A Guided Tour to the Plane-Based Geometric Algebra PGA

Leo Dorst & Steven De Keninck
University of Amsterdam

Version 2.0– March 14, 2022

Planes are the primitive elements for the constructions of objects and operators in Euclidean geometry. Triangulated meshes are built from them, and reflections in multiple planes are a mathematically pure way to construct Euclidean motions.

A geometric algebra based on planes is therefore a natural choice to unify objects and operators for Euclidean geometry. The usual claims of 'completeness' of the GA approach leads us to hope that it might contain, in a single framework, all representations ever designed for Euclidean geometry - including normal vectors, directions as points at infinity, Plücker coordinates for lines, quaternions as 3D rotations around the origin, and dual quaternions for rigid body motions; and even spinors.

This text provides a guided tour to this *algebra of planes PGA*. It indeed shows how all such computationally efficient methods are incorporated and related. We will see how the PGA elements naturally group into blocks of four coordinates in an implementation, and how this more complete understanding of the embedding suggests some handy choices to avoid extraneous computations. In the unified PGA framework, one never switches between specialized representations for subtasks, and this obviously saves any time spent on data conversions.

Relative to other treatments of PGA, this text is rather light on the mathematics. Where you see careful derivations, they involve the aspects of *orientation and magnitude*. These features have been neglected by authors focussing on the mathematical beauty of the projective nature of the algebra. But since orientation and magnitude are very relevant for practical usage, we must incorporate them into the framework.

This text started as a replacement for Chapter 11 of the 2007 book 'Geometric Algebra for Computer Science', written at a time when PGA was underappreciated. At 100 pages, it has become rather more; but we do assume some familiarity with the standard geometric algebra of the chapters that came before. Even if you are totally new to GA, you will still get the main gist of the power of PGA, and that may motivate you to study the basics of general GA. Welcome to the future!

PGA 2.0: Sign Changes Relative to Earlier Versions

There was always a dilemma on how to choose our signs in a standardization of PGA. In the GA community, Leo came to PGA from CGA, where a plane is encoded as $p = \mathbf{n} + \delta\infty$, and there is a reciprocal vector o such that $o \cdot \infty = -1$. Steven, working in computer graphics, found it more natural to explain PGA as a deepening of homogeneous coordinates, and encode a plane as $p = \mathbf{n} - \delta e_0$. The minus sign there derives from the homogenization of the plane equation $\mathbf{x} \cdot \mathbf{n} = \delta$. In that choice of signs, one would use a reciprocal e_0^r (where needed) satisfying $e_0^r \cdot e_0 = 1$.

There is no essential difference, of course. The extra dimension plays essentially a computationally administrative role, the interpretable entities will always be Euclidean. But it is pleasant to have a memorable iconic form of one's equations, especially when we want to encourage using signs to encode a consistent 'oriented' version of geometry. Originally, we took the CGA option, merely replacing ∞ by e in our notation.

Yet our first target audience for applications is computer graphics, motion capture and dynamics simulations. We find that in practice the software connection to homogeneous coordinates is more important than the CGA connection. So we have now decided to change to the second option, and we used that in our new tutorial on Dynamics in PGA [1]. The present text needed to be updated to be consistent with that usage.

From now on, we use e_0 as basis vector for the extra dimension in PGA (Leo still prefers a more indexfree mathematical notation like $\boldsymbol{\epsilon}$, but let us go all Steven's way for now). Mostly, this is a simple replacement setting $e_0 = -e$ relative to the 'before 2.0' versions of this tutorial; but where duality is involved, our pseudoscalar changes from $e\mathbf{I}_d$ to $e_0\mathbf{I}_d$; that is effectively an additional sign change which needs to be propagated through the formulas.

One of the consequences is that the `join` now is $A \vee B = \star^{-1}(\star A \wedge \star B)$ (rather than with swapped arguments); the oriented line from P to Q is still $P \vee Q$. Moreover, some issues arise when interpreting orientations, since the regularly 'oriented plane at infinity ∞' is how effectively $-e_0$.

Either choice is a connection to some historical convention; we now opt to go for Steven's crowd pleaser, since it increases chances of adoption for the framework. Leo apologizes for not doing this from the start...

Please refer to this chapter as:

Leo Dorst & Steven De Keninck
A Guided Tour to the Plane-Based Geometric Algebra PGA
2022, version 2.0
Available at `http://www.geometricalgebra.net`
and `http://bivector.net/PGA4CS.html`.

Contents

1 A Minimal Algebra for Euclidean Motions

1.1 PGA: One More Dimension Suffices

While the 3D vector space model $\mathbb{R}_3$ (of Chapter 10 in [2]) can nicely model directions, it is considered to be inadequate for use in 3D computer graphics. The reason is primarily a desire to treat points and vectors as different objects; after all, they are transformed differently by translations. Instead, graphics uses an extension of linear algebra known as 'homogeneous coordinates', which is often described as augmenting a 3-dimensional vector $\mathbf{v}$ with coordinates $(v_1, v_2, v_3)^\top$ to a 4-vector $(v_1, v_2, v_3, 1)^\top$. This extension helps to discriminate point location vectors from direction vectors, and makes nonlinear operations such as translations and projective transformations implementable as linear mappings.

This 'homogeneous model' can be described in terms of a geometric algebra of one more dimension, with an explicit extra basis vector. That is how we did it in our book [2], in its Chapter 11. Then we went on to show the shortcomings of this model: for *while translations become linear, they do not become rotors*. That means that the great advantage of geometric algebra, namely 'structure preservation of universally applicable operators', fails in this model. We then went on to Conformal Geometric Algebra (CGA), turning translations into rotors by going up 2 dimensions (rather than just 1 as in the homogeneous model) in our representational space. That worked as desired.

The conformal model is indeed useful, and its inclusion of spheres and circles as primitives truly makes it feel like a natural Euclidean geometry. But as has been remarked many times, we get too much: going up 2 dimensions in this manner actually lands us in a space in which the rotors are *conformal transformations*, rather than merely the *Euclidean motions* we were looking for. Viewing the 3D rigid body motions with their six degrees of freedom (dof) as special rotors in a space that allows 10-dof rotors gives implementational and computational issues. It works, but one would have hoped to do Euclidean geometry of flat offset elements like lines and planes with less.

And indeed, one can. Parallel to the development of CGA, Charles Gunn proposed that an algebra that he denoted $\mathbb{R}^*_{3,0,1}$ could have the rigid body motions precisely as its rotors, and that it should be the natural algebra for Euclidean geometry. Its useful structure had earlier been exposed by Selig [3] as the Clifford algebra of points, lines and planes. Gunn called it PGA, for

Projective Geometric Algebra. That name is somewhat unfortunate, since it suggests to us that it is the algebra that can generate projective transformations by rotors; whereas in fact the 'projective' refers to the homogeneous representation it acts on. However, as we will see, PGA is the geometric algebra of (hyper)planes; so if we read PGA as *'Plane-Based Geometric Algebra'*, the moniker is fine.[1]

To represent translations as rotors, one needs null vectors (i.e., vectors squaring to zero, which make exponentiation produce linear terms). In CGA, there are two null vectors, and they combine to make the pseudoscalar invertible. In PGA, with only one null vector, the pseudoscalar is a null blade, and *not* invertible. It was the unconventional nature of duality in PGA that hampered its acceptance by the GA community (which had collectively jumped onto CGA by 2007). Rather than division by the pseudoscalar, Gunn introduced a J-map between $\mathbb{R}^*_{d,0,1}$ (where the vectors represent hyperplanes) and its dual space $\mathbb{R}_{d,0,1}$ (where the vectors represent points). It seemed that one needed both those spaces to make things work; and then one effectively had the same number of basis elements as the CGA model of $\mathbb{R}_{d+1,1}$, which dualizes more conveniently and has its round primitives like spheres and circles. So the community held on to CGA, and Gunn's message was not heard. His admonition that we confused 'duality' and 'polarity' (a metric form of duality) was considered too mathematically prissy, and viewed in the same vein as the unfortunate tradition of considering 'linear algebra without metrics' as the fundamental representation, with metrics as an afterthought (which has made our everyday metric applications of LA needlessly involved, since we almost always have a metric anyway).

We will now develop PGA $\mathbb{R}_{d,0,1}$ immediately as an algebra of planes. When this is done from the start, there is no real need to see it as dual to anything (just as CGA, an algebra of spheres, does not have a dual star in its notation $\mathbb{R}_{d+1,1}$). PGA can stand on its own, mostly.

1.2 How to Choose a Geometric Algebra?

For those who are new to geometric algebra, let us briefly summarize the organizational principle behind the choice of a geometric algebra for a class

[1]Leo himself dissuaded Charles early on from calling the PGA $\mathbb{R}^*_{d,0,1}$ 'EGA', for Euclidean Geometric Algebra, when he did not yet fully understand what it did. That might not have been a bad name, after all. Sorry Charles...

of geometry problems. It is: *choose the smallest algebra in which your symmetries (the set of transformations under which your objects move) become orthogonal transformations.* The reason behind this insistence on orthogonality is that geometric algebra has a particularly efficient way to represent orthogonal transformations (as versors, aka rotors or spinors), which provides automatically covariant combinations of primitives and operators (and that saves a lot of code). There are several routes to achieving this versor representation.

- One can observe an important conserved quantity, such as Euclidean distance, and choose the inner product of the representational space accordingly. Since orthogonal transformations preserve the inner product, they make up your invariant motions.

  For Euclidean geometry, this leads to CGA (Conformal Geometric Algebra), as explained in Chapters 13-16 of [2]; points are represented by null vectors since they have zero distance to themselves. And for 3D projective geometry it leads to $\mathbb{R}_{3,3}$, see [4], the geometric algebra of lines.

- One can start from Cartan-Dieudonné's theorem, which states that in n-D space all orthogonal transformations can be made by at most n reflections. Set up the representative space such that *the reflecting elements are its vectors.*

  This is the recipe that for Euclidean geometry leads to PGA, using reflections in (hyper)planes to generate all Euclidean motions. In this manner, one can also construct an algebra for conformal geometry, by taking spherical inversion as one's reflection; then CGA results.

- A third method which has been followed starts from geometric primitives in coordinate form, and basically assigns a dimension to each coordinate, with a metric that makes the coordinate algebra come out right. This tends to lead to high-dimensional algebras like $\mathbb{R}_{9,6}$ for 3D quadrics [5], and the versors then generate a host of superfluous symmetries beyond the ones of interest.

  We are yet to be convinced of this approach by a controlled example with a clearly relevant set of versors.

PGA is an algebra in which the vectors represent Euclidean planes, and it therefore describes Euclidean geometry according to the second principle above. In 3D, four planes are enough to encode any Euclidean motion. Thus we expect a 4-dimensional basis for the PGA of 3D Euclidean space. We will see how the elements that represent a quadruple reflection act exactly as unit dual quaternions (which are currently the most sophisticated way to encode Euclidean motions in computer graphics), and that our embedding implements them most efficiently. As we will show, we can arrange the PGA representation of all primitives and basic motions of 3D geometry naturally in blocks of 4 coordinates which are processed identically, and thus align well with SIMD/GPU architecture (glance ahead at Table 5).

Usually, 'projective' or 'homogeneous' is interpreted as 'modulo a scale factor' – a bad habit inherited from projective geometry in mathematics. If instead we are slightly more strict and interpret our elements as 'identical modulo a *positive* scale factor', we obtain an *oriented geometry* that enables consistent encoding of the sides of planes, the propagation direction of rays, oriented distances, etc. Doing so will refine the original presentation of PGA. Moreover, in many instances in this text, we will also find useful geometrical meaning in the value of the weighting factors themselves, often as a numerical indication of how non-degenerate a geometric situation is (the more orthogonal a line intersects a plane, the more stable the intersection point is under small perturbations, etc.).

There is more to Euclidean geometry than moving point sets around. PGA tells us all – if we listen.

1.3 Playing with PGA

Because PGA turns out to be a subalgebra of CGA [6] (with the null vector in the extra dimension e_0 directly corresponding to $-\infty$), you can use the free GAviewer software provided with [2] to visualize it. After downloading the library to generate the book figures (from the book website: `www.geometricalgebra.net/figures.html`), just type in '`init(2)`' as first command to start up the CGA model (with appropriate basis vectors and visualization). You may want to add a line '`e = -ni;` ' to allow you to use the notation of the present text. Just never use any expressions involving the '`no`' basis vector in their definition. That is an element of CGA which PGA does not have, and omitting it will reduce the primitives to flat subspaces. Since GAviewer draws such flat subspaces whether they have been specified 'directly' or 'dually' (and the latter corresponds to PGA), not only does it compute your results, but it also draws them automatically. That is how we generated many of the figures in this replacement chapter.

However, that GAviewer software is 15 years old by now. You may also use more modern computational and visualization alternatives such as the JavaScript tool `ganja.js` (though currently in 2021, that does not yet indicate the orientation of elements).

2 Meet the Primitives of PGA

Let us first simply use Euclidean direction vectors to denote the location $\mathbf{x}$ of a point relative to the origin, and define a plane with unit normal vector $\mathbf{n}$, at an oriented distance δ from the origin. The equation of this plane is

$$\mathbf{x} \cdot \mathbf{n} = \delta. \tag{1}$$

The idea behind homogeneous coordinates is to rewrite this equation in terms of an inner product for a vector $x \equiv [\mathbf{x}^\top, 1]^\top$ in a space of one more dimension. Let us denote the basis vector for this extra orthogonal dimension momentarily by ϵ_o, then $x = \mathbf{x} + \epsilon_o$. We could then (naively) assume a Euclidean metric in this extended space, and represent the plane with its $\mathbf{n}$ and δ parameters as $[\mathbf{n}^\top, -\delta]^\top$, i.e., as a vector $n = \mathbf{n} - \delta\epsilon_o$. That would rewrite the equations into the homogeneous form

$$x \cdot n = 0. \tag{2}$$

There is a zero on the right-hand side, so any multiple of x or n would represent the same plane. (That is the reason behind the mathematical term 'homogeneous' for this representational trick.)

The Euclidean metric assumption is not really required to obtain this functionality. One could also denote a point representative as a *column vector*, and the plane representative as a *row vector*, and employ a matrix product

$$\begin{bmatrix} n_1 & n_2 & \cdots & -\delta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}. \tag{3}$$

This is in fact a non-metric approach, with the plane being a 1-form n acting on the 1-vector x (as a mathematician might phrase it). But of course the concept of a normal (i.e., perpendicular) vector to characterize the direction of a plane is convenient, so the Euclidean parts of x and n (i.e., $\mathbf{x}$ and $\mathbf{n}$) do feel metric. It is the extra dimension ϵ_o for the point that is really awkward to consider as part of a Euclidean metric. For instance, one should not take the inner product of two point representatives, which would involve a term $\epsilon_o \cdot \epsilon_o$ that is ungeometrical. Perhaps it should be zero – but that would make the metric no longer Euclidean. This is where PGA takes over.

$\cdot$	e_0	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$
e_0	0	0	0	0
$\mathbf{e}_1$	0	1	0	0
$\mathbf{e}_2$	0	0	1	0
$\mathbf{e}_3$	0	0	0	1

Table 1: *The inner product in PGA.*

2.1 PGA: Planes as Vectors

To set up PGA we focus on the representation of Euclidean *planes* as basic, rather than of Euclidean *points*. (The reason for focusing on the representation of Euclidean planes is that we are going to represent Euclidean motions as reflections in planes, as we stated in the introduction, and it is handy to have our reflectors be vectors.) Given a plane with unit normal vector $\mathbf{n}$ at directed distance δ from the origin (measured in the direction $\mathbf{n}$, so $\delta\mathbf{n}$ is a location on the plane), we pick as homogeneous representative the vector [2]

$$n = \mathbf{n} - \delta\, e_0. \tag{4}$$

To have such a hyperplane as a vector, we clearly need to set up a space of $d+1$ dimensions. First there are the d Euclidean normal directions, which we may characterize by the d basis vectors $\mathbf{e}_i$ of an orthonormal basis. Moreover, this basis is augmented with an extra orthogonal basis vector we denote as e_0 (we write it non-bold to remind ourselves that it is not a vector in Euclidean space). We pick a metric for this space in which $\mathbf{e}_i \cdot \mathbf{e}_j = \delta_{ij}$, and in which $\mathbf{e}_i \cdot e_0 = 0$. And we choose $e_0 \cdot e_0 = 0$, so e_0 is a *null vector*. These relationships are summarized in the inner product table (Table 1).

With this metric, the squared norm $\|n\|^2 = n \cdot n$ of a vector representing a hyperplane is identical to the squared norm $\|\mathbf{n}\|^2 = \mathbf{n} \cdot \mathbf{n}$ of its Euclidean normal vector. Therefore a hyperplane having a unit normal vector squares

[2]If you have read an earlier version of the present text, older than 2.0, you may realize that we have changed the sign of the basis vector of the extra dimension. The current sign agrees better with expectations from homogeneous coordinates, the earlier $e = -e_0$ agreed better with the signs from CGA where it is analogous to ∞. We now believe that the CS community is better served by familiar signs. Of course this changes many things; we also changed the order of arguments in the `join` operator, to make everything internally sensible.

to 1.

$$\text{normalized (hyper)plane} : n = \mathbf{n} - \delta e_0, \quad \text{with } \mathbf{n} \in \mathbb{R}^d \text{ and } n^2 = \mathbf{n}^2 = 1. \quad (5)$$

Our convention will be to denote the Euclidean elements (from the algebra of directions of Chapter 10 of [2]) in **bold** font, and elements in the $(d+1)$-dimensional representation algebra by math *italic*.

The *dot product* between two hyperplanes also has a straightforward geometric interpretation, similar to the dot product between Euclidean direction vectors

$$n \cdot m = \mathbf{n} \cdot \mathbf{m} = \|\mathbf{n}\| \, \|\mathbf{m}\| \, \cos(\phi). \quad (6)$$

So independent of their locations, the dot product of the hyperplanes provides information on the magnitude of the mutual angle ϕ between their normals (though not on its sign, which would require also knowing the value of $n \wedge m$).

Some degenerate cases of the hyperplane representation $n = \mathbf{n} - \delta e_0$ may improve our understanding.

- A purely Euclidean hyperplane $n = \mathbf{n}$ passes through the origin. Since the origin should have no special geometric significance, this may not seem a sensible special case. However, hand computations and proofs around the origin are more easily done (fewer terms!), and the translational invariance of our framework (to be introduced later) will then make them valid anywhere.

- The purely Euclidean hyperplanes $\mathbf{e}_i$ are the *coordinate hyperplanes*, perpendicular to the corresponding coordinate directions. These have no special geometric significance, but they will permit us to establish relationships with more classical coordinate-based representations.

- The purely non-Euclidean (hyper)plane $n = -\delta e_0$ can be seen as a (hyper)plane in which the distance to the origin outweighs any directional aspects. It is orthogonal to all Euclidean (hyper)planes: $e \cdot (\mathbf{n} - \delta e_0) = 0$; and it cannot be normalized, since $e_0 \cdot e_0 = 0$.

  As we will see, in 3D this plane e_0 contains all vanishing points, i.e., the points in which parallel lines meet. The vanishing points will act as pure directions in Euclidean space.

  In projective geometry, e_0 is called the *ideal plane*, or the *improper plane*. Since it is made up of vanishing points, we propose calling it the *vanishing plane* (in attempt to make it sound more useful and less obscure). You can imagine it as the 'celestial sphere' at infinity, with the vanishing points as its stars.

- You have probably noticed that this is all similar to the usual homogeneous coordinates, where a point of the form $[\mathbf{d}^\top, 0]^\top$ is treated as a direction, and $[\mathbf{0}^\top, \delta]^\top$ as a (hyper)plane at infinity. As we will see, the novelty of PGA lies in the *combination* of the planar elements, rather than in their *representation.*

In almost all of the remainder of this text, we revert to the 3D terminology of planes, lines and points for convenience of discussion. But we will occasionally lift our results to d dimensions (with their hyperplanes) to display the general results for future reference, and sometimes descend to 2D (whose hyperplanes are lines) when the results are pleasantly simple there, or differ in sign (for good reasons!).

2.2 Intersecting Planes: 3D Lines as 2-blades

The *outer product* of two planes in 3D represents the line they have in common. The basic reason behind this is the identity (e.g., from Chapter 3 of [2])

$$x \cdot (m \wedge n) = (x \cdot m)\, n - (x \cdot n)\, m, \tag{7}$$

which implies that a dot product with a (non-zero) 2-blade[3] is zero if and only if the dot product with each of the constituents is zero.

So if we can set up our algebra such that '$x \cdot n = 0$' means that '*a point represented by x is on a plane represented by n*', then the algebraic inheritance relationships are just right to make the 2-blade $m \wedge n$ be the representative of their intersection, i.e., their common line. There is not even a need to spell out how we do this explicitly (by specifying precisely how we represent points) for this to be true. The inheritance property eq.(7) of the outer product is sufficient justification. Therefore we dodge this issue of point representation for now; we get back to it later in section 2.5. Having planes is sufficient to define lines (and even points, as we will see).

[3]A k-blade is an element that can be written as the outer product of k vectors; it represents a k-dimensional oriented and weighted subspace of the homogeneous representational space.

When we use the outer product as our `meet` operation, we should be aware of its somewhat unusual nature. The `meet` is a piecewise-linear algebraic abstraction of the intersection operation, see Chapter 4 of [2]. Since it must change sign as planes move from having a positive angle to a negative angle, the `meet` of coincident planes is zero. It is therefore *not* a blade representation of the actual intersection of the point sets (whereas the meet of a plane with itself is zero, the intersection would have been the plane itself). But the linearity properties of the `meet` will make it much easier to incorporate the essence of intersection into the algebra. When the `meet` is zero, there is a degeneracy among its arguments, which you can resolve in a factored-out subspace (see Chapter 5 of [2]).

Since we plan to use the `meet` of planes $p_1 \wedge p_2$ as our line representation, let us look at it in more detail, to confirm that it contains the relevant characterizations of the line, in terms of direction and position.

meet line of planes p_1 and p_2 :

$$\begin{aligned} L &= p_1 \wedge p_2 \\ &= (\mathbf{n}_1 - \delta_1 e_0) \wedge (\mathbf{n}_2 - \delta_2 e_0) \\ &= \mathbf{n}_1 \wedge \mathbf{n}_2 - e_0 \wedge (\delta_1 \mathbf{n}_2 - \delta_2 \mathbf{n}_1) \\ &= \mathbf{n}_1 \wedge \mathbf{n}_2 - e_0 \wedge \big((\delta_1 \mathbf{n}_2 - \delta_2 \mathbf{n}_1)/(\mathbf{n}_1 \wedge \mathbf{n}_2)\big)\,(\mathbf{n}_1 \wedge \mathbf{n}_2) \\ &= \mathbf{n}_1 \wedge \mathbf{n}_2 - e_0 \mathbf{d}\,(\mathbf{n}_1 \wedge \mathbf{n}_2) \\ &= (1 - e_0\,\mathbf{d})\,(\mathbf{n}_1 \wedge \mathbf{n}_2), \end{aligned} \tag{8}$$

with $\mathbf{d} = (\delta_1 \mathbf{n}_2 - \delta_2 \mathbf{n}_1)/(\mathbf{n}_1 \wedge \mathbf{n}_2)$ the support vector of the resulting line, pointing orthogonally to a specific point on it from the origin.[4]

The line representation consists of two terms.

- *Directional (Tangential):* The first term $\mathbf{n}_1 \wedge \mathbf{n}_2$ is purely Euclidean, it describes the direction of the line by specifying a Euclidean plane $\mathbf{n}_1 \wedge \mathbf{n}_2$ orthogonal to its vector direction. (We will later have reason to prefer referring to this part as the *tangent aspect* of the line.)

- *Positional:* The second term $-e_0\,\mathbf{d}\,(\mathbf{n}_1 \wedge \mathbf{n}_2) = -e_0\,\big(\mathbf{d} \cdot (\mathbf{n}_1 \wedge \mathbf{n}_2)\big)$ implicitly contains the positional aspect of the line: it denotes where

[4]This particular form of $\mathbf{d}$ is related to *reciprocal frames* as $\mathbf{d} = \delta_1 \mathbf{n}_1^r + \delta_2 \mathbf{n}_2^r$. We provide a structural exercise 12.2:1, to play with this handy technique (which was explained in Chapter 5 in [2]).

the line is in space. It needs to have the tangential term $\mathbf{n}_1 \wedge \mathbf{n}_2$ as a factor, for L to be a line (if not, it is a *screw*, see Section 5.6).

Such a directional term and positional term occur in all our elements representing offset subspaces in PGA (though they may be zero). In hindsight, they also occurred in the plane $\mathbf{n} - \delta e_0$, where $\mathbf{n}$ was the (normal) direction and δ indicated the position.

Above, the positional aspect is by the *orthogonal support vector* $\mathbf{d}$, permitting the final multiplicative factorization in terms of the geometric product. But any other location $\mathbf{p}$ of a point on the line could have been used as well.

2.3 Choosing an Orientation Convention

We have to choose how we interpret the orientation of this element representing a line, i.e., we should assign a Euclidean direction vector $\mathbf{u}$ to the 2-blade direction element $\mathbf{n}_1 \wedge \mathbf{n}_2$. At this point, there is no fundamental reason to choose one orientation over the other, this is just choosing an interface between the algebraic relations and how we choose to interpret them geometrically. For compatibility it seems reasonable to use the analogy with the classical 3D cross product (in a right-handed space), and set $\mathbf{u} = \mathbf{n}_1 \times \mathbf{n}_2$; so the `meet` of planes with normals $\mathbf{e}_1$ and $\mathbf{e}_2$ (in that order!) would correspond to a line with direction vector $\mathbf{u} = \mathbf{e}_3$. We will see later that this is indeed a good choice, which ties in well with the difference of points defining a direction.

With this convention, we have

$$\mathbf{u} \equiv \mathbf{n}_1 \times \mathbf{n}_2 = (\mathbf{n}_1 \wedge \mathbf{n}_2)^\star = (\mathbf{n}_1 \wedge \mathbf{n}_2)\, \mathbf{I}_3^{-1}, \tag{9}$$

where $\mathbf{I}_3 = \mathbf{e}_1 \mathbf{e}_2 \mathbf{e}_3$ is the Euclidean volume blade (the pseudoscalar). A general point on the line is $\mathbf{p} = \mathbf{d} + \lambda\, \mathbf{u}$. We can thus represent the line when we have been given its direction $\mathbf{u}$ and some location $\mathbf{p}$ (not necessarily $\mathbf{d}$) as[5]

$$\begin{aligned} L &= \mathbf{u}\mathbf{I}_3 - e_0\, \mathbf{p} \cdot (\mathbf{u}\mathbf{I}_3) \\ &= \mathbf{u}\mathbf{I}_3 - e_0\, (\mathbf{p} \wedge \mathbf{u})\mathbf{I}_3 \\ &= \big(\mathbf{u} - e_0\, (\mathbf{p} \wedge \mathbf{u})\big)\, \mathbf{I}_3. \end{aligned} \tag{10}$$

[5]Though later, in Section 3.3.2, we will do this more naturally ‘in-algebra’ using the `join`, and obtain $\mathbf{u} \cdot P = \mathbf{u} \cdot (O + \mathbf{p}\mathcal{I})$.

In 3D, you recognize in its components on the bivector basis $\{\mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}\}$ the coefficients of the traditional direction vector $\mathbf{u}$; and the components on the bivector basis $\{e_0\mathbf{e}_1, e_0\mathbf{e}_2, e_0\mathbf{e}_3\}$ are the coefficients of the traditional moment vector $(\mathbf{p} \wedge \mathbf{u})^\star = \mathbf{p} \times \mathbf{u}$. Together, they corresponds to components of the traditional Plücker representation of lines, which would be $[-\mathbf{u}, \mathbf{p} \times \mathbf{u}]$ [6]

2.4 Vanishing Lines: the `meet` of Parallel Planes

Effectively, the outer product $p_1 \wedge p_2$ of two plane vectors p_1 and p_2 acts as the `meet` operation, providing intersections through the inheritance property eq.(7). That even works for parallel planes, though the resulting line is unusual: it contains no locational aspects, but it encodes the common normal vector of the two planes.

line common to parallel planes p_1 and p_2:

$$\begin{aligned} L &= p_1 \wedge p_2 \\ &= (\mathbf{n} - \delta_1 e_0) \wedge (\mathbf{n} - \delta_2 e_0) \\ &= (\delta_2 - \delta_1)\, e_0 \wedge \mathbf{n}. \end{aligned} \tag{11}$$

The final rewriting shows that the same result can be viewed as the outer product (intersection) of the weighted origin plane $(\delta_2 - \delta_1)\,\mathbf{n}$ and the vanishing plane $-e_0$.

Any plane $p = \mathbf{n} - \delta e_0$ with the same normal $\mathbf{n}$, when intersected with the special plane e_0, results in an element proportional to $e_0 \wedge \mathbf{n} = e_0\,\mathbf{n}$. Thus ideal 2-blades represent purely directional aspects of geometry; the positional part does not feature. In 2D PGA $e_0 \wedge \mathbf{n}$ would represent the *vanishing point* of parallel lines with normal direction $\mathbf{n}$. So for 3D, we suggest for elements of the form $e_0\,\mathbf{n}$ the term *vanishing line* common to all parallel planes with normal $\mathbf{n}$.

[6] This is the Shoemake convention that a line from $\mathbf{p}$ to $\mathbf{q}$ has Plücker coordinates $[\mathbf{p}-\mathbf{q}, \mathbf{p}\times\mathbf{q}]$. Plücker coordinates in a geometric algebra context are treated in Chapter 12 of [2]. The 3D PGA representation is equivalent to those six Plücker numbers, but on a geometrically meaningful basis that relates it to planar intersection. Also, the PGA representation naturally extends to lines in other dimensions besides 3.

2.5 3D Points as 3-blades

By the same reasoning as when we interpreted the `meet` of two planes as a line due to eq.(7), the `meet` of three general planes in 3D should define their *common intersection point.* In GA, we have the identity

$$x \cdot (p_1 \wedge p_2 \wedge p_3) = (x \cdot p_1)\, p_2 \wedge p_3 + (x \cdot p_2)\, p_3 \wedge p_1 + (x \cdot p_3)\, p_1 \wedge p_2. \quad (12)$$

Thus for linearly independent plane vectors p_i, a point x on all three planes p_i (so that $x \cdot p_i = 0$) has zero dot product with the trivector $p_1 \wedge p_2 \wedge p_3$, the `meet` of the planes. Therefore elements that behave like Euclidean points are actually contained in the algebra PGA that represents planes as vectors: they are simply the 3-blades, the outer product of three vectors. Yes you read that right: in 3D PGA, *points are trivectors.*

We would of course expect the Euclidean coordinates (and even the homogeneous coordinates) of a point to show up as the coefficients of the 3-blade that represents it. That parametrization is most easily seen when we construct a point at location $\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$ by intersecting three orthogonal planes p_i parallel to the coordinate planes $\mathbf{e}_i$ at the properly signed distances $x_i = \mathbf{x} \cdot \mathbf{e}_i$.

$$\begin{aligned} X &= p_1 \wedge p_2 \wedge p_3 \\ &= (\mathbf{e}_1 - x_1 e_0) \wedge (\mathbf{e}_2 - x_2 e_0) \wedge (\mathbf{e}_3 - x_3 e_0) \\ &= \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 - e_0 \wedge (x_1\, \mathbf{e}_2 \wedge \mathbf{e}_3 + x_2\, \mathbf{e}_3 \wedge \mathbf{e}_1 + x_3\, \mathbf{e}_1 \wedge \mathbf{e}_2) \\ &= \mathbf{I}_3 - e_0 \wedge (\mathbf{x} \cdot \mathbf{I}_3) \\ &= (1 - e_0\, \mathbf{x})\, \mathbf{I}_3. \end{aligned} \quad (13)$$

If the rewriting is a bit quick for you, consult exercise 12.2:2. In the second line, we recognize the coefficients $(1, x_1, x_2, x_3)$ of a (unit weight) point in homogeneous coordinates, but on an unusual basis $\{\mathbf{e}_{123}, e_0\mathbf{e}_{32}, e_0\mathbf{e}_{13}, e_0\mathbf{e}_{21}\}$. It should be clear that the unusual basis need not imply computational overhead in coordinate-based computations. It is in a sense merely algebraic bookkeeping of geometrical relationships with the other primitives (lines and planes), allowing compact structural computations between them. (And that administrative functionality is indeed a perfectly acceptable and pragmatic computational view of geometric algebra in general.)

The 'square' of a point of this form, the scalar $X\widetilde{X}$ equals[7]

$$X\widetilde{X} = (1-e_0\,\mathbf{x})\,\mathbf{I}_3\,\widetilde{\mathbf{I}}_3\,(1+e_0\,\mathbf{x}) = (1-e_0\,\mathbf{x})\,(1+e_0\,\mathbf{x}) = 1-e_0\,\mathbf{x}+e_0\,\mathbf{x}+0 = 1. \tag{14}$$

(We will later see that the translation-invariance of PGA elements permits us to take the simpler approach of only computing at the origin: $X\widetilde{X} = \mathbf{I}_3\,\widetilde{\mathbf{I}}_3 = 1$. Done.) The point X in eq.(13) is thus of unit weight. If we had intersected non-orthogonal unit planes, we would have obtained a 'weaker' intersection point, of less weight.

In d dimensions, a similar result to eq.(13) ensues (which you are asked to prove in the exercise 12.2:3):

$$X = (1-e_0\mathbf{x})\,\mathbf{I}_d = \mathbf{I}_d + \mathbf{x}e_0\mathbf{I}_d \equiv O + \mathbf{x}\mathcal{I},$$

expressing the result finally in terms of the origin $O \equiv \mathbf{I}_d$ and the pseudoscalar $\mathcal{I} \equiv e_0\mathbf{I}_d$. Clearly the coordinates of the usual Euclidean 1-vector $\mathbf{x}$ relative to O determine coordinates of the PGA d-blade X representing it, and vice versa. This d-D form also suggests why we used $X\,\widetilde{X}$ as the 'square' of a point, rather than X^2: the norm is then independent of d.

2.6 Vanishing Points

When we take the `meet` of a line $L = p_1 \wedge p_2$ with the special plane $-e_0$ at infinity, we obtain elements of the form

$$\begin{aligned} V &= -e_0 \wedge p_1 \wedge p_2 \\ &= -e_0 \wedge (\mathbf{n}_1 - e_0\delta_1) \wedge (\mathbf{n}_2 - \delta_2 e_0) \\ &= -e_0 \wedge (\mathbf{n}_1 \wedge \mathbf{n}_2). \end{aligned} \tag{15}$$

Such a point at infinity is common to all lines with directional aspect $\mathbf{n}_1 \wedge \mathbf{n}_2$, independent of their location: it is their *vanishing point*. If you would draw that set of lines in perspective, V would be an actual point location in the image. The projective nature of PGA's representation of space means that such vanishing points are also just elements of the algebra. But contrary to common use in projective geometry, we will make a distinction between V and $-V$; for we do want our lines to be oriented – as light rays are.

[7]The reverse $\widetilde{X}$ of an element X is obtained by reversing all its geometric product or outer product factors. For an element of grade X, that gives a sign of $(-1)^{x(x-1)/2}$.

It takes a bit of getting used to the dual indication of directions here. If you want the vanishing point $V_{\mathbf{u}}$ lying in a Euclidean direction represented by vector $\mathbf{u}$, then you need to set $V = -e_0\mathbf{u}\mathbf{I}_3 = \mathbf{u}(e_0\mathbf{I}_3) = \mathbf{u}\mathcal{I}$, in accordance with eq.(9).

The vanishing points can also be seen as the difference between two normalized points

$$Q-P = (1-e_0\mathbf{q})\,\mathbf{I}_3-(1-e_0\mathbf{p})\,\mathbf{I}_3 = -e_0(\mathbf{q}-\mathbf{p})\,\mathbf{I}_3 = -e_0\mathbf{u}\mathbf{I}_3 = \mathbf{u}(e_0\mathbf{I}_3), \quad (16)$$

with $\mathbf{u} \equiv \mathbf{q}-\mathbf{p}$ the regularly used direction vector of the line from P to Q. Even though points are represented in PGA as 3-blades, such relationships are therefore completely similar to homogeneous coordinates, where the difference between two points is a direction vector.

2.7 Four Planes Speak Volumes About Oriented Distances

We can now return to the equation that determines whether a point X lies on a plane p. With the point represented as a trivector describing the intersection of three planes, this simply is the algebraic statement

$$p \wedge X = 0. \quad (17)$$

Let us verify this in computational detail to convince you.

$$\begin{aligned} 0 &= (\mathbf{n}-\delta e_0)\wedge(\mathbf{I}_3 - e_0\,\mathbf{x}\,\mathbf{I}_3) \\ &= -\delta e_0\mathbf{I}_3 + e_0\big(\mathbf{n}\wedge(\mathbf{x}\,\mathbf{I}_3)\big) \\ &= (\mathbf{n}\cdot\mathbf{x}-\delta)\,e_0\,\mathbf{I}_3, \end{aligned} \quad (18)$$

so indeed this retrieves the expected $\mathbf{n}\cdot\mathbf{x} = \delta$ for a point with location $\mathbf{x}$ to be on the line.

Contrast this with the usual homogeneous coordinate approach which we recalled in our introduction (Section 2): '*when a homogeneous point x lies in a homogeneous plane n, then $x \cdot n = 0$*'. As we saw in the introductory Section 2, if we want to distinguish points and planes as vectors, we have to put them in different spaces (with ϵ_o and ϵ_o^r as extra basis vectors, and having $\epsilon_o^r \cdot \epsilon_o = 1$), or treating one of them as a vector and the other as a covector. PGA is thus simply more frugal and algebraically (c)leaner, in its setting up of only one $(d+1)$-dimensional representational space, and then using its outer product. (And for purists, the non-metric outer product $\wedge$ is more appropriate to encode an incidence relationship than the metric dot product anyway.)

As the above computation shows, for a general plane p relative to the point X, the 4-blade $p \wedge X$ is not zero, but equal to a multiple $\mathbf{n} \cdot \mathbf{x} - \delta$ of the PGA pseudoscalar $\mathcal{I}_3$ defined as

$$\mathcal{I}_3 \equiv e_0 \, \mathbf{I}_3 = e_0 \wedge \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3.$$

(In d-dimensional PGA, with its $d+1$-dimensional representation space, we will denote the pseudoscalar $(d+1)$-blade by $\mathcal{I}_d$.) That multiple is actually the orthogonal oriented distance between point and plane (if both are normalized), so we can use $p \wedge X$ to compute the distance.

2.8 Linear Combinations of Primitives

In classical linear algebra, we are used to having the vector equation of a line, to generate points along it as $\mathbf{x} = \mathbf{q} + \lambda \mathbf{u}$. That is a convenient way to draw a linear orbit. Such capabilities are not lost in PGA, but they take a slightly different form.

- **Sliding a point along a line:**
  A point is a trivector, and a direction is a vanishing point. If one has the line 2-blade L available, then its vanishing point is $-e_0 L$; alternatively a vanishing point can be constructed from a direction vector $\mathbf{u}$ as $\mathbf{u}\mathcal{I}_3$.

  One can generate a starting point lying on the line L through meeting the line by a plane p as $Q = p \wedge L$. The 'trivector equation of a line' is then
  $$X = Q - \lambda \, (e_0 \, L). \tag{19}$$
  This equation can also generate trivector points parallel to L, starting from any point Q (not necessarily on the line), as in standard LA or homogeneous coordinates.

- **Sliding a line along a plane:**
  In a similar way, you can slide a line $L = p \wedge q$ linearly along a plane p containing it
  $$M = L - \mu \, (e_0 \, p). \tag{20}$$

But beware: this only works if $e_0 \wedge p \wedge L = 0$. Geometrically, this condition means that the line should be parallel to the plane p used to determine the sliding. This constraint is an instance of the subtlety that 3D lines (such as L and $e_0 p$ in this case) cannot be added arbitrarily to produce another line; we will see why in Section 5.6.

- **Sliding a plane along a volume:**
  We can also slide a plane, which is rather obvious when you realize that in $p = \mathbf{n} - \delta e_0$, the δe_0 adds the displacement from the origin. But in view of the above, this follows the general pattern: we can form a family of parallel planes as

  $$x = p - \nu\,(e_0\,1). \tag{21}$$

  Geometrically, we could view this as adding a bit of the quantity $e_0 1$, representing the (rather trivial) positional aspect of the 3D unit volume element in PGA.

Other linear combinations, already known from common practice in homogeneous coordinates, continue to be sensible constructions in PGA. But in the case of lines, some care is required!

- **Centroids of points:**
  The unit-weight point between two normalized points P and Q is $(P + Q)/2$. The *centroid* of two or more non-normalized points is their sum: $C = \sum_i P_i$; it is weighted with the total weight of the individual points. The centroid location is obtained through normalization (dividing by the coefficient of $\mathbf{I}_3$, which is $C^\star \equiv C \cdot \mathbf{I}_3^{-1} = C \cdot \widetilde{\mathbf{I}}_3$).

  Obviously, as in the usual way, you can use *affine combinations* of points to parametrize the points on a line segment, or in a triangle or tetrahedron. In the 4-dimensional PGA of 3D space, trivectors behave just like vectors in their linear properties.

- **Bisector of two planes:**
  The plane that bisects two given normalized oriented planes p_1 and p_2 is $b = p_1 + p_2$. If the planes intersect in a finite line, the resulting plane is indeed a bisection of the angular difference. If the planes have a vanishing (ideal) line in common, the resulting plane b is halfway in distance between the two: a ‘translational bisector’.

Note that which of the two possible bisectors you get depends on the orientation of the plane: what is generated is halfway between the 'smallest move' to align the two planes in orientation or location. So the other bisector is just the difference $p_1 + (-p_2) = p_1 - p_2$ (with a suitably chosen orientation; you might want $p_2 - p_1$ instead). Exercise: what happens with oppositely oriented normalized parallel planes?!

- **Bisector of two lines:** If two normalized lines L_1 and L_2 are in the same plane, then $M = L_1 + L_2$ is their bisector. Just as for planes, this unifies bisection in angle and in distance (if the lines are parallel), and the orientation of the lines determines which of the two possible bisectors results.

  But when the lines are not coplanar, the result is not a line: algebraically, the 2-blades add up to a bivector that cannot be written as a 2-blade. This is a screw, and we will meet it later in Section 5.6. For now, restrict your addition of lines to 2D subspaces (where it is convenient for handling edges, bisectors and the like within a spatial polygon).

Steven made an interesting 'Wedge Game' in 2D PGA, in which your assignment is to make increasingly more involved constructions in planar geometry. It is most suitable for tablets or phones (since pinching and spreading of elements is then available to denote their combinations) and may be found at `https://enkimute.github.io/ganja.js/examples/example_game_wedge.html`. The game uses some of the constructions above, but you should only play it after also absorbing the `join` operation in the next Section 3 (which enables you to construct the line connecting two points, and related constructions).

2.9 Example: Solving Linear Equations

In standard linear algebra, a system of linear equations

$$\left\{ \begin{array}{lcl} a_{11}x + a_{12}y + a_{13}z + a_{10} & = & 0 \\ a_{21}x + a_{22}y + a_{23}z + a_{20} & = & 0 \\ a_{31}x + a_{32}y + a_{33}z + a_{30} & = & 0 \end{array} \right. \tag{22}$$

is solved by determinants (by Cramer's rule), matrix inversion, or numerical methods. Our `meet` operation allows for a directly geometrical perspective.

Each of the equations represents a plane, and the solution is therefore required to lie on the `meet` of those planes. Let p_i be the plane for row i, so equal to the PGA vector

$$p_i = a_{i1}\,\mathbf{e}_1 + a_{i2}\,\mathbf{e}_2 + a_{i3}\,\mathbf{e}_3 + a_{i0}\,e_0. \tag{23}$$

Then the solution of the linear system for $\mathbf{x} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$ is fully represented as the point X computed by

$$X = p_1 \wedge p_2 \wedge p_3. \tag{24}$$

Computing this quantity allows the coordinates of the solution $\mathbf{x}$ to be read off immediately, as we saw when discussing eq.(13).

This form of the solution of a set of linear equations works in two directions: it is a compact way to express or compute the solution; and, conversely, numerical methods developed for solving equations can lead to efficient implementations of the outer product. This is beginning to be investigated, both for the exact case, and for overdetermined equations and their least square solutions. In one of the early papers [7], this simple method was compared to Cholesky decomposition. In 3D, the outer product method takes 60% more operations, but since it can be implemented in parallel using SIMD/GPU architecture, it is still a contender.

Of course vanishing points are also legitimate solutions to linear systems. And when the equations are degenerate, the outcome of the outer product is zero. In that case the point as a solution is underdetermined, the actual solution may be a line, just as if there were only two equations. In PGA an expression like $p_1 \wedge p_2$ specifies that line solution compactly, and as we have seen then determines the usual line parameters. In the context of linear equations, this 2-blade solution can be converted to the more classical form of (a specific solution plus an amount of kernel). We hope to report on this soon [8].

2.10 Summary

PGA $\mathbb{R}_{d,0,1}$ is an algebra with d Euclidean dimension, and a null dimension (which we denoted by e_0). We use it to encode planes in d-dimensional Euclidean space as vectors in our representational space.

We introduced the first GA product in PGA: the `meet`, the linearized oriented intersection operation encoded as the outer product $\wedge$. We showed in 3D how that this product gives us the representation of lines as 2-blades and points as trivectors, and identified the correspondence to their classical parametrizations.

When planes are a `meet` with the null plane e_0, we obtain the 'ideal' elements (which we called 'vanishing points' etc.): the intersection line of parallel planes, or the intersection point of parallel lines (which acts as a 1D direction element). These are all a natural part of this algebra of planes.

3 The `join` in PGA

The `meet` intersects planes to produce reduced elements like lines and points. We also want to construct elements by merging elements, such as permitting two points to determine a line. That operation is called the `join`. It is a union, linearized in similar fashion to how the `meet` linearize intersection. In fact, the two operations are dual to each other.

3.1 The Join as Dual to the Meet

The `join` (Grassmann's *regressive product*) was designed to be dual to the `meet` (see e.g., Chapter 5 in [2]). In PGA, where $\wedge$ is an intersection (and reminds us of the set intersection $\cap$), it is customary to denote the `join` by a $\vee$ (which handily reminds us of set union $\cup$).

The `join` was designed carefully by Grassmann to satisfy the *Common Factor Axiom* (CFA), which gives a consistent relationship to the `meet` (see [9]): [8]

$$(\mathbf{A} \wedge \mathbf{B}) \vee (\mathbf{B} \wedge \mathbf{C}) = \mathbf{B} \vee (\mathbf{A} \wedge \mathbf{B} \wedge \mathbf{C}). \tag{25}$$

To make this equation valid without extraneous grade-dependent signs or weights (as we would desire for our oriented geometry), we should define the `join` in a fully Euclidean geometric algebra as

$$(\mathbf{A} \vee \mathbf{B})^\star = \mathbf{B}^\star \wedge \mathbf{A}^\star. \tag{26}$$

We prove this consequence of the CFA in exercise 12.2:5. If we are in an n-dimensional algebra where undualization is possible, it would follow from eq.(26) that

$$\mathbf{A} \vee \mathbf{B} = \big(\mathbf{B}^\star \wedge \mathbf{A}^\star\big)^{-\star}. \tag{27}$$

Usually (and always in [2]), dualization here is division by the invertible pseudoscalar $\mathbf{I}_n$, so undualization of eq.(26) is multiplication by it. Using the contraction $\rfloor$, which is adjoint to the outer product, we can even rewrite this to $\mathbf{B}^\star \rfloor \mathbf{A}$, so $\vee$ is like 'do $^\star\rfloor$' on the swapped arguments. This was how we treated it in [2] in general (though since we were then focusing on a non-dual

[8]In the version of the present text lower than 2.0, we used $(\mathbf{B} \wedge \mathbf{C}) \vee (\mathbf{A} \wedge \mathbf{B}) = (\mathbf{A} \wedge \mathbf{B} \wedge \mathbf{C}) \vee \mathbf{B}$, to make the `join` argument order easy to remember. Our change of sign of e to $e_0 = e$ now has the same logic take the opposite form for the `join`, to still have $P \vee Q$ be the line from P to Q.

representation, this actually corresponded to the `meet` rather than the `join`.) But in the present text, we will stay at the more obviously symmetric level of duality between `meet` and `join` involving the undualized outer product of duals.

Thus defined, the `join` of two elements in a space of dimension n is of grade

$$\begin{aligned} \text{grade}(\mathbf{A} \vee \mathbf{B}) &= \text{grade}(\mathbf{A}) + \text{grade}(\mathbf{B}) - n \quad \textit{(if non-negative)} \\ &= \text{grade}(\mathbf{A} \wedge \mathbf{B}) - n \qquad\qquad \textit{(if non-negative).} \end{aligned} \tag{28}$$

If the resulting grade is negative, the join is zero – the outer product will have made it so.

The `join` is clearly *linear* and *associative*, but may obtain an extra sign when its arguments are swapped.

$$\mathbf{B} \vee \mathbf{A} = (-1)^{(n-a)(n-b)} \mathbf{A} \vee \mathbf{B}, \tag{29}$$

with $a = \text{grade}(\mathbf{A})$ and $b = \text{grade}(\mathbf{B})$. This minus sign equals the parity of n when both $\mathbf{A}$ and $\mathbf{B}$ are even, and of $n+1$ when both $\mathbf{A}$ and $\mathbf{B}$ are odd, and there is no sign in the other cases.[9]

By invoking the duality, which implicitly involves orthogonality, the `join` requires a metric to be defined in this manner. Mathematically inclined authors like Browne [9] develop the `join` non-metrically, as a counteroperation to the `meet`, and this works. They then later show more compact expressions when they allow themselves to have a metric. All very beautiful, of course, but for our goal of representing Euclidean motions, we decided to use the metric approach immediately.[10]

3.2 Constructing the PGA Join

In PGA we cannot quite convert eq.(26) to an explicit definition of the form of eq.(27), since we cannot undualize so simply. For the pseudoscalar $\mathcal{I}$ of PGA is 'null'; the customary $\mathcal{I}^{-1} = \widetilde{\mathcal{I}}/(\mathcal{I}\widetilde{\mathcal{I}})$ just does not compute.

[9] Note that if you would change the pseudoscalar of your space for some reason, a swap of its sign leads to a sign swap of the regressive product, since the pseudoscalar occurs an odd number of times in the definition.

[10] Leo has spelled out the correspondence of the present text with Browne's notation in a separate appendix [10].

So we momentarily transcend PGA; we realize that PGA is 'unbalanced' in this sense, but that it can be balanced in a larger space with an additional basis vector e_0^r reciprocal to e_0 (so that $e_0^r \cdot e_0 = 1$).[11] Then the pseudoscalar $\mathcal{I}$ has a reciprocal $\mathcal{I}^r$ such that $\mathcal{I}^r\mathcal{I} = 1$, and dualization can be performed. It turns out that the final result of this roundabout way of constructing a `join` for the `meet` is completely expressible in the regular PGA we started with. So we can then take the resulting expression as the intrinsic definition of our `join` in the confidence that we will be consistent. We are especially aiming to getting the signs correct, for we do want a geometric algebra in which the orientations are used in a meaningful manner.

With this in mind, we follow a similar pattern to eq.(27) of defining the regressive product as dual to the outer product in PGA and set

$$A \vee B = \big(B^* \wedge A^*\big)^{-*}. \tag{30}$$

Note that this uses the full metric duality in the extended space we have set up, not merely the Euclidean duality. We use as pseudoscalar $\mathcal{I}$ for PGA: $\mathcal{I} = e_0 \wedge \mathbf{I}_d$. Dualization can then be performed only in the combined space that balances the algebra, by means of the reciprocal pseudoscalar $\mathcal{I}^r = \widetilde{\mathbf{I}}_d \wedge e_0^r$. Undualization involves $\mathcal{I}$. It will be natural to reduce results by means of duality in the Euclidean subalgebra of PGA. To distinguish the two dualities, we will denote full PGA duality by $(\ )^*$ and Euclidean duality by $(\ )^\star$:

$$A^* \equiv A\,\mathcal{I}^r \quad \text{whereas} \quad \mathbf{A}^\star \equiv \mathbf{A}\,\mathbf{I}_d^{-1} = \mathbf{A}\widetilde{\mathbf{I}}_d. \tag{31}$$

The context usually clarifies which one is required, so we made the difference subtle enough not to draw too much attention to itself.

Our k-vector elements in PGA $\mathbb{R}_{d,0,1}$ will always be of the form

$$A = \mathbf{T}_A + e_0\,\mathbf{P}_A, \tag{32}$$

with $\mathbf{T}_A$ and $\mathbf{P}_A$ from a d-D Euclidean GA $\mathbb{R}_d$, with its pseudoscalar $\mathbf{I}_d$ and the usual Euclidean dualization (relative to that pseudoscalar) of $\mathbf{T}_A{}^\star \equiv \mathbf{T}_A\,\widetilde{\mathbf{I}}_d$ etc. We call $\mathbf{T}$ the *tangent space aspect*, and $\mathbf{P}$ the *positional aspect* (hence the mnemonics).

Let us first compute the (full) dual of a PGA blade; it is an element of the dual space to PGA.

$$A^* = A \cdot \mathcal{I}^r = (\mathbf{T}_A + e_0\,\mathbf{P}_A)\cdot(\widetilde{\mathbf{I}}_d\,e_0^r) = \mathbf{T}_A^\star\,e_0^r + \widehat{\mathbf{P}_A^\star}, \tag{33}$$

[11]If you want to compute with this reciprocal e_0^r in GAviewer, where e_0 is represented as `-ni`, then define '`er = no;` '.

Here $\widehat{X} = (-1)^{\text{grade}(X)}\,X$, the grade involution, which accounts for the sign swaps involved in moving the Euclidean k-vector $\mathbf{T}_A$ 'to the other side of e_0'. Undualizing an expression of this form then involves dotting with the pseudoscalar $\mathcal{I}_d = e_0\mathbf{I}_d$ (and reverts to the original blade).

We compute for the `join`

$$\begin{aligned}
A \vee B &= \big(\ B^* \wedge A^*\ \big)^{-*} \\
&= \Big((\mathbf{T}_B^\star\, e_0^r + \widehat{\mathbf{P}_B^\star}) \wedge (\mathbf{T}_A^\star\, e_0^r + \widehat{\mathbf{P}_A^\star})\Big)^{-*} \\
&= \Big((\mathbf{T}_B^\star\ \wedge \mathbf{P}_A^\star)\, e_0^r + (\widehat{\mathbf{P}_B^\star} \wedge \mathbf{T}_A^\star)\, e_0^r + \widehat{\mathbf{P}_B^\star} \wedge \widehat{\mathbf{P}_A^\star}\Big)^{-*} \\
&= (\mathbf{T}_B^\star\ \wedge \mathbf{P}_A^\star)\,\mathbf{I}_d + (\widehat{\mathbf{P}_B^\star} \wedge \mathbf{T}_A^\star)\,\mathbf{I}_d + e_0\,(\mathbf{P}_B{}^\star \wedge \mathbf{P}_A{}^\star)\,\mathbf{I}_d \qquad (34) \\
&= (\mathbf{T}_B^\star\ \wedge \mathbf{P}_A^\star)^{-\star} + (\widehat{\mathbf{P}_B^\star} \wedge \mathbf{T}_A^\star)^{-\star} + e_0\,(\mathbf{P}_B{}^\star \wedge \mathbf{P}_A{}^\star)^{-\star} \\
&= \mathbf{T}_A \vee \mathbf{P}_B + (-1)^d\ \widehat{\mathbf{P}_A} \vee \mathbf{T}_B + e_0\,(\mathbf{P}_A \vee \mathbf{P}_B). \qquad (35)
\end{aligned}$$

Halfway this derivation, you see how this reduces the join in PGA to a combination of operations and duals in the Euclidean subalgebra; in the final line we even rewrote that fully in terms of the Euclidean `join` (cheekily overloading the $\vee$ notation since the PGA `join` of purely Euclidean elements is zero anyway).

The resulting expression can be computed completely within the algebra $\mathbb{R}_{d,0,1} = \mathbb{R}_d \oplus e_0$, without running into any problems with its non-invertible pseudoscalar $\mathcal{I}_d \equiv e_0\mathbf{I}_d$. Since it is completely computable in $\mathbb{R}_{d,0,1}$, it is part of its algebra PGA. We have found our `join` operation!

So indeed, PGA provides a fully geometrically significant algebra of flats in a space of one more dimension than the Euclidean dimension d. Having said that, the defining expressions for the `join` in PGA do look somewhat contrived – but at least you understand where they came from.

Later, in Section 9.2 when we have the Hodge dual (a specially constructed form of dualization which exists entirely with the algebra and denoted by a prefix $\star$ symbol), we can rewrite and compute the `join` without splitting it into a sum of terms, as

$$A \vee B = \star^{-1}\big(\star A \wedge \star B\big). \qquad (36)$$

The swapping of arguments relative to the other dualization above exactly compensates for the signs resulting from the Hodge dualization, as we will show in Section 9.2.

We will see that the Hodge duals involve merely the selection of certain components on a dual basis, with some appropriate sign changes. Then eq.(36) shows that the expensive part to compute in a `join` operation is really only an outer product – which is a sparse product anyway due to its property of 'repeated factors give zero'. You may view `join` and `meet` as comparably elementary operations of PGA.

3.3 Examples of the Join

3.3.1 A Line as the Join of Two Points

Let us verify the `join` of two point 3-blades in 3D: $P = \mathbf{I}_3 - e_0\,\mathbf{p}\mathbf{I}_3$ and $Q = \mathbf{I}_3 - e_0\,\mathbf{q}\mathbf{I}_3$.

$$\begin{aligned}
P \vee Q &= \Big(\mathbf{I}_3^\star \wedge (-\mathbf{p}\mathbf{I}_3)^\star\Big)^{-\star} + \Big(\widehat{(-\mathbf{q}\mathbf{I}_3)}^\star \wedge \mathbf{I}_3{}^\star\Big)^{-\star} + e_0\,\Big((-\mathbf{q}\mathbf{I}_3)^\star \wedge (-\mathbf{p}\mathbf{I}_3)^\star\Big)^{-\star} \\
&= -(1 \wedge \mathbf{p})\,\mathbf{I}_3 + (\mathbf{q} \wedge 1)\,\mathbf{I}_3 + e_0\,(\mathbf{q} \wedge \mathbf{p})\,\mathbf{I}_3 \\
&= (\mathbf{q} - \mathbf{p})\,\mathbf{I}_3 - e_0\,(\mathbf{p} \wedge \mathbf{q})\,\mathbf{I}_3 \\
&= (\mathbf{q} - \mathbf{p})\,\mathbf{I}_3 - e_0\,\big(\tfrac{1}{2}(\mathbf{p} + \mathbf{q}) \wedge (\mathbf{q} - \mathbf{p})\big)\,\mathbf{I}_3. \qquad (37)
\end{aligned}$$

We see indeed that the tangent aspect is the 2-blade $(\mathbf{q}-\mathbf{p})\,\mathbf{I}_3$ orthogonal to the direction vector of the line. The direction of the line is thus $\mathbf{u} = \mathbf{q}-\mathbf{p}$, so *$P \vee Q$ is the line from P to Q.* Its magnitude is proportional to the oriented distance between the points. Comparing the positional aspect in the second term with eq.(10) shows that the line passes through the centroid $(\mathbf{p}+\mathbf{q})/2$ of the points (as expected!).

3.3.2 Constructing a Line from a Point P and a Direction $\mathbf{u}$

Joining a point $P\mathbf{I}_3 - e_0\mathbf{p}\mathbf{I}_3$ to the vanishing point $V_\mathbf{u} = -e_0\mathbf{u}\mathbf{I}_3$ in vector direction $\mathbf{u}$ gives:

$$\begin{aligned}
P \vee V_\mathbf{u} &= \big(\mathbf{I}_3 - e_0\,\mathbf{p}\mathbf{I}_3\big) \vee (-e_0\mathbf{u}\mathbf{I}_3) \\
&= 0 + \big(\widehat{(-\mathbf{u}\mathbf{I}_3)}^\star \wedge \mathbf{I}_3{}^\star\big)^{-\star} + e_0\,\big((-\mathbf{u}\mathbf{I}_3)^\star \wedge (-\mathbf{p}\mathbf{I}_3)^\star\big)^{-\star} \\
&= (\mathbf{u} \wedge 1)\,\mathbf{I}_3 + e_0\,(\mathbf{u} \wedge \mathbf{p})\,\mathbf{I}_3 \\
&= \mathbf{u}\mathbf{I}_3 - e_0\,(\mathbf{p} \wedge \mathbf{u})\mathbf{I}_3 \\
&= \mathbf{u} \cdot (\mathbf{I}_3 - e_0\mathbf{p}\mathbf{I}_3) \\
&= \mathbf{u} \cdot P. \qquad (38)
\end{aligned}$$

Comparing with eq.(10), the components of this expression are again like those of the Plücker coordinates $[-\mathbf{u}, \mathbf{p}\times\mathbf{u}]$ of this line, on the bases $\{\mathbf{e}_{32}, \mathbf{e}_{13}, \mathbf{e}_{21}\}$ and $\{\mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03}\}$.

3.3.3 Constructing a Plane as the Join of Three Points

Let us `join` a third point R to a line $P \vee Q$ of eq.(37).

$$\begin{aligned} &P \vee Q \vee R = \\ &= \ \Big((\mathbf{q}-\mathbf{p})\,\mathbf{I}_3 - e_0\,(\mathbf{p}\wedge\mathbf{q})\,\mathbf{I}_3\Big) \vee (\mathbf{I}_3 - e_0\,\mathbf{r}\mathbf{I}_3) \\ &= \ \big(1 \wedge (-\mathbf{p}\wedge\mathbf{q})\big)^{-\star} + \big(-\widehat{\mathbf{r}} \wedge (\mathbf{q}-\mathbf{p})\big)^{-\star} + e_0\,\big(\mathbf{r}\wedge\mathbf{p}\wedge\mathbf{q}\big)^{-\star} \\ &= \ -(\mathbf{q}\wedge\mathbf{r} + \mathbf{r}\wedge\mathbf{p} + \mathbf{p}\wedge\mathbf{q})\,\mathbf{I}_3 - e_0\,(\mathbf{p}\wedge\mathbf{q}\wedge\mathbf{r})\,\mathbf{I}_3 \\ &= \ (\mathbf{p}^r + \mathbf{q}^r + \mathbf{r}^r + e_0)\,(\mathbf{p}\wedge\mathbf{q}\wedge\mathbf{r})^\star. \end{aligned} \tag{39}$$

We used the reciprocal frame $\{\mathbf{p}_i^r\}$ of the vectors $\{\mathbf{p}_i\}$ (being $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$) to write this more compactly, see exercise 12.2:1; here $\mathbf{p}_i^r \cdot \mathbf{p}_j = \delta_{ij}$. Since the final factor $(\mathbf{p}\wedge\mathbf{q}\wedge\mathbf{r})^\star$ is a scalar, the result is indeed a vector, and therefore represents a plane. The scalar is twice the area of the triangle formed by the three points.

It is easy to show that the point P lies on the plane $P \vee Q \vee R$:

$$(\mathbf{p}^r+\mathbf{q}^r+\mathbf{r}^r+e_0)\wedge\big((1-e_0\,\mathbf{p})\,\mathbf{I}_3\big) = -e_0\,\big((\mathbf{p}^r+\mathbf{q}^r+\mathbf{r}^r)\cdot\mathbf{p}\big)\,\mathbf{I}_3 + e_0\mathbf{I}_3 = 0, \tag{40}$$

and similarly for the other points. So the `join` indeed computes the correct plane containing all three points.

3.3.4 The Join of Two Lines (and the Importance of Contraction)

When we `join` two lines, which are 2-blades, we should expect a scalar outcome. Let us parametrize the lines by their direction vector $\mathbf{u}$ and moment vector $\mathbf{m} = \mathbf{p}\times\mathbf{u}$ (which has to satisfy $\mathbf{m}\cdot\mathbf{u} = 0$).

Then we compute

$$\begin{aligned} L_1 \vee L_2 &= (\mathbf{u}_1^{-\star} + e_0\,\mathbf{m}_1) \vee (\mathbf{u}_2^{-\star} + e_0\,\mathbf{m}_2) \\ &= \big(\mathbf{u}_2 \wedge \mathbf{m}_1^\star + \widehat{\mathbf{m}_2^\star} \wedge \mathbf{u}_1\big)^{-\star} + e_0\,(\mathbf{m}_2^\star \wedge \mathbf{m}_1^\star)^{-\star} \\ &= \mathbf{u}_2\cdot\mathbf{m}_1 - \mathbf{u}_1\cdot\mathbf{m}_2 + 0. \end{aligned} \tag{41}$$

We will see later (in Section 8.6) that this scalar outcome is related to a volume spanned by the lines involving their direction vectors $\mathbf{u}_1$, $\mathbf{u}_2$ and their perpendicular distance, and that its sign is related to their relative chirality (the 'handedness' of their relative positioning).

3.4 Summary

The (linearized) union of planes in d-dimensional Euclidean space is encoded by the `join`. The `join` is dual to the `meet`; developing that relationship carefully leads to computable expressions for the `join` in eq.(35) or eq.(36). Its computational complexity is similar to the `meet` (since duals can be implemented as signed coordinate selections).

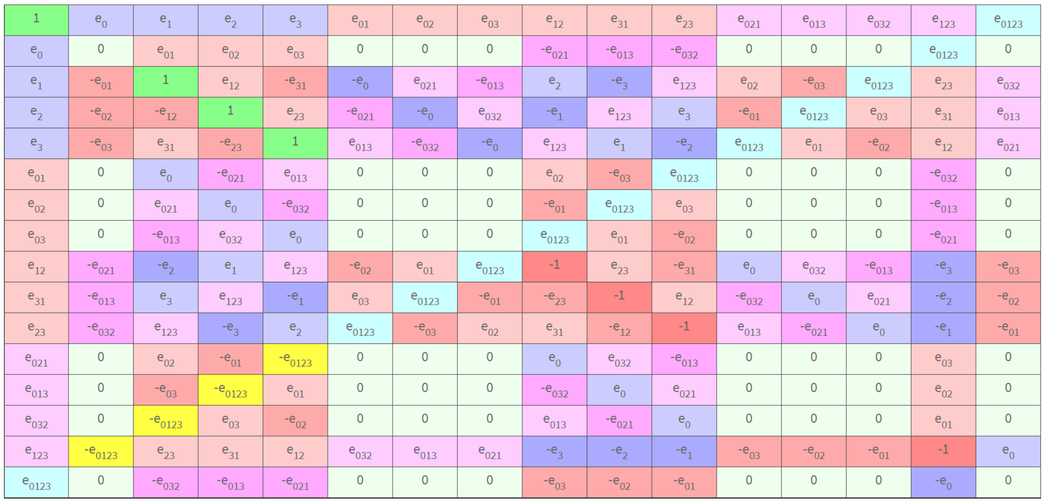
The `join` allows us to connect points to make a line, or a line and point to construct a plane, etc. It is as fundamental as the `meet` for the expressiveness of PGA.

We kept the orientation and magnitude information consistent by carefully designing the relationship between `meet` and `join` around the Common Factor Axiom.

4 The Geometric Product and PGA Norms

4.1 The Geometric Product of PGA

As a geometric algebra, PGA also has a geometric product. The multiplication table (or Cayley table) of this geometric product is given in Figure 1. We will use it to form the versors representing the Euclidean transformations, but also to study the relationships between geometric elements, in the next section.



1	e_0	e_1	e_2	e_3	e_{01}	e_{02}	e_{03}	e_{12}	e_{31}	e_{23}	e_{021}	e_{013}	e_{032}	e_{123}	e_{0123}
e_0	0	e_{01}	e_{02}	e_{03}	0	0	0	$-e_{021}$	$-e_{013}$	$-e_{032}$	0	0	0	e_{0123}	0
e_1	$-e_{01}$	1	e_{12}	$-e_{31}$	$-e_0$	e_{021}	$-e_{013}$	e_2	$-e_3$	e_{123}	e_{02}	$-e_{03}$	e_{0123}	e_{23}	e_{032}
e_2	$-e_{02}$	$-e_{12}$	1	e_{23}	$-e_{021}$	$-e_0$	e_{032}	$-e_1$	e_{123}	e_3	$-e_{01}$	e_{0123}	e_{03}	e_{31}	e_{013}
e_3	$-e_{03}$	e_{31}	$-e_{23}$	1	e_{013}	$-e_{032}$	$-e_0$	e_{123}	e_1	$-e_2$	e_{0123}	e_{01}	$-e_{02}$	e_{12}	e_{021}
e_{01}	0	e_0	$-e_{021}$	e_{013}	0	0	0	e_{02}	$-e_{03}$	e_{0123}	0	0	0	$-e_{032}$	0
e_{02}	0	e_{021}	e_0	$-e_{032}$	0	0	0	$-e_{01}$	e_{0123}	e_{03}	0	0	0	$-e_{013}$	0
e_{03}	0	$-e_{013}$	e_{032}	e_0	0	0	0	e_{0123}	e_{01}	$-e_{02}$	0	0	0	$-e_{021}$	0
e_{12}	$-e_{021}$	$-e_2$	e_1	e_{123}	$-e_{02}$	e_{01}	e_{0123}	-1	e_{23}	$-e_{31}$	e_0	e_{032}	$-e_{013}$	$-e_3$	$-e_{03}$
e_{31}	$-e_{013}$	e_3	e_{123}	$-e_1$	e_{03}	e_{0123}	$-e_{01}$	$-e_{23}$	-1	e_{12}	$-e_{032}$	e_0	e_{021}	$-e_2$	$-e_{02}$
e_{23}	$-e_{032}$	e_{123}	$-e_3$	e_2	e_{0123}	$-e_{03}$	e_{02}	e_{31}	$-e_{12}$	-1	e_{013}	$-e_{021}$	e_0	$-e_1$	$-e_{01}$
e_{021}	0	e_{02}	$-e_{01}$	$-e_{0123}$	0	0	0	e_0	e_{032}	$-e_{013}$	0	0	0	e_{03}	0
e_{013}	0	$-e_{03}$	$-e_{0123}$	e_{01}	0	0	0	$-e_{032}$	e_0	e_{021}	0	0	0	e_{02}	0
e_{032}	0	$-e_{0123}$	e_{03}	$-e_{02}$	0	0	0	e_{013}	$-e_{021}$	e_0	0	0	0	e_{01}	0
e_{123}	$-e_{0123}$	e_{23}	e_{31}	e_{12}	e_{032}	e_{013}	e_{021}	$-e_3$	$-e_2$	$-e_1$	$-e_{03}$	$-e_{02}$	$-e_{01}$	-1	e_0
e_{0123}	0	$-e_{032}$	$-e_{013}$	$-e_{021}$	0	0	0	$-e_{03}$	$-e_{02}$	$-e_{01}$	0	0	0	$-e_0$	0

Figure 1: *The Cayley table of 3D PGA. In the text we denote* e_0 *by* e_0*.*

4.2 The Euclidean norm

The natural concept of the norm of an element A in geometric algebras is the usual

$$\|A\| = \sqrt{A * \widetilde{A}} = \sqrt{\langle A\,\widetilde{A}\rangle_0}, \tag{42}$$

where $*$ is the scalar product, and $\langle\ \rangle_0$ selects the grade-0 part of its argument; the two are equivalent. This definition works well in spaces where the basis vectors do not have negative squares: the reversion always makes similar elements cancel each other with a positive or zero scalar, so the square root exists. This definition therefore seems to suffice for PGA.

4.3 Vanishing or Infinity Norm

However, when an element A is 'null' (i.e., has a zero square), the standard norm eq.(42) is rather useless. For instance, when we have an element of the form $\delta \mathbf{n} e_0$ as the result of the `meet` of two planes with normal $\mathbf{n}$ at a distance δ apart, we would like to extract the distance δ from the result. But the norm of the null result is always zero.

There is a simple method which can be implemented easily: null elements have an e_0 as a factor; (1) isolate the null elements, (2) strike out that factor e_0, and (3) compute the Euclidean norm of what remains. Algebraically, it can be defined by employing the *reciprocal* e_0^r of e_0, as a dot product (so that the parts not containing e_0 automatically do not contribute).

$$\|A\|_\infty = \|e_0^r \cdot A\|. \tag{43}$$

This is called the *infinity norm*, or *ideal norm*; for us it would be consistent to call it the *vanishing norm*.

Note that this definition eq.(43) is not algebraically a part of PGA proper, since we need to invoke the reciprocal e_0^r of e_0, which is in the dual space of PGA. When in Section 9 we have the Hodge dual, we can employ that to denote the infinity norm as

$$\|A\|_\infty = \|\star A\|, \tag{44}$$

since the Hodge dual also has the effect of removing the e_0 part from A and producing something of which a Euclidean norm can be taken. (It also adds a multiplication factor e_0 to an already Euclidean part, automatically excluding it from contributing to the norm.)

Actually, since the squared norm definition destroys the orientational sign of its arguments, one might prefer to treat the Euclidean element $e^r \cdot A$ directly; in the above example, $\delta\mathbf{n}$ is a geometrically significant quantity, whereas $|\delta|$ by itself is not. In an implementation, finding this corresponding Euclidean part, this is simple enough: just take the components of A on a basis $\mathbf{e}_{0J}$ and consider them as if they were those of an element on the basis $\mathbf{e}_J$. Done.[12] We will treat the issue of oriented quantities in more detail in Section 8, as a more subtle form of quantitative geometry.

[12]In terms of the Hodge dual of Section 9, the extraction of the Euclidean factor $\mathbf{X}$ from an element $e_0\mathbf{X}$ is done by $\mathbf{X} = \mathbf{I}_d(\star(e_0\mathbf{X}))^\sim$.

4.4 Using Norms

We should emphasize that a general element of PGA of the form $A = \mathbf{T}_A + e_0\,\mathbf{P}_A$ typically has both a regular norm (returning a magnitude for $\mathbf{T}_A$) and an infinity norm (returning a magnitude for $\mathbf{P}_A$):

$$\|A\| = \|\mathbf{T}_A + e_0\,\mathbf{P}_A\| = \|\mathbf{T}_A\|, \tag{45}$$

$$\|A\|_\infty = \|\mathbf{T}_A + e_0\,\mathbf{P}_A\|_\infty = \|e_0\,\mathbf{P}_A\|_\infty = \|\mathbf{P}_A\|. \tag{46}$$

These norms can be used as a compact way of computing the length and area of an edge loop, or the area and volume of a triangle mesh [11, 12].

In 2D PGA, define an *edge loop* as a concatenation of consistently oriented weighted lines between points P_i, each edge L_i being the `join` of two consecutive points $L_i = P_i \vee P_{i+1}$ (with the last edge formed by joining the last point to the first). The points here are in the 2D subspace, so they are bivectors of the form $(1 - e_0\mathbf{x})\,\mathbf{I}_2$, see exercise 12.2:3; the edges are vectors of 2D PGA. Then the *contour length* c and *area* a of this loop are simply

$$c = \sum_i \|L_i\| \quad \text{and} \quad a = \tfrac{1}{2!}\,\|\sum_i L_i\|_\infty. \tag{47}$$

Note the subtlety in placement of the norm delimiters relative to the summation! These formulas also work for a loop of points lying in a common plane in d-dimensional space, where points are represented as d-vectors; so you can use them for polygons in 3D.

Similarly, we define a closed triangle mesh through its faces f_i characterized by oriented weighted planes constructed by joining the three vertices of each triangle, in an order that is everywhere consistent with the mesh orientation: $f_i = P_{i1} \vee P_{i2} \vee P_{i3}$. The points should be 3D PGA trivectors, so that the faces are vectors in 3D PGA. Then the *area* a and *volume* v of this mesh are simply

$$a = \tfrac{1}{2!}\sum_i \|f_i\| \quad \text{and} \quad v = \tfrac{1}{3!}\,\|\sum_i f_i\|_\infty. \tag{48}$$

Both formulas clearly make essential use of the capability of PGA to have its geometrical elements be weighted and oriented. Their proof may be found in [12].

4.5 Summary

We introduced two types of norm in PGA, to enable the measurement of magnitudes of finite and ideal elements. In an application of these norms, we demonstrated compact PGA expressions for integral measures on discrete meshes.

We remarked that for an oriented geometry, we may need to be more more subtle than just taking a norm (but this is new terrain).

5 Bundles and Pencils as Points and Lines

Everyone who encounters PGA for the first time is taken aback that points in 3D should be represented by trivectors. It seems only natural that points should be vectors, and apparently some people feel very strongly about that. Yet having points as trivectors is the natural consequence of identifying the outer product with an intersection operation. In this section, we will try to make the 'points as trivector' view more acceptable, and show how this leads to some truly compact constructions for the relationships between geometrical primitives.

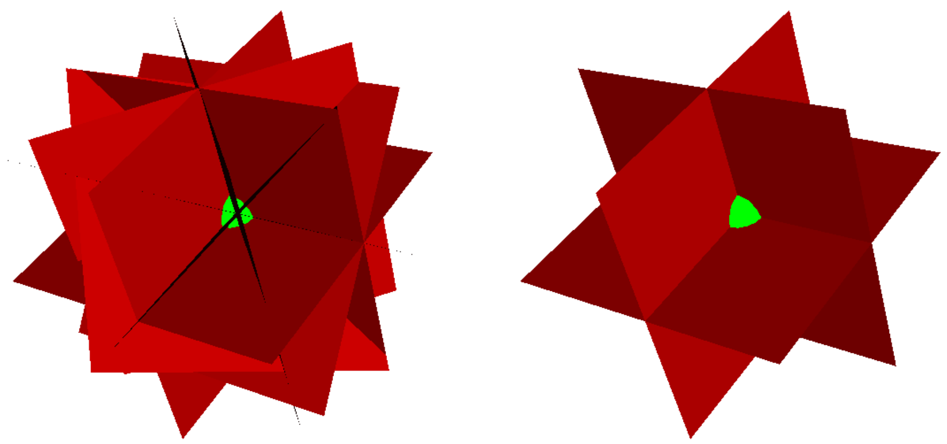


Figure 2: *Visualization of the bundle of planes at a point X, and a basis of three planes to represent all. The one depicted happens to be orthonormal, but that is not necessary.*

5.1 The Bundle of Planes at a Point

Many planes pass through a 3D point X with location $\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$. These planes can all be characterized on a 3D basis of planes through that point. We could choose, for instance, planes parallel to the coordinate planes (orthogonal to the coordinate directions) for such a basis: $p_1 = \mathbf{e}_1 - x_1 e_0$ and $p_2 = \mathbf{e}_2 - x_2 e_0$ and $p_3 = \mathbf{e}_3 - x_3 e_0$, with $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ an orthonormal basis for convenience.

Any linear combination of such planes forms a plane that also passes through the point at location $\mathbf{x}$ (you may feel more comfortable verifying this using your familiar homogeneous coordinates: if $x \cdot p_1 = 0$ and $x \cdot p_2 = 0$, then $x \cdot (\alpha p_1 + \beta p_2) = 0$, so the weighted sum of p_1 and p_2 is a new plane that also contains x). There is thus a 3-D linear space of planes, all passing through the point at location $\mathbf{x}$; this is called the *bundle of planes* at that point. In geometric algebra, whenever we have a 3-D linear space, we can characterize it by a 3-blade T, such that

$$p \wedge T = 0 \quad \iff \quad p \text{ is a member of the subspace characterized by } T. \tag{49}$$

In a Grassmannian sense, the 3-blade T *is* the subspace - as we explained in Chapter 2 of [2], or see [9]. In fact, it is more quantitatively specific than a mere characterization of a set of vectors, since it also has an orientation and a magnitude, beside the spatial attitude.

Since the trivector

$$T_{\mathbf{x}} \equiv p_1 \wedge p_2 \wedge p_3 \tag{50}$$

contains *all* the planes passing through the point X at location $\mathbf{x}$, it pinpoints that point. We could let it *be* the point. Indeed, if we have all operations like `meet`, `join` and so on to work with such a representation, there is no real need for a separate data structure to describe the point. So let us simply identify trivector $T_{\mathbf{x}}$ with the point X at location $\mathbf{x}$, and see where that leads us.

And immediately, using the trivector $X = T_{\mathbf{x}}$ to represent the bundle of planes through X allows some standard techniques from geometric algebra to acquire a powerful geometric interpretation. For instance, the *orthogonal projection* of a vector $\mathbf{x}$ onto a subspace characterized by a blade $\mathbf{A}$ is (see Chapter 6 of [2])

$$\text{orthogonal projection of } \mathbf{x} \text{ to subspace } \mathbf{A}\text{:} \;\; P_{\mathbf{A}}[\mathbf{x}] \equiv (\mathbf{x} \cdot \mathbf{A})/\mathbf{A}. \tag{51}$$

(Why? Because $(\mathbf{x} \cdot \mathbf{A})/\mathbf{A} = \frac{1}{2}\,(\mathbf{x}\,\mathbf{A}\,\mathbf{A}^{-1} - \widehat{\mathbf{A}}\mathbf{x}\mathbf{A}^{-1}) = \frac{1}{2}\,(\mathbf{x} - \widehat{\mathbf{A}}\mathbf{x}\mathbf{A}^{-1})$ is half the sum of $\mathbf{x}$ and the reflection of $\mathbf{x}$ in $\mathbf{A}$, see Table 7.1 in [2]).

We can therefore project an arbitrary plane p to the bundle of planes passing through X, and the resulting $p' \equiv (p \cdot X)/X$ must pass through X, since it is part of the subspace of planes that do. Moreover, it is the *orthogonal* projection onto the bundle, in the sense that the result p' is in the bundle, but the difference $p' - p$ is orthogonal to the bundle. The only plane

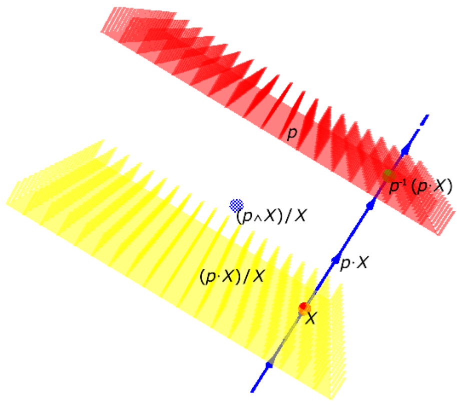


Figure 3: *Computational relationships between a point X and a plane p. The weighted e_0-plane $(p \wedge X)/X$ is denoted by a weighted dotted point at the origin.*

orthogonal to the bundle (i.e., to all planes in the bundle) is the vanishing plane $-e_0$ (check that indeed $p \cdot e_0 = 0$ holds for any plane p). Therefore the difference $p' - p$ is purely a multiple of e_0, with no Euclidean component in their difference; it follows that the two planes p and p' must have the same Euclidean normal. This implies that the two planes do not only have the same attitude, but also the same orientation ('handedness', useful for distinguishing back and front), and they even have the same weight. So they are truly parallel, even in an oriented sense.

We have thus found a simple way of finding the parallel plane through the point X, to a given oriented plane p, just by employing the projection $P_X[\,]$.

$$\text{parallel plane to } p \text{ through point } X\text{:}\quad P_X[p] \equiv (p \cdot X)/X. \tag{52}$$

Such is the power of geometric algebra: the same structural operators (here orthogonal projection) acquire new meanings in new embeddings.

Another way of looking at the equation is: $p \cdot X$ is of grade 2, contained in X, and $p \cdot (p \cdot X) = (p \wedge p) \cdot X = 0$; therefore $p \cdot X$ is the line through X orthogonal to p. Let us keep this nice intermediate result:

$$\text{line orthogonal to plane } p \text{ and through point } X\text{:}\quad L = p \cdot X. \tag{53}$$

In the projection eq.(52), the subsequent division of $p \cdot X$ by X, which is the local pseudoscalar, takes the orthogonal complement of this line (at X), and therefore results in the plane orthogonal to that line - which is the parallel of the oriented plane p (since we have been careful about the signs: once a product with X, and once a division, 'next to' each other). It is actually rather remarkable that the division by X does this orthogonal complement, i.e., dualization, locally at the location X, independent of the origin. As we will see later, in Section 9, projective duality in PGA does *not* do this.

The sister operation to orthogonal projection is, in general, the *orthogonal rejection*

$$\text{orthogonal rejection of } \mathbf{x} \text{ to subspace } \mathbf{A}\text{:} \quad R_{\mathbf{A}}[\mathbf{x}] \equiv (\mathbf{x} \wedge \mathbf{A})/\mathbf{A}. \tag{54}$$

The outcome of this operation is a vector of which the dot product with $\mathbf{A}$ is zero, so it is orthogonal to $\mathbf{A}$. As we have just seen, in PGA the *rejection* $p' - p = (p \wedge X)/X$ is proportional to the vanishing plane e_0. The constant of proportionality is geometrically meaningful: if the planes are normalized, it is the oriented distance of p to X.

$$\text{oriented distance (times } e_0\text{) from plane } p \text{ to point } P\text{:} \quad R_X[p] \equiv (p \wedge X)/X. \tag{55}$$

Let us check the signs, since we want an algebra of oriented flats. With the conventions we have chosen, the plane $\mathbf{n} - \delta e_0$ is a distance δ along $\mathbf{n}$ from the origin $O = \mathbf{I}_3$. Since $R_{\mathbf{I}_3}[\mathbf{n} - \delta e_0] = (0 - \delta e_0 \wedge \mathbf{I}_3)/\mathbf{I}_3 = -\delta e_0 - \delta(-e_0)$, we should indeed interpret the outcome δ as the oreineted distance from plane to point ('how much to move the plane into the normal direction to get onto the point'), rather than from point O to plane p (which would be $-\delta$).[13] Those signs and orderings are hard to remember, so later we will prefer to test the 'relative orientation' using a probe, to answer questions of the type: 'is this point at the same side as or at the opposite side of the plane, relative to this other point?'.

[13]When we introduced the oriented plane at infinity as a limiting process for large positive distance δ of a finite plane $\mathbf{n} - \delta e_0$, it clearly is more properly represented as $-e_0$ rather than e_0.

5.2 The Geometric Product of Plane and Point

Together, the projection and rejection form a decomposition of the plane p relative to the point X:

$$p = (p\,X)/X = (p \cdot X)/X + (p \wedge X)/X. \tag{56}$$

With the above interpretations, this can be read as: 'the arbitrary plane p can be thought of as being constructed from a plane through the point X, moved parallel to itself by adding the right amount of vanishing plane'. It makes good geometric sense, and the algebraic form it takes is pleasantly straightforward. But actually, this way of translating elements is not that convenient in implementations, since it is operand dependent. We will do better later when we design the universal translation operation by versor sandwiching in Section 6.

An alternative way of reading eq.(56) is: given the PGA quantity $p\,X$ and the point X, you can reconstruct a unique plane p. And since we also have $X = p^{-1}\,(p\,X)$, we could also have constructed the point X if we had been given this PGA quantity $p\,X$ and the plane p. Somehow, then, the geometric product $p\,X$ contains *all* geometric relationships between p and X.

Let us investigate this in slightly more detail. When we develop pX in grades, we obtain a 2-blade and a 4-blade, which can therefore be separately retrieved. The 2-blade evaluates to

$$\begin{aligned}\langle p\,X\rangle_2 = p \cdot X &= (\mathbf{n} - \delta e_0) \cdot \big(\mathbf{I}_3 - e_0\,(\mathbf{x} \cdot \mathbf{I}_3)\big)\\ &= \mathbf{n} \cdot \mathbf{I}_3 + e_0\,\big(\mathbf{n} \cdot (\mathbf{x} \cdot \mathbf{I}_3)\big)\\ &= \mathbf{n}\mathbf{I}_3 - e_0\,\big(\mathbf{x} \cdot (\mathbf{n}\mathbf{I}_3)\big)\\ &= \mathbf{n}\mathbf{I}_3 - e_0\,(\mathbf{x} \wedge \mathbf{n})\,\mathbf{I}_3.\end{aligned} \tag{57}$$

We have seen this before: it is the line L in the $\mathbf{n}$-direction passing through location $\mathbf{x}$, i.e. it is the line L through X perpendicular to plane p, with a normal that obeys the right-hand convention relative to the 2-dimensional orientation of the plane. And the 4-blade term evaluates to

$$\begin{aligned}\langle p\,X\rangle_4 = p \wedge X &= (\mathbf{n} - \delta e_0) \wedge \big(\mathbf{I}_3 - e_0\,(\mathbf{x}\mathbf{I}_3)\big)\\ &= -e_0\,\big(\delta \mathbf{I}_3 - \mathbf{n} \wedge (\mathbf{x}\mathbf{I}_3)\big)\\ &= (\mathbf{n} \cdot \mathbf{x} - \delta)\,\mathcal{I}_3.\end{aligned} \tag{58}$$

This quantity $(\mathbf{n} \cdot \mathbf{x} - \delta)$ is the distance from the point to the plane. It now makes geometrical sense why the element $p\,X$ allows mutual retrieval of p given X, or of X given p.

- If we give you $p\,X$, i.e., '*an oriented line L through X (without telling you which point on the line X actually is) and an oriented point-plane distance δ*', and then give you p (which must be perpendicular to the line in the correct orientation), then you can reconstruct X by moving the stated distance δ against the direction of L from the intersection point of line L and plane p.
- If we give you $p\,X$, i.e., '*an oriented line L through X (without telling you which point on the line X actually is) together with an oriented point-plane distance δ*', and then give you X (which must be on the line), then you can reconstruct p by moving the stated distance δ along L from X, and establishing the oriented orthogonal plane with normal vector in the line direction.

Always when we compute a geometric product between two invertible primitives, we can reconstruct either of them given that product and the other element. The realization of what you would need to perform such a reconstruction will give you a good intuition of what the different grades of the geometric product can mean geometrically. Those grade parts are always handy constructions by themselves, as we saw above for $p \cdot X$ and $p \wedge X$.

Constructions involving two geometric elements were presented in the 'cheat sheet' of recipes at SIGGRAPH 2019 [11], from which we list some in Table 2. With the above explanation as your guide, you could consider each of the entries as an exercise in your ability to connect algebra and geometry. It will always be consistent, and often provide satisfying insights.

5.3 Relocating Any Flat by Orthogonal Projection

Our primitives are made by outer products of planes. We know how to relocate any plane p to a point X, namely by the orthogonal projection $P_X[p] = (p \cdot X)/X$ by eq.(52). But since orthogonal projection is a linear transformation, we can extend it as an *outermorphism*: the orthogonal projection of an outer product of terms is the outer product of the orthogonal projections. (See Chapter 4 of [2] for a treatment of outermorphisms.)

This implies that if we want to move a line L to pass through a point X, we can pick any two planes p_1 and p_2 that define the line as their intersection, orthogonally project those, take their outer product, and we will have the line passing through X, which is thus $P_X[p_1] \wedge P_X[p_2]$. For an orthogonal projection, we even have the convenience that the orthogonal projection operation extends in its mathematical form: $(p \cdot X)/X$ becomes $((p_1 \wedge p_2 \wedge \cdots) \cdot X)/X$, we can just replace the argument in the function for vectors (see Section 4.2 of [2]).

As a consequence, to position any flat A at location X, we simply compute:[14]

$$\text{flat } A \text{ positioned at point } X\text{:} \quad P_X[A] \equiv (A \cdot X)/X. \tag{59}$$

This formula gives us a convenient way to make flats: just construct them at the origin, where they will be purely Euclidean, and project them to their destination. If you want to move them again to a different point, project them to that point, for $P_Q \circ P_P = P_Q$: the last repositioning counts.

Such 'repositioning by orthogonal projection' is more convenient than repositioning by the classical translation operator, for which you would first need to compute the relative position of 'where you are' to 'where you want to end up' and use that as an argument for the translation operator.

It is interesting to study what happens when we try to relocate to the vanishing elements, which contain a factor e_0. Since $P_X[e_0] = 0$, such a factor in a flat will make the whole result 0. You cannot reposition elements from infinity back to the finite Euclidean domain. Sounds reasonable! And it does show in detail how $P_X[A]$ 'forgets' the positional aspect $\mathbf{P}_A$ of its argument $A = \mathbf{T}_A + e_0\mathbf{P}_A$, while constructing a new positional aspect for the X-location, from $\mathbf{T}_A$ and $\mathbf{x}$.

Later we will meet more general elements which are the sum of blades of different grades. The same principle applies to each of the individual terms, so we can move a general element of the algebra to a point X in this manner: by orthogonal projection onto the bundle of planes that is the point X.

5.4 Projecting a Point onto a Plane

Projecting a point onto a plane also uses the general orthogonal projection operation construction. You might guess to use $(X \cdot p)/p$, and this fine if you are using as your definition of the dot product: the minimum grade of the geometric product (as is common!). In the context of our book [2], the dot product in this chapter is actually always the left contraction $\rfloor$, which is zero when a larger grade is contracted onto a smaller one. Therefore we

[14]If you want this to work properly for a scalar A, make sure that the dot product you use has the property $\alpha \cdot X = \alpha\, X$ for a scalar α; when in doubt, use the left contraction, see [2].

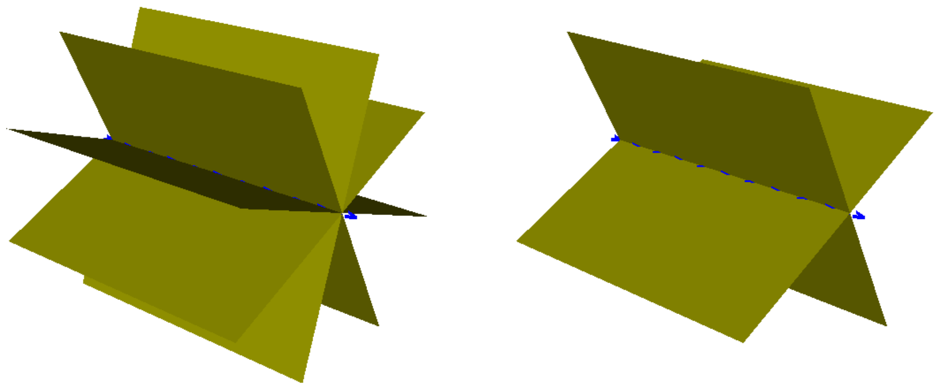


Figure 4: *Visualization of the pencil of planes containing a line L, and a basis of two planes to represent all. The one depicted happens to be orthonormal, but that is not necessary.*

have preferred to denote it as the equivalent $p\backslash(p \cdot X) \equiv p^{-1}(p \cdot X)$ in our Table 2.

5.5 The Pencil of Planes Around a Line

We have seen how a line is encoded as a 2-blade, by considering it as the `meet` of two planes

$$L = p_1 \wedge p_2. \tag{60}$$

This 2-blade characterizes a 2-dimensional subspace of planes: the *pencil of planes around line* L, see Figure 4. Any plane also containing the line L can be written as a linear combination of p_1 and p_2; any plane that does not contain the line cannot.

Again the orthogonal projection of a plane p onto this subspace L is a geometrically sensible plane:

$$\text{component plane of } p \text{ containing line } L\text{:} \;\; P_L[p] \equiv (p \cdot L)/L. \tag{61}$$

Here 'component' is not a very descriptive term. Perhaps we should revive Grassmann's own word 'shadow': $P_L[p]$ is the 'L-aligned perpendicular shadow' of p. The pattern is the same as with projecting on a bundle: the resulting element must be a plane, since projection preserves grade. It must be a plane in the pencil, since it is an element of L. And orthogonality of the difference plane $p' - p$ to L means that the resulting normal vector is the rejection of the plane normal $\mathbf{n}$ by the line direction vector $\mathbf{u}$.

> Example: for a line $L = \mathbf{u}\mathbf{I}_3$ in direction $\mathbf{u}$ through the origin, and a plane $p = \mathbf{n} - \delta e_0$, we have $P_L[p] = \big((\mathbf{n} - \delta e_0) \cdot \mathbf{u}\mathbf{I}_3\big)(\mathbf{u}\mathbf{I}_3)^{-1} = (\mathbf{n} \wedge \mathbf{u})/\mathbf{u} \equiv R_{\mathbf{u}}[\mathbf{n}]$. So the normal vector of the resulting plane is perpendicular to the direction vector $\mathbf{u}$ of the line.

The rejection $(p \wedge L)/L$ produces another plane. It is orthogonal to all planes in the pencil (for that is what a rejection by a subspace does: returning the perpendicular to all vectors in it). It contains $p \wedge L$, which is the intersection point of p and L

$$\text{intersection point of plane } p \text{ and line } L\text{:}\ \ p \wedge L. \tag{62}$$

(Exercise: show that indeed $\big((p \wedge L)/L\big) \wedge (p \wedge L) = 0$.) With that, the geometry of the rejection is clear:

$$\text{plane perpendicular to } L\text{, through intersection with plane } p\text{:}$$
$$R_L[p] \equiv (p \wedge L)/L. \tag{63}$$

If the intersection happens to be a vanishing point, the rejection is a vanishing plane, whose weight is the oriented distance from the line L to p. Exercise: show this!

> Example: for a line $L = \mathbf{u}\mathbf{I}_3$ in direction $\mathbf{u}$ through the origin, and a plane $p = \mathbf{n} - \delta e_0$, we have $R_L[p] = \big((\mathbf{n} - \delta e_0) \wedge \mathbf{u}\mathbf{I}_3\big)(\mathbf{u}\mathbf{I}_3)^{-1} = (\mathbf{n} \cdot \mathbf{u})/\mathbf{u} - \delta e_0 = P_{\mathbf{u}}[\mathbf{n}] - \delta e_0$. (Be somewhat careful interpreting the location of this plane, since it is no longer normalized!) Shifting the origin of our coordinates to the intersection point (so that $\delta = 0$) shows the interpretation of this plane.

The sum of projection and rejection relative to the line L again reconstructs the plane p. Figure 5 shows the planes involved.

5.6 Non-Blade Bivectors are Screws

So far, we have decomposed a plane relative to a point, and a plane relative to a line. When we would seek to use similar techniques to decompose a line

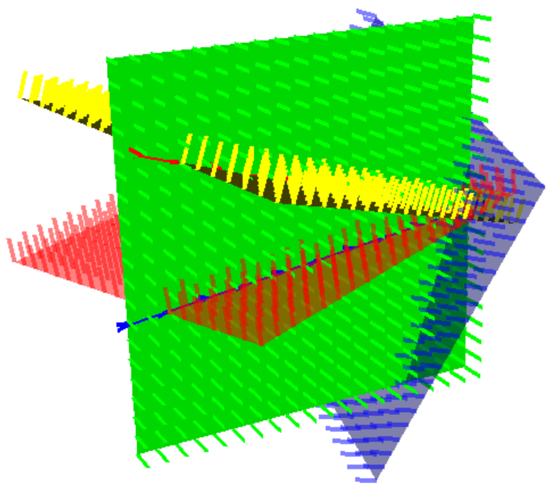


Figure 5: *All relationships of a red plane p and a red line L. In green: $p \cdot L$, in yellow: $(p \cdot L)/L$, blue plane: $(p \wedge L)/L$, blue line: $p \backslash (p \cdot L)$. The intersection point is $p \wedge L$.*

relative to another line, we need more terms – and the geometric product automatically provides them:

$$L\,M = \underbrace{L \cdot M}_{\text{grade } 0} + \underbrace{L \times M}_{\text{grade } 2} + \underbrace{L \wedge M}_{\text{grade } 4} \tag{64}$$

The parts of grade 0 and grade 4 can contain only one scalar weight each, these are related to the relative angle and distance of the lines (for normalized lines, for others the linearity gives an overall scaling based on their 'strength' or 'velocity'). The *commutator product* (defined as $L \times M = \frac{1}{2}(L\,M - M\,L)$) must provide all the rest of the relative geometric positioning. You would expect the orthogonal line connecting L and M to be a part of this; and indeed it is, but there is more. In contrast to the inner and outer product, the commutator product of two 2-blades does not produce another blade, but a 2-vector (a.k.a. *bivector*). And bivectors in the 4-dimensional space of 3D PGA are unusual.

We remind you that blades can be written algebraically as the outer product of vectors, and that is the algebraic reason we can geometrically interpret them as the intersection `meet` of planes by eq.(7). Hence in PGA they represent off-origin flat subspaces of different grades. The commutator product will produce a sum of terms of grade 2, like the other products produce sums of elements of their own appropriate grades. But in the 4D representational space, a sum of k-blades is only guaranteed to be a k-blade in the 4D representational space for $k = 0, 1, 3, 4$; not for grade 2.

It is therefore important to distinguish between *2-vectors* and *2-blades*. Since 2-blades square to a scalar, testing a grade 2 element B for bladeness is equivalent to confirming that B^2 only contains scalar parts; this boils down to verifying that

$$\text{bladeness test for 2-vector: check whether } \; B \wedge B = 0. \tag{65}$$

The 2-blades come in two varieties[15] in 3D PGA:

- If $B^2 < 0$, the 2-blade B represents a true line, the intersection of two Euclidean planes. When exponentiated, such elements will generate a *rotation* around that line, as we will see in Section 7.
- If $B^2 = 0$, the 2-blade B represents an 'ideal' line, the intersection of some Euclidean plane with the vanishing plane e_0 at infinity. Since such lines contain vanishing points, we call them 'vanishing lines'. When exponentiated, such elements will also generate a rotation around that vanishing line. But historically, we prefer to refer to that as a *translation*, in a direction perpendicular to the Euclidean plane involved.

Therefore, the 2-blades are the generators of simple motions: rotations and translations. The 2-vectors can also be exponentiated, but they then generate a *screw motion*, the most general kind of Euclidean motion in 3D. This can be decomposed (according to Chasles' theorem) into a rotation around a line and a translation along that line. We will treat this later in Section 6.5; let us now establish that there are two unique special lines hiding within every 2-vector.

In particular, we show that we can split any bivector B into a weighted sum of a unit line L and its unit dual vanishing line $L\mathcal{I}$:

$$B = \alpha\, L + \beta\, (L\mathcal{I}). \tag{66}$$

[15] $B^2 > 0$ does not occur in PGA (Euclidean PGA) – though it happens in Hyperbolic PGA [13]!

(Remember that $\mathcal{I} \equiv e_0 \mathbf{I}_3$ is the pseudoscalar of the 4D representational space of 3D PGA.) Here L and $L\mathcal{I}$ are blades, so they satisfy $L \wedge L = 0$ and $(L\mathcal{I}) \wedge (L\mathcal{I}) = 0$. They also commute: due to $L\mathcal{I} = \mathcal{I}L$, we have $L(L\mathcal{I}) = L^2\mathcal{I} = \mathcal{I}L^2 = (\mathcal{I}L)L = (L\mathcal{I})L$. They are geometrically orthogonal, since $L \cdot (L\mathcal{I}) = (L \wedge L)\,\mathcal{I} = 0$.

We now compute the α, β and L from eq.(66). Since $L\mathcal{I} = \mathcal{I}L$, we can factorize the equation to $B = (\alpha + \beta\mathcal{I})\, L$. This is a kind of polar decomposition of B, with a *self-reverse* factor, see [14]. By squaring, we can eliminate the L (since $L\widetilde{L} = -L^2 = 1$) and directly solve for α and β:

$$B\widetilde{B} = B \cdot \widetilde{B} + \tfrac{1}{2}(B\,\widetilde{B} - \widetilde{B}\,B) + B \wedge \widetilde{B} = B \cdot \widetilde{B} + B \wedge \widetilde{B} = \alpha^2 + 2\alpha\beta\,\mathcal{I}. \quad (67)$$

Terms of corresponding grades should be equal at both sides, so

$$\alpha = \sqrt{B \cdot \widetilde{B}} \quad \text{and} \quad \beta\mathcal{I} = \frac{B \wedge \widetilde{B}}{2\sqrt{B \cdot \widetilde{B}}}. \quad (68)$$

Note that this does not work if $\alpha = 0$, which happens precisely when $B\widetilde{B} = 0$. In that case B was already a 2-blade, of the form $\beta L\mathcal{I}$.

For $B\widetilde{B} \neq 0$, we compute

$$L\mathcal{I} \;=\; B\mathcal{I}/\alpha = \frac{B\mathcal{I}}{\sqrt{B \cdot \widetilde{B}}} \quad (69)$$

$$L \;=\; (B - \beta L\mathcal{I})/\alpha = \frac{B}{\sqrt{B \cdot \widetilde{B}}}\big(1 - \tfrac{1}{2}\frac{B \wedge \widetilde{B}}{B \cdot \widetilde{B}}\big). \quad (70)$$

So combining it all, we have decomposed the 2-vector B as a sum of two commuting 2-blades

$$B = \underbrace{B\big(1 - \tfrac{1}{2}\frac{B \wedge \widetilde{B}}{B \cdot \widetilde{B}}\big)}_{\text{Euclidean line}} + \underbrace{\tfrac{1}{2}B\,\frac{B \wedge \widetilde{B}}{B \cdot \widetilde{B}}}_{\text{vanishing line}}\;, \quad (71)$$

unless $B \cdot \widetilde{B} = 0$; then B is already a vanishing line.

Example: Split the non-blade bivector $B = (\mathbf{e}_1 - \mathbf{e}_3) \wedge \mathbf{e}_2 + e_0 \wedge \mathbf{e}_3$ into commuting blade parts. We compute

$$\begin{aligned} B \wedge \widetilde{B} &= \big((\mathbf{e}_1 - \mathbf{e}_3) \wedge \mathbf{e}_2 + e_0 \wedge \mathbf{e}_3\big) \wedge \big(\mathbf{e}_2 \wedge (\mathbf{e}_1 - \mathbf{e}_3) + \mathbf{e}_3 \wedge e_0\big) \\ &= -2\, e_0\, \mathbf{e}_1\, \mathbf{e}_2\, \mathbf{e}_3 \neq 0, \end{aligned} \quad (72)$$

so B is indeed not a blade. Then

$$\begin{aligned} B \cdot \widetilde{B} &= \big((\mathbf{e}_1 - \mathbf{e}_3) \wedge \mathbf{e}_2 + e_0 \wedge \mathbf{e}_3\big) \cdot \big(\mathbf{e}_2 \wedge (\mathbf{e}_1 - \mathbf{e}_3) + \mathbf{e}_3 \wedge e_0\big) \\ &= (\mathbf{e}_1 - \mathbf{e}_3) \cdot (\mathbf{e}_1 - \mathbf{e}_3) = 2. \end{aligned} \tag{73}$$

It then follows by substitution that

$$\begin{aligned} B &= (\mathbf{e}_1 - \mathbf{e}_3) \wedge \mathbf{e}_2 + e_0 \wedge \mathbf{e}_3 \\ &= \Big((\mathbf{e}_1 - \mathbf{e}_3) \wedge \mathbf{e}_2 + e_0 \wedge (1.5\,\mathbf{e}_3 - \mathbf{e}_1)\Big) - \tfrac{1}{2} e_0\,(\mathbf{e}_1 + \mathbf{e}_3) \\ &= (1 + e_0\,\mathbf{e}_2/2)(\mathbf{e}_1 + \mathbf{e}_3)^\star - \tfrac{1}{2} e_0\,(\mathbf{e}_1 + \mathbf{e}_3). \end{aligned} \tag{74}$$

It is easy to establish that the two terms are blades, and that they indeed commute.

The decomposition into two commuting 2-blades will aid us in the screw interpretation of B; those 2-blades will provide the screw axis, and its pitch, in Section 7.1.

5.7 Summary

We developed many useful formulas to combine the flat elements of our algebra, by employing the metric aspects implicit in the geometric product. Application of standard techniques like orthogonal projection and decomposition led to an understanding of the principles behind compact constructions and their encoding.

For example, we found that we can place an element at any location X by just projecting it onto the trivector representing that location in eq.(59). The cheat sheet of Table 2 lists some of the more common constructions.

We also pointed out that 3D lines are unusual: when you add them arbitrarily you get non-lines (screws). But any bivector of PGA can be decomposed orthogonally into a line and an orthogonal vanishing line by eq.(71).

geometric relationships					expression
Plane	perpendicular to	plane p	and containing	line L	$p \cdot L$
Point	intersection of	plane p	and	line L	$p \wedge L$
Line	perpendicular to	plane p	and containing	point P	$p \cdot P$
Distance	(times $\mathcal{I}$) from	point P	towards	plane p	$p \wedge P$
Plane	perpendicular to	line L	and containing	point P	$L \cdot P$
Plane	parallel to	plane p	and containing	point P	$(p \cdot P)/P$
Line	parallel to	line L	and containing	point P	$(L \cdot P)/P$
Point	repositioned from	point X	and placed at	point P	$(X \cdot P)/P$
Plane	projected from	plane p	and containing	line L	$(p \cdot L)/L$
Point	projected from	point P	and lying on	line L	$L\backslash(L \cdot P)$
Point	projected from	point P	and lying on	plane p	$p\backslash(p \cdot P)$
Plane	as reflection of	plane p	in the reflector	point X	$-X\,p/X$
Line	as reflection of	line L	in the reflector	point X	$X\,L/X$
Point	as reflection of	point P	in the reflector	point X	$-X\,P/X$
Line	from	point P	to	point Q	$P \vee Q$
Line	from	point P	to direction	$V_{\mathbf{u}}$	$P \vee V_{\mathbf{u}}$
Plane	through	points	P, Q, R		$P \vee Q \vee R$
Distance	between	point P	and	point Q	$\|P \vee Q\|$
Chirality	(times $\mathcal{I}$) between	line L	and	line M	$L \wedge M$

Table 2: *A 'cheat sheet' with useful nuggets in 3D PGA. Here p is a plane, L a line, and P, X points. If you are not interested in scaling factors, you can replace the divisions by reversions; this will be computationally cheaper, but still propagate the orientation signs consistently. See also a similar table by Gunn and De Keninck [11]; and Selig [3](his Section 4.9) effectively contains many of the same relations (though in a commutator/anti-commutator notation).*

6 Euclidean Motions through Planar Reflections

We have designed PGA as an algebra of planes since we want to describe Euclidean motions by means of planar reflections. This follows the general Cartan-Dieudonné Theorem:

> *All orthogonal transformations in an n-D symmetric bilinear space can be constructed from at most n reflections.*

It is immediately obvious that in 3D we can use two planar reflections to construct the two typical motions: rotations and translations. Using more planes combines such basic motions. But with 3D planes residing in a 4D representational space, Cartan-Dieudonné states that there will not be motions requiring more than 4 reflections. These multiple reflections have been given names:

- 0 reflections: the *identity* operation
- 1 reflection: a *planar reflection*
- 2 reflections: when the planes are parallel, this is a *translation*; when intersecting, a *rotation* around the common line; when perpendicular, a line reflection
- 3 reflections: this adds an extra reflection to the previous case, and generates a *glide reflection* when two of the three are parallel, and a *rotoreflection* when none are; when the three planes are perpendicular, this produces a point reflection
- 4 reflections: the general Euclidean rigid body motion, a *screw motion.* We can factorize a general motion always as a rotation around an axis and a translation along that same axis (Chasles' theorem). We will see that this description by means of four real reflections in real planes corresponds, in its coordinate form, directly to the often-used 19th century description of 3D rigid body motions by *dual quaternions* (which are often needlessly viewed as 'complex' in all senses of the word).

Let us find the versors corresponding to these multiple reflections in planes.

6.1 Multiple Reflection as Sandwiching

For the algebra of Euclidean directions, reflection of a direction vector $\mathbf{x}$ in a plane through the origin with normal vector $\mathbf{n}$ is the reflected direction $-\mathbf{n}\,\mathbf{x}\,\mathbf{n}^{-1}$ (if this is new to you, consult the earlier chapters of [2] or some other general text). This *sandwiching* operation:

$$\mathbf{x} \;\mapsto\; -\mathbf{n}\,\mathbf{x}\,\mathbf{n}^{-1} \tag{75}$$

then extends in two ways.

- It can be applied to any other element X of GA that we construct as linear combinations of geometric products (and we have no desire to make anything else)
$$X \;\mapsto\; \mathbf{n}\,\widehat{X}\,\mathbf{n}^{-1}, \tag{76}$$
where $\widehat{X} = (-1)^{\mathrm{grade}(X)}X$, the grade involution, introduces a minus sign for the odd grade elements, and no sign for the even grade elements.

- We can perform sandwiching by repeated reflection in other elements. Associativity allows us to group those in *versors*, which are geometric products of invertible vectors. In a Euclidean geometric algebra, this looks like (see Chapter 7 of [2]) a *sandwiching product*
$$\begin{aligned} \mathbf{x} \;&\mapsto\; (-1)^k\,(\mathbf{n}_k\,\mathbf{n}_{k-1}\cdots\mathbf{n}_1)\,\mathbf{x}\,(\mathbf{n}_1^{-1}\cdots\mathbf{n}_{k-1}^{-1}\,\mathbf{n}_k^{-1}) \\ &=\; (-1)^k\,(\mathbf{n}_k\,\mathbf{n}_{k-1}\cdots\mathbf{n}_1)\,\mathbf{x}\,(\mathbf{n}_k\,\mathbf{n}_{k-1}\cdots\mathbf{n}_1)^{-1} \\ &=\; \widehat{V}\,\mathbf{x}\,V^{-1}, \end{aligned} \tag{77}$$
with the versor V defined as the product of k reflectors (and $\widehat{V} \equiv (-1)^kV$). For motions, we are mostly interested in applying even versors; then there are no signs to worry about.

- The inversion might seem somewhat expensive to perform – it is defined as $V^{-1} = \widetilde{V}/(V\widetilde{V})$. But in PGA $\mathbb{R}_{d,0,1}$, the denominator $V\widetilde{V}$ is always positive since all Euclidean vectors have positive squares. So if you are only interested in results modulo a positive scaling factor (and in computer graphics, that is often the case), then in the sandwich product you can replace the inverse by the much cheaper reverse $\widetilde{V}$ (which is merely a grade-dependent sign: a minus sign for the parts of grade 2 and 3).

Versors work in a general geometric algebra. Multiple reflections preserve the dot product between vectors of the representational space; therefore *versors are orthogonal transformations* of that space. By a proper choice of representational space relative to our usual 'task space', those should be precisely the motions we are interested in. In PGA, with its representational linear space in which Euclidean planes are represented as vectors, the Euclidean motions will thus be represented as orthogonal transformations.

6.2 One Planar Reflection

Let us reflect a general point $X = (1 - e_0\mathbf{x})\,\mathbf{I}_3$ in a general plane $p = \mathbf{n} - \delta e_0$ (where we normalize $\mathbf{n}^2 = 1$ for convenience). The inverse of that plane p is the plane itself, since $p^2 = 1$.

$$\begin{aligned}
X' &\equiv -p\,X\,p^{-1} \\
&= (\mathbf{n} - \delta e_0)\,(1 - e_0\mathbf{x})\,\mathbf{I}_3\,(\mathbf{n} - \delta e_0) \\
&= (\mathbf{n} - \delta e_0)\,(1 - e_0\mathbf{x})\,(\mathbf{n} + \delta e_0)\,\mathbf{I}_3 \\
&= (\mathbf{n} - \mathbf{n}\,e_0\,\mathbf{x} - \delta e_0)\,(\mathbf{n} + \delta e_0)\,\mathbf{I}_3 \\
&= (\mathbf{n}^2 - \mathbf{n}\,e_0\,\mathbf{x}\,\mathbf{n} - \delta e_0\,\mathbf{n} + \mathbf{n}\,\delta e_0)\,\mathbf{I}_3 \\
&= \big(1 - e_0\,(2\delta\mathbf{n} - \mathbf{nxn})\big)\,\mathbf{I}_3 \\
&= \Big(1 - e_0\,\big(2(\delta - \mathbf{x}\cdot\mathbf{n})\,\mathbf{n} + \mathbf{x}\big)\Big)\,\mathbf{I}_3 \\
&\equiv (1 - e_0\,\mathbf{x}')\,\mathbf{I}_3.
\end{aligned} \tag{78}$$

The two ways of spelling out the result are illustrated in Figure 6. It shows that the simple 'algorithm' $X' = -p\,X\,p^{-1}$ indeed reflects the point X properly.

Since reflection is a linear operation, and therefore can be extended as an outermorphism, a more basic approach would have been to reflect a general plane; then the reflection of the trivector of point X would have followed.

So let us reflect one plane in another. Without loss of generality we can take the origin on our reflecting plane, so that it takes the convenient form $p = \mathbf{n}$. Then the reflection of a plane $\mathbf{m} - \delta e_0$ is

$$\begin{aligned}
-p\,(\mathbf{m} - \delta e_0)\,p^{-1} &= -\mathbf{n}\,\mathbf{m}\,\mathbf{n}^{-1} + \delta\,p\,e_0\,p^{-1} \\
&= -\mathbf{n}\,\mathbf{m}\,\mathbf{n}^{-1} - \delta\,e_0.
\end{aligned} \tag{79}$$

The normal vector is reflected, but the distance (from our reference point on p along the new normal) does not change. A parallel plane with $\mathbf{m} =$

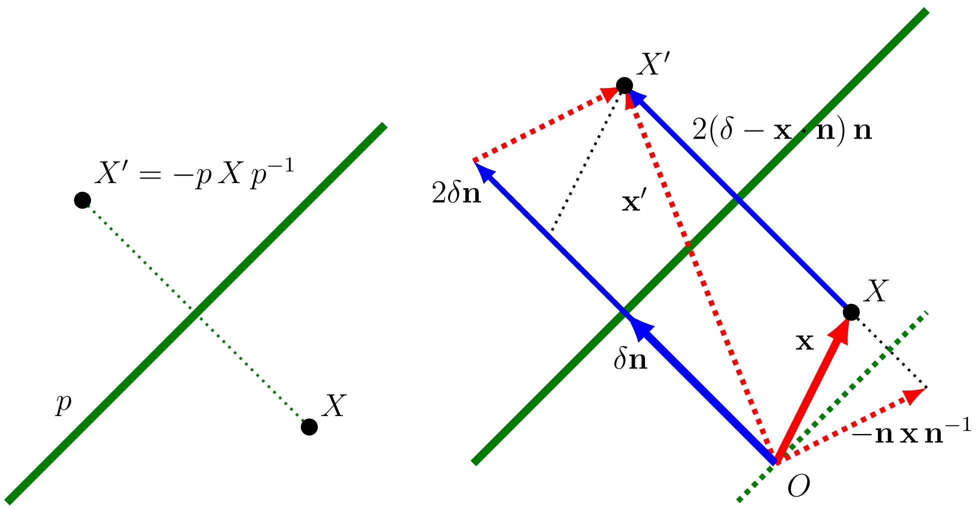


Figure 6: *A reflection of a point X into a single plane p (see on edge, in green) is in PGA simply computed as $X' = -p\,X\,p^{-1}$ (left). When worked out in detail, that simple equation implements some fairly intricate geometry. The resulting position $\mathbf{x}'$ can be made by adding a reflection $-\mathbf{n}\,\mathbf{x}\,\mathbf{n}^{-1}$ to twice the support vector $\delta\mathbf{n}$, or by twice adding, to the point position $\mathbf{x}$, a relative distance vector $(\delta - \mathbf{x}\cdot\mathbf{n})\,\mathbf{n}$. The result is the same, but the algebraic 'code' is much more compact in PGA.*

$\mathbf{n}$ becomes $-\mathbf{m} - \delta e_0$, so with opposite normal and at the other side of the mirror. If we conveniently take our arbitrary origin along the line of intersection of the planes, then $\delta = 0$ and we see that this common line remains invariant under reflection.

We emphasize that we should not just omit the minus sign in the definition. If we reflect a plane parallel to the mirroring plane, we would expect its normal to flip sign, so that 'back' and 'front' are interchanged after reflection. With the minus sign as indicated, this is precisely what happens:

$$p \quad \mapsto \quad -\,p\,p\,p^{-1} = -p. \tag{80}$$

And of course a plane p' perpendicular to the mirror (so with $p' \cdot p = 0$) retains its normal, as it should: $-p\,p'\,p^{-1} = p\,p^{-1}\,p' = p'$.

Many other sources on PGA unfortunately tend to neglect this sign in the basic projection (perhaps because of the projective origins of PGA's use of the

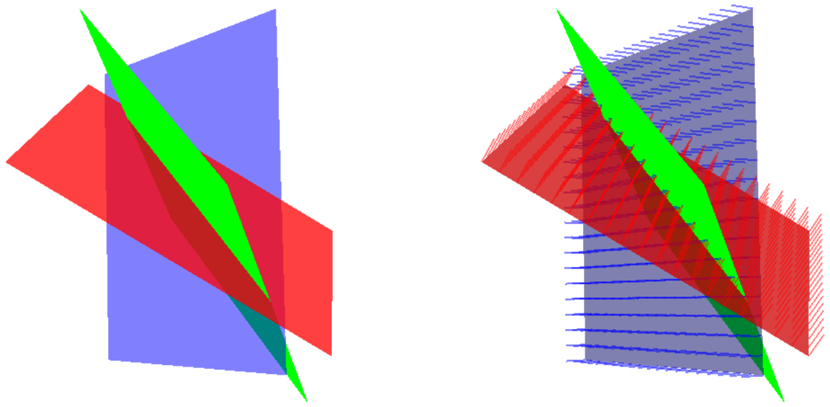


Figure 7: *Left: the red plane is reflected in the green plane, to produce the blue plane. Right: in oriented geometry, the oriented red plane, with normal vectors denoting its outside 'front', reflects to become the oriented blue plane. The orientation of the green mirror is irrelevant in the reflection formula, and not indicated.*

vanishing elements?). This is a pity, since with just a bit more care we obtain a richer and more usefully consistent *oriented geometry*. As we remarked before, you may be interested only in results modulo a positive scale factor – then you can use $\widetilde{p} = p$ instead of p^{-1}, to save some computational effort. But do *not* get into the habit of leaving out the minus sign in equations like eq.(80).[16]

For completeness, let us check that the vanishing elements work as expected under reflection. First, the vanishing plane e_0 (the plane at infinity) cannot be used to perform a reflection $X \mapsto e_0 X e_0^{-1}$, since e_0^{-1} does not exist (after all, $e_0^2 = 0$). Reflecting something finite in the vanishing plane e_0 would have gotten us beyond infinity, so it makes sense that we cannot. Second, we can reflect e_0 in the plane p, but the result is e_0, since $-pe_0p^{-1} = e_0pp^{-1} = e_0$. You cannot reflect back from infinity: it is invariant. The element e_0 thus

[16]Actually, there is a subtlety here: onde could also defend that without the minus sign, we describe the reflection of an 'internally oriented' plane. Elements constructed by a `join` are of this type: a `join` line between two points is reflected a a `join` line between the reflected points; a meet line reverses its orientation (as it must to encode the proper reflected rotation around it. We are currently (2022) investigating this possibility, initial insights in [1].

acts algebraically just as we would have expected geometrically.

6.3 Reflections in Two Planes

The geometric product of two planes describes a double reflection. Let us study what form such a product can take. We can take normalized planes without loss of generality (since in the sandwich product we will use to apply the result to elements, the inverse that also occurs cancels any scale factor).

$$\begin{aligned} p_2\, p_1 &= (\mathbf{n}_2 - \delta_2 e_0)\,(\mathbf{n}_1 - \delta_1 e_0) \\ &= (\mathbf{n}_2 \cdot \mathbf{n}_1) + \mathbf{n}_2 \wedge \mathbf{n}_1 + e_0\,(\delta_1 \mathbf{n}_2 - \delta_2 \mathbf{n}_1). \end{aligned} \tag{81}$$

There are some cases:

- **Translation:** If the planes are parallel, with $\mathbf{n}_1 = \mathbf{n}_2 = \mathbf{n}$ as common unit vector, then the above evaluates to the *translator*

  $$T_\mathbf{t} \equiv 1 + (\delta_1 - \delta_2) e_0 \wedge \mathbf{n} \equiv 1 - e_0 \mathbf{t}/2, \tag{82}$$

  with $\mathbf{t}$ defined as $2(\delta_2 - \delta_1)\mathbf{n}$, double the separation vector of the planes. In the sandwiching with this element $T_\mathbf{t}$, any element translates over $\mathbf{t}$. Just to show how this produces the prefix operators we have seen before, let us translate the point at the origin in d-dimensional space, represented by $\mathbf{I}_d$:

  $$\begin{aligned} (1 - e_0\mathbf{p}/2)\,\mathbf{I}_d\,(1 - \mathbf{p}e_0/2) &= (1 - e_0\mathbf{p}/2)\,\big(\mathbf{I}_d + \mathbf{p}(-1)^d \mathbf{I}_d e_0/2\big) \\ &= (1 - e_0\mathbf{p}/2)\,(1 - e_0\mathbf{p}/2)\,\mathbf{I}_d \\ &= (1 - e_0\mathbf{p})\,\mathbf{I}_d, \end{aligned} \tag{83}$$

  retrieving the point representation of eq.(13). On the other hand, translating an origin plane $\mathbf{n}$ gives

  $$\begin{aligned} (1 - e_0\mathbf{p}/2)\,\mathbf{n}\,(1 - \mathbf{p}e_0/2) &= \mathbf{n} - e_0\,\mathbf{p}\,\mathbf{n}/2 - \mathbf{n}\,e_0/2 + e_0\,\mathbf{p}\,\mathbf{n}\,\mathbf{p}\,e_0/4 \\ &= \mathbf{n} - (\mathbf{p} \cdot \mathbf{n})\,e_0 \end{aligned} \tag{84}$$

  which indeed retrieves the plane representation $\mathbf{n} - \delta e_0$, with δ the signed distance from the origin to the plane along the normal vector $\mathbf{n}$.

  A translator can alternatively be made as the ratio of two points, see Exercise 12.2:10, or of two parallel lines.

- **Rotation at the Origin:** Planes through the origin are not geometrically special (because the origin is not a geometrically significant point). But let us see what kind of a motion two planes without distance δ_i produce:

$$R_{\phi\mathbf{I}} \equiv p_2\, p_1 = \mathbf{n}_2 \cdot \mathbf{n}_1 + \mathbf{n}_2 \wedge \mathbf{n}_1 = \cos(\phi/2) - \sin(\phi/2)\,\mathbf{I}. \tag{85}$$

  This is a purely Euclidean *rotor* for the rotation over ϕ, double the angle $\phi/2$ between the planes. Note that $\mathbf{I}^2 = -1$.

  We can decompose $\mathbf{I}$ on the coordinate 2-blade basis $\{\mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}\}$, each of which squares to -1, and which have combination properties like $\mathbf{e}_{23}\mathbf{e}_{31} = \mathbf{e}_{21} = -\mathbf{e}_{12}$. In fact, setting $\mathbf{i} = \mathbf{e}_{23}$, $\mathbf{j} = \mathbf{e}_{31}$, $\mathbf{k} = \mathbf{e}_{12}$, the algebraic properties are isomorphic to those of a *quaternion.* So the real 3D rotation operation $R_{\phi\mathbf{I}}$ is algebraically like a quaternion. But its derivation as the product of two planes feels quite different – and a lot more real and natural. We did not need to introduce complex numbers to define it, just ratios of real planes through the origin.

- **Rotation around a Line:** Rotation around an arbitrary normalized line L is also possible. To see the simple form it takes, let us construct it by a translation back to the origin, then a rotation at the origin, and placing it back again, so making $T_{\mathbf{t}}\, R_{\phi\mathbf{I}} T_{\mathbf{t}}^{-1}$. But the bivector $\mathbf{I}$ around which this rotates should of course be parallel to the original line L. Or conversely, L is the $\mathbf{t}$-translated version of $\mathbf{I}$, so $L = T_{\mathbf{t}}\, \mathbf{I}\, T_{\mathbf{t}}^{-1}$. Performing the translation, the resulting operation is effectively the translated version of the rotor $R_{\phi\mathbf{I}}$, and can be expressed directly in terms of the line L as

$$R_{\phi L} \equiv \cos(\phi/2) - \sin(\phi/2)\, L. \tag{86}$$

  So in 3D PGA we can simply use a line L to produce a rotation around it.[17] Note that $L^2 = -1$, just as $\mathbf{I}^2 = -1$.

  This is in fact algebraically isomorphic to how a *dual quaternion* (or *biquaternion*) represents a 3D rotation. It uses elements from the full 2-blade basis $\{\mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}, \mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03}\}$ (where we rewrote e_0 as $\mathbf{e}_0$ for symmetry and compactness). The three elements involving $e_0 = \mathbf{e}_0$

[17] In the next section, we will dare to rewrite this in exponential form as $R_{\phi L} \equiv \exp(-\phi\, L/2)$.

square to 0, the others square to -1, and their mutual product properties are precisely right for the dual quaternion functionality (as you may verify in the Cayley table of Figure 1).

Dual quaternions therefore do not need to be introduced as doubly mysterious entities, over an abstract algebra that contains both three complex numbers and three dual numbers, somehow magically applicable to 3D motions. The $R_{\phi L}$ above is simply a rotation operation around a real line L, obtainable as the product of two real reflection planes having that line as their intersection. When such a product of planes is applied in a sandwich product, it produces a rotation on any element. When the planes are parallel, the rotation is around a vanishing line – we would usually call that a translation. Get real!

6.4 Three Reflections in 3D; Point Reflection

This case contains just one extra reflection beyond the previous case, and is of less interest on a first pass through PGA.

We could study some things like glide motions, and factorize them ingeniously. In 2D such things have been done in the context of planar symmetry groups. In 3D, this would produce the crystallographic groups. It is an interesting exercise to rewrite the CGA treatment of crystallographic groups from [15] into PGA!

There is a special case, though, that deserves attention: if the three reflecting planes are perpendicular, their geometric product is identical to their outer product. Therefore this is a *reflection in a point.* Devices with such three orthogonal planes are called retroreflectors - Apollo 11 put one on the Moon, to determine its distance by bouncing laser rays off it, from the Earth.

We can most easily study the point reflection at the origin, for the point $O = \mathbf{I}_d$ (yes, we do the d-dimensional case). Then a general blade of the form $X = \mathbf{T}_X + e_0\,\mathbf{P}_X$ of grade x reflects as

$$\begin{aligned} X &\mapsto (-1)^{xd}\,\mathbf{I}_d\,X\,\widetilde{\mathbf{I}}_d \\ &= (-1)^{xd}\,\mathbf{I}_d\,(\mathbf{T}_X + e_0\,\mathbf{P}_X)\,\widetilde{\mathbf{I}}_d \\ &= (-1)^{xd}\,\big((-1)^{x(d+1)}\,\mathbf{T}_X + (-1)^{d+(x-1)(d+1)}\,e_0\,\mathbf{P}_X\big) \\ &= (-1)^{x}\,(\mathbf{T}_X - e_0\,\mathbf{P}_X). \end{aligned} \tag{87}$$

By factoring out the scaling by $(-1)^x$, we see that the positional aspect has changed sign: so the result is a similar blade, but at the opposite side of O, and for odd X also oriented in the opposite direction. We also see that the point reflection is a way of producing points of negative weight: in odd-dimensional spaces like 3D, reflection the point $X = O = \mathbf{I}_3$ in O gives $-O$. Even elements (i.e., even x) retain their orientation sign after point reflection.

For those of you familiar with the grade involution $\widehat{X} \equiv (-1)^x X$, the point reflection result can be written as involution of the Euclidean parts only: $\mathbf{T}_X + e_0\,\mathbf{P}_X \;\mapsto\; \widehat{\mathbf{T}_X} + e_0\,\widehat{\mathbf{P}_X}$.

6.5 Four Reflections: General Euclidean Motions (Oh and Incidentally Dual Quaternions)

According to Cartan-Dieudonné, in the 4D representational space of 3D PGA, four reflections will generate the most general orthogonal transformation. So using the GA representation, the product of four general plane vectors should produce a versor for the most general rigid body motion. We can rearrange these terms in different ways. In the usual classical representation when using homogeneous coordinates, one typically views the general motion as a rotation at the origin followed by a translation. In PGA, this would be (reading right to left) a factorization in terms of two planes intersecting in a line through the origin, followed by two planes intersecting in a vanishing line.

$$\big((\mathbf{t} - e_0/2)\,\mathbf{t}\big)\,\big(\mathbf{n}_2\,\mathbf{n}_1\big) \quad \propto \quad (1 - e_0\mathbf{t}/2)\,(\cos(\phi/2) - \sin(\phi/2)\,\mathbf{I}) \tag{88}$$

with $\phi\mathbf{I}$ the bivector angle from $\mathbf{n}_1$ to $\mathbf{n}_2$. The final form of representation is unique, though the four vector factors can be chosen with quite some leeway (as long as they produce the same $\phi\mathbf{I}$ and $\mathbf{t}$). The element we have constructed is a rigid body motion versor, often called a *motor*.

But we have just seen how we perform rotations around a general line L, so rotations at the origin are no longer as special as they are in the classical view originating in linear algebra. Therefore we could also characterize the general motion as a rotation around a general line L followed by a translation in the direction of that line (this is in fact Chasles' theorem on decomposition of 3D Euclidean motions). The former is the operation $R_{\phi L}$, the latter translation effectively 'rotates' around the vanishing line $L\mathcal{I}$, so we could denote it as $R_{\delta L\mathcal{I}}$ (we will see the exact reason behind this notation in Section 7). Then the general motor is represented as the product

$$R_{\delta L\mathcal{I}}\, R_{\phi L} = R_{\phi L}\, R_{\delta L\mathcal{I}}. \tag{89}$$

We can use either order, since the two operations commute. This is a natural parametrization of a general motor: by one normalized oriented line L, an oriented angle ϕ and an oriented distance δ. It will take an even more convenient form in the exponential representation, in Section 7. This combination generates a general element on the basis $\{1, \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}, \mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03}, \mathbf{e}_{0123}\}$, with three basis bivector elements squaring to -1, and three squaring to 0. They are, again, dual quaternions, but now much less mysterious as 'real screws'.

6.6 The PGA Square Root of a Motor

Motors can be defined as the ratio of a (normalized) target blade B and a (normalized) start element blade A (obviously of the same grade), so as B/A – or as products of such ratios. (This uses the division by a blade, which equals multiplication by its inverse $A^{-1} = \widetilde{A}/(A\widetilde{A})$; for a unit blade therefore simply equal to its reverse.) However, B/A is not the motor that can be used in sandwiching to turn A into B, since it moves over twice the required transformation. This deficiency is easily seen (taking normalized blades for convenience): $(BA^{-1})A(AB^{-1}) = BAB^{-1} \neq B$.

To move A to become B, we actually need $V = \sqrt{B/A}$, as a sketchy but plausible computation shows: $\sqrt{BA^{-1}}A\sqrt{BA^{-1}}^{\,-1} = \sqrt{BA^{-1}}A\sqrt{A^{-1}B} = B$. (We can later improve on this derivation when we have the exponential and logarithm of a motor, but this will do for now.) Therefore we need the capability to take the square root of a motor.

In PGA, the square root of a normalized motor M (i.e., a motor that satisfies $M\,\widetilde{M} = 1$) is

$$\text{square root of motor } M\text{:} \qquad \sqrt{M} = \frac{1+M}{\sqrt{2(1+\langle M\rangle)}}\left(1 - \frac{\langle M\rangle_4}{2(1+\langle M\rangle)}\right), \tag{90}$$

where $\langle M\rangle$ denotes the scalar part of M, and $\langle M\rangle_4$ the 4-vector part. This was derived in [14] on CGA's roots; but PGA's are the same since it is a subalgebra of CGA. You may verify eq.(90) directly, by squaring and using the normalization condition $M\,\widetilde{M} = 1$ to rearrange terms.

The case of $M = -1$ is excluded, but this is not a problem geometrically: since the action of M and $-M$ is the same we lose no significant meaning by demanding, for instance, $\langle M\rangle \geq 0$ before applying the square root formula.

Apart from a scalar normalization eq.(90) is in essence

$$\sqrt{M} \quad \propto \quad (1+M)\,(1+\langle M\rangle - \tfrac{1}{2}\langle M\rangle_4), \tag{91}$$

which may be more efficient in applications where positive weighting factors are not of interest. You would then use the reverse of this motor $\widetilde{\sqrt{M}}$ at the right hand side of the sandwiching – it is obtained by replacing M by $\widetilde{M}$ in the first factor.

We use the term *simple motor* for a motor that has no grade 4 part. It is typically made as the ratio of two unit vectors. In the exponential form we will treat later, they are characterized as the exponential of a 2-blade (as opposed to a 2-vector). Because of the lack of a grade-4 part, we have for simple motors

$$\text{square root of simple motor:}\ \sqrt{M} = \frac{1+M}{\sqrt{2(1+\langle M\rangle)}}\ \propto\ 1+M. \tag{92}$$

In 2D PGA, all motors are simple (there is literally no space for them not to be!), so this square root formula applies there universally. In spaces of more than 5 dimensions, terms of grade 6 would appear and one would need to generalize the square root of eq.(90). But for now, we will not go there.

Having square roots of motors is not only required for determining the correct motor from a blade ratio; it is also convenient to have when interpolating motions. Each motion can be halved by its square root, and repeated halving gives a simple form of interpolation by subdivision. A more general interpolation, employing n-th roots, can be performed after we have determined motor logarithms in Section 7.3.

6.7 Example: Universal Motors

As a simple demonstration of the universality of PGA when acting on geometric elements, consider Figure 8. It contains a green triangle T with three vertices, the trivector points T_1, T_2 and T_3. There is a yellow plane p, constructed from a point Q and a normal vector $\mathbf{n}$ as $p = (\mathbf{n}\cdot Q)/Q$. A red line L (normalized) is running in direction $\mathbf{u}$, so it can be constructed from a point on it and the vanishing point $\mathbf{u}\mathcal{I}_3$ (as in eq.(38)).

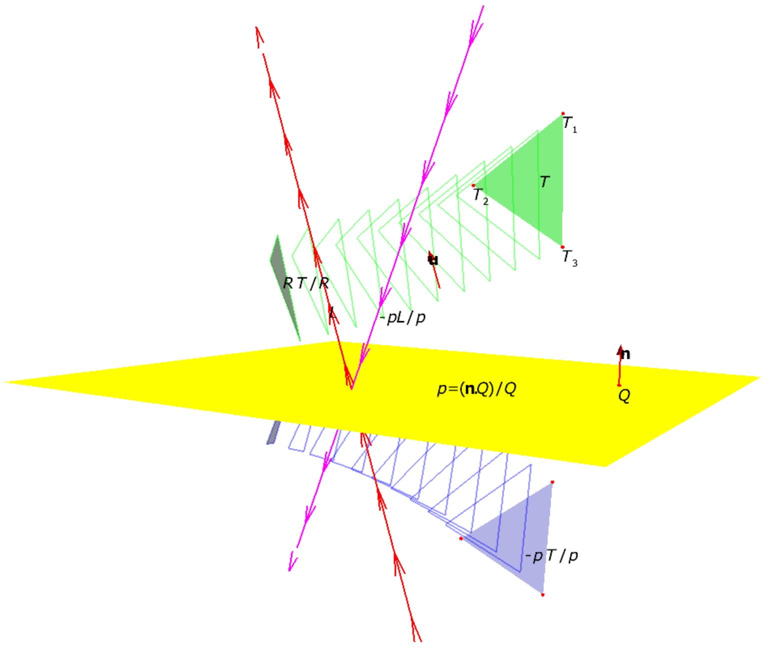


Figure 8: *A simple example of PGA versors in action, in a GAviewer visualization.*

The green triangle is rotated around line L by a rotation motor $R = \exp(-\phi L/2)$. In PGA, this is done by rotating each of its points T_i to become RT_i/R, and recomposing the result. You may simply overload R to work on the triangle structure specifying its vertices (PGA trivectors) and edges (PGA 2-blades), so we may denote this as $R\,T/R$. We have interpolated the rotation using as a motor the n-th root of R, i.e., $\sqrt[n]{R} = R^{1/n} = \exp(\log(R)/n)$, using the logarithm that will be treated in Section 7.3 - although you could also use repeated square roots to iteratively halve the motion.

The plane p is capable of reflecting the green triangle T to become the blue triangle $-p\,T/p$. In fact, we can reflect all green triangles in this manner. The blue triangle has then effectively been rotated around the reflected line $-p\,L/p$, by the motor $\exp(-\phi(-p\,L/p)/2) = -p\,(\exp(-\phi L/2))/p = -p\,R/p$. So motors obey the same versor transformation rules as geometrical primitives. *The motor of a reflection is a reflection of the motor.* With a suitable data structure, we can therefore apply the sandwiching operation by p simply to *all* elements ‘above’ the plane, to get the elements ‘below’ - even including the counterrotation that acts there.

You can view this setup flexibly in 3D and play with it (by changing the defining geometrical parameters) in `ganja` at `https://enkimute.github.io/ganja.js/examples/coffeeshop.html#chapter11_motors`. Since the code is javascript, you can modify the scene, confirming how easy it is to transform in PGA.

6.8 Example: Reconstructing a Motor from Exact Point Correspondences

Let us assume that we have three normalized labeled points A, B, C (not on a line), and that we know where an exact motor (i.e., rigid body motion) sends each of them, to A', B', C', with the same weights. We would like to reconstruct that motor. Here is how we could proceed:

1. First align A to A'. This is done by the motor
$$V_A = \sqrt{A'/A}, \tag{93}$$
which thus achieves $A' = V_A\, A\, \widetilde{V}_A$ (it is of course a translation, see Exercise 12.2:10). When we apply this motor to B' and C', they have become slightly better aligned, and now only the new relative positions to A' matter. The new subproblem thus involves aligning $B_A \equiv V_A\, B\, \widetilde{V}_A$ with B' and $C_A \equiv V_A\, C\, \widetilde{V}_A$ with C'.

2. Now align B_A with B' while preserving the result on A. That requires an additional motor that aligns the line $A' \vee B_A$ with the line $A' \vee B'$, through
$$V_B = \sqrt{(A' \vee B')/(A' \vee B_A)}. \tag{94}$$
This motor V_B leaves A' invariant, and rotates B_A into B'.

   The total motor so far is then $V_B\, V_A$, which aligns both A and B to A' and B', and transforms C to $C_{BA} \equiv V_B\, V_A\, C\, \widetilde{V}_A\, \widetilde{V}_B$ (or C_A to $C_{BA} \equiv V_B\, C_A\, \widetilde{V}_B$).

3. Finally, align C_{BA} to C' by aligning the plane $A' \vee B' \vee C_{BA}$ with $A' \vee B' \vee C'$, keeping the line $A' \vee B'$ and the point A' invariant. This requires the motor
$$V_C = \sqrt{(A' \vee B' \vee C')/(A' \vee B' \vee C_{BA})}. \tag{95}$$

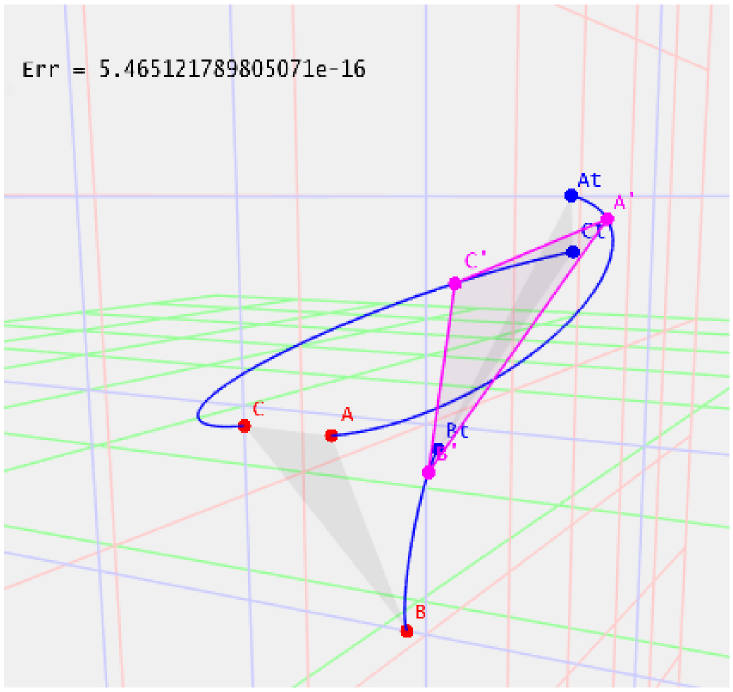

Err = 5.465121789805071e-16

Figure 9: *A screenshot of the motor estimation algorithm implemented in* `ganja`.

The motor V_C leaves both A' and B' invariant, and rotates C_{BA} to C' around the line $A' \vee B'$.

4. The total alignment motor is now $V = V_C\, V_B\, V_A$. Done!

At each step, only one new element needs to be aligned; by decomposing the alignment blade as a geometric product of orthogonal factors (which can always be done) you can show that the ratio boils down to a ratio of vectors. Therefore the motors involved in each step are simple motors: pure translations and pure rotations around an axis (no screw motions). The square root of each of the normalized motors M is thus simply the normalization of $(1 + M)$.

A free sample of this algorithm may be found in the `ganja` coffeeshop: `https://enkimute.github.io/ganja.js/examples/coffeeshop.html#chapter11_motor_reconstruction`. Figure 9 shows a screenshot.

Very similar algorithms to the above can be designed to find the motor that transforms three corresponding plane pairs (easy), or three line pairs (harder!).[18] See our structural exercise 12.2:11.

6.9 Summary

General Euclidean transformations can be generated by multiple reflections in planes. In PGA this produces *versors*, the usual geometric algebra of orthogonal transformations.

We explicitly derived the versors for some of the basic motions, and connected those real constructions briefly to the complex encoding by quaternions or dual quaternions which is more commonly taught. Incidentally we saw that in PGA a *pure rotation* is a rotation around any line, rather than around a line through the origin (as it is in linear algebra).

We showed how versors are applied universality to all elements, primitives and operators alike, by a simple sandwiching construction eq.(77). This enormously simplifies and reduces implementation code.

Versors are easy to make from elements by geometric product ratios. But the ratio of elements does not quite give the versor transforming one to the other, you need to take a square root. So we exposed the square root formula for PGA in eq.(90), including the simplified form eq.(91) it takes when magnitudes are less important in your application.

[18]In [16], a structurally identical algorithm was used to perform alignment of directions in 3D (so using only the rotors in the 3D GA of Chapter 10 in [2]). It resulted in an efficient formula that was directly usable in the more traditional quaternion formulation, and that appeared to be new. The geometrical approach, viewing a quaternion as the ratio of two planes, paid off! It may be possible to derive a similar closed form expression for our final result, giving a compact formula for the dual quaternion that aligns the triple of points (though it appears at first try that it has too many terms to be memorable).

7 The Exponential Representation

We have constructed the rigid body motions as products of planes, or ratios of flats. But in practice it is often more convenient to synthesize them as exponentials, since that parametrizes them straightaway by geometrically relevant elements: direction of motion for a translation, axis and angle for a rotation, and screw and pitch for a general rigid body motion. This approach works for the 'proper' motions, not for the reflections. So in this section our versor contains an *even* number of terms, and is thus an even multi-vector, a member of the even subalgebra $\mathbb{R}^{+}_{3,0,1}$ of 3D PGA.

7.1 Exponentiation of Invariants Gives Motions

We have seen how a rotation versor can be formed as the product of two reflection planes $V = p_2\, p_1$, and that it rotates around their common line. That line is represented as $L = p_1 \wedge p_2$, and it would be nice if we could write the versor in terms of it: for many pairs of planes could have that unique line L in common, and we prefer our parametrizations as unambiguous as possible. In geometric algebra in general, we can perform exponentiation (as explained in Chapter 7 of [2]) of bivectors to produce even versors. (This is intimately related to the classical description of motions through their Lie algebra, but we will just absorb this rather than spell it out, referring to e.g., [17] for details.)

We will employ the standard notation of even versor application, in which a normalized bivector B produces a versor $V_\tau = \exp(-\tau\, B/2)$. (This versor is even a *rotor*, since $V\widetilde{V} = 1$, and therefore $V^{-1} = \widetilde{V}$.) This versor V_τ is applied to an arbitrary element X by a sandwich product, to produce an orbit parametrized by τ as

$$X_\tau = V_\tau\, X\, V_\tau^{-1} = e^{-\tau B/2}\, X\, e^{\tau B/2}. \tag{96}$$

As we have seen when constructing versors by multiple reflections, such a transformation is linear in X, and it preserves grades. Moreover, versors transform the geometric product *covariantly*:

$$V\,(a\,b)\,V^{-1} = (V\,a\,V^{-1})\,(V\,b\,V^{-1}), \tag{97}$$

and since outer and inner products are linear combinations of geometric products, it does the same for them. Products involving duality (like the `join`) also transform covariantly under even versors, since they have determinant $+1$, and therefore do not affect the sign of the pseudoscalar. The sandwiching product is thus an obviously structure-preserving way of transforming elements.

Let us show such exponentiation for rotations around the line $L = p_1 \wedge p_2$. We normalize L such that $\|L\| = 1$. Such a line squares to $L^2 = -1$. This is most easily seen if we construct it as the intersection of two orthogonal normalized planes (so $p_1^2 = p_2^2 = 1$ and $p_1 \cdot p_2 = 0$). Then $p_1 \wedge p_2 = p_1\, p_2$, and

$$L^2 = (p_1 \wedge p_2)\,(p_1 \wedge p_2) = p_1\, p_2\, p_1\, p_2 = -p_1\, p_1\, p_2\, p_2 = -(p_1)^2\,(p_2)^2 = -1. \quad (98)$$

Therefore using L in an exponentiation produces trigonometric functions, in the usual way:

$$\begin{aligned} \exp(-\phi\, L/2) &= 1 - \tfrac{1}{1!}\phi L/2 + \tfrac{1}{2!}(\phi L/2)^2 - \tfrac{1}{3!}(\phi L/2)^3 + \cdots \\ &= \cos(\phi/2) - \sin(\phi/2)\, L. \end{aligned} \quad (99)$$

Note that the line L remains invariant under this operation, and that *the motor is the exponential of the invariant*. That turns out to be very useful general principle to construct specific motors!

Translation versors (of the form of eq.(82)) involve direction elements like $e_0\mathbf{t}$, which can be seen as a vanishing line, the locations where the plane $\mathbf{t}$ (with normal vector $\mathbf{t}$) meets the vanishing plane e_0. Such elements square to zero, since $e_0^2 = 0$:

$$(e_0\mathbf{t})^2 = e_0\, \mathbf{t}\, e_0\, \mathbf{t} = -\mathbf{t}^2\, e_0^2 = 0. \quad (100)$$

Therefore we can also make translation versors by exponentiation

$$\begin{aligned} \exp(-e_0\mathbf{t}/2) &= 1 - \tfrac{1}{1}e_0\mathbf{t}/2 + \tfrac{1}{2}(e_0\mathbf{t}/2)^2 + \cdots \\ &= 1 - e_0\mathbf{t}/2, \end{aligned} \quad (101)$$

completely conforming to the form they took when constructed from reflection in two parallel planes in eq.(82).

When translating orthogonally to a plane $p = \mathbf{n} - \delta e_0$ with normal vector $\mathbf{n}$, the vanishing line $e_0\mathbf{n} = e_0 p$ remains invariant. And the corresponding translation operator is indeed the exponential of this invariant: $\exp(-e_0 p)$.

Composing a general rigid body motion from an origin rotation (with $L = \mathbf{I} = \mathbf{e}_1\,\mathbf{e}_2$ on an orthonormal basis) followed by a translation yields a versor that is commonly called a *motor*:

$$\begin{aligned} M &= \exp(-e_0\mathbf{t}/2)\,\exp(-\phi\mathbf{I}/2) \\ &= (1 - e_0\mathbf{t}/2)\,(\cos(\phi/2) - \sin(\phi/2)\,\mathbf{I}) \\ &= \cos(\phi/2) - \sin(\phi/2)\,\mathbf{I} - e_0\mathbf{t}/2\,\cos(\phi/2) + \sin(\phi/2)\,e_0\,\mathbf{t}\,\mathbf{I}/2. \quad (102) \end{aligned}$$

A general motor M therefore contains grades 0, 2 and 4. Note that you never have to write out such expressions explicitly, we just worked this out in components to give you a feeling for the internal structure. But we have already mentioned that there is a better parametrization of motors, according to Chasles' decomposition, in eq.(89): a rotation around a line, combined with a translation along that same line. In exponential form, that becomes

$$M = R_{\delta L\mathcal{I}}\,R_{\phi L} = \exp(\delta L\mathcal{I}/2)\,\exp(-\phi L/2) = \exp(-\phi L/2 + \delta L\mathcal{I}/2), \quad (103)$$

where the combination into a single exponent is permitted since the L and $L\mathcal{I}$ commute. Both the Euclidean line L and the vanishing line $L\mathcal{I}$ are invariant under this rotor M, so this is again of the form 'exponential of the invariant'. The ratio δ/ϕ is the *pitch* of the resulting screw motion, indicating the amount of translation per turn.

7.2 Child's Play

Let us enjoy an intuitive example showing how the continuity of our experience in rotation and translation is faithfully represented in PGA, and how natural it is to exponentiate geometrically invariant elements to produce motors.

Mia-Lou wants to ride her tricycle. She starts at the origin, facing in the $\mathbf{e}_1$ direction. Let us denote the plane perpendicular to the ground plane $\mathbf{e}_3$ and containing her rear axle by r, and the plane perpendicular to the ground plane containing her front axle by f. The latter plane depends on her steering angle ϕ, so let us denote it as $f(\phi)$. With the trike assumed to be of unit length, we have in coordinates:

$$r = \mathbf{e}_1 \quad \text{and} \quad f(\phi) = (\mathbf{e}_1\,\cos\phi + \mathbf{e}_2\,\sin\phi) - e_0\cos\phi. \quad (104)$$

The two planes meet in $r \wedge f(\phi)$, and this is the center of her motion: it stays invariant as she moves with constant steering angle ϕ. Therefore Mia-Lou's

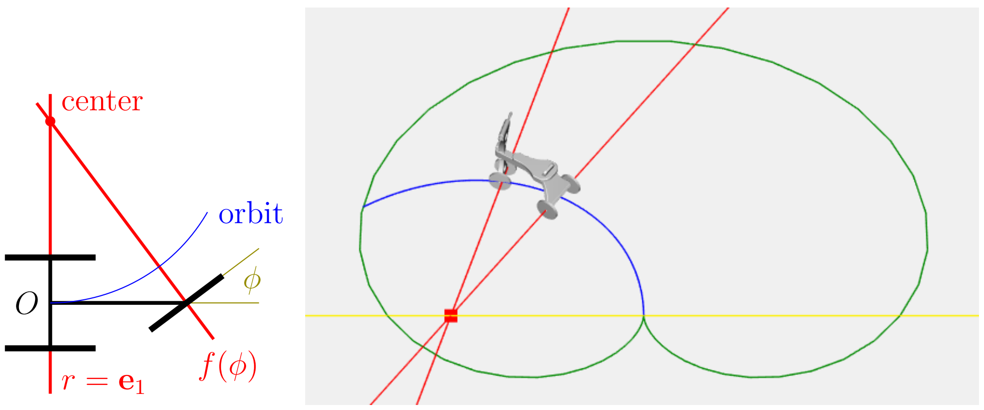


Figure 10: *Geometry and orbits of a tricycle. When moved one time-unit under the motor $M_\phi = \exp(r \wedge f(\phi)/2)$, the endpoints $M_\phi O \widetilde{M}_\phi$, for various steering angles ϕ, appear to form a cardioid.*

motor is a power of:

$$\exp(r \wedge f(\phi)) = \exp(\mathbf{e}_1\mathbf{e}_2 \sin\phi - \mathbf{e}_1 e_0 \cos\phi). \tag{105}$$

We will not normalize the bivector argument and then introduce an angle measure, but rather take the magnitude of the meet as the amount of motion (which can be further parametrized by a time t, as in $\exp(r \wedge f(\phi)t)$, to travel fractional amounts).

It is clear from this formulation that there are no discontinuities in the motor. When $\phi = 0$, the motor is $\exp(e_0\mathbf{e}_1)$, a purely translational motion over 2 units of length. The invariant center is now a vanishing point (at infinite distance to her left or to her right). As ϕ increases, the motion becomes rotational around a finite turning center at her left, with an ever decreasing turning radius. Then, when Mia-Lou has her handlebars at an angle of $\pi/2$, the motion is purely rotational around the pivot point on the rear axle, by motor $\exp(\mathbf{e}_1\mathbf{e}_2)$ over an angle of 2 radians. If we motorize her front wheel (so that she can even move when the angle exceeds $\pi/2$), she moves backwards around a turning point at her right. A 180-degree turn of the handlebars has her driving straight back to her father, by the motor $\exp(-e_0\mathbf{e}_1)$.

And the same sequence applies to her steering to the right, by negative angles ϕ. Starting from steering straight ahead, the center of rotation thus moves smoothly from infinity to either her left or right, and even to the point on the rear axle for a pure rotation when $\phi = \pi/2$. At no point is there a need for a conditional statement to switch motion state – the one motor $\exp(r \wedge f(\phi))$ covers all motions without discontinuity.

Figure 10 shows a screenshot of a demo of Mia-Lou's trip. It is a nice geometrical fact that a unit motion in the motor $\exp(r \wedge f(\phi))$ moves the original midpoint of the rear axle to an endpoint that lies on a cardioid. You might try to prove that!

You can play with this demo in `ganja` at the link `https://enkimute.github.io/ganja.js/examples/coffeeshop.html#chapter11_tricycle&fullscreen`. Wait half a minute for it to load; the slider to adjust the steering angle is at the top.

7.3 The PGA Motor Logarithm

As eq.(103) shows, a general motor is the exponential of a bivector. If we want to interpolate the motion, we need that bivector; to retrieve it, given the motor, we require a logarithm. For the non-commuting elements of PGA, this requires some care. First, since a versor V and its negative $-V$ have the same effect, let us assume that our versors have a non-negative scalar part; it makes the essence of the logarithms simpler to express.[19]

A *simple motor* is the exponential of a 2-blade. The logarithm of a simple motor of the form exp(2-blade) is easy to extract by the considering its scalar and 2-blade components: it is proportional to the 2-blade by a factor that can be deduced from the magnitudes of the terms. In PGA, there are two cases:

- $R = \exp(B)$ with $B^2 < 0$ (rotation).
  Then define $\|B\| = \sqrt{-B^2}$, and write $B = \|B\|\check{B}$, so that $\check{B}^2 = -1$. Expand the motor to $R = \cos(\|B\|) + \sin(\|B\|)\check{B}$ and it is then clear that the principal logarithm is, in terms of the k-grade selection operator $\langle\ \rangle_k$,

$$\log(R) = \operatorname{atan}\Big(\frac{\|\langle R\rangle_2\|}{\langle R\rangle_0}\Big)\,\frac{\langle R\rangle_2}{\|\langle R\rangle_2\|} \equiv \operatorname{atan}\Big(\frac{\langle R\rangle_2}{\langle R\rangle_0}\Big), \tag{106}$$

[19]There are some more advanced reasons to discriminate between V and $-V$, in certain applications of multiple connected objects, but we will not use that in this text. Consult Chapter 7 of [2] if you are interested.

where the second rewriting effectively defines a GA atan function on bivector arguments, to return a bivector. In this case, $\log(R)$ is proportional to the invariant rotation axis.

- $T = \exp(B)$ with $B^2 = 0$ (translation).
  We can expand this motor to $T = 1 + B$, so the logarithm is simply

$$\log(T) = \langle T\rangle_2 = T - 1. \tag{107}$$

  This logarithm is proportional to the invariant vanishing line.

But for a general motor, of the form exp(bivector), the basic puzzle is that the individual components of a general bivector do not commute; therefore we cannot write the exponential of their sum as a sum of exponentials (which would give a product of simple motors and make their logarithms easy). The solution is to write a general bivector as the sum of two commuting 2-blades; then we can perform the split, and are left with a product of commuting simple motors. We have already performed this split in Section 5.6, but we are only half done: we should retrieve the 2-blade parts not from B, but from $V = \exp(B)$.

Studying the expansion of the motor in factorized form in terms of the unit lines L and $L\mathcal{I}$ shows how we can proceed:

$$\begin{aligned} V &= \exp(B) \\ &= \exp(\alpha L + \beta L\mathcal{I}) \\ &= \exp(\alpha L)\,\exp(\beta L\mathcal{I}) \\ &= (\cos\alpha + \sin\alpha\, L)\,(1 + \beta L\mathcal{I}) \\ &= \cos\alpha + \sin\alpha\, L + \beta\cos\alpha\, L\mathcal{I} - \beta\sin\alpha\,\mathcal{I} \\ &= \cos\alpha\big(1 + \tan\alpha\, L + \beta\, L\mathcal{I} - \beta\tan\alpha\,\mathcal{I}\big). \end{aligned} \tag{108}$$

In this expansion, almost all information for the logarithm is contained in the bivector part of the versor, especially since we can retrieve the coefficients of its decomposition into commuting 2-blades by eq.(69). We then merely apply some function to the weights to retrieve the actual parameters of rotation and translation. This leads to the *motor logarithm*

$$\begin{aligned}
W &= \langle V\rangle_2/\langle V\rangle_0, & (109)\\
\alpha &= \operatorname{atan}\left(\sqrt{W\cdot\widetilde{W}}\right), & (110)\\
\beta\mathcal{I} &= \frac{W\wedge\widetilde{W}}{2\sqrt{W\cdot\widetilde{W}}}, & (111)\\
L &= \frac{W}{\sqrt{W\cdot\widetilde{W}}}\left(1-\tfrac{1}{2}\frac{W\wedge\widetilde{W}}{W\cdot\widetilde{W}}\right), & (112)\\
\log(V) &= \left(\alpha+\beta\mathcal{I}\right)L. & (113)
\end{aligned}$$

Again, this does not work when $W^2 = {\langle V\rangle_2}^2 = 0$, but we have seen that the logarithm in that purely translational case is extremely simple, see eq.(107). It also fails when $\langle V\rangle_0 = 0$, but then $\alpha = \pi/2$ and the rest of the formulas can use $W = V$. The result of the logarithm is now the sum of the invariant rotation axis and the invariant vanishing line (or 'translation axis').

In 2D PGA, motors are simpler, because they are always already the exponent of a 2-blade (since in the 3-dimensional representation space, all 2-vectors are 2-blades). In 2D PGA the logarithm thus simplifies to

$$W = \langle V\rangle_2/\langle V\rangle_0 \quad\text{and}\quad \log(V) = \operatorname{atan}\left(\sqrt{W\cdot\widetilde{W}}\right)\frac{W}{\sqrt{W\cdot\widetilde{W}}}, \qquad (114)$$

except when $W\cdot\widetilde{W} = 0$ (the translation case); then $\log(V) = V - 1$.

When extracting logarithms, please realize that a versor is always of the form $V = \exp(-B/2)$, so that retrieval of the actual generating B (if that is what you are looking for) requires $B = -2\,\log(V)$.

7.4 Summary

The exponential form of an even versor allows direct and unique parametrization of an elementary (screw) motion by its invariant bivectors (basically, the lines that are not affected).

Conversely, retrieving the invariants requires a logarithm. We derived that in some detail, to a fairly compact final result eq.(109)-eq.(113) The elementary motion can then be interpolated.

8 Oriented Euclidean Geometry

The expressions we compute are not only purely geometrically interpretable in terms of positions and attitudes, but they also have magnitudes and signs. In a purely projective approach to PGA, we would neglect those; but they turn out to give useful quantitative information, and also permit us to define 'inside/outside' and chirality ('handedness') consistently. We have so far drawn little attention to this possibility, since we wanted to show the geometry first.

In image synthesis applications, the neglect of positive weighting factors usually makes good sense: it does not change *where* you draw things. However, the *signs* of products consistently propagate 'direction of travel', 'reflection from the outside', 'consistent normal directions', et cetera, and therefore should be kept for interpretation and generation.

In 3D scene analysis, on the other hand, the numerical factors are highly indicative of quantitative aspects of the scene, related to useful measures and stability: 'do these three points `join` to form a reliable plane', 'is this intersection point sensitive to perturbation', etc.

In all we have done so far, we carefully kept the signs and weighting factors. We now should provide some ways to interpret them. You will find that some of these meanings sound annoyingly hard to remember; we are apparently not too naturally sensitive to such aspects of geometry. But if you want to compute with them in a consistent manner, geometric algebra gives you the tools.

Whichever field you are in, get into the good habit of *not* being sloppy with the signs, even if you do decide to ignore the magnitudes. Signs give you an oriented geometry in which you will not need to reconstruct orientations afterwards, but can carry them along consistently in your computations.

8.1 Ordering

Some of the signs we will obtain are not so much due to the orientations of the objects we combine, but to the order in which we combine them. Let us get that out of the way first, notably for the `meet` and `join` operations.

- **Ordering the `meet`.**
  Since the `meet` is an outer product, the effect of changing the order of

two arguments A and B with grades a and b, respectively, is

$$A \wedge B = (-1)^{ab}\, B \wedge A. \tag{115}$$

Thus a line that is the `meet` of two planes p_1 and p_2 obtains the opposite orientation if we compute it as the `meet` of p_2 and p_1. Our eq.(9) shows that the direction vector of the line $p_1 \wedge p_2$ is the 3D cross product $\mathbf{n}_1 \times \mathbf{n}_2$ of the normal vectors of the planes, *in that order* (for our right-handed coordinate system). The magnitude is the sine of the angle between the normal vectors, which is indeed an odd function, so as the planes 'hinge across' each other, the sign changes.

- **Ordering the `join`.**
  The sign incurred in swapping the arguments of the `join` (let us call it the '*swap sign*') depends on the dimension of the space it is computed in. For PGA, that space is $(d+1)$-dimensional, and we have

  $$A \vee B = (-1)^{(d+1-a)(d+1-b)}\, B \vee A. \tag{116}$$

  Thus a line that is the `join` of two points P and Q (which are d-blades of PGA) satisfies $P \vee Q = -Q \vee P$, in any dimension. A line can thus be given an orientation by joining points in a chosen order. But beware: those points themselves should be translated versions of the point at the origin $\mathbf{I}_d$, i.e., they should be *positively proportional* to the Euclidean pseudoscalar that was chosen for the space. Since negative points can result from operations (such as reflection or `meet`), this may be an issue in your subsequent constructions.

Looking at these swapping signs, we thus find that in 3D PGA (so for $d = 3$), the swap signs of `meet` and `join` are the same. Note that any combination involving a 3D line L (which is a 2-blade) will be symmetric:

$$A \wedge L = L \wedge A \;\text{ and }\; A \vee L = L \vee A, \;\text{ for a line } L. \tag{117}$$

For quick reference, we show the swap signs for `meet` and `join` in Table 3. Where these signs are negative, we have to be careful in how we combine the elements, since the signs we obtain will be partly non-geometrical, merely caused by the way we phrased the question of 'how they met', and 'when they joined'. That does not make the answers useless, but definitely potentially confusing. In 3D the rule is: *only if both arguments are odd, there will be a minus sign in the swapping for both* `join` *and* `meet`.

$\wedge, \vee$	0	1	2	3	4
0	$+$	$+$	$+$	$+$	$+$
1	$+$	$-$	$+$	$-$	$+$
2	$+$	$+$	$+$	$+$	$+$
3	$+$	$-$	$+$	$-$	$+$
4	$+$	$+$	$+$	$+$	$+$

$\vee_{2D}$	0	1	2	3
0	$-$	$+$	$-$	$+$
1	$+$	$+$	$+$	$+$
2	$-$	$+$	$-$	$+$
3	$+$	$+$	$+$	$+$

Table 3: *Signs incurred when swapping the arguments of n-D* `meet` *and 3D* `join`*, and of the 2D* `join` *(denoted $\vee_{2D}$). The signs for the 2D* `meet` *are as for the 3D* `meet`*.*

8.2 A Line and a Plane Meet

When (in 3D) a line L and a plane p meet in a point, the sign of their `meet` is independent of their order. So that sign has a geometric significance independent of convention. Moreover, though the geometric location of the common point depends on the precise location of line and plane, both the scalar factor (containing its 'strength' of meeting) and its sign (denoting the 'kind' of incidence) do not depend on their location, but only on their relative attitude. We can therefore most easily acquire an intuition of the sign and magnitude of the `meet` by simply computing how it looks at the origin.

So we calculate the `meet` of a line with direction $\mathbf{u}$ through the origin (which is $L = \mathbf{u}\mathbf{I}_3$), with a plane with normal $\mathbf{n}$ through the origin (which is $p = \mathbf{n}$). Let us take both as being normalized, so $\mathbf{u}^2 = 1$ and $\mathbf{n}^2 = 1$. Then we compute

$$p \wedge L = \mathbf{n} \wedge (\mathbf{u}\mathbf{I}_3) = (\mathbf{n} \cdot \mathbf{u})\,\mathbf{I}_3. = (\mathbf{n} \cdot \mathbf{u})\,O, \tag{118}$$

where $O = \mathbf{I}_3$ is the normalized point at the intersection point, in this case the origin. We thus find that there is a 'meeting strength' equal to the cosine between line direction $\mathbf{u}$ and the plane normal $\mathbf{n}$: the `meet` is strongest when the line is perpendicular to the plane, and zero when the line lies within the plane (which is a degenerate situation for this linearized form of intersection). We also see that the intersection point is positively proportional to $O = \mathbf{I}_3$ when the line direction and plane normal are in similar directions (positive cosine), and negative when opposite. The distinction is required for applications like ray tracing, where the plane would denote a locally flat piece of boundary and one would want to know whether an incoming ray could validly hit it from the outside, or invalidly from the inside.

So this is how negative points get into our algebra: the signs indicate how the points were made. Be aware of such possible signs in subsequent computations with those points; you may want to normalize them to positive before you use them, so that you can keep the orientations of composite objects clear in their meaning.

The sign is unavoidable, since changing the order of the arguments has no effect. So if we would prefer to have a convention in which the intersection point of ray coming in from the outside of a planar boundary is positive (i.e., to be a translated positive multiple of $O = \mathbf{I}_3$), then we would have to enforce an encoding convention in which $\mathbf{n}$ is the 'inside pointing normal'.

8.3 The Join of Two Points

As we saw above, the `join` of two points is a line whose orientation (the sign of its direction vector) depends on their order, in any dimension, and this is precisely as we would like it. The line determined by P and Q, which is $P \vee Q$, should have the opposite sign of the line $Q \vee P$ determined by Q and P. In this case, the magnitude of the `join` of two normalized points is proportional to their distance (as we compute below); including the orientation, we could say it is their signed distance.

We repeat the explicit expression for the line $P \vee Q$ of eq.(37):

$$P \vee Q = (\mathbf{q} - \mathbf{p})\,\mathbf{I}_3 - e_0\left(\tfrac{1}{2}(\mathbf{p} + \mathbf{q}) \wedge (\mathbf{q} - \mathbf{p})\right)\mathbf{I}_3. \tag{119}$$

For simplicity (and without loss of generality due to the translation covariance of the `join` in PGA), let us take $\mathbf{p} = \mathbf{0}$ so that $P = O = \mathbf{I}_3$:

$$O \vee Q = \mathbf{q} \cdot \mathbf{I}_3 = \mathbf{q} \cdot O. \tag{120}$$

This is a line through the origin with direction vector $\mathbf{q}$. To be precise, the resulting line has the same direction as $\mathbf{q}$, with a magnitude equal to the distance of O to Q. So the 2-blade $O \vee Q$ represents the line from O to Q, with a 'strength' (or 'velocity') $\|\mathbf{q}\|$.

8.4 The Meet of Two Planes

The `meet` of two planes also results in a line, this time oriented in the way we might expect in a space with right-handed pseudoscalar. Again the order matters: since both planes are odd, there is a swap sign. Let us check the result at the origin, with two planes $p_1 = \mathbf{n}_1$ and $p_2 = \mathbf{n}_2$, both normalized. Then

$$p_1 \wedge p_2 = \mathbf{n}_1 \wedge \mathbf{n}_2 = \big((\mathbf{n}_1 \wedge \mathbf{n}_2)\,\mathbf{I}_3^{-1}\big)\,\mathbf{I}_3 = (\mathbf{n}_1 \times \mathbf{n}_2)\,\mathbf{I}_3, \tag{121}$$

with $\times$ the 3D cross product. The final rewriting was our way of describing the direction of a line by means of a Euclidean vector, as explained in eq.(9).

8.5 The Meet of Plane and Point: Inside/Outside

When a plane and a point `meet`, the result is a PGA 4-vector, as we saw in Section 2.7. Yes, even a point *not* on the plane has a `meet` with it; for the `meet` is not a geometric intersection of point sets.

For simplicity, let us first do the case where $P = O$, the point at the origin. There is a swapping sign (both are odd) so the order matters.

$$p \wedge O = (\mathbf{n} - \delta e_0) \wedge \mathbf{I}_3 = -\delta\, e_0\, \mathbf{I}_3 = -\delta\, \mathcal{I}. \tag{122}$$

The plane representation $\mathbf{n} - \delta e$ is of a plane with normal vector $\mathbf{n}$ and a support vector of $\delta\mathbf{n}$, in the positive $\mathbf{n}$-direction; thus the point at location $\delta\mathbf{n}$ is on the plane.

We now find that we can retrieve δ as the strength (or rather, weakness) of the `meet` of plane with point O at the origin: it is the proportionality factor with the pseudoscalar $\mathcal{I}$. We can interpret this factor therefore as the distance we have to travel from the plane to encounter the point O; so O lies at distance $-\delta$ along $\mathbf{n}$.

For a plane p and general point X, we get $p \wedge X = (\mathbf{n} \cdot \mathbf{x} - \delta)\,\mathcal{I}$, and the interpretation is the same: the factor of the pseudoscalar denotes how much we should move p to get onto X. If it is positive, X lies on the side of the plane that $\mathbf{n}$ points to. If we wish to call that positive side the outside of the planar border, then $\mathbf{n}$ should be the 'outward pointing normal'.

8.6 The Meet of Two Lines: Chirality

The `meet` of two lines has no swapping sign, and is therefore a geometric property independent of any convention on ordering (though it will depend on the choice of pseudoscalar orientation). It is related to chirality: the 'sense of turning' of one line about the other.

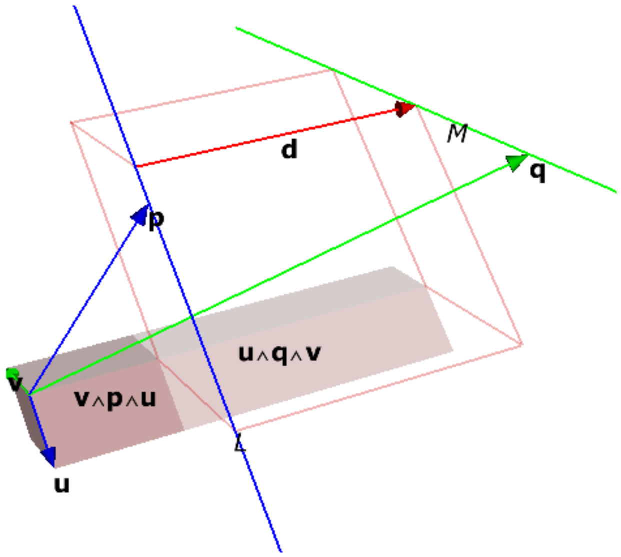


Figure 11: *The* `meet` *of two lines* $L = (\mathbf{u}\mathbf{I}_3) - e_0(\mathbf{p} \cdot (\mathbf{u}\mathbf{I}_3))$ *and* $M = \mathbf{v}\mathbf{I}_3{}^{-\star} - e_0(\mathbf{q} \cdot (\mathbf{v}\mathbf{I}_3))$ *equals* $-e_0 \wedge (\mathbf{q} - \mathbf{p}) \wedge \mathbf{u} \wedge \mathbf{v} = -e_0\, \mathbf{d}\, (\mathbf{u} \wedge \mathbf{v})$. *The proportionality factor is interpreted as the oriented volume spanned by their direction vectors and their orthogonal separation vector* **d**. *It is independent of the order of the arguments of the* `meet`. *As the lines cross over, or change direction, the sign of this volume changes; hence it can be used as the relative orientation of the lines. For the lines in this figure, this chirality (the factor of* $-e_0$*) is negative (assuming the usual right-handed space orientation). (This is Figure 11.6 from [2].)*

Let us parametrize the lines with direction and location vectors, to correspond more directly to the classical results.

$$\begin{aligned} L \wedge M &= \big(\mathbf{u}\mathbf{I}_3 - e_0(\mathbf{p} \cdot \mathbf{u}\mathbf{I}_3)\big) \wedge \big(\mathbf{v}\mathbf{I}_3 - e_0(\mathbf{q} \cdot \mathbf{v}\mathbf{I}_3)\big) \\ &= -e_0 \big((\mathbf{p} \cdot \mathbf{u}\mathbf{I}_3) \cdot \mathbf{v} - \mathbf{u}\mathbf{I}_3 \cdot (\mathbf{q} \wedge \mathbf{v})\big)^{-\star} \\ &= -e_0 \,(\mathbf{v} \wedge \mathbf{p} \wedge \mathbf{u} + \mathbf{q} \wedge \mathbf{v} \wedge \mathbf{u})\mathbf{I}_3^2 \\ &= -e_0 \wedge (\mathbf{q} - \mathbf{p}) \wedge \mathbf{u} \wedge \mathbf{v} \\ &= -e_0\, \mathbf{d}\, (\mathbf{u} \wedge \mathbf{v}) \\ &= -\delta\, \sin(\phi)\, \mathcal{I}, \end{aligned} \tag{123}$$

where we factorized orthogonally by the orthogonal separation vector $\mathbf{d} \equiv$

$\big((\mathbf{q}-\mathbf{p})\wedge\mathbf{u}\wedge\mathbf{v}\big)/(\mathbf{u}\wedge\mathbf{v})$ (from L to M), see Figure 11.

The result of $L \wedge M$ is proportional to the pseudoscalar $\mathcal{I}$, by a signed quantity of which the magnitude is the product of the line separation δ (from L to M) and the sine of the angle between $\mathbf{u}$ and $\mathbf{v}$ (from the L-direction vector $\mathbf{u}$ to the M-direction vector $\mathbf{v}$).

That factor (and with it the `meet` result itself) is therefore an interesting measure that combines spatial and angular distances: it is zero when the lines intersect, but also when the lines are parallel (which is of course also an intersection, in a common vanishing point).

The *chirality* is the sign of the proportionality factor of $-e_0$ (the oriented plane at infinity), which is also the sign of the scalar $(\mathbf{d}\wedge\mathbf{u}\wedge\mathbf{v})/\mathbf{I}_3$. Thus if we follow line L in its direction $\mathbf{u}$ and then find ourselves moving around line M such that $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{d}$ form a frame of the same orientation as $\mathbf{I}_3$, we call the chirality positive; and if not, negative (or zero). Figure 11 shows an example of negative chirality in a space with right-handed pseudoscalar.

8.7 Summary

In this text we really tried to use the capability of PGA to propagate orientation signs carefully, since they are so useful in practice. We showed that this is possible, but it takes some getting used to. To interpret them properly, we need to distinguish between 'geometrical signs' and incidental signs from construction order. But both are useful tools.

The emphasis on these aspects is rather new. We believe PGA is a good algebraic framework to encode the oriented projective geometry of Stolfi [18], and we would xpect this field to develop more than we could indicate in this introductory text.

9 Projective Duality in PGA

The original meaning of the P in PGA is 'projective' [13], since it was derived in the context of a projectivization of Euclidean space (i.e., adding points at infinity as regular elements). As usual in a projective approach, there is a natural duality between points and hyperplanes, between k-blades and $(n-k)$-blades. To fully represent this duality relationship, one needs to maintain both PGA and its dual space (or combine both in one), with both e_0 and e_0^r in their bases – as we temporarily did when motivating the `join` in Section 3. In this full space the duality between points and hyperplanes can be represented naturally; both types of elements are represented distinctly and explicitly.

When limiting ourselves to only the PGA space spanned by $\{e_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ (as we do in this text to show that a 4D space suffices for Euclidean geometry), a somewhat awkward 'shadow' of this full duality remains. We have avoided its use in this text, since we were able to work well enough without it. Full duality is much nicer, and if you need to maintain both spaces anyway, you had better use the full 5-dimensional CGA $\mathbb{R}_{4,1}$ and get spheres and circles and conformal transformations for free (see Chapters 14-16 of [2]). The 4-dimensional PGA $\mathbb{R}_{3,0,1}$ is a subalgebra of this full CGA $\mathbb{R}_{4,1}$.

But you may encounter the partial duality of PGA in other treatments and as programming shortcut, so we give a brief introduction to its definition and usage.

9.1 Complement Duality and Hodge $\star$

Some authors use a form of duality that is related to the duality relationship between the PGA of planes and its dual space in which vectors represent points. A proper way of defining this duality would be projectively [9]. We could determine, for a blade X, what more we need to span the pseudoscalar $\mathcal{I} = e_0\,\mathbf{I}_3$ of PGA:

$$\text{\textit{'complement-dual'} } \overline{X} \text{ of } X \text{ defined through:} \quad X \wedge \overline{X} = \mathcal{I}. \tag{124}$$

(This 'complement-dual' is our term, to distinguish it from other forms.) Although mathematically proper, there are two practical problems with this definition:

- The complement dual is not unique. One can always add an element X' to $\overline{X}$ for which $X \wedge X' = 0$. This reflects the fact that one can set up many dual relationships between projective planes and points, all of which make the incidence relationships of `meet` and `join` work. But we know that we will want to move to Euclidean geometry, so we would like to select a unique, *metric* correspondence. Normally one would do this by additionally demanding $X \cdot \overline{X} = 0$. Unfortunately the null elements of PGA would still not make that unique, either.

- The complement dual is not linear. When we multiply X by α, we see that its complement dual $\overline{X}$ is divided by α. Therefore the dual of a sum is not the sum of the duals. This is very annoying in metric applications; we prefer to keep our framework as linear as possible.

So this 'complement duality' is not convenient. But we can define a related form of duality by applying complement-duality to the basis blades, and then extending that to be linear. This is done by just copying the corresponding coefficients, using the coefficient of E_i then for $\overline{E_i}$. This is the principle behind the *Hodge star operator* $\star$. This construction will be dependent on the choice of basis, which is unfortunate (and much less satisfying than duality in CGA!). In a homogeneous model, that basis includes a choice of where the origin is. So a plane will indeed be dual to a certain point, but always relative to some chosen origin, and some choice of basis directions. At least we should choose to make the directional aspect of the duality related to the usual orthogonality of a Euclidean dual.

In geometric algebra, we can introduce to the Hodge dual more tidily than via such a kludgy definition on basis elements. We view the Hodge dual as a refinement of the complement dual eq.(124) to enforce uniqueness, linearity and Euclidean orthogonality. We do so by replacing the outer product by a geometric product, and guaranteeing linearity by an appropriate unsigned scalar factor. Thus our Hodge dual $\star$ is defined for a PGA blade S through

$$S\,(\star S) = (\mathbf{S}\,\widetilde{\mathbf{S}})\,\mathcal{I} \quad \text{for a blade } S, \tag{125}$$

and extended to multivectors by viewing them as sums of blades. Here $\mathbf{S}$ is the Euclidean factor (i.e., non-e factor) of the blade S. In PGA $\mathbb{R}_{d,0,1}$, we will use as our pseudoscalar $\mathcal{I} = e_0\,\mathbf{I}_d$.

We purposely denoted eq.(125) in terms of a variable named S (for 'simple'), to help remind you that S is a blade. Do *not* use this formula for general elements X as the Hodge's 'extension by linearity'. For instance, though the Hodge dual is linear and thus satisfies $\star(X+Y) = \star X + \star Y$, that property does not hold for the specific blade-based definition eq.(125) if you naively set $S = X + Y$! We ourselves have fallen into this trap, repeatedly...[20]

For a s-blade element S that is a Euclidean blade $\mathbf{S}$ or of the form $e_0\mathbf{S}$ we thus find

$$\star\mathbf{S} = \widetilde{\mathbf{S}}\, e_0\, \mathbf{I}_d = e_0\, \mathbf{S}\, \mathbf{I}_d\, (-1)^{s(s+1)/2} \tag{126}$$

$$\star(e_0\mathbf{S}) = \widetilde{\mathbf{S}}\, \mathbf{I}_d = \mathbf{S}\, \mathbf{I}_d\, (-1)^{s(s-1)/2} \tag{127}$$

It is perhaps more consistent to rewrite these duals in terms of the Euclidean dual $\mathbf{S}^\star = \mathbf{S}\widetilde{\mathbf{I}_d}$ rather than the 'undual' $\mathbf{S}\mathbf{I}_d$. In 2D and 3D such a rewrite produces an extra minus sign, and the result in those dimensions is

$$\text{2D, 3D PGA:} \quad \star\,\mathbf{S} = e_0\, \mathbf{S}^\star\, (-1)^{(s-1)(s-2)/2} \tag{128}$$

$$\text{2D, 3D PGA:} \quad \star\,(e_0\mathbf{S}) = \mathbf{S}^\star\, (-1)^{(s+1)(s+2)/2}. \tag{129}$$

For quick reference, in 3D PGA (and 2D PGA) we then have the following formulas for the blades of various grades (with α a scalar, $\mathbf{v}$ a Euclidean vector, $\mathbf{B}$ a Euclidean bivector, $\mathbf{T}$ a Euclidean trivector):

$$\begin{array}{rclcrcl} \star\alpha & = & -e_0\alpha^\star & \quad & \star e_0\alpha & = & -\alpha^\star \\ \star\mathbf{v} & = & e_0\mathbf{v}^\star & & \star e_0\mathbf{v} & = & -\mathbf{v}^\star \\ \star\mathbf{B} & = & e_0\mathbf{B}^\star & & \star e_0\mathbf{B} & = & \mathbf{B}^\star \\ \star\mathbf{T} & = & -e_0\mathbf{T}^\star & & \star e_0\mathbf{T} & = & \mathbf{T}^\star \end{array} \tag{130}$$

Since dualization is effectively accessing coordinates from a different basis element, with possibly a sign, it is computationally a trivial operation, and very fast. When you are implementing this Hodge dual using a basis, you can even cleverly choose your basis elements to group the signs in blocks of four, see Table 4. This leads to a fast SIMD/GPU implementation of duality.

Undualization is a bit subtle, for when we apply the Hodge dual twice, a sign may appear. From the above (and some sign-chasing), the double Hodge dual of any blade S in $\mathbb{R}_{d,0,1}$ obtains the sign of

$$\star(\star S) = (-1)^{\text{grade}(S)\, d}\, S, \quad \text{for a blade } S \tag{131}$$

so that we may define a *Hodge undualization* $\star^{-1}$ by

$$\star^{-1}(S) = (-1)^{\text{grade}(S)\, d} \star S, \quad \text{for a blade } S \tag{132}$$

[20]However, since in 3D Euclidean geometry all k-vectors are blades, you may use the formula for any element of uniform grade in 3D GPA.

$\star 1$	$=$	$\mathbf{e}_{0123}$	$\star \mathbf{e}_{0123}$	$=$	1
$\star \mathbf{e}_0$	$=$	$\mathbf{e}_{123}$	$\star \mathbf{e}_{123}$	$=$	$-\mathbf{e}_0$
$\star \mathbf{e}_1$	$=$	$\mathbf{e}_{032}$	$\star \mathbf{e}_{032}$	$=$	$-\mathbf{e}_1$
$\star \mathbf{e}_2$	$=$	$\mathbf{e}_{013}$	$\star \mathbf{e}_{013}$	$=$	$-\mathbf{e}_2$
$\star \mathbf{e}_3$	$=$	$\mathbf{e}_{021}$	$\star \mathbf{e}_{021}$	$=$	$-\mathbf{e}_3$
$\star \mathbf{e}_{01}$	$=$	$\mathbf{e}_{23}$	$\star \mathbf{e}_{23}$	$=$	$\mathbf{e}_{01}$
$\star \mathbf{e}_{02}$	$=$	$\mathbf{e}_{31}$	$\star \mathbf{e}_{31}$	$=$	$\mathbf{e}_{02}$
$\star \mathbf{e}_{03}$	$=$	$\mathbf{e}_{12}$	$\star \mathbf{e}_{12}$	$=$	$\mathbf{e}_{03}$

Table 4: *Complement-duality (the Hodge star) on the basis blades of 3D PGA relative to pseudoscalar $\mathcal{I} = e_0 \wedge \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \equiv \mathbf{e}_{0123}$. The particular basis elements are chosen to make sign patterns as symmetric as possible in blocks of 4, to enable fast implementation using standard 4D computer graphics SIMD/GPU hardware. Dualization simply becomes transfer of coordinates to a dual basis.*

No sign in even-D, but in odd-D (including 3D!) we have $\star^{-1}(X) = \star\widehat{X}$.

The resulting Hodge dual is an orthogonally selected form of complement-duality. It implicitly introduces a Euclidean duality in which dual elements are indeed orthogonal in the usual sense. Defining a Hodge dual is just a different (and rather implicit) way of defining a metric. In fact, Browne [9] introduces the basic operations in the order: `meet` then `join` then complement-duality then interior product based on the complement. In this way, he ends up with a Euclidean dot product in what started as a metric-free Grassmann algebra, getting rather close to geometric algebra at the end of his Volume 1.[21]

We have chosen here for the order: `meet`, metric complement, then `join` in terms of Euclidean complement. This approach no longer appears to need a non-Euclidean form of duality explicitly - though in an essentially projective approach that motivates PGA (partly), projective duality between points and (hyper-)planes always has a barely hidden presence, and can be a source of inspiration.

[21]For those interested, an external appendix [10] connects our text to that of Browne's Grassmann Algebra approach.

9.2 The Join by Hodge Duals

We derived the `join` and its signs from a careful consideration of the external dual space of PGA in Section 3.2. Having an 'internal' form of duality (the Hodge dual) suggests that we might be able to write the `join` in PGA rather more simply. And indeed, by carefully chasing the signs of the Hodge duals, we can show that

$$A \vee B = \star^{-1}(\star A \wedge \star B). \tag{133}$$

The proof may be found in exercise 12.2:12.

When you implement the Hodge dual as the coordinate manipulation indicated above, this dual outer product formula for the `join` leads to the fastest code. In `ganja`, the `join` is implemented in this manner at compile time, as one of the basic operations of PGA.

> **Example:** Hodge-based `join` of two points.
> With $P = \mathbf{I}_3 - e_0\,\mathbf{p}\,\mathbf{I}_3$ and $Q = \mathbf{I}_3 - e_0\,\mathbf{q}2\,\mathbf{I}_3$, we find $\star P = -e_0 - \mathbf{p}$ and $\star Q = -e_0 - \mathbf{q}$. Then
>
> $$\begin{aligned} P \vee Q &= \star^{-1}(\star P \wedge \star Q) \\ &= \star^{-1}\big(e_0 \wedge (\mathbf{q} - \mathbf{p}) + \mathbf{p} \wedge \mathbf{q}\big) \\ &= \star\big(e_0(\mathbf{q} - \mathbf{p}) + \mathbf{q} \wedge \mathbf{p}\big) \\ &= -(\mathbf{q} - \mathbf{p})^\star + e_0(\mathbf{q} \wedge \mathbf{p})^\star \\ &= (\mathbf{q} - \mathbf{p})\,\mathbf{I}_3 - e_0\,(\mathbf{p} \wedge \mathbf{q})\,\mathbf{I}_3. \end{aligned} \tag{134}$$
>
> This indeed agrees with this `join` $P \vee Q$ as in eq.(37).

9.3 Summary

Duality is a fundamental aspect of the geometry of flats (planes, lines, points). In PGA it is more awkward than we are used to in more balanced geometric algebras.

We tried to navigate this lightly. Our main goal was to stay within the basic algebra $\mathbb{R}_{d,0,1}$, since for 3D Euclidean geometry that is 4D and matches current SIMD/GPUs well. We showed how the Hodge dual is a workable form of duality under these constraints, even leading to a compact and cheap `join` formula eq.(133). A clever choice of basis elements for PGA, in which all signs change in blocks of four, enables fast computation with duals (see Table 4).

10 Performance Considerations

10.1 PGA versus CGA

Having to take the dual view (with the vectors representing planes, not points) and possessing a non-invertible pseudoscalar: these were both hurdles that kept the GA community from accepting PGA for about 10 years. As we have tried to show in this tutorial, the former is only a conceptual issue, and the latter hardly surfaces (and where it does, it is not a programming issue, just one of algebraic purity). CGA still has the advantage over PGA of representing *round* elements like spheres, circles and tangents as blades; but traditional graphics algorithms use the *flats* (planes, lines, directions), and for them PGA is a sufficiently natural environment. We have tried to convince the practitioners to use CGA for years; only in robotics, where the sphere computations really simplify the inverse kinematics, has CGA gained some ground.

If flats are indeed enough in your application, the '1-up' PGA is your algebra. And then there is a significant advantage in computational performance in *not* using the '2-up' CGA.

- For the geometric product of a full 3D PGA multivector (from $\mathbb{R}_{3,0,1}$), 192 multiplications and 176 additions are required.
- For the geometric product of a full 3D CGA multivector (from $\mathbb{R}_{4,1}$), 1024 multiplications and 992 additions are required.

This makes PGA more than 5 times faster compared to CGA, for a general geometric product. But we should say that one rarely needs to compute a full geometric product of general multivectors, since sensible geometric elements are blades or versors; still, it indicates that there is a significant difference.

10.2 PGA versus LA

However, for applications in computer graphics, the crucial comparison to make is between PGA and the classic matrix/vector techniques from linear algebra (LA). PGA builds on the representations (homogeneous points, Plücker coordinates, linear equations, (dual-) quaternions) already widely used in the computer graphics community. As a result, adopting existing frameworks to incorporate PGA is often a trivial task, and in many cases PGA can coexist with classical techniques.

The advantage of embedding in PGA, for even in the highly optimized graphics space, is that extra performance gains can be made.

- For the composition of Euclidean motions, the classic 4×4 matrix product (requiring 64 multiplications, 48 additions and 16 floats of storage) can be replaced by the geometric product between motors (requiring only 48 multiplications, 40 additions and 8 floats of storage).

- Additionally, PGA motors offer improved numerical stability, eliminating the need for re-orthogonalization or re-unitarization, and provide a covariant approach - thus removing costly calculations of matrix inverses or adjoints.

Other advantages of the general GA formalism apply equally to PGA, with the availability of exception-free incidence relations and well-defined duality of elements and their relationships further reducing code complexity.

10.3 Memory Layout

A carefully selected memory layout allows a perfect split in 4 blocks of 4 floats that matches today's practices (and SIMD/GPU hardware) exactly.

The trick is to organize the basis elements *not* simply by grade (from scalar, via vectors, bivectors and trivectors to the 4-dimensional pseudoscalar), but to interleave the scalar and pseudoscalar as in Table 5, arranging the basis of the even subalgebra into a contiguous group. When we do so, the coordinate aspects neatly coincide with geometrical concepts. This layout not only arranges the basic objects (plane, line, point) in blocks of consecutive coordinates, but also does so for the versors of rotation and rigid body motion. Translation, which officially requires a basis $\{1, \mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03}\}$ does not quite follow this pattern. But since the scalar part of a translation versor is always equal to 1 anyway, we only need to pick up the other three elements to construct that versor – and those are nicely consecutive in the layout.

We easily recognize the more classical representations in the resulting scheme. Points and planes are represented homogeneously with four floats, while lines are stored using what are effectively Plücker coordinates. Both quaternions and dual quaternions are using 'classic' memory layouts, and hence are available from PGA elements without any conversion - simply point to the correct starting floats.

If additionally the actual basis elements are chosen smartly, so that complement duality sign changes are 'per grade' as in Table 4 (which has the same signs per block), this provides a perfect match to current practice in computer graphics. (If by contrast we had chosen for the trivector basis the seemingly natural $\{\mathbf{e}_{123}, \mathbf{e}_{023}, \mathbf{e}_{031}, \mathbf{e}_{012}\}$, for instance, not all would have the same sign under the complement duality of Table 4, and this would take extra operations to fix.)

10.4 Speed-up of Sandwich-like Products

Many of the formulas presented in this text are sandwiching-like products with a repeating factor: geometric transformations are $X \mapsto (-1)^{vx} V\, X\, V^{-1}$, the projection is $X \mapsto (X \cdot A)/A$ and the rejection $x \mapsto (x \wedge A)/A$ (which can be written in sandwiching form modulo some signs, see Exercise 12.2:9). These products are all structure-preserving (transforming points to points, lines to lines, etc.), and the repetition of a factor means that they are perfect candidates for symbolic optimization.

Optimizing these sandwiches can often provide substantial gains (for example, the projection of a line on a point $(L \cdot P)/P$ can be performed using just 29 multiplications compared to 42 if done fully). High performance implementations [19] of PGA typically include shortcuts for these common sandwiching constructions.

10.5 Inverses

Graphics engineers to whom PGA (and GA in general) is new may be taken aback by the large number of divisions typically used in these algebras, notably in the sandwiching products above. However, in many cases multivector inverses are not explicitly needed.

Even versors constructed directly as the exponential of their invariant bivectors are of the form $\exp(-B/2)$. For this kind of versor (aka a rotor), $V\,\widetilde{V} = 1$. Therefore their inverse is exactly equal to their reverse, which is trivial to compute: just a minus sign for the parts of grade 2 and 3.

As we already indicated in Section 6.1, for the blades and general versors, the inverse is defined as $A^{-1} = \widetilde{A}/(A\widetilde{A})$. In PGA, the denominator is always positive (for elements for which the inverse exists at all). So if you are interested in geometry modulo a positive scaling factor, you are allowed to replace

grade 1	**grade 0**	**grade 2**		**grade 4**	**grade 3**
vector $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$	scalar 1	E-bivector $\mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}$	e_0-bivector $\mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03}$	pseudo $\mathbf{e}_{0123}$	trivector $\mathbf{e}_{123}, \mathbf{e}_{032}, \mathbf{e}_{013}, \mathbf{e}_{021}$
plane (normal equation)		**line** (Plücker coordinates)			**point** (homogeneous coordinates)
		screw (Lie algebra)			
plane reflector	**rotator** (unit quaternion)		**translator**		**point reflector**
	motor (unit dual quaternion, Lie group)				
1 reflection	**0/2/4 reflections**				**3 reflections**

Table 5: *The components of the elements of 3D PGA can be stored as different kinds of blades, often in quadruples. If done in a smart arrangement, this can use SIMD/GPU hardware for graphics efficiently. E denotes Euclidean elements; V denotes elements on the vanishing plane. (Actually, the translator is placed here for symmetry, its scalar is always 1 so does not need to be stored, and we set its pseudoscalar to zero. With this artificial symmetry comes efficiency.) In gray, we have indicated classical concepts now structurally embedded in PGA. Minor caveats: not all linear combinations of the bivectors are lines, only those that are factorizable by the outer product. Also, the point reflector is indeed 3 planar reflections; but in general, 3 planar reflections generate a more general transformation. You can order this table on a mug at* `https://bivector.net/merch.html`.

the inverse by the reverse. You then still retain the benefit of consistently oriented computations.

For example, the projection formula $(X \cdot A)\, A^{-1}$. For a normalized blade, this is equivalent to $(X \cdot A)\, \widetilde{A}$ which only requires the reversion operator (swapping signs of grades 2 and 3). For non-normalized blades, we can do the same: replace the inversion by the reversion, and you may draw the result, or compute on with it, or decide to normalize it before further computations (some interpretations depend on normalization).

Normalized points, (proper) lines and planes in PGA are their own inverses (up to a grade-dependent sign: $-$ for point and line, $+$ for plane). As such, these inversions or divisions in our cheat sheet of Table 2 do not imply the performance penalty they would in the matrix/vector approach.

11 To PGA! (and Beyond...)

This text intends to convey the basic structure of PGA, to help you with accepting its rather unusual nature (yes, 3D points are trivectors!), and to show how things you already use (but that are somewhat extraneous to the basic LA approach, like dual quaternions) are actually fully integrated into PGA.

We hope that this write-up encourages you to try this out for yourself. It does work; now make it work for you. Once you are hooked, there is no going back; rather, you may decide to even go beyond the algebra of planes PGA to the algebra of spheres CGA, which does for conformal transformations what PGA did for Euclidean motions. The modeling principles are similar, and described in Chapters 14-16 of [2]; it is just that in CGA, spheres are the vectors, and then the rest follows. In this context, planes are special spheres (with infinite radius), so in Computer Science terms CGA is a fully backwards compatible extension of PGA (see also [6]). Mathematically, PGA is a subalgebra of CGA.

But for now, start with PGA, and enjoy!

Acknowledgement

Our gratitude to Chris Doran, Tomas Bosschieter and Stephen Mann for thorough proofreading of earlier versions of this text. Any errors remaining have been introduced by us afterwards. Please tell Leo about them at `l.dorst@uva.nl`.

12 Exercises

12.1 Drills

1. Show that $e_0 \wedge \mathbf{e}_1 + \mathbf{e}_2 \wedge \mathbf{e}_3$ is a 2-vector, but not a 2-blade. Try to factorize it by the outer product!

2. Hail Caesar! Compute $A \vee e_0$.

3. Show that meeting the celestial sphere (going to heaven?) reduces you to a null object: $(A \wedge e_0)^2 = 0$ for all A.

4. Joining the universe will not affect you: show that $\mathcal{I} \vee A = A = A \vee \mathcal{I}$. Apparently, you were already part of it...

12.2 Structural Exercises

1. **Reciprocal Frames (with Section 2.2)**
   In some of our equations, combinations of vectors occur that are most easily interpreted if we are aware of reciprocal frames. Let us take a side view of the two intersecting planes, used in Section 2.2 to generate the line representation, see Figure 12.

   The location of the intersection of the two lines can be found by the vertex of a parallelogram with sides parallel to the lines, and hence in directions perpendicular to $\mathbf{n}_1$ and $\mathbf{n}_2$. If we construct two vectors $\mathbf{n}_1^r$ and $\mathbf{n}_2^r$ which satisfy $\mathbf{n}_1 \cdot \mathbf{n}_1^r = 1$ and $\mathbf{n}_2 \cdot \mathbf{n}_2^r = 1$ and $\mathbf{n}_1 \cdot \mathbf{n}_2^r = 0$ and $\mathbf{n}_2 \cdot \mathbf{n}_1^r = 0$, then we can use these as a new basis for those directions in the plane spanned by $\mathbf{n}_1$ and $\mathbf{n}_2$. These equations show that $\mathbf{n}_1^r$ is perpendicular to $\mathbf{n}_2$, with a sensibly chosen direction and length by its relationship to $\mathbf{n}_1$, and conversely for $\mathbf{n}_2^r$, see Figure 12.

   (a) Solve these equations, i.e., give expressions for $\mathbf{n}_1^r$ and $\mathbf{n}_2^r$.

   (b) Establish the *reciprocal frame relationships*: the $\mathbf{n}_i^r$-coordinate of a point $\mathbf{x}$ is $\mathbf{x} \cdot \mathbf{n}_i$ (and vice versa, the $\mathbf{n}_i$-coordinate of $\mathbf{x}$ is $\mathbf{x} \cdot \mathbf{n}_i^r$).

   (c) Show how the reciprocal frame vectors indeed explain the positional term of the line representation eq.(8). Note that this term arose automatically, so you do not actually need to know the reciprocal frame; it happens anyway.

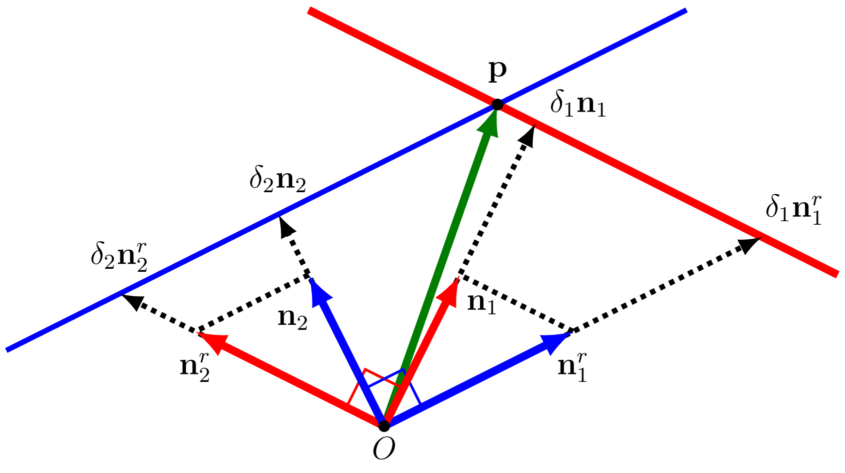


Figure 12: *The intersection line of two planes encoded by the reciprocal frame of their normal vectors, as seen head on in the plane spanned by their normal vectors* $\mathbf{n}_1$ *and* $\mathbf{n}_2$*. The intersection point* $\mathbf{p}$ *is most easily seen as the sum of weighted reciprocal vectors* $\mathbf{p} = \delta_1 \mathbf{n}_1^r + \delta_2 \mathbf{n}_2^r$*.*

Solution: Using geometric algebra, we can easily find expressions for the vectors $\mathbf{n}_i^r$. We compute $\mathbf{n}_1^r (\mathbf{n}_1 \wedge \mathbf{n}_2) = \mathbf{n}_1^r \cdot (\mathbf{n}_1 \wedge \mathbf{n}_2) = (\mathbf{n}_1^r \cdot \mathbf{n}_1)\, \mathbf{n}_2 - (\mathbf{n}_1^r \cdot \mathbf{n}_2)\, \mathbf{n}_1 = \mathbf{n}_2$, so that $\mathbf{n}_1^r = \mathbf{n}_2 / (\mathbf{n}_1 \wedge \mathbf{n}_2)$. Similarly $\mathbf{n}_2^r = \mathbf{n}_1/(\mathbf{n}_2 \wedge \mathbf{n}_1) = -\mathbf{n}_1/(\mathbf{n}_1 \wedge \mathbf{n}_2)$. You can find more about these reciprocal frames in Section 3.8 of [2], such as the general form in d dimensions. For the intersection point we obtain

$$\mathbf{p} = (\mathbf{p} \cdot \mathbf{n}_1)\, \mathbf{n}_1^r + (\mathbf{p} \cdot \mathbf{n}_2) \mathbf{n}_2^r = \delta_1 \mathbf{n}_1^r + \delta_2 \mathbf{n}_2^r. \tag{135}$$

We can also see this from similar triangles indicated in the figure, and recognize these terms in eq.(8). Since Figure 12 is a perpendicular cross section of the 3D situation, $\mathbf{p}$ is indeed the orthogonal support vector of the resulting line (which sticks orthogonally out of the page), as we mentioned in the main text.

2. **Simplification of Expressions (with Section 2.5)**
Check why the following is allowed in rewriting the point representation

of eq.(13):

$$\begin{aligned}\mathbf{p}\cdot\mathbf{I}_3 &= (p_1\mathbf{e}_1+p_2\mathbf{e}_2+p_3\mathbf{e}_3)\cdot(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3)\\ &= p_1\mathbf{e}_2\mathbf{e}_3 - p_2\mathbf{e}_1\mathbf{e}_3 + p_3\mathbf{e}_1\mathbf{e}_2\\ &= p_1\mathbf{e}_{23}+p_2\mathbf{e}_{31}+p_3\mathbf{e}_{12}. \qquad (136)\end{aligned}$$

And for $\mathbf{A}$ purely Euclidean: $e_0\,\mathbf{A} = e_0\cdot\mathbf{A} + e_0\wedge\mathbf{A} = 0 + e_0\wedge\mathbf{A}$.

3. **Point Representation in d-dimensional Space (with Section 2.5)**

Show that in d dimensions, the point representation is $P = (1-e_0\mathbf{p})\,\mathbf{I}_d$, by performing the `meet` of d well-chosen planes. Mind the signs!

Solution:

$$\begin{aligned}P &= \bigwedge_{i=1}^{d}(\mathbf{e}_i - (\mathbf{p}\cdot\mathbf{e}_i)e_0)\\ &= \mathbf{I}_d - \sum_{i=1}^{d}\mathbf{e}_1\wedge\mathbf{e}_2\wedge\cdots(\mathbf{p}\cdot\mathbf{e}_i)e_0\wedge\cdots\wedge\mathbf{e}_d\\ &= \mathbf{I}_d - e_0\wedge\sum_{i=1}^{d}(-1)^i\mathbf{e}_1\wedge\mathbf{e}_2\wedge\cdots(\mathbf{p}\cdot\mathbf{e}_i)\wedge\cdots\wedge\mathbf{e}_d\\ &= \mathbf{I}_d - e_0\wedge(\mathbf{p}\cdot\mathbf{I}_d)\\ &= (1-e_0\mathbf{p})\,\mathbf{I}_d. \qquad (137)\end{aligned}$$

This can be interpreted as a prefix translation operator applied to the point at the origin represented by the d-blade $\mathbf{I}_d$. It is in prefix form since $e_0\mathbf{p}$ anti-commutes with $\mathbf{I}_d$ in any dimension: $e_0\mathbf{I}_d = \widehat{\mathbf{I}}_d e_0$ and $\mathbf{p}\mathbf{I}_d = -\widehat{\mathbf{I}}_d\mathbf{p}$. Therefore the translation versor sandwiching product can be rewritten in a one-sided manner: $(1-e_0\mathbf{p}/2)\,\mathbf{I}_d(1+e_0\mathbf{p}/2) = (1-e_0\mathbf{p}/2)^2\,\mathbf{I}_d = (1-e_0\mathbf{p})\,\mathbf{I}_d$.

4. **Classical Homogeneous Point Representation (Section 2.5)**
Show that balancing the algebra with the extra vector e_0^r reciprocal to e_0 (so that $e_0\cdot e_0^r = 1$), and then doing the dualization as a dot product with the reciprocal pseudoscalar $\mathcal{I}^r \equiv \mathbf{e}_3\wedge\mathbf{e}_2\wedge\mathbf{e}_1\wedge e_0^r$, leads to a fairly recognizable form of the usual point representation p in homogeneous

coordinates, which would allow us to have $p \cdot n = 0$ as the homogeneous representation of a plane n.

Solution: $p \equiv P \rfloor \mathcal{I}^r = \mathbf{p} + e_0^r$ and $p \cdot n = \mathbf{p} \cdot \mathbf{n} - \delta$. So this is just what you get in homogeneous coordinates, if you consider e_0^r as the homogeneous coordinate bases vector.

5. **The Common Factor Axiom (with Section 3.1)**
Using the scalar product $*$, in Chapter 3 of [2] we define the left contraction $\rfloor$ from the outer product $\wedge$ by

$$(X \wedge A) * B \equiv X * (A \rfloor B), \quad \text{for all } X. \tag{138}$$

It is really important to use the contraction rather than a general inner product for this exercise, not least because of its properties on scalars: $1 \rfloor B = B$ (rather than 0, as in some other inner products, see [20]). We can use this relationship to derive the *Common Factor Axiom* (CFA) for blades:

$$(A \wedge B) \vee (B \wedge C) = (A \wedge B \wedge C) \vee B. \tag{139}$$

(It makes no difference whether we put B to the left or the right on the right hand side of this equation, see Exercise 12.1.4.)

Let $\text{grade}(A \wedge B \wedge C) = n$, and take duality $(\ )^*$ in that n-D space. Then we will find that the demand to have no extraneous argument-dependent signs in the CFA we should define it as:

$$(X \vee Y)^* \equiv Y^* \wedge X^*, \tag{140}$$

or equivalently (avoiding the undualization which is so awkward to define in degenerate spaces):

$$X \vee Y \equiv Y^* \rfloor X \tag{141}$$

(though this requires a reciprocal in the dualization). Verify the following derivation of the CFA! (Remember that the scalar product is only non-zero if its two arguments have the same grade, and the `join` we compute has the grade of B; therefore so does X.)

$$\begin{aligned}
&X * \big((A \wedge B) \vee (B \wedge C)\big) \\
&= \; X * \big((B \wedge C)^* \rfloor (A \wedge B)\big) \\
&= \; \big(X \wedge (B \wedge C)^*\big) * (A \wedge B) \\
&= \; (A \wedge B) * \big(X \wedge (B \wedge C)^*\big) \\
&= \; (B \wedge A) * \big(X \rfloor (B \wedge C)\big)^* (-1)^{ab} \\
&= \; B * \Big(A \rfloor \big(X \rfloor (B \wedge C)\big)^*\Big) \, (-1)^{ab} \\
&= \; B * \Big(A \wedge \big(X \rfloor (B \wedge C)\big)\Big)^* \, (-1)^{ab} \\
&= \; B * \Big(\big(X \rfloor (B \wedge C)\big) \wedge A\Big)^* \, (-1)^{a(b+c)} \\
&= \; B * \Big(\big(X \rfloor (B \wedge C)\big) \rfloor (A^*)\Big) \, (-1)^{a(b+c)} \\
&= \; \Big(B \wedge \big(X \rfloor (B \wedge C)\big)\Big) * (A^*) \, (-1)^{a(b+c)} \\
&\overset{!}{=} \; \big(B \wedge (X \rfloor B) \wedge C\big) * (A^*) \, (-1)^{a(b+c)} \\
&= \; (X \rfloor B) \, \Big((B \wedge C) * (A^*)\Big) \, (-1)^{a(b+c)} \\
&= \; (X * B) \, \Big((B \wedge C) \rfloor (A^*)\Big) \, (-1)^{a(b+c)} \\
&= \; (X * B) \, (B \wedge C \wedge A)^* \, (-1)^{a(b+c)} \\
&= \; X * \big((A \wedge B \wedge C)^* \rfloor B\big) \\
&= \; X * \big(B \vee (A \wedge B \wedge C)\big)
\end{aligned}$$

Hint: In the last few steps (after $\overset{!}{=}$), it is important to recognize which elements are scalars and pseudoscalars, and creatively replace products to rewrite equivalent expressions for those. Note that the inverse of the dual is not required, so we should be able to make this work in the degenerate PGA (with some carefully defined duality). We do so in Section 3 of the main text.

6. **Bladeness Verification (with Section 5.6)**
Prove that L and $L\mathcal{I}$ of eq.(69) indeed are blades, by explicitly computing $L \wedge L$ and $(L\mathcal{I}) \wedge (L\mathcal{I})$. Start with the latter, it is easier.

Solution: For the latter that is a straightforward use of the duality properties of inner and outer product :

$$\begin{aligned}(L\mathcal{I}) \wedge (L\mathcal{I}) &\propto (B\,\mathcal{I}) \wedge (B\,\mathcal{I}) \\ &= \big((B\,\mathcal{I}) \cdot B\big)\,\mathcal{I} \\ &= \big(B \cdot (B\,\mathcal{I})\big)\,\mathcal{I} \\ &= (B \wedge B)\,\mathcal{I}^2 = 0. \end{aligned} \tag{142}$$

The former is more involved. Using the symmetry of the outer product for blades, and the fact that $(B \wedge B)^2 = 0$:

$$\begin{aligned} L \wedge L &\propto \big(B\,(B\cdot B) - \tfrac{1}{2}B(B\wedge B)\big) \wedge \big(B\,(B\cdot B) - \tfrac{1}{2}B(B\wedge B)\big) \\ &= (B\wedge B)\,(B\cdot B)^2 + \tfrac{1}{2}\big(B\,(B\wedge B)\big) \wedge \big(B\,(B\wedge B)\big) \\ &\quad - B \wedge \big(B\,(B\wedge B)\big)\,(B\cdot B) \\ &= (B\wedge B)\,(B\cdot B)^2 + \tfrac{1}{2}(B\wedge B)^3 - (B\cdot B)^2\,(B\wedge B) \\ &= 0. \end{aligned} \tag{143}$$

7. **The Commutator of Lines (with Section 5.6)**
The commutator product of two 2-blades is in general a 2-vector. Therefore the commutator of two lines need not be a line. Compute it, and specify under what conditions it is a line.

Solution: (We use $\widetilde{M}$ to avoid some annoying signs, but this is not essential.)

$$\begin{aligned} L \times \widetilde{M} &= \big(\mathbf{u}\mathbf{I}_3 - e_0(\mathbf{p}\cdot\mathbf{u}\mathbf{I}_3)\big) \times \big(\mathbf{v}\mathbf{I}_3 - e_0(\mathbf{q}\cdot\mathbf{v}\mathbf{I}_3)\big)^{\sim} \\ &= \mathbf{u}\wedge\mathbf{v} - e_0\,\big(\mathbf{u}\cdot(\mathbf{q}\wedge\mathbf{v}) - \mathbf{v}\cdot(\mathbf{p}\wedge\mathbf{u})\big) \end{aligned} \tag{144}$$

Note that this is in general not a blade, for its outer square is not zero:

$$\begin{aligned} (L\times\widetilde{M}) \wedge (L\times\widetilde{M}) &= (\mathbf{u}\cdot\mathbf{v})\,e_0\wedge\mathbf{u}\wedge(\mathbf{p}-\mathbf{q})\wedge\mathbf{v} \\ &= (\mathbf{u}\cdot\mathbf{v})\,e_0\wedge\mathbf{u}\wedge\mathbf{d}\wedge\mathbf{v}. \end{aligned} \tag{145}$$

Only when the lines have a point in common (which may be a vanishing point), or when the lines have orthogonal directions, this does equal zero.

8. **Repositioning by Orthogonal Projection (with Section 5.3)**
Apply the projection eq.(59) on a plane $\mathbf{n}$, to show its correctness in

that case. Then reason why that implies that it will also be correct for all geometric primitives constructed from plane by means of `meet` operations.

Solution:

$$\begin{aligned}
(\mathbf{n}\cdot X)/X &= \big(\mathbf{n}\cdot((1-e_0\mathbf{x})\mathbf{I}_3)\big)/\big((1-e_0\mathbf{x})\mathbf{I}_3\big) \\
&= \big(\mathbf{n}\wedge(1-e_0\mathbf{x})\big)\,(1+e_0\mathbf{x}) \\
&= (\mathbf{n}+e_0\,\mathbf{n}\wedge\mathbf{x})\,(1+e_0\mathbf{x}) \\
&= \mathbf{n}+e_0\,\mathbf{n}\wedge\mathbf{x}-e_0\mathbf{n}\mathbf{x} \\
&= \mathbf{n}-e_0\,(\mathbf{n}\cdot\mathbf{x}). \qquad (146)
\end{aligned}$$

And now think ‘outermorphism’...

9. **Contraction Sandwiching (with Section 5.3)**
We saw in Section 5.3 how to project an element onto a point, to produce a positioning operation. You can rewrite this operation as a ‘contraction sandwich’, using the left and right contraction (to show the analogy with geometric product sandwiching when applying a versor), but this involves some extra signs. We have to use the contraction in this exercise to make the projection universally correct in its geometrical meaning, even for scalars and pseudoscalars. Show that:

$$T_{\mathbf{x}}[\mathbf{A}_k] = (\mathbf{A}_k \rfloor X)\rfloor X^{-1} = (-1)^{k(d+1)}\,X\lfloor \mathbf{A}_k \rfloor X^{-1} \qquad (147)$$

The two contractions in the dot sandwiching can be performed in either order.

Solution: the sign difference derives from (see Section 3.4 of [2]):

$$\begin{aligned}
X_m\lfloor Y_n &= (\widetilde{Y}_n\rfloor \widetilde{X}_m)^{\sim} \\
&= (-1)^{n(n-1)/2+m(m-1)/2+(m-n)(m-n-1)/2}\,X_m\rfloor Y_n \\
&= (-1)^{(m+1)n}\,X_m\rfloor Y_n. \qquad (148)
\end{aligned}$$

While the dot sandwich is an interesting structural rewriting, reminiscent of versor sandwiching, the projection formulation is more easily remembered (it avoids the awkward sign factor). Also, it is more obviously idempotent.

10. **Translator as a Ratio of Points (with Section 6.3)**
Show that the ratio of two similarly weighted points Q and P gives a translator over twice their distance.

Solution: $\big((1 - e_0\mathbf{q})\mathbf{I}_3\big)/\big((1 - e_0\mathbf{p})\mathbf{I}_3\big) = (1 - e_0\mathbf{q})(1 + e_0\mathbf{p}) = 1 - e_0(\mathbf{q} - \mathbf{p})$. The actual translator is then the square root of this, which is $1 - e_0(\mathbf{q} - \mathbf{p})/2$.

11. **Exact Line Correspondence Algorithm (with Section 6.8)**
Design an algorithm using the basic ideas of Section 6.8 to find the motor that aligns the minimum number of line correspondences in 3D. Your first step aligns the first set of lines; what are the remaining degrees of freedom? How would you construct the next ratio of primitives to disambiguate those while preserving the first alignment?

12. **Hodge Dual Join in d Dimensions (with Section 9.2)**
Compute the sign in a PGA for d-dimensional space (so $\mathbb{R}_{d,0,1}$) relating $A \vee B$ to $\star(\star A \wedge \star B)$, to generalize eq.(133).

Solution: Let the pseudoscalar be $e_0\,\mathbf{I}_d$. We employ the Hodge dualization equations eq.(126) and eq.(127). Then we compute the terms involving $\mathbf{P}_A$, of grade $(a-1)$, and $\mathbf{P}_B$, of grade $(b-1)$, in $\star A$ and $\star B$. They are

$$\star(e_0\mathbf{P}_A) = \mathbf{P}_A\,\mathbf{I}_d\,(-1)^{(a-1)(a-2)/2} \tag{149}$$

and similare for $\star^{-1}(e_0\mathbf{P}_B)$. When using the proposed $A \vee B = \star^{-1}(\star A \wedge \star B)$, these are wedged together to produce a Euclidean element of grade $(a + b - d - 2)$, and then the Hodge undual based on

$$\star^{-1}(\mathbf{S}) = e_0\mathbf{S}\widetilde{\mathbf{I}}_d\,(-1)^{(d-s)(d-s-1)/2}$$

produces a term of the form (modulo sign):

$$e_0\big((\mathbf{P}_A\mathbf{I}_d) \wedge (\mathbf{P}_B\mathbf{I}_d)\big)\mathbf{I}_d = e_0\left(\mathbf{P}_A{}^\star \wedge \mathbf{P}_B{}^\star\right)\mathbf{I}_d. \tag{150}$$

which according to eq.(34) should be fully correct apart from swapping A and B on the right. The overall signs are, modulo 2: $\frac{1}{2}(a-1)(a-2)$ $+\frac{1}{2}(b-1)(b-2)$ $+\frac{1}{2}(d-(a+b-d-2))(d-(a+b-d-2)-1)$

$+\frac{1}{2}d(d-1) = (d-(a-1))(d-(b-1)) \mod 2$. Thus

$$\begin{aligned}(e_0\mathbf{P}_A \vee e_0\mathbf{P}_B) &= e_0\left(\mathbf{P}_A{}^\star \wedge \mathbf{P}_B{}^\star\right)\mathbf{I}_d\,(-1)^{(d-(a-1))(d-(b-1))}\\ &= e_0\left(\mathbf{P}_B{}^\star \wedge \mathbf{P}_A{}^\star\right)\mathbf{I}_d\\ &= e_0\,(\mathbf{P}_A \vee \mathbf{P}_B),\end{aligned}$$

where the last $\vee$ is the Euclidean `join`. and this precisely corresponds to the expected term in the `join` of eq.(34). The other terms in the `join` (involving the tangent aspects $\mathbf{T}_A$ and $\mathbf{T}_B$) turn out to be similar in sign (you may not enjoy checking this), so we have found in $\mathbb{R}_{d,0,1}$:

$$A \vee B = (B^* \wedge A^*)^{-*} = \star^{-1}(\star A \wedge \star B) \tag{151}$$

for any A and B, in any dimension. Thus both the reciprocal duality from full PGA and the Hodge duality can each be used to compute the `join`, but you should beware of the order. In computational implementations, the Hodge is more frequently used, and it is pleasant that application order is reading order.

References

[1] L. Dorst, "May the forque be with you, dynamics in PGA," 2022, available at https://bivector.net/PGADYN.html.

[2] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra for Computer Science: An Object-oriented Approach to Geometry.* Morgan Kaufman, 2009. [Online]. Available: http://www.geometricalgebra.net

[3] J. M. Selig, "Clifford algebra of points, lines and planes," *Robotica*, vol. 18, pp. 545–556, 2000. [Online]. Available: https://openresearch.lsbu.ac.uk/item/87v54

[4] L. Dorst, "3D oriented projective geometry through versors of $\mathbb{R}^{3,3}$," *Advances in Applied Clifford Algebras*, vol. 26, pp. 1137–1172, 2016. [Online]. Available: https://link.springer.com/article/10.1007/s00006-015-0625-y

[5] S. Breuils, V. Nozick, A. Sugimoto, and E. Hitzer, "Quadric conformal geometric algebra in $\mathbb{R}(9,6)$," *Adv. Appl. Clifford Algebras*, vol. 28, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s00006-018-0851-1

[6] C. Doran, "Euclidean geometry and geometric algebra," June 2020, blogpost https://geometry.mrao.cam.ac.uk/2020/06/euclidean-geometry-and-geometric-algebra/.

[7] S. De Keninck and L. Dorst, "Geometric Algebra Levenberg-Marquardt," in *Advances in Computer Graphics*, M. Gavrilova, J. Chang, N. Thalmann, E. Hitzer, and H. Ishikawa, Eds. Cham: Springer International Publishing, 2019, pp. 511–522. [Online]. Available: https://www.springerprofessional.de/en/geometric-algebra-levenberg-marquardt/16798234

[8] ——, "Hyperwedge," 2020, (submitted to ENGAGE 2020).

[9] J. Browne, *Grassmann Algebra. Volume 1: Foundations.* Barnard Publishing, 2012. [Online]. Available: https://grassmannalgebra.com/

[10] L. Dorst, "Notational correspondence between Browne's "Grassmann algebra" and "PGA: Plane-based geometric algebra"," 2020, appendical note on `bivector.net`.

[11] S. De Keninck and C. G. Gunn, "PGA cheat sheets for the SIGGRAPH 2019 course: `3DPGA.pdf` and `2dpga.pdf`," to be found at https://bivector.net/.

[12] S. De Keninck and L. Dorst, "PGA: clean up your mesh! (working title)," 2020, in preparation.

[13] C. Gunn, "Geometry, kinematics, and rigid body mechanics in Cayley-Klein geometries," Ph.D. dissertation, TUBerlin, 2011. [Online]. Available: http://page.math.tu-berlin.de/~gunn/Documents/Papers/Thesis-final.pdf

[14] L. Dorst and R. Valkenburg, "Square root and logarithm of rotors in 3D conformal geometric algebra using polar decomposition," in *Guide to Geometric in Practice*, L. Dorst and J. Lasenby, Eds. Springer-Verlag, 2011, pp. 81–104. [Online]. Available: https://www.springer.com/gp/book/9780857298102

[15] E. Hitzer and D. Ichikawa, "Representation of crystallographic subperiodic groups in Clifford's geometric algebra," *Adv. Appl. Clifford Algebras*, vol. 23, pp. 887–906, 2013. [Online]. Available: https://link.springer.com/article/10.1007/s00006-013-0404-6

[16] D. Fontijne and L. Dorst, "Reconstructing rotations and rigid body motions from exact point correspondences through reflections," in *Guide to Geometric in Practice*, L. Dorst and J. Lasenby, Eds. Springer-Verlag, 2011, pp. 63–78. [Online]. Available: https://www.springer.com/gp/book/9780857298102

[17] C. Doran, D. Hestenes, F. Sommen, and N. Van Acker, "Lie groups as spin groups," *J. Math. Phys.*, vol. 34, no. 8, pp. 3642–3669, 1993. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.530050

[18] J. Stolfi, *Oriented Projective Geometry.* Academic Press, 1991.

[19] J. Ong, "jeremyong/klein: v2.2.1," March 2020, with Steven De Keninck and Faaux. [Online]. Available: https://zenodo.org/record/3711285

[20] L. Dorst, "The inner products of geometric algebra," in *Applications of Geometric Algebra in Computer Science and Engineering*, L. Dorst, C. Doran, and L. Lasenby, Eds. Boston: Birkhäuser, 2002, pp.

35–46. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4612-0089-5_2

Index