in conformal space therefore encodes the (squared) Euclidean distance of the original points! Since points have distance zero to themselves, they are represented by null vectors; and since Euclidean transformations should preserve the inter-point distance, they should preserve the dot product.

Euclidean transformations are represented as orthogonal transformations

in CGA.[1] This is more specific than their representation as a certain strange class of linear transformations in the usual homogeneous model, and it permits us to design

[1] We have simplified slightly; the general representation of a point at $\mathbf{x}$ in CGA is a scalar multiple of x in (1); the scalar factor is the scalar $-e_\infty\cdot x$ (as you may verify), and this can be consistently

a more effective computational framework tailored to this property. Matrices are actually not that great for representing orthogonal transformations, but fortunately there is something better, as we will see in the next section.

First, let us determine what the vectors in the 5D representation space signify geometrically. Suppose that we want to represent a sphere with center C and radius ρ in Euclidean space. A point X on such a sphere would satisfy $\|\mathbf{x} - \mathbf{c}\|^2 = \rho^2$ (using Euclidean vectors). Using (2), this can be written in terms of the dot product of the representative vectors x and c as $x \cdot c = -\frac{1}{2}\rho^2$; and using $-e_\infty \cdot x = 1$, we can even group into $x \cdot (c - \frac{1}{2}\rho^2 e_\infty) = 0$. The vector $\sigma = \alpha(c - \frac{1}{2}\rho^2 e_\infty)$ is the most general vector we can make in the conformal space (it has five parameters), and written in this form we recognize it as representing a sphere with center c, radius ρ, and "weight" α through the equation $x \cdot \sigma = 0$. You may verify that $\|\sigma\|^2 = \alpha^2\rho^2$ (even "imaginary spheres" with $\rho^2 < 0$ are included) and that a point is just a sphere with radius zero, represented by a null vector (for which $\|x\|^2 = 0$). A plane is the degenerate case of a sphere, and it is represented by a vector of the form $\pi = \alpha(\mathbf{n} + \delta e_\infty)$ (which has no e_o-component and therefore satisfies $\pi \cdot e_\infty = 0$). Here $\mathbf{n}$ is the unit normal vector of the plane, δ is its oriented distance from the origin, and α a weight. So:

the vectors in conformal space represent weighted spheres and planes.

In this tutorial, we will mostly use unit weights, focusing on the merely geometrical aspects of the representation. In our *notation*, we will use bold for the elements of the conformal model that are in its n-D Euclidean subspace, and nonbold for elements residing in the full $(n + 2)$-D representational space or its geometric algebra. Since there is a clear correspondence between elements of Euclidean geometry and their conformal representation, we will drop the distinction between X and x, and talk about a point x at location $\mathbf{x}$.

2.2 Trick 2: Orthogonal Transformations as Multiple Reflections in a Sandwiching Representation

In mathematics, the Cartan–Dieudonné theorem states that all orthogonal transformations can be represented as multiple reflections. In linear algebra, this fact is not used much, since reflections are represented awkwardly and therefore unsuitable as atoms of representation. If we want to reflect a Euclidean vector $\mathbf{x}$ in a plane through the origin with normal vector $\mathbf{a}$, this is the linear transformation

$$\mathbf{x} \mapsto \mathbf{x} - 2(\mathbf{x} \cdot \mathbf{a})\mathbf{a}/(\mathbf{a} \cdot \mathbf{a}). \tag{3}$$

interpreted as the *weight* of the point. The squared distance between weighted points is computed by normalizing first as $(x/(-e_\infty \cdot x)) \cdot (y/(-e_\infty \cdot y))$. Euclidean transformations should then not affect this formula; this implies that they are the specific orthogonal transformations that preserve the special vector e_∞.

It does not look elementary at all, and within linear algebra the dot products cannot be simplified.

We now introduce a clever trick: we consider the dot product (which is symmetric) as merely the symmetrical part of a more fundamental product between vectors. That product (invented by Clifford in 1872) is called the *geometric product* and denoted by a space. So we rewrite:

$$\mathbf{a} \cdot \mathbf{x} = \frac{1}{2}(\mathbf{a}\mathbf{x} + \mathbf{x}\mathbf{a}). \tag{4}$$

This more fundamental product is defined to be bilinear and associative but not necessarily commutative. We see that $\|\mathbf{x}\|^2 = \mathbf{x} \cdot \mathbf{x} = \mathbf{x}\mathbf{x} = \mathbf{x}^2$, so that the square of a vector under the geometric product is a scalar. We extend the geometric product to scalars (and later to other elements). Scalars commute under the geometric product, so $\alpha x = x\alpha$ for vector x and scalar α. A vector $\mathbf{x}$ has a unique inverse $\mathbf{x}^{-1}$ under the geometric product, defined through $\mathbf{x}\mathbf{x}^{-1} = 1 = \mathbf{x}^{-1}\mathbf{x}$ and therefore found explicitly as

$$\textit{inverse of a vector}: \quad \mathbf{x}^{-1} = \mathbf{x}/\big(\mathbf{x}^2\big).$$

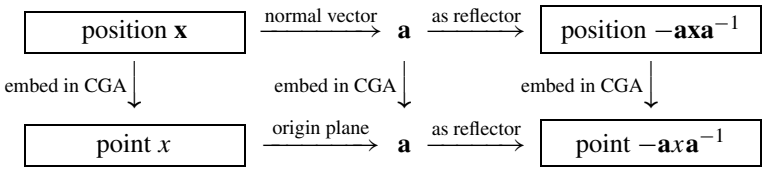
Now we see how this simplifies the reflection representation:

$$\begin{aligned} \textit{reflection in origin hyperplane with normal } \mathbf{a}: \quad \mathbf{x} \mapsto\ & \mathbf{x} - 2(\mathbf{x} \cdot \mathbf{a})\mathbf{a}/(\mathbf{a} \cdot \mathbf{a}) \\ =\ & \mathbf{x} - (\mathbf{x}\mathbf{a} + \mathbf{a}\mathbf{x})\mathbf{a}^{-1} \\ =\ & -\mathbf{a}\mathbf{x}\mathbf{a}^{-1}. \end{aligned} \tag{5}$$

The reflection of $\mathbf{x}$ in the origin hyperplane with normal vector $\mathbf{a}$ is therefore simply a "sandwiching" of $\mathbf{x}$ by $\mathbf{a}$ and $\mathbf{a}^{-1}$ (with a minus sign). In this form, the fundamental nature of reflections for the representation of transformations is more obvious.

You may rightly object that we have not really reflected a point x, but only its Euclidean part $\mathbf{x}$. Let us try to extend the formula to the point x, using the explicit representation (1). Postulating distributivity of the geometric product, we get $-\mathbf{a}x\mathbf{a}^{-1} = -\mathbf{a}(e_o + \mathbf{x} + \frac{1}{2}\|\mathbf{x}\|^2 e_\infty)\mathbf{a}^{-1} = -\mathbf{a}e_o\mathbf{a}^{-1} - \mathbf{a}\mathbf{x}\mathbf{a}^{-1} - \frac{1}{2}\|\mathbf{x}\|^2\mathbf{a}e_\infty\mathbf{a}^{-1}$. Evaluating this requires computing what $-\mathbf{a}e_o\mathbf{a}^{-1}$ and $-\mathbf{a}e_\infty\mathbf{a}^{-1}$ are. We realize from definition (4) and the dot product table that $-\mathbf{a}e_o\mathbf{a}^{-1} = -(\mathbf{a}e_o)\mathbf{a}^{-1} = -(2\mathbf{a} \cdot e_o - e_o\mathbf{a})\mathbf{a}^{-1} = 0 + e_o\mathbf{a}\mathbf{a}^{-1} = e_o$. Of course, you would expect this geometrically: the point at the origin does not change after the reflection. Similarly for e_∞, as you may verify. Further realize that $\| - \mathbf{a}\mathbf{x}\mathbf{a}^{-1}\|^2 = (-\mathbf{a}\mathbf{x}\mathbf{a}^{-1})(-\mathbf{a}\mathbf{x}\mathbf{a}^{-1}) = \mathbf{a}\mathbf{x}\mathbf{x}\mathbf{a}^{-1} = \mathbf{x}^2(\mathbf{a}\mathbf{a}^{-1}) = \|\mathbf{x}^2\|$—obviously, since reflection is an orthogonal transformation. Combining all this, we find $-\mathbf{a}x\mathbf{a}^{-1} = e_o - \mathbf{a}\mathbf{x}\mathbf{a}^{-1} + \frac{1}{2}\| - \mathbf{a}\mathbf{x}\mathbf{a}^{-1}\|^2 e_\infty$, which is precisely the representation of a point at the reflected location. Therefore a point x is reflected by transfer of the Euclidean formula (3), as $x \mapsto -\mathbf{a}x\mathbf{a}^{-1}$. This

structural principle may be illustrated as the commutative diagram



If we perform a second reflection in another origin hyperplane, with normal vector **b**, this should be the mapping

$$x \mapsto = -\mathbf{b}\left(-\mathbf{a}x\mathbf{a}^{-1}\right)\mathbf{b}^{-1} = (\mathbf{ba})x(\mathbf{ba})^{-1},$$

using the associativity of the geometric product in the rewriting. Geometrically, a double reflection is a rotation (see Fig. 2), so the operator (**ba**) represents a rotation operator (in an axis through the origin, determined as the intersection of the planes **a** and **b**). In this manner, we can generate all orthogonal transformations as sandwiching products by elements that are themselves the geometric product of vectors. These elements are called *versors*. A delightful property of versors is that they do not only apply to vectors, but also directly to other geometric elements like lines and circles. Let us first make those geometric elements part of our algebra.

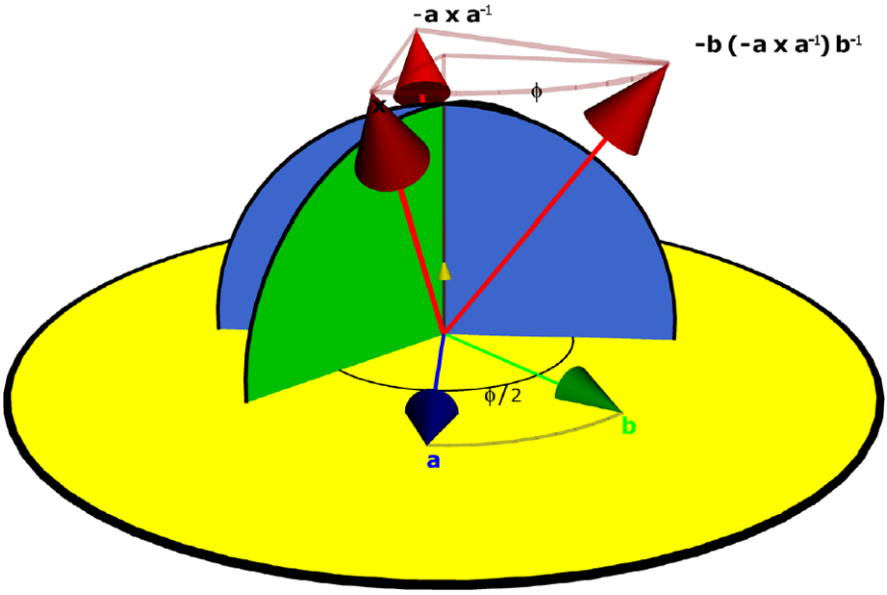


Fig. 2 A reflection in two successive planes is equivalent to a rotation over double their separating angle, around the line of their intersection (in 3D)

2.3 Trick 3: Constructing Elements by Anti-Symmetry

When we introduced the geometric product for vectors, we used only its symmetric part (that was the dot product). But of course there is an anti-symmetric part as well. Let us denote that by $\wedge$ and call it the *outer product*. For vectors, it is defined as

$$\mathbf{x} \wedge \mathbf{a} = \frac{1}{2}(\mathbf{xa} - \mathbf{ax}).$$

It is clear that $\mathbf{x} \wedge \mathbf{a} = -\mathbf{a} \wedge \mathbf{x}$, so that $\mathbf{x} \wedge \mathbf{x} = 0$.

To interpret this new element $\mathbf{x} \wedge \mathbf{a}$ geometrically, let us use some classical linear algebra and take $\mathbf{x}$ and $\mathbf{a}$ as direction vectors. If we take an orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2\}$ in the plane spanned by $\mathbf{x}$ and $\mathbf{a}$, and choose it such that $\mathbf{x} = \|\mathbf{x}\|\mathbf{e}_1$, then $\mathbf{a}$ can be written as $\mathbf{a} = \|\mathbf{a}\|(\cos(\phi)\mathbf{e}_1 + \sin(\phi)\mathbf{e}_2)$ with ϕ the angle from $\mathbf{x}$ to $\mathbf{a}$. We evaluate:

$$\begin{aligned}\mathbf{x} \wedge \mathbf{a} &= \big(\|\mathbf{x}\|\mathbf{e}_1\big) \wedge \big(\|\mathbf{a}\|\big(\cos(\phi)\mathbf{e}_1 + \sin(\phi)\mathbf{e}_2\big)\big) \\ &= \|\mathbf{x}\|\|\mathbf{a}\|\big(\cos(\phi)\mathbf{e}_1 \wedge \mathbf{e}_1 + \sin(\phi)\mathbf{e}_1 \wedge \mathbf{e}_2\big) \\ &= \|\mathbf{x}\|\|\mathbf{a}\|\sin(\phi)\mathbf{e}_1 \wedge \mathbf{e}_2,\end{aligned}$$

for being the sum of two bilinear products, the outer product is itself bilinear. We recognize in $\|\mathbf{x}\|\|\mathbf{a}\|\sin(\phi)$ the signed area of the oriented parallelogram spanned by $\mathbf{x}$ and $\mathbf{a}$ (in that order) and can therefore interpret $\mathbf{e}_1 \wedge \mathbf{e}_2$ as the algebraic specification of the unit area element in the $(\mathbf{e}_1, \mathbf{e}_2)$-plane. We call this a *unit 2-blade*.We then interpret the 2-blade $\mathbf{x} \wedge \mathbf{a}$ of direction vectors as the full specification of the geometric area element spanned by $\mathbf{x}$ and $\mathbf{a}$ (in that order) in terms of its magnitude, orientation, and geometrical attitude (i.e., spatial stance). Only the shape is not determined, for you can easily verify that, for instance, $\mathbf{x} \wedge (\mathbf{a} + \lambda\mathbf{x}) = \mathbf{x} \wedge \mathbf{a}$ so that $\mathbf{x}$ and $\mathbf{a} + \lambda\mathbf{x}$ span the same element as $\mathbf{x}$ and $\mathbf{a}$. For parallel direction vectors $\mathbf{x}$ and $\mathbf{a}$, the outer product $\mathbf{x} \wedge \mathbf{a}$ is zero, so the commutativity relationship $\mathbf{xa} = \mathbf{ax}$ is the algebraic way of expressing parallelness of vectors. Orthogonality of vectors is expressed as $\mathbf{xa} = -\mathbf{ax}$, or $\mathbf{x} \cdot \mathbf{a} = 0$.

The outer product can be extended over more vector terms, always as the anti-symmetric sum. This is done by permuting the geometric products and endowing even permutations with a plus and odd permutations with a minus. For instance:

$$\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} = \frac{1}{3!}(\mathbf{abc} - \mathbf{bac} + \mathbf{bca} - \mathbf{cba} + \mathbf{cab} - \mathbf{acb})$$

(but this algebraic equation is a very inefficient way of computing the value of the outer product; the equivalent $\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} = \frac{1}{2}(\mathbf{abc} - \mathbf{cba})$ is already better). It can be shown that the outer product thus defined is associative and multilinear. To make it fully defined over all elements, we can extend it to scalars simply by defining $\alpha \wedge \mathbf{a} = \alpha\mathbf{a}$ for scalar α and vector $\mathbf{a}$.

The outer product of k vector factors is called a *k-blade*, and the number of vector factors k is called its *grade*. Geometrically, a k-blade is a quantitative representation of a weighted, oriented k-dimensional subspace of the space its vectors reside in,

and its signed magnitude is an oriented hypervolume. For instance, if you would compute the outer product of three direction vectors in 3D space, you would find that the coordinates of the vectors combine to a familiar signed scalar multiple of the unit volume: $\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} = \det([\![\mathbf{a}\ \mathbf{b}\ \mathbf{c}]\!])\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3$. This volume is zero when the vectors are co-planar, and therefore $\mathbf{x} \wedge (\mathbf{a} \wedge \mathbf{b}) = 0$ can be solved for $\mathbf{x}$ as $\mathbf{x} = \lambda\mathbf{a} + \mu\mathbf{b}$. Again, the 2-blade $\mathbf{a} \wedge \mathbf{b}$ is seen to be a single computational element representing the plane spanned by the direction vectors $\mathbf{a}$ and $\mathbf{b}$.

In the conformal model, the outer product of vectors representing points a and b takes on a different geometric interpretation, even though its algebra is the same. In CGA, the blade $a \wedge b$ *represents an oriented point-pair*, in the sense that the set of points x satisfying $x \wedge a \wedge b = 0$ is either $x = a$ or $x = b$. (Comparing to the derivation just given, we do get $x = \lambda a + \mu b$, as before, but to be a point in CGA, x has to satisfy $x \cdot x = 0$ by (2), as do a and b. Some algebra then leads to $\lambda\mu(a \cdot b) = 0$, and this implies $\lambda = 0$ and/or $\mu = 0$.) Similarly, $a \wedge b \wedge c$ *represents the oriented circle through the points a, b, and c*, and the outer product of four points $a \wedge b \wedge c \wedge d$ represents an oriented *sphere*. We call these elements *rounds*. If the points are in degenerate positions, or if one of them is the point at infinity e_∞, an oriented *flat* results (in 3D, these are: a line $a \wedge b \wedge e_\infty$, a plane $a \wedge b \wedge c \wedge e_\infty$, or a "flat point" $a \wedge e_\infty$). Showing these facts without too much computation requires the technique of dual representation, introduced next.

2.4 Trick 4: Dual Specification of Elements Permits Intersection

A subspace can be characterized by the outer product, but it is often convenient to take a "dual" approach, not specifying the vectors in it but the vectors orthogonal to it. We have already seen this for spheres: the orthogonality demand $x \cdot (c - \frac{1}{2}\rho^2 e_\infty) = 0$ solves for x lying on a sphere with center c and radius ρ. Duality is a fundamental concept of geometric algebra and requires no more than complementation relative to the volume of the vector space, through division.

An n-dimensional vector space cannot have nonzero blades of a grade exceeding n. A nonzero blade of the maximum grade n is called a *pseudoscalar* for the space. It is common to normalize this to a unit pseudoscalar and to denote it by $\mathbf{I}_n$ or I_n. The choice of the sign of the unit pseudoscalar amounts to choosing a reference orientation for the space. In a 3D Euclidean space of direction vectors with an orthonormal basis, $\mathbf{I}_3 = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 (= \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3)$ picks the standard "right-handed" orientation. In the conformal model space, a suitable pseudoscalar is $I_{4,1} = e_o \wedge \mathbf{I}_3 \wedge e_\infty$. The inverse of the unit pseudoscalar in 3D Euclidean space is $\mathbf{I}_3^{-1} = -\mathbf{I}_3$ (verify that $\mathbf{I}_3\mathbf{I}_3^{-1} = 1$!). In the conformal space, $I_{4,1}^{-1} = e_o \wedge \mathbf{I}_3^{-1} \wedge e_\infty = -I_{4,1}$.

One can find the blade representing the orthogonal complement of any subspace through right-dividing its blade A by the pseudoscalar, as AI_n^{-1}. This is called the *dual* of A and denoted A^*:

$$\textit{dualization}: \quad A^* = AI_n^{-1}. \tag{6}$$

For instance, the dual of the 2-blade $\mathbf{e}_1 \wedge \mathbf{e}_2$ in 3D-space is $(\mathbf{e}_1 \wedge \mathbf{e}_2)(\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3)^{-1} = -(\mathbf{e}_1\mathbf{e}_2)(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3) = \mathbf{e}_3$. This is indeed the normal vector of the $(\mathbf{e}_1, \mathbf{e}_2)$-plane using the right-hand rule. The familiar 3D cross product of vectors can be made in CGA as $\mathbf{x} \times \mathbf{a} = (\mathbf{x} \wedge \mathbf{a})\mathbf{I}_3^{-1}$, though its use should be avoided.

Duality permits us to intersect subspaces. Let us denote the *intersection* (or meet) of blades A and B as $A \cap B$; then we can define it in terms of outer product and dual as

$$\textit{dual specification of meet}: \quad (A \cap B)^* = B^* \wedge A^*, \tag{7}$$

where the duality is to be taken relative to the smallest-grade blade containing both A and B (this is known as their join, and the intersection as their meet). If one simply takes duality relative to the full space, a meet can become zero in degenerate situations. (More about these operations and their efficient implementation in [4].)

An extension of the inner product beyond vector arguments can be developed as a product in its own right, with its own set of algebraic rules. When done properly, it is consistent with the rest of the framework in the sense that

$$\textit{extension of inner product}: \quad A \cdot B \equiv \left(A \wedge B^*\right)^{-*}, \tag{8}$$

with duality relative to a blade containing the join (one usually takes the pseudoscalar I_n).[2] This inner product has properties like

$$x \cdot (a \wedge b) = (x \cdot a)b - (x \cdot b)a. \tag{9}$$

The inner product is especially convenient to define orthogonal projection of subspaces as

$$\textit{orthogonal projection of } X \textit{ onto } B: \quad X \mapsto \left(X \cdot B^{-1}\right) \cdot B.$$

For flats, this corresponds to the usual orthogonal projection but it is more general: for instance, projecting a line onto a sphere produces a great circle.

Knowing duality also permits us to interpret elements like $\mathbf{a} \wedge \mathbf{b}$. In CGA, $\mathbf{a}$ and $\mathbf{b}$ are the dual representations of planes through the origin, for the points on these planes satisfy $x \cdot \mathbf{a} = 0$ and $x \cdot \mathbf{b} = 0$. Therefore by (7), the 2-blade $\mathbf{a} \wedge \mathbf{b}$ should be the dual representation of their intersection line. Points x on that line should then satisfy $x \cdot (\mathbf{a} \wedge \mathbf{b}) = 0$, and expanding according to (9) shows that this indeed holds. You may verify that the point at infinity e_∞ is on the line $(\mathbf{a} \wedge \mathbf{b})^{-*}$.

We now have enough to show that in CGA, $S = a \wedge b \wedge c \wedge d$ represents the sphere through the four points a, b, c, d. The geometry is illustrated in Fig. 3. By antisymmetry of $\wedge$, we can subtract any factor from the others without changing the value of S. We use a to produce $S = a \wedge (b - a) \wedge (c - a) \wedge (d - a)$. To find out what $(b - a)$ represents, solve $x \cdot (b - a) = 0$. This evaluates to $x \cdot a = x \cdot b$, and because of (2), this means that x has the same distance to a and b. So $(b - a)$

[2] This inner product is called the *left contraction* and denoted "⌋" in [2]. It differs in details from the inner product used in [1].

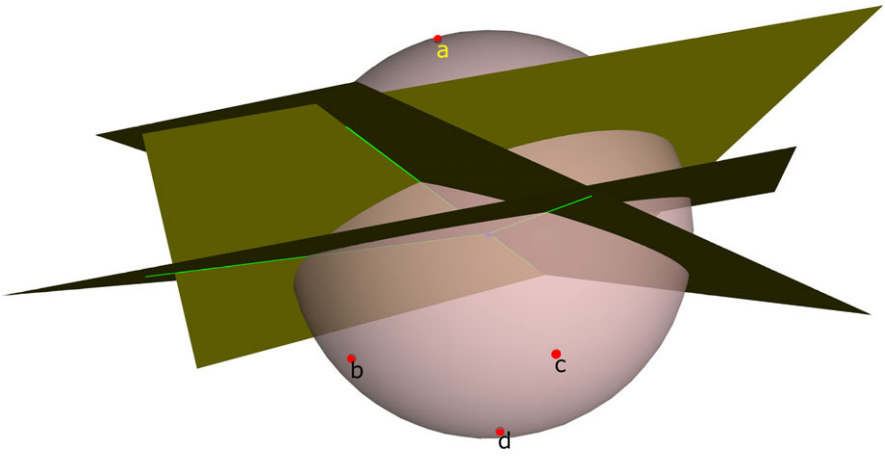


Fig. 3 The proof that $a \wedge b \wedge c \wedge d$ represents a sphere involves the intersection of the midplanes $b - a$, $c - a$, and $d - a$

is the dual representation of the midplane between a and b. Therefore $(b - a) \wedge (c - a) \wedge (d - a)$ is the dual representation of the intersection of three midplanes. These planes intersect in two points: the center of the sphere m and the point at infinity e_∞, so $(b - a) \wedge (c - a) \wedge (d - a)$ is proportional to $(m \wedge e_\infty)^*$. Then we find $S \propto a \wedge (m \wedge e_\infty)^* = (a \cdot (m \wedge e_\infty))^* = (m - \frac{1}{2}\rho^2 e_\infty)^*$ with $a \cdot m = -\frac{1}{2}\rho^2$. So indeed S is the dual of a dual sphere representation and therefore a sphere. This also gives a very compact way to compute center and radius of a sphere given by four points: they are simply the appropriate components of $(a \wedge b \wedge c \wedge d)^*$.

3 Bonus: The Elements of Euclidean Geometry as Blades

Closure of the operations of outer product and duality produces a suite of blades representing recognizable elements of Euclidean geometry. We have seen many examples of this already, and the full list is given in Table 2 from [2] (where n is the dimension of the Euclidean space, $\mathbf{E}$ a purely Euclidean element of appropriate grade, $\mathbf{E}^\star$ denotes the Euclidean dual $\mathbf{E}\mathbf{I}_n^{-1}$, and $T_\mathbf{p}$ denotes the translation versor over $\mathbf{p}$, see (11)). Care has been taken to orient the blades and their duals consistently.

The square of a normalized round gives its radius squared, and this may be negative. Such "imaginary rounds" occur naturally, for instance, when intersecting two spheres that are further apart than the sum of their radii. Because only the squared radius occurs in the conformal model, these elements are tractable in a real algebra. Tangents are in fact rounds of zero radius, indicative of their infinitesimal size. A tangent 2-blade occurs, for instance, as the grade 3 element that is the meet of two touching spheres. In this context, a weighted point may be viewed as a localized tangent scalar.

Table 2

Element	*Standard form X*	*Defining properties*
Direction	$\mathbf{E} \wedge e_\infty$	$e_\infty \wedge X = 0; e_\infty \cdot X = 0$
Dual direction	$-\mathbf{E}^\star \wedge e_\infty$	$e_\infty \wedge X = 0; e_\infty \cdot X = 0$
Flat	$T_\mathbf{p}(e_o \wedge \mathbf{E} \wedge e_\infty)T_\mathbf{p}^{-1}$	$e_\infty \wedge X = 0; e_\infty \cdot X \neq 0$
Dual flat	$T_\mathbf{p}(\mathbf{E}^\star(-1)^{n-\text{grade}(\mathbf{E})})T_\mathbf{p}^{-1}$	$e_\infty \wedge X \neq 0; e_\infty \cdot X = 0$
Tangent	$T_\mathbf{p}(e_o \wedge \mathbf{E})T_\mathbf{p}^{-1}$	$e_\infty \wedge X \neq 0; e_\infty \cdot X \neq 0; X^2 = 0$
Dual tangent	$T_\mathbf{p}(e_o \wedge \mathbf{E}^\star(-1)^n)T_\mathbf{p}^{-1}$	$e_\infty \wedge X \neq 0; e_\infty \cdot X \neq 0; X^2 = 0$
Round	$T_\mathbf{p}((e_o + \frac{1}{2}\rho^2 e_\infty) \wedge \mathbf{E})T_\mathbf{p}^{-1}$	$e_\infty \wedge X \neq 0; e_\infty \cdot X \neq 0; X^2 \neq 0$
Dual round	$T_\mathbf{p}((e_o - \frac{1}{2}\rho^2 e_\infty) \wedge \mathbf{E}^\star(-1)^n)T_\mathbf{p}^{-1}$	$e_\infty \wedge X \neq 0; e_\infty \cdot X \neq 0; X^2 \neq 0$

It is especially notable that the various uses and meanings of "vector with direction $\mathbf{u}$" from applied linear algebra get their own "algebraic data structures":

- *a point at location* $\mathbf{u}$ is represented by the CGA vector $e_o + \mathbf{u} + \frac{1}{2}\mathbf{u}^2 e_\infty$
- *a free vector* is represented by the translation invariant 2-blade $\mathbf{u} \wedge e_\infty$
- *a normal vector* is the vector $p \cdot (\mathbf{u} \wedge e_\infty)$ and can shift on a localized plane
- *a force vector* is represented by the 3-blade $p \wedge \mathbf{u} \wedge e_\infty$ and can shift along a line
- *a tangent vector* $\mathbf{u}$ at p is the localized 2-blade $p \cdot (p \wedge \mathbf{u} \wedge e_\infty)$

All these automatically move appropriately under Euclidean versors, without a programmer needing to specify that they should (by giving them their own "classes" and "methods," as is required in common practice in classical software, even when based on homogeneous coordinates).

4 Bonus: Euclidean Motions Through Sandwiching

We have seen how all orthogonal transformations can be made as multiple reflections and that a single reflection is represented by an invertible vector $\mathbf{a}$ as the transformation $x \mapsto -\mathbf{a}x\mathbf{a}^{-1}$. Now that we know what the vectors in the conformal model represent, we can easily generate the versors for common motions. Euclidean motions are generated by multiple reflections in planes, and we have seen that those are dually represented by vectors of the form $\pi = \mathbf{n} + \delta e_\infty$ that satisfy $e_\infty \cdot \pi = 0$.

- *Rotation in a plane through the origin*: If we take two unit dual planes at the origin $\mathbf{n}_1$ and $\mathbf{n}_2$ with a relative angle of $\phi/2$ from $\mathbf{n}_1$ to $\mathbf{n}_2$, the double reflection first in $\mathbf{n}_1$ and then in $\mathbf{n}_2$ is represented as

$$R_{\mathbf{I}\phi} = \mathbf{n}_2\mathbf{n}_1 = \mathbf{n}_2 \cdot \mathbf{n}_1 + \mathbf{n}_2 \wedge \mathbf{n}_1 = \cos(\phi/2) - \mathbf{I}\sin(\phi/2). \tag{10}$$

  When used in a sandwiching operation, this is a rotation over the angle ϕ around the dual line given by the unit 2-blade $\mathbf{I}$ (proportional to $\mathbf{n}_1 \wedge \mathbf{n}_2$). Such a 2-blade has the property $\mathbf{I}^2 = -1$. To show this, introduce an orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2\}$,

write $\mathbf{I} = \mathbf{e}_1 \wedge \mathbf{e}_2 = \mathbf{e}_1\mathbf{e}_2$, and compute using the associativity property of the geometric product: $(\mathbf{e}_1 \wedge \mathbf{e}_2)(\mathbf{e}_1 \wedge \mathbf{e}_2) = (\mathbf{e}_1\mathbf{e}_2)(\mathbf{e}_1\mathbf{e}_2) = -\mathbf{e}_2\mathbf{e}_1\mathbf{e}_1\mathbf{e}_2 = -\mathbf{e}_2\mathbf{e}_2 = -1$. In this real geometric algebra, we therefore naturally get elements that square to -1. In 3D, there is a basis for 2-blades consisting of the elements $\mathbf{I} = \mathbf{e}_1 \wedge \mathbf{e}_2$, $\mathbf{J} = \mathbf{e}_2 \wedge \mathbf{e}_3$, and $\mathbf{K} = \mathbf{e}_3 \wedge \mathbf{e}_1$, each squaring to -1 and having multiplicative relationships like $\mathbf{IJ} = -\mathbf{JI} = -\mathbf{K}$. These are of course isomorphic to the elementary quaternions which have proven so useful for 3D rotation computations. In geometric algebra, they are introduced in a real manner as products of vectors, fully integrated with the real elements they operate on. We will soon see that they can rotate any element, and derive the versor for a rotation around a general line in Sect. 6.

- *Translation*: A translation over a vector $\mathbf{t}$ is generated by reflection in two dual planes separated by a vector $\mathbf{t}/2$, resulting in the element: $(\mathbf{t} + \frac{1}{2}\mathbf{t} \cdot \mathbf{t}e_\infty)\mathbf{t} = \mathbf{t}^2(1 - \mathbf{t}e_\infty/2)$. Since a scalar multiple generates the same motion in the sandwiching product with the inverse, we prefer to define

$$\textit{versor for translation over } \mathbf{t}\text{:} \quad T_\mathbf{t} \equiv 1 - \mathbf{t}e_\infty/2. \tag{11}$$

  You can check that the point representation (1) is indeed related to the point at the origin e_o by translation over $\mathbf{x}$, since $x = T_\mathbf{x} e_o T_\mathbf{x}^{-1}$.
- *General rigid body motion*: A general rigid body motion can be constructed in the usual manner as a rotation followed by a translation. In CGA, an alternative is to make it directly as the reflection in two lines, which produces a screw motion (see [2]).
- *Uniform scaling*: Although not strictly a rigid body motion, the Euclidean similarity transformation of uniform scaling can be made by subsequent reflection in two dual spheres at the origin such as $e_o - \frac{1}{2}\rho_1^2 e_\infty$ and $e_o - \frac{1}{2}\rho_2^2 e_\infty$. After some simplification, the scaling versor for a uniform scaling by e^γ is found to be

$$S_\gamma \equiv \cosh(\gamma/2) + \sinh(\gamma/2) e_o \wedge e_\infty.$$

More versors can be generated by reflection in spheres, notably for the conformal operation of a *transversion*—details may be found elsewhere [2].

5 Bonus: Structure Preservation and the Transfer Principle

All constructions of elements were based on the linear combinations of geometric products, since the other products are ultimately expressible in that manner. Therefore, when we act on them with a versor V in the sandwiching product, all constructions transform covariantly. For the outer product, this means that equations hold like the following:

$$V(a \wedge b)V^{-1} = \left(VaV^{-1}\right) \wedge \left(VbV^{-1}\right).$$

The same structure-preserving property holds for *all* operations we introduced, be they spanning, inner product, or duality (relative to a transformed pseudoscalar). In words: "*the transformation of a construction equals the construction of the transformed elements*." This fact is very convenient, for it implies that we can simply construct something at the origin and then move it into place to find the general form (hence our preference for origin-based specification in the table above). And composite elements move by the same versor as points do: the translation versor $T_{\mathbf{t}}$ universally translates points, lines, planes, spheres, or tangent elements. As we mentioned, there is no longer any need for data structures distinguishing between "position vectors" which feel translations and "direction vectors" which do not; all is automatically administrated in the algebraic behavior of the corresponding elements. This is an enormous advantage relative to the classical homogeneous model for the development of structural code, either by hand or using a code generator [3].

This principle is also extremely useful in derivations. Let us, for instance, use it to prove the general formula for the reflection of a line Λ in a dual plane π as $\Lambda \mapsto -\pi\Lambda\pi^{-1}$, simply from the 1-D direction reflection formula (5). A line Λ_0 with direction $\mathbf{u}$ through the origin is given as $\Lambda_0 = e_o \wedge \mathbf{u} \wedge e_\infty$, and a dual plane π_0 through the origin with normal vector $\mathbf{n}$ as $\pi_0 = \mathbf{n}$. The reflection of the direction $\mathbf{u}$ is affected by (5) as $\mathbf{u} \mapsto \mathbf{u}' \equiv -\mathbf{n}\mathbf{u}\mathbf{n}^{-1} = -\pi_0\mathbf{u}\pi_0{}^{-1}$. The reflected line is then $\Lambda_0' = e_o \wedge \mathbf{u}' \wedge e_\infty$. Now we note that due to the algebraic commutation (i.e., the geometric orthogonality) of the bold Euclidean and the nonbold extra dimensions e_o and e_∞, we have $-\pi_0 e_o \pi_0{}^{-1} = -\mathbf{n}e_o\mathbf{n}^{-1} = e_o$ and $-\pi_0 e_\infty \pi_0{}^{-1} = -\mathbf{n}e_\infty\mathbf{n}^{-1} = e_\infty$. Therefore we can "pull out" the reflection operator to act on the whole line Λ_0 by (5):

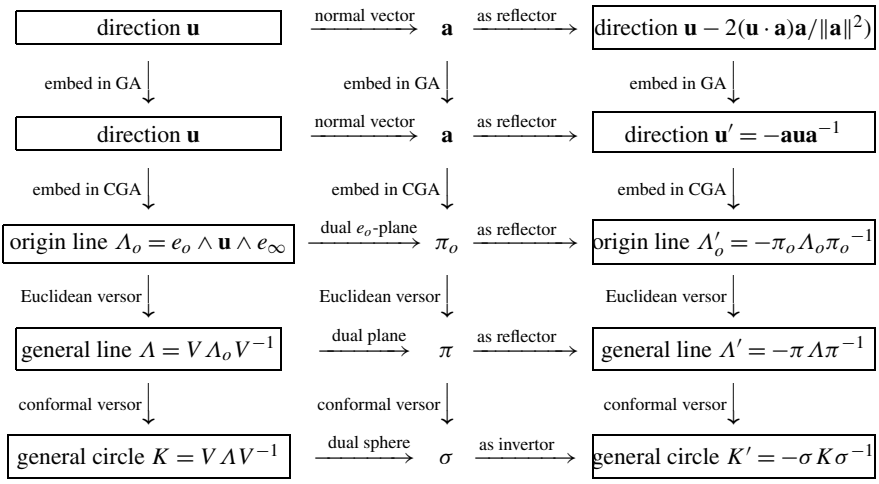
$$\begin{aligned}\Lambda_0' &= \left(-\pi_0 e_o \pi_0{}^{-1}\right) \wedge \left(-\pi_0 \mathbf{u} \pi_0{}^{-1}\right) \wedge \left(-\pi_0 e_\infty \pi_0{}^{-1}\right) \\ &= -\pi_0 (e_o \wedge \mathbf{u} \wedge e_\infty)\pi_0{}^{-1} = -\pi_0 \Lambda_0 \pi_0{}^{-1}.\end{aligned}$$

This is still only true at the origin, but we can move this construction by a motion versor V to an arbitrary location. All elements change to their general form $\pi = V\pi_0 V^{-1}$, $\Lambda = V\Lambda_0 V^{-1}$, and the reflection transformation preserves its structure since $V\Lambda_0' V^{-1} = (V\pi_0 V^{-1})(V\Lambda_0 V^{-1})(V\pi_0 V^{-1}) = \pi\Lambda\pi^{-1}$. Therefore the general reflection formula of a line in a plane is simply

$$\textit{reflection of a line } \Lambda \textit{ in the dual plane } \pi\text{:} \quad \Lambda \mapsto -\pi\Lambda\pi^{-1}.$$

This includes all aspects of location, direction, and orientation. Note that this computation reflects a general line in a general plane without computing its intersection point—try doing that using linear algebra! (If you need the intersection point of line and plane, it is $\pi \cdot \Lambda$, by straightforward application of the universal meet opera-

tion (7) and duality (6), (8).)



We can even apply an arbitrary conformal versor and change the reflecting dual plane π into a dual sphere σ, and the line L into a circle K; the result is a spherical inversion operation. (As a further extension, another application of the structure preservation property shows that the reflection in σ of a general element X is $X \mapsto (-1)^{\text{grade}(X)}\sigma X \sigma^{-1}$.)

The conformal model renders all transitions trivial in this transfer, all the way from a reflection of a Euclidean direction vector at the origin to the inversion of a general circle in a general sphere. Such is the power of a structure-preserving framework!

6 Trick 5: Exponential Representation of Versors

Even-graded versors, made by an even number of reflections, represent motions that can be performed continuously and in small amounts. In Euclidean and Minkowski spaces, *all even-graded versors can be written as the exponentials of bivectors*. The bivector specification of an even versor is often more directly related to the geometry of the situation than the "product of vectors" method.

As an example of the exponential rewriting, take the rotation $R_{\mathbf{I}\phi}$ over the angle ϕ, parallel to the $\mathbf{I}$-plane as treated in (10),

$$R_{\mathbf{I}\phi} = \cos(\phi/2) - \sin(\phi/2)\mathbf{I} = e^{-\mathbf{I}\phi/2}.$$

It is the property $\mathbf{I}^2 = -1$ that makes the exponential rewriting permitted:

$$
\begin{aligned}
e^{-\mathbf{I}\phi/2} &= 1 + \frac{1}{1!}(-\mathbf{I}\phi/2)^1 + \frac{1}{2!}(-\mathbf{I}\phi/2)^2 + \cdots \\
&= \left(1 - \frac{1}{2!}(\phi/2)^2 + \cdots\right) + \left(\frac{1}{1!}(\phi/2)^1 - \frac{1}{3!}(\phi/2)^3 + \cdots\right)\mathbf{I} \\
&= \cos(\phi/2) - \sin(\phi/2)\mathbf{I}.
\end{aligned}
$$

The translation versor of (11) can also be written in this exponential form; but since it involves the bivector $\mathbf{t} \wedge e_\infty$, the expansion truncates after two terms (fundamentally due to $e_\infty^2 = 0$):

$$T_\mathbf{t} = 1 - \mathbf{t} \wedge e_\infty/2 = e^{-\mathbf{t}\wedge e_\infty/2}.$$

A rotation around a general 3D unit line Λ over ϕ is now generated by the versor:

$$\textit{rotation around } \Lambda \textit{ over } \phi: \quad R_{\Lambda,\phi} = e^{\Lambda^*\phi/2}$$

Proof This follows from the simply derived structural property

$$V \exp(B) V^{-1} = \exp\bigl(V B V^{-1}\bigr)$$

and the transfer property applied as follows. First recognize that the rotation axis of the origin rotation $R_{\mathbf{I}\phi}$ is the line $\Lambda_0 = \mathbf{I}^* = -\mathbf{I}^{-*}$, so the origin rotation is $\exp(\Lambda_0{}^*\phi/2)$. Then transfer this by a translation T to the actual location of the desired axis Λ, which changes $\Lambda_0{}^*$ to $T(\Lambda_0{}^*)T^{-1} = (T\Lambda_0 T^{-1})/(T I_{4,1} T^{-1}) = \Lambda^*$ since the pseudoscalar $I_{4,1}$ involved in the dualization is translation invariant. Done. □

General rigid body motions can of course also be made, for instance, by the usual method of combining an origin rotation with a translation. You find that the result can be written as the exponential of a general conformal bivector on the basis $\{\mathbf{e}_1 \wedge \mathbf{e}_2, \mathbf{e}_2 \wedge \mathbf{e}_3, \mathbf{e}_3 \wedge \mathbf{e}_1, \mathbf{e}_1 \wedge e_\infty, \mathbf{e}_2 \wedge e_\infty, \mathbf{e}_3 \wedge e_\infty\}$, giving the six degrees of freedom required. Since this space of bivectors is linear, it can be used for motion interpolation. To interpolate between two poses characterized by the versors M_0 and M_1, find their bivectors $B_0 = \log(M_0)$ and $B_1 = \log(M_1)$. Now apply a standard vector interpolation method to smoothly change B_0 into B_1 through intermediate bivectors B_i; then use the versors $\exp(B_i)$ to generate the interpolated poses. To execute this procedure, one needs to find the bivector corresponding to a given versor; such "versor logarithms" may be found in [2].

Linearization of versor motions for extrapolation or estimation is also possible and requires geometric calculus. When performed (see [1]), the first order change in an element X that is moved by a changing versor $V(\tau)$ from a standard element X_0 as $X(\tau) = V(\tau)X_0V(\tau)^{-1}$ is

$$
\begin{aligned}
X(\tau + d\tau) &= X(\tau) + \bigl(\Omega(\tau)X(\tau) - X(\tau)\Omega(\tau)\bigr) \\
\text{with } \Omega(\tau) &= \left(\frac{d}{d\tau}V(\tau)\right)V(\tau)^{-1}.
\end{aligned}
$$

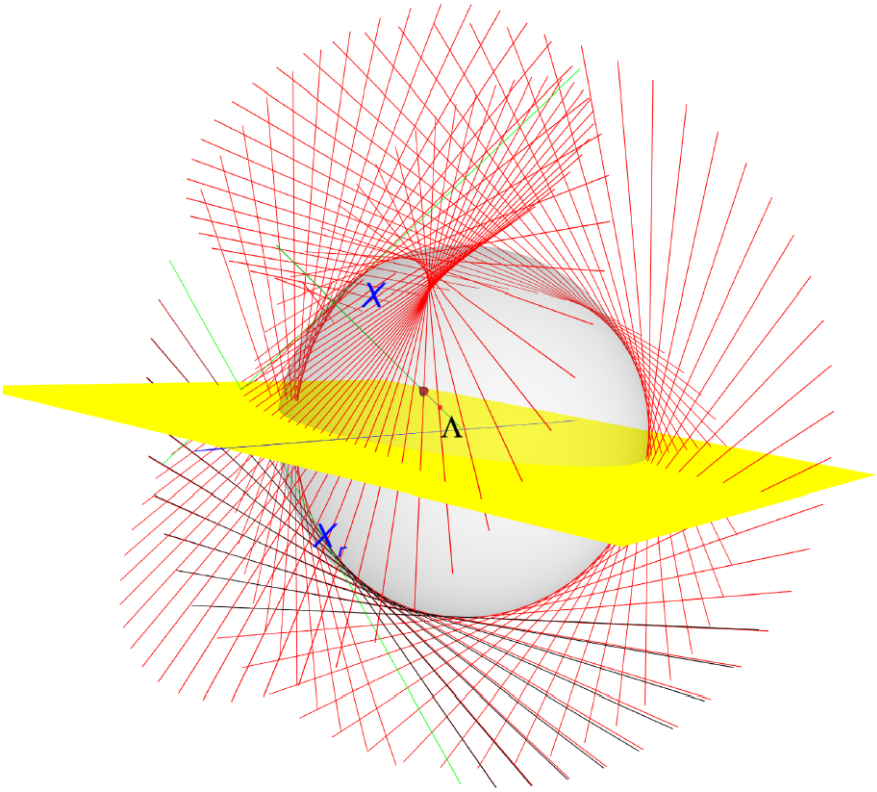


Fig. 4 The mirror Π rotates ϕ round a line Λ, and a line X is reflected in it. Using a local first-order linearization of the reflection versor, one can derive the perturbation of the reflected line to second order (*in black*) to be the rotation with versor $\exp(-\phi((\Lambda \cdot \Pi)/\Pi)^*)$, i.e., around the projection of Λ onto Π with angle $2\phi\cos(\Pi, \Lambda)$. For details, see [2]

If V is normalized, Ω is a bivector, and its commutator product with $X(\tau)$ preserves the grade. This linearization of geometrical perturbations is very useful in applications, see Fig. 4. The full geometric calculus is truly powerful, and one can differentiate relative to an arbitrary element of the algebra (such as a blade or a versor). We cannot treat that here, and the reader is referred to introductions like [2] and [1].

7 Trick 6: Sparse Implementation at Compiler Level

Implementation of CGA may seem to be expensive. After all, to treat a 3D space, we embed into a 5D representational space and use the geometric algebra of that, which involves a 2^5-D basis of constructible elements of all grades. Yet the use we make of this space is restricted, and the elements are therefore somehow sparse.

Ultimately, the main purpose of the algebraic organization is to keep track automatically of the administration of the meaning of the coordinates of points, lines, planes, spheres, etc., simultaneous with performing the quantitative computations. That is in a sense a Boolean selection task of the algebra, which one would intuitively expect not to be too expensive. Indeed it has proved possible to limit the overhead of the use of CGA to about 10% relative to the best available coordinate code programmed classically. For the computer science techniques that achieve this, consult [2] and [3]. A warning: before you start using CGA in commercial applications, be aware that it is covered by a *US patent* [6].

References

1. Doran, C., Lasenby, A.: Geometric Algebra for Physicists. Cambridge University Press, Cambridge (2000)
2. Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science, An Object-Oriented Approach to Geometry. Morgan Kaufman, San Mateo (2007). Second revised printing 2009. See also www.geometricalgebra.net
3. Fontijne, D.: Efficient implementation of geometric algebra. Ph.D. Thesis, U. of Amsterdam (2007). See also www.science.uva.nl/~fontijne
4. Fontijne, D., Dorst, L.: Efficient algorithms for factorization and join of blades. This volume
5. Hestenes, D.: New tools for computational geometry and the rejuvenation of screw theory. This volume
6. Hestenes, D., Rockwood, A., Li, H.: System for encoding and manipulating models of objects. US Patent 6,853,964, granted 8 February 2005