PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Research

Article submitted to journal

Subject Areas:
Discrete Differential Geometry, Mesh Processing

Keywords:
Projective Geometric Algebra, Moments of k-complexes, Jacobi Method

Author for correspondence:
Steven De Keninck
e-mail: s.a.j.dekeninck@uva.nl

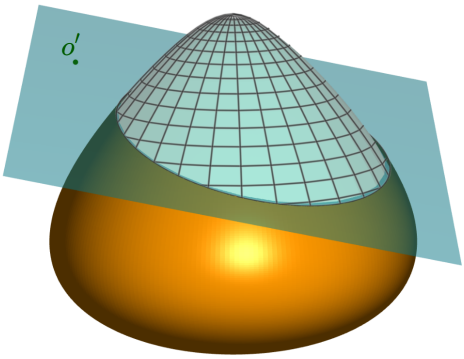
Clean up your Mesh!

Part 1: Plane and simplex

Steven De Keninck[1], Martin Roelfs[2], Leo Dorst[3] and David Eelbode[4]

[1,3]University of Amsterdam
[2,4]University of Antwerp

We revisit the geometric foundations of mesh representation through the lens of Plane-based Geometric Algebra (PGA), questioning its efficiency and expressiveness for discrete geometry. We find how k-simplices (vertices, edges, faces, ...) and k-complexes (point clouds, line complexes, meshes, ...) can be written compactly as joins of vertices and their sums, respectively. We show how a single formula for their k-magnitudes (amount, length, area, ...) follows naturally from PGA's Euclidean and Ideal norms. This idea is then extended to produce unified coordinate-free formulas for classical results such as volume, centre of mass, and moments of inertia for simplices and complexes of arbitrary dimensionality. Finally we demonstrate the practical use of these ideas on some real-world examples.



$$V = \frac{1}{6}\sum_{i\in\partial}(v_{i_0} \vee v_{i_1} \vee v_{i_2} \vee o')$$

THE ROYAL SOCIETY PUBLISHING

1. Introduction

In the geometric representation of discrete 3D meshes, it sometimes seems that there are as many 3D mesh representations as there are engineering problems involving them. For each application, a variety of topological (e.g. half-edge, adjacency matrix, triangle lists, ...), representational (e.g. position vectors, plane equations, edge vectors, ...), and performance (e.g. octrees, bsp, bvh, ...) options need to be considered. In this paper we focus on the geometric representation, and consider the efficacy of the multivectors of PGA, plane-based geometric algebra, for the representation of simplices and complexes. We believe that this encompasses most of the others, in a compact, coordinate-free framework. While we refer the reader to [5,9,10] for a more complete introduction to PGA, we shall briefly review the basics used in this article.

The main contributions of this paper are formulas for the 0th (amount, length, area, ...), 1st (center of mass) and 2nd moments (inertia) of arbitrary k-simplices and complexes. An overview of these results is given in table 1, where the header compactly summarizes the main results to be proven in this work, while the rows present specific formulas that can be implemented directly using any modern GA library that supports PGA, e.g. [3,4,11,13]. The methodology distinguishes itself from previous methods using exterior calculus [2] or GA [6] through its inclusion of ideal elements (residing at infinity, but finitely represented in this projective algebra), which endows PGA with a Euclidean and ideal norm, neatly corresponding to magnitudes of simplices and their boundaries. Moreover, these quantities are naturally invariant under the transformations of the Euclidean group $\mathrm{E}(n)$.

This paper is concerned with motivating and proving these results, with important geometric intuitions being given throughout. We show an involved practical example of measuring the volume of an airplane tank mesh, where PGA provides a very compact solution. Table 1 shows that the magnitudes of complexes of various dimensions follow a straightforward pattern when expressed in PGA.

k-simplex $S_k = v_0 \vee \ldots \vee v_k$ point, edge, triangle, tetrahedron, ...	k-complex $C_k = \sum S_k$ point set, polygon, mesh, ...	k-magnitude $\frac{1}{k!}\|S_k\|\,,\ \frac{1}{k!}\|C_{k-1}\|_\infty$ amount, length, area, volume, ...	k-com $\frac{\sum_\partial (\sum v_i + o)(S_{k-1} \vee o)}{(k+1)!}$ center of mass
Vertex v		$(x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3 + \mathbf{e}_0)^*$	
Edge E		$v_1 \vee v_2$	
Triangle F		$v_1 \vee v_2 \vee v_3$	
Tetrahedron T		$v_0 \vee v_1 \vee v_2 \vee v_3$	
Vertex count		$\|\sum v_i\|$	
Edge length		$\|E\|$	
Face area		$\frac{1}{2}\|F\|$	
Tetrahedron volume		$\frac{1}{6}\|T\|$	
Planar polygon area (via ∂E)		$\frac{1}{2}\|\sum E_i\|_\infty = \frac{1}{2}\sum(E_i \vee o)$	
Mesh area (via ∂F)		$\frac{1}{2}\sum\|F_i\|$	
Mesh volume (via ∂F)		$\frac{1}{6}\|\sum F_i\|_\infty = \frac{1}{6}\sum(F_i \vee o)$	
Area of sum of missing ∂F		$\frac{1}{2}\|\sum F_i\|$	
Mesh center of mass (via ∂F)		$\frac{1}{24}\sum(v_{i_1} + v_{i_2} + v_{i_3} + o)(F_i \vee o)$	

Table 1: Magnitudes of simplices and complexes.

2. The Geometric View of Clifford Algebras

This paper shows a use of 3D PGA $\mathbb{R}_{3,0,1}$, Plane-based Geometric Algebra, which is a novel model of Euclidean geometry [10]. As a geometric algebra, it is an interpretation of Clifford algebra, which mathematically might be described as a tensor algebra modulo the ideal $x \otimes x - Q(x)$, with $Q(\,)$ the chosen quadratic form (this effectively states that any square of a vector x can be replaced by the scalar $Q(x)$). However, Clifford algebra's geometric interpretation is more directly explained by introducing an equivalent 'geometric algebra' (GA) through elementary symmetries.

In the practice of GA, given the desire to work in a geometry (such as the Euclidean geometry of this paper), we set up a vector space with a quadratic form (metric) such that the elementary symmetry operations are representable as orthogonal transformations of the representational space. This is advantageous since GA has an effective representation of orthogonal transformations, by spinor-like elements called k-reflections. Let us show briefly how this geometric view indeed produces a practical Clifford algebra.

(a) Transformations as k-reflections

According to the Cartan-Dieudonné theorem, all orthogonal transformations in a d-dimensional space W with quadratic form Q can be written as at most d reflections in hyperplanes of W. So let us consider a hyperplane reflection of a vector $x \mapsto M(x)$, in a hyperplane through the origin with (non-null) normal vector a. From linear algebra, we easily derive the expression

$$M(x) = x - 2(x \cdot a)a/(a \cdot a),$$

where we introduced the dot product notation $a \cdot a \equiv Q(a)$ (which by polarization gives $a \cdot b = \frac{1}{2}(Q(a+b) - Q(a) - Q(b))$).

In geometric algebra, we consider the dot product of vectors as the symmetric part of the *geometric product* (denoted by a half-space), $a \cdot b = \frac{1}{2}(a\,b + b\,a)$. When we substitute this into $M(x)$, we get a more compact expression if we assume this product to be bilinear, distributive, associative (but not commutative), commuting with scalars:[1]

$$M(x) = x - 2(x \cdot a)\,a/(a \cdot a) = x - (x\,a + a\,x)\,a/a^2 = x - x - a\,x\,a/a^2 = -a\,x\,a^{-1}. \qquad (2.1)$$

In this final form, the multiple reflections of Cartan-Dieudonné are easily performed. Two consecutive reflections in planes a and b lead to the $x \mapsto b\,a\,x\,a^{-1}\,b^{-1} = (ab)\,x\,(ab)^{-1}$. Therefore the element (ab) of the geometric algebra denotes a double reflection – in the 3D algebra $\mathbb{R}_3$, this would be a rotation around the origin (and in fact isomorphic to a quaternion).

For a d-dimensional space W, its orthogonal transformations can thus be represented as elements consisting of the geometric product of k (which is at most d) invertible vectors, called a *k-reflection*. Such a k-reflection V is to be applied to a vector x by the 'sandwiching' product $x \mapsto (-1)^k V\,x\,V^{-1}$. We will extend this to general elements below.

(b) Choosing a Geometric Algebra

By a sensible choice of representational space, we can represent a variety of geometries. For the Euclidean geometry of this paper, we need to represent rotations and translations (and their combinations as screw motions) as multiple reflections. Since these operators can be made as multiple planar reflections, the GA approach dictates that we choose a representational space whose vectors represent planes in Euclidean space.

The familiar homogeneous plane equation $ax + by + cz + d = 0$ involves the homogeneous 4-D 'plane-vector' $[a, b, c, d]^T$. The space of such vectors is the representational space of 3D PGA, plane-based geometric algebra.

[1] ... which makes this geometric product in fact identical to the Clifford product from the formal tensor-based Clifford algebra definition.

We need to establish a metric so that the multiple reflections correctly represent the Euclidean transformations, and it turns out that this is $\mathbb{R}_{3,0,1}$, This signature $(3,0,1)$ implies that we can make an orthogonal basis $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ such that:

$$\begin{aligned} \mathbf{e}_i \cdot \mathbf{e}_i &= 1 \quad \text{for } i = 1,2,3 \\ \mathbf{e}_i \cdot \mathbf{e}_0 &= 0 \quad \text{for } i = 0,1,2,3 \\ \mathbf{e}_i \cdot \mathbf{e}_j &= 0 \quad \text{for } i,j = 0,1,2,3 \text{ and } i \neq j \end{aligned}$$

So $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ form an orthogonal basis of three Euclidean basis vectors and $\mathbf{e}_0$ is an orthogonal null vector for which $\mathbf{e}_0^2 = 0$. Using this basis, we would represent the plane with Euclidean unit normal vector $\vec{n}$ at signed distance δ from the origin, which has the equation $\vec{n} \cdot \vec{x} - \delta = 0$, as a multiple of the PGA vector $\vec{n} - \delta\mathbf{e}_0$.

The elements obtained by multiplying such vectors can be used to algebraically represent Euclidean transformations, and with an even number of factors are isomorphic to the dual quaternions, viewed as multiple reflections in planes.

For instance, the 2-reflection in two parallel planes $p_1 = \vec{n} - \delta_1\mathbf{e}_0$ and $p_2 = \vec{n} - \delta_2\mathbf{e}_0$ (with $\vec{n}^2 = 1$) produces the PGA element

$$p_2\, p_2 = (\vec{n} - \delta_2\mathbf{e}_0)\,(\vec{n} - \delta_1\mathbf{e}_0) = 1 - (\delta_2 - \delta_1)\mathbf{e}_0\vec{n} \equiv 1 - \mathbf{e}_0\,\vec{t},$$

where we defined $\vec{t} \equiv (\delta_2 - \delta_1)\vec{n}$, the distance vector between the planes. Then applying the element p_2p_1 in a sandwich product to a plane $\vec{m}$ at the origin gives:

$$(1 - \mathbf{e}_0\,\vec{t})\,\vec{m}\,(1 + \mathbf{e}_0\,\vec{t}) = \vec{m} - 2(\vec{m} \cdot \vec{t})\,\mathbf{e}_0,$$

which is the vector representing the $\vec{m}$-plane translated over $2\,\vec{t}$. This is indeed what we would expect from a double reflection in parallel planes. The product between two non-parallel planes produces a rotation operator, over twice their relative angle and with their common line as its axis (see e.g. [14]). The product of four (or more) vectors then encodes a general 3D Euclidean motion.

(c) Derived Products and their Semantics

For this geometric use of Clifford algebra, it is convenient to define some more 'products' derived from the fundamental geometric product. We have seen how the dot product of two vectors is the symmetric part of their geometric product. Their anti-symmetric part is the *wedge product* $\wedge$:

$$a\,b = a \cdot b + a \wedge b = \tfrac{1}{2}(a\,b + b\,a) + \tfrac{1}{2}(a\,b - b\,a).$$

This wedge product establishes a Grassmann subalgebra within the geometric algebra. It can be extended by antisymmetry to multiple factors, and then forms elements called k-blades. In PGA, the k-blade $a \wedge b$ of two plane vectors a and b is a 2-blade representing their intersection (though note that $a \wedge a = 0$, it is a linearized form of the geometric intersection operation). It is often called the '*meet product*' when used in this manner. We can make geometrical points in 3D PGA as 3-blades, since they are the intersection of 3 planes. The element $\mathbf{e}_{0123} \equiv \mathbf{e}_0 \wedge \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3$ is chosen as the *pseudoscalar* of $\mathbb{R}_{3,0,1}$; it anti-commutes with all vectors.

In addition to the polarity obtained by multiplying with the pseudoscalar, we can define a (Hodge) dual in PGA, which we will denote by * (the exact definition will follow later in section 3(a)). It associates, in a linear and invertible manner, a $(4-k)$-blade with each k-blade. The dual of the 3-blade point from PGA then becomes a 1-vector, in a more recognizable form reminiscent of the usual homogeneous coordinate representation. With this Hodge duality, we can also define the *join product* (aka regressive product) :

$$(A \vee B)^* = A^* \wedge B^*. \tag{2.2}$$

In 3D PGA, this product allows two 3-blade points to be joined to form a 2-blade line connecting them. We will show the explicit parametrization (and its connection to Plücker coordinates) later in section 3(c),(d).

The dot product can also be extended to work on multivectors, in a manner that obeys its quantitative duality with the wedge product. This is a bit involved (see e.g. [9]), and since in PGA it is more natural to use the join instead, we will forego the details.

Ultimately, an element we can produce using these derived products and their sums is a *multivector*, i.e., a weighted sum of geometric products of basis vectors in the 2^4-dimensional basis of $\mathbb{R}_{3,0,1}$. Algebraically, any such multivector X transforms in exactly the same way (i.e., equivariantly) under $2k$-reflections, namely as the derived product sometimes called *sandwiching*:

$$X \mapsto V\,X\,V^{-1}.$$

This is easy to see: linearity of the geometric product allows us to look at the basis blades only. A term like $\mathbf{e}_1\mathbf{e}_2$ would be sandwiched to $V(\mathbf{e}_1\mathbf{e}_2)V^{-1} = V\mathbf{e}_1(V^{-1}V)\,\mathbf{e}_2V^{-1} = ((-1)^{2k}V\mathbf{e}_1V^{-1})((-1)^{2k}V{\mathbf{e}_2}^{-1}V)$, which is indeed the geometric product of the transformed basis vectors. This argument extends trivially to any number of basis vector factors. In PGA, these $2k$-reflections (aka as 'even' transformations) represent rotations , translations and their compositions, and they are all we need in this paper (no net reflections). [2]

The equivariance property of the GA k-reflection representation of transformations makes it superior to the matrix representation of linear algebra, and it has been used to great advantage (most recently in Machine Learning [15,17]).

(d) Coordinate-free Constructions

In 3D PGA, a plane, line, point, reflection, rotation, translation, transflection, rotoreflection and screw are thus each written as an appropriate multivector

$$X$$

without the need for coefficients, indices (abstract or otherwise), or augmentation to indicate co- or contravariant transformation rules. All sensible (i.e., equivariant) geometrical operations between such elements can be specified and performed at this level.

- An (all even or all odd) multivector X transforms under the action of a normalized product of k vectors V, universally as $\pm VXV^{-1}$.
- The transformation between any same grade types A and B is always $\sqrt{BA^{-1}}$. That $2k$-reflection transforms A to B when used in a sandwich product on A.
- Any B can be orthogonally projected onto any invertible A to produce $(B \cdot A)A^{-1}$.
- The unique element at the intersection of blades A and B is always its *meet* $A \wedge B$, while the *join* $A \vee B$ produces the unique element that contains both. (With the caveat that degeneracy may produce a zero result, see [7].)
- The *logarithm* of any $2k$-reflection $\log V = E$ produces a bivector, and conversely the *exponential* of a bivector $V = \exp(E)$ a $2k$-reflection. This ties directly into the Lie algebra of the motions. In 3D PGA, the logarithm of a simple rotation is its axis; for a translation that axis is of the form $\mathbf{e}_0\,\vec{t}$, and resides in the plane at infinity. The logarithm of a general 3D Euclidean motion is its screw bivector.

In this paper, we stay at the multivector level of specification as much as we can, but will occasionally derive the classical coordinate expression to show the correspondence.

3. Euclidean Primitives and their Split

Table 2 shows the geometric interpretation of the relevant multivectors of PGA: those of single grades (k-blades and k-vectors) and elements with only even or only odd grades.

[2]For general k-reflections, we can split $X = X_+ + X_-$ in its even grade parts X_+ and its odd grade part X_-. Then the transformation formula is: $X \mapsto VX_+V^{-1} + \hat{V}X_-V^{-1}$. Incidentally, our derived products only produce all even or all odd multivectors when applied successively to vectors.

	s	bivector						pss	vector			trivector			
elements	1	$\mathbf{e}_{23}$	$\mathbf{e}_{31}$	$\mathbf{e}_{12}$	$\mathbf{e}_{01}$	$\mathbf{e}_{02}$	$\mathbf{e}_{03}$	$\mathbf{e}_{0123}$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_0$	$\mathbf{e}_{032}$	$\mathbf{e}_{013}$	$\mathbf{e}_{021}$	$\mathbf{e}_{123}$
		line$_o$ $\mathfrak{so}(3)$			line$_\infty$ $\mathfrak{t}(3)$				plane				point / direction			
		line / screw $\mathfrak{se}(3)$							$a\mathbf{e}_1+b\mathbf{e}_2+c\mathbf{e}_3+d\mathbf{e}_0$				$(x\mathbf{e}_1+y\mathbf{e}_2+z\mathbf{e}_3+w\mathbf{e}_0)^*$			
transformations	quaternion $SO(3)$				translation $T(3)$				plane-reflection $E(3)$				point-reflection $E(3)$			
	dual quaternion $SE(3)$															
	2-reflection								1-reflection							
	4-reflection even multivectors $\mathbb{R}^+_{3,0,1}$								3-reflection odd multivectors $\mathbb{R}^-_{3,0,1}$							

Table 2: $\mathbb{R}_{3,0,1}$ basis, elements, transformations. Here "s" denotes scalar, "pss" denotes pseudoscalar. Subscript "o" means at the (arbitrary) origin, "∞" at infinity (or ideal).

(a) The Euclidean Split and Dualization

PGA is thus a projective way of modeling Euclidean geometry, departing from its planes modelled as vectors. Its null basis vector $\mathbf{e}_0$ can be interpreted as the *plane at infinity*, also know as the *ideal plane*. Rotations around an ideal line in this ideal plane model translations, which to us Euclidean beings feel rather different from regular rotations around a finite axis. Also, finite points feel different from directions, which in PGA are modelled as ideal points (i.e., points at infinity, on the ideal plane).

When we establish useful formulas, this distinction between the 'finite' and the 'ideal' is important, even though such elements may be united within a given PGA multivector to represent proper geometry. So let us introduce the *Euclidean split*, breaking up our multivectors into Euclidean and Ideal terms:

$$A = A_E + \mathbf{e}_0 A_I \ , \tag{3.1}$$

where the *Euclidean term* is A_E and the Ideal term contains A_I, which we will refer to as the *Ideal factor*. Both A_E and A_I are elements of the subalgebra $\mathbb{R}_3$. In an implementation, it is easy to make this split: one just selects which parts are given on the 2^3-dimensional Euclidean basis (involving only the vectors $\mathbf{e}_1\mathbf{e}_2, \mathbf{e}_3$ and their products), and which on the 2^3 basis elements also having a factor $\mathbf{e}_0$.

For the definition of the join, we need a PGA dualization operation. The *Hodge dual* (aka right compliment dual) of any basis-blade $\mathbf{e}_J$ with J a multi-index is defined to satisfy $\mathbf{e}_J\, \mathbf{e}^*_J = \mathbf{e}_{0123}$. By demanding linearity, this determines the Hodge dual of any multivector.

Using the Euclidean split, the Hodge (un)dual of a general multivector A can be written using the *grade involution* (hat[3]), the *reversion* (tilde[4]) and the Euclidean 3D pseudo-scalar $I_3 = \mathbf{e}_{123}$ as

$$A^* = \widetilde{A_I}\, I_3 + \mathbf{e}_0\, \widehat{\widetilde{A_E}}\, I_3, \qquad A^{-*} = -\widehat{\widetilde{A_I}}\, I_3 + \mathbf{e}_0\, \widetilde{A}_E\, I_3, \tag{3.2}$$

where A^{-*} denotes the undual. One may recognize the Euclidean duals of A_E and A_I in these expressions. We can trivially verify that $(A^*)^{-*} = (A^{-*})^* = A$.

Proof. To demonstrate that A^* of eq. (3.2) is indeed the right complement dual in PGA, it suffices to show that $AA^* = A^{-*}A = \mathbf{e}_{0123}$, for all basis blades $A = \mathbf{e}_J$ of $\mathbb{R}_{3,0,1}$. This can be done by direct computation, when one remembers that for basis blades we have $A_E A_I = 0$, since they are either purely Euclidean or purely Ideal. □

[3] $\widehat{X}$, the grade involution of X, inverts the sign of the odd grade parts. Note that $\widehat{AB} = \widehat{A}\widehat{B}$.
[4] $\widetilde{X}$, the *reversion* of X, inverts the sign of grades 2 and 3 (modulo 4). Note that $\widetilde{AB} = \widetilde{B}\widetilde{A}$.

We can now give the split definition of the PGA meet and join, reducing them to computations in the Euclidean subalgebra $\mathbb{R}_3$:

$$(A_E + \mathbf{e}_0\, A_I) \wedge (B_E + \mathbf{e}_0\, B_I) = (A_E \wedge B_E) + \mathbf{e}_0\, (\widehat{A_E} \wedge B_I + A_I \wedge B_E) \tag{3.3}$$

and

$$\begin{aligned}&(A_E + \mathbf{e}_0\, A_I) \vee (B_E + \mathbf{e}_0\, B_I)\\ &= \left((\widehat{B_I} I_3) \wedge (A_E I_3)\right) I_3^{-1} - \left((B_E I_3) \wedge (A_I I_3)\right) I_3^{-1} - \mathbf{e}_0 \left((B_I I_3) \wedge (A_I I_3)\right) I_3^{-1}.\end{aligned} \tag{3.4}$$

The terms in the result are effectively joins in $\mathbb{R}_3$, see eq. (2.2) and also [9].

(b) Norm and Ideal Norm.

The Euclidean split eq. (3.1) leads us to define two useful PGA norms. The Euclidean norm of a PGA element A is equal to the norm of its Euclidean term A_E and given simply by

$$\|A\| \equiv \sqrt{\tilde{A} A} = \|A_E\|.$$

Because $\mathbf{e}_0^2 = 0$, the A_I factor does not impact the Euclidean norm. We will however be interested in the magnitude of A_I, so we define the ideal norm as the norm of that ideal factor, and give three equivalent alternatives to compute it:

$$\|A\|_\infty \equiv \|A_I\| = \|A \vee o\| = \|A^*\|.$$

Here o is our notation for the point at the origin $o = \mathbf{e}_0^* = I_3$. Using o, the ideal factor A_I may be extracted from A as $A \vee o = o \vee \widehat{A}$ (since $A_E \vee \mathbf{e}_0^* + (\mathbf{e}_0 A_I) \vee \mathbf{e}_0^* = A_I$), and we will often rewrite it in this form.

Note that in Euclidean PGA, both these norms are positive semi-definite and hence cannot be used to extract signed magnitudes. An exception is when $A \vee o$ is a scalar, this scalar then representing the signed ideal magnitude (this happens when A is a 1-vector, i.e., represents a hyperplane). In that case, we use the signed expression $A \vee o$ instead of $\|A\|_\infty$.

We can normalize elements through dividing them by the appropriate norm. A general element with non-zero Euclidean term is usually normalized by the Euclidean norm, and we denote this normalization by an overline. So for a plane $p = \alpha(\vec{n} - \delta \mathbf{e}_0)$ with $\vec{n}^2 = 1$, we have $\overline{p} = \vec{n} - \delta \mathbf{e}_0$ (and note that then $\overline{p}^2 = \vec{n}^2 = 1$). A point v at location $\vec{v}$ is normalized to the form $\overline{v} = I_3 - \mathbf{e}_0 \vec{v} I_3$, and $\vec{v}^2 = -1$, as you would expect from a point reflection operator in 3D. We will mostly work with normalized points and vertices in this paper, so to unclutter the formulas we will drop the overline for them. If an element has only an ideal term, we would normalize it by the infinity norm.

(c) From Planes to Lines and Points

In this section we show how in PGA the outer product of planes generates lines and points as blades, and that the Euclidean split of those blades contains useful and recognizable information on relevant magnitudes, in preparation for their use in mesh processing.

As we have seen, a homogeneous linear equation representing a plane is embedded as a PGA vector:

$$ax + by + cz + d = 0 \overset{\text{embed}}{\Longrightarrow} p = a\mathbf{e}_1 + b\mathbf{e}_2 + c\mathbf{e}_3 + d\mathbf{e}_0 \overset{\text{split}}{\Longrightarrow} \vec{n} + \mathbf{e}_0 d \tag{3.5}$$

where the Euclidean split in the last rewrite exposes the normal vector $\vec{n} = a\mathbf{e}_1 + b\mathbf{e}_2 + c\mathbf{e}_3$ and $-d$, the plane's signed distance from the origin (scaled with the norm $\sqrt{a^2 + b^2 + c^2}$ of the normal

vector). The outer product of two such planes in Euclidean split form,

$$p_1 \wedge p_2 = (\vec{n}_1 + \mathbf{e}_0 d_1) \wedge (\vec{n}_2 + \mathbf{e}_0 d_2) = \vec{n}_1 \wedge \vec{n}_2 + \mathbf{e}_0(d_1 \vec{n}_2 - d_2 \vec{n}_1) \tag{3.6}$$

reveals, on a bivector basis, the well known cross product components $[\vec{n}_1 \times \vec{n}_2 \,;\, d_1 \vec{n}_2 - d_2 \vec{n}_1]$ for the Plücker coordinates of their intersection line. The norm and ideal norm for such a line

$$\begin{aligned} \|p_1 \wedge p_2\| &= \|\vec{n}_1 \wedge \vec{n}_2\| = \|\vec{n}_1\| \|\vec{n}_2\| \sin\theta \\ \|p_1 \wedge p_2\|_\infty &= \|d_1 \vec{n}_2 - d_2 \vec{n}_1\| \end{aligned} \tag{3.7}$$

encode the relevant angles and distances. For normalized planes (with $p_1^2 = p_2^2 = 1$), the Euclidean norm produces the sine of the angle between the planes, while the ideal norm produces the distance between the planes for parallel planes, and the absolute distance from the line to the origin otherwise.

We can construct a point 3-vector v at a location $\vec{v}$ by intersection the 3 coordinate planes $p_i = \mathbf{e}_i - v_i \mathbf{e}_0$ specifying the point coordinates v_i. This results in

$$v = \mathbf{e}_{123} - \mathbf{e}_0(v_1 \mathbf{e}_{23} + v_2 \mathbf{e}_{31} + v_3 \mathbf{e}_{12}) = \mathbf{e}_{123} - \mathbf{e}_0 \vec{v} \mathbf{e}_{123} = (\mathbf{e}_0 + \vec{v})^*. \tag{3.8}$$

The result clearly shows how the usual homogeneous point representation of the parametrized form $(1, \vec{v})$ embeds naturally into PGA through the Hodge dual.

In PGA, the numerical representation of a vertex does not depend on the choice of origin o. For if we choose as origin instead of $o = \mathbf{e}_{123} = I_3$ a point $q = I_3 - \mathbf{e}_0 \vec{q} I_3$ at location $\vec{q}$, the vertex v relative to that new location would have position vector $\vec{v} - \vec{q}$. Vertex v would therefore be constructed as $v = (I_3 - \mathbf{e}_0 \vec{q} I_3) - \mathbf{e}_0(\vec{v} - \vec{q}) I_3 = I_3 - \mathbf{e}_0 \vec{v} I_3$, i.e., it would be literally unchanged. Specifying vertices numerically in PGA notation therefore does *not* mean that we have chosen a fixed origin: the vertex's PGA 'coordinates' can be used relative to any origin. The corresponding relative position vector appears automatically as one performs such an 'origin split' on the invariant vector v.

Using this representation of a point, it is easy to verify that $v \vee p = 0$ for a plane $p = \vec{n} - \delta \mathbf{e}_0$ indeed reproduces the normal plane equation $\vec{v} \cdot \vec{n} - \delta = 0$.

(d) From Points to Lines and Planes

The PGA *join* of points generates lines and planes as blades. Again, the Euclidean split of those blades contains useful and recognizable information on relevant magnitudes, in which we recognize common expressions. The involved coordinate expressions commonly used to compute those are therefore just retrievable aspects of the join of PGA points.

A normalized point v, at Euclidean position vector $\vec{v} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$ is embedded as a PGA dual vector. i.e., a 3-blade

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \overset{\text{embed}}{\Longrightarrow} v = (\mathbf{e}_0 + \vec{v})^* \overset{\text{split}}{\Longrightarrow} I_3 - \mathbf{e}_0 \vec{v} I_3. \tag{3.9}$$

The join between two points v_0, v_1 (assumed normalized unless otherwise mentioned) produces the carrier line connecting them:

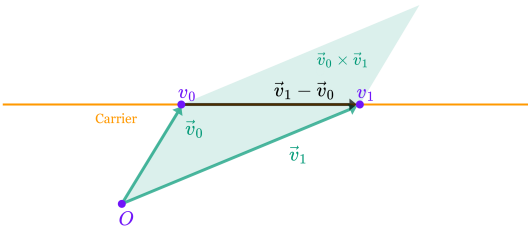
$$v_0 \vee v_1 = (\vec{v}_1 - \vec{v}_0) I_3 + \mathbf{e}_0(\vec{v}_1 \wedge \vec{v}_0) I_3. \tag{3.10}$$

This expression on a 6-dimensional 2-vector basis is directly related to the Plücker coordinates $(\vec{v}_1 - \vec{v}_0, \vec{v}_0 \times \vec{v}_1)$, using the fact that the cross product between two vectors is Euclidean dual to their outer product: $\vec{v}_0 \times \vec{v}_1 = (\vec{v}_1 \wedge \vec{v}_0) I_3$. It shows direction vector and moment of the line.

It immediately follows that the norm and ideal norm of such a carrier line

$$\begin{aligned} \|v_0 \vee v_1\| &= \|\vec{v}_1 - \vec{v}_0\|, \\ \|v_0 \vee v_1\|_\infty &= \|\vec{v}_0 \wedge \vec{v}_1\| = \|\vec{v}_0\|\|\vec{v}_1\| \sin\theta. \end{aligned} \tag{3.11}$$

denote respectively the distance between the joined points, and twice the area of the triangle formed by our two points and the origin, since $\|v_0 \vee v_1\|_\infty = \|v_0 \vee v_1 \vee o\|$. The figure below illustrates all the relevant quantities.



Let us join in a third point v_2, giving the quantitative representation of a 3D carrier plane:

$$v_0 \vee v_1 \vee v_2 = ((\vec{v}_2 - \vec{v}_0) \wedge (\vec{v}_1 - \vec{v}_0))I_3 + \mathbf{e}_0(\vec{v}_0 \wedge \vec{v}_1 \wedge \vec{v}_2)I_3. \tag{3.12}$$

As before, both norms contain valuable metric information. We rewrite slightly:

$$\begin{aligned} \|v_0 \vee v_1 \vee v_2\| &= \|(\vec{v}_2 - \vec{v}_0) \wedge (\vec{v}_1 - \vec{v}_0)\|, \\ &= \|\vec{v}_0 \wedge \vec{v}_2 + \vec{v}_2 \wedge \vec{v}_1 + \vec{v}_1 \wedge \vec{v}_0\| \\ \|v_0 \vee v_1 \vee v_2\|_\infty &= \|\vec{v}_0 \wedge \vec{v}_1 \wedge \vec{v}_2\| = \|(\vec{v}_0 \times \vec{v}_1) \cdot \vec{v}_2\| \,. \end{aligned} \tag{3.13}$$

For the Euclidean norm we see the sum of the bivectors (and their associated signed areas) for each (consistently ordered) pair of vectors. This results in twice the area of the triangle defined by our points as can be seen in fig. 1 on the next page. As the ideal norm, we recognize the scalar triple product, known to produce the volume of the parallelepiped spanned by our three vectors, or, more geometrically, 6 times the volume of the tetrahedron defined by our three points and the origin.

Joining with a fourth point v_3 gives the last carrier in the series, which is a scalar:

$$v_0 \vee v_1 \vee v_2 \vee v_3 = \big((\vec{v}_3 - \vec{v}_0) \wedge (\vec{v}_2 - \vec{v}_0) \wedge (\vec{v}_1 - \vec{v}_0)\big)\, I_3, \tag{3.14}$$

which equals $3! = 6$ times the oriented volume of the tetrahedron spanned by the points. All these results are summarized in table 3, which illustrates the clarity offered by PGA. The pattern even continues beyond 3D.

magnitude	**Vector Calculus**	**PGA**
amount of $[v_0]$	?	$\|v_0\|$
length of edge $[v_0, v_1]$	$\|\vec{v}_1 - \vec{v}_0\|$	$\|v_0 \vee v_1\|$
area of triangle $[v_0, v_1, v_2]$	$\frac{1}{2}\|(\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0)\|$	$\frac{1}{2}\|v_0 \vee v_1 \vee v_2\|$
volume of tetrahedron $[v_0, \ldots, v_3]$	$\frac{1}{3!}\|[(\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0)] \cdot (\vec{v}_3 - \vec{v}_0)\|$	$\frac{1}{3!}\|v_0 \vee v_1 \vee v_2 \vee v_3\|$
magnitude of 5-cell $[v_0, \ldots, v_4]$	?	$\frac{1}{4!}\|v_0 \vee v_1 \vee v_2 \vee v_3 \vee v_4\|$

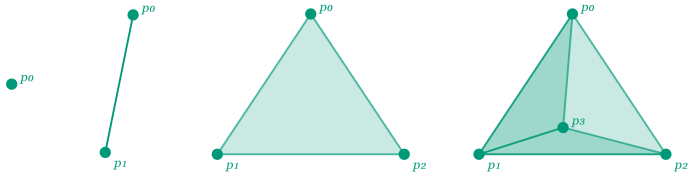
Table 3: Various k-magnitudes as computed with vector calculus compared to the PGA norms.

4. Simplices and Complexes

Not only the Clifford Algebra, but also the idea of a general k-simplex as a dimensional generalization of vertices, edges, triangles, tetrahedrons, et cetera was first introduced by W.K. Clifford [1], albeit not in the context of geometry but to solve a problem in probability. While he called it a "prime confine", in this paper we use the more modern name *k-simplex*. Additionally we will use the term *k-complex* to describe a collection of k-simplices. Moreover, a k-complex is called *complete* if for each of its k-simplices it also contains the constituent $(k-i)$-simplices for $1 \leq i \leq k$. For example, for each triangle in a triangle mesh it also contains the edges and the vertices of that triangle. Unless stated otherwise, we assume all complexes to be complete complexes for the remainder of this paper. The boundary of a k-simplex is a $(k-1)$-complex, and the boundary of a boundary of a closed manifold vanishes.

(a) k-simplices, Carriers, Carrier Gauges and Simplicial Extension

A k-simplex σ_k is a polytope that is the convex hull of its affinely independent $k+1$ vertices, i.e. the simplest shape defined by a given ordered set of vertices $\sigma_k = [v_0, \ldots, v_k]$.[5] In order of increasing k these are a vertex, an edge, a triangle, and a tetrahedron.



We define the *k-carrier* of a k-simplex σ_k as its k-dimensional infinite extension, with an appropriate k-magnitude assigned to it. The carrier of an edge is a line with an edge length measure, the carrier of a triangle is a plane with an area measure, etc. Moreover, we define the *simplicial extension* of a $(k-1)$-simplex as the addition of a single vertex v_k, which we call the apex, creating a k-simplex. Let us begin with some geometric observations about k-magnitudes (length, area, volume, ...) of k-simplices.

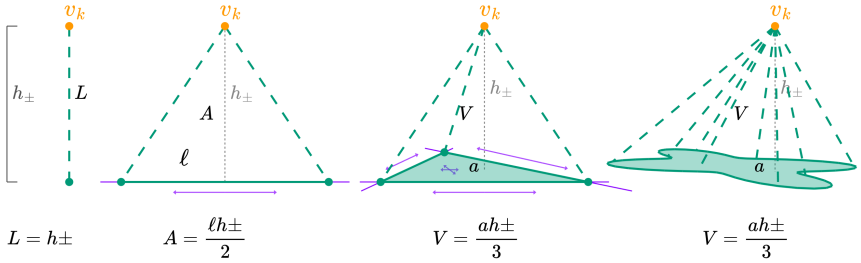


Figure 1: The simplicial extension of a simplex and the associated (signed) k-magnitudes. The carrier gauges of the $(k-1)$-carrier are shown as purple arrows.

In fig. 1, note that the k-magnitude of each k-simplex is not affected by the precise position of the apex v_k (nor any of the other vertices). Instead, it depends only on the $(k-1)$-magnitude of the $(k-1)$-carrier and the height of the apex vertex v_k to the $(k-1)$-carrier. For example, any edge of a triangle can be moved over its carrier line without changing the triangle area. Similarly, each triangle of a tetrahedron can be moved on its carrier without changing the tetrahedron's volume. This degree of freedom, which we will call a *carrier gauge*, must be present in any suitable formal definition of the carrier of a k-simplex.

As we have already seen how the join product of points creates spaces of higher dimensions that contain those points in section 3(d), we define the k-carrier of a k-simplex as follows:

[5] The square brackets employ the k-chain notation from algebraic topology [12].

Definition 4.1 (k-carrier of a k-simplex $\sigma_k = [v_0, \ldots, v_k]$)**.**

$$S(\sigma_k) = v_0 \vee \cdots \vee v_k$$

When there is no ambiguity, we may write $S_k = S(\sigma_k)$.

By construction S_k is the desired carrier space, but we must still verify that it (i) captures the correct magnitude, (ii) captures carrier gauges, and (iii) is oriented.

Firstly, in order to show that definition 4.1 correctly measures the magnitude of a k-simplex, consider the k-simplex $\sigma_k = [v_0, \ldots, v_{k-1}, v_k]$, where we take v_k to be the apex vertex. Using barycentric coordinates, v_k can be expressed as

$$v_k = \sum_{i=0}^{k-1} \alpha_i v_i + h\vec{u}_k^* \,,$$

where $\sum_i \alpha_i = 1$ and $\vec{u}_k^*$ is a new affinely independent normalized infinite point. Since $v_i \vee v_i = 0$,

$$S_k = v_0 \vee \cdots \vee v_{k-1} \vee v_k = S_{k-1} \vee (h\vec{u}_k^*)$$

and hence $\|S_k\| = \|S_{k-1}\|\|h\|$.

Secondly, the invariance of S_k under transformations of the base S_{k-1} follows from the equivariance of the join product. Specifically, let $V \in \mathrm{SE}(k-1)$ be a Euclidean motion in the carrier space S_{k-1}. Then $v_i' = V v_i V^{-1}$ are the transformed vertices within the carrier space S_{k-1}, but the equivariance allows us to write

$$S_{k-1}' = v_0' \vee \ldots \vee v_{k-1}' = V S_{k-1} V^{-1} = S_{k-1} \,,$$

and hence S_k is invariant under transformations that occur in its subcarriers.

Thirdly, the carrier S_k must share the orientation of the k-simplex. In other words, swapping two neighboring vertices in a simplex swaps the orientation, i.e. $[\ldots, v_i, v_j, \ldots] = -[\ldots, v_j, v_i, \ldots]$. This is captured faithfully by the anti-symmetry of the join product under swapped operands, i.e. $\cdots v_i \vee v_j \cdots = -\cdots v_j \vee v_i \cdots$, and hence a natural property of the carrier.

In conclusion, S_k has the simple interpretation as the oriented carrier of the simplex σ_k, and its Euclidean norm represents the k-magnitude (amount, length, area, ...) of the parallelotope formed by σ_k, and it exhibits invariance under carrier gauges. This motivates the following definition of the $k \geq 0$ dimensional magnitude (amount, length, area, volume, ...) of a k-simplex σ_k (vertex, edge, triangle, tetrahedron, ...), where we introduced a factor $1/k!$ to correct for the over-counting in S_k:

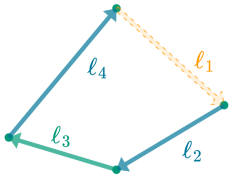
Definition 4.2 (Magnitude of a k-simplex σ_k)**.**

$$\operatorname{mag} \sigma_k \coloneqq \frac{1}{k!}\|S_k\| = \frac{1}{k!}\|v_0 \vee \cdots \vee v_k\| \,.$$

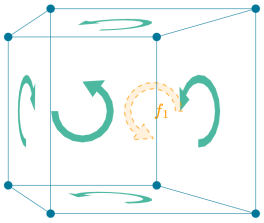
The computation of the k-magnitude of a simplex σ_k is therefore straightforward. There is however, an alternative method to compute the magnitude of a k-simplex from its boundary, which leverages PGA's inclusion of elements at infinity.

(b) Mind the gap

Consider a 2D polygon or a 3D mesh. When a polygon is closed, the sum of its edge vectors is trivially the zero vector. Similarly, for a closed mesh, the sum of its face 2-vectors will be the zero 2-vector. A direct consequence of this is that if a single edge or face is missing, its magnitude must absolutely equal the magnitude of the sum of the remaining boundary elements.



$$\|\ell_1\| = \|\ell_2 + \ell_3 + \ell_4\|$$



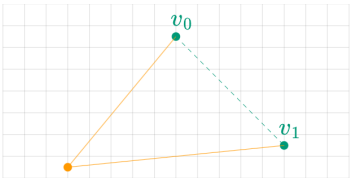
$$\|f_1\| = \|(f_2 + \ldots + f_n)\|$$

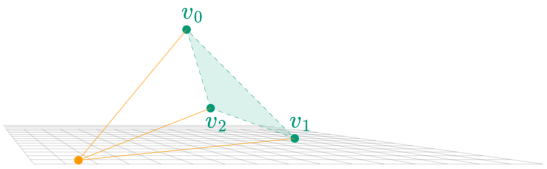
We can use this idea to find an alternate form to calculate the magnitude of a k-simplex σ_k, by considering it the gap of a $k+1$ simplex formed by the simplicial extension of σ_k with an arbitrary point.

Before we continue, we must introduce the boundary of a simplex, a standard concept in algebraic topology [12]. The boundary of a k-simplex $\sigma_k = [v_0, \ldots, v_k]$ is defined by

$$\partial\sigma_k = \sum_{i=0}^{k} (-1)^i [v_0, \ldots, \cancel{v_i}, \ldots, v_k] \,, \tag{4.1}$$

where $\cancel{v_i}$ denotes the missing term.





The figure above (left) then suggests that we can consider any edge $[v_0, v_1]$ as the missing edge, the *gap*, of a triangle formed by the v_0, v_1 and a new point v_2. We can then use the closure of $[v_0, v_1, v_2]$ to express the gap $[v_0, v_1]$ as the sum of the remaining edges. Moreover, by picking the origin o as the extra point v_2, we can factor and recognize the ideal norm $\|x\|_\infty = \|o \vee x\|$ of our boundary complex:

$$\|v_0 \vee v_1\| = \|\overbrace{o \vee v_0 + v_1 \vee o}^{\text{2 edges of triangle}}\| = \|o \vee (v_1 - v_0)\| = \overbrace{\|v_1 - v_0\|_\infty}^{\text{boundary vertices}} \,. \tag{4.2}$$

Thus, the length of the edge $[v_0, v_1]$ can be computed on the edge as $\|v_0 \vee v_1\|$ or using its boundary $\partial[v_0, v_1] = [v_1] - [v_0]$ as $\|v_1 - v_0\|_\infty$. Similarly, in the right of the figure, any triangle $[v_0, v_1, v_2]$ can be viewed as the missing face of a tetrahedron formed by v_0, v_1, v_2 and a new point v_3, which we take to be the origin o:

$$\begin{aligned} \|v_0 \vee v_1 \vee v_2\| &= \|\overbrace{o \vee v_0 \vee v_1 + o \vee v_1 \vee v_2 + o \vee v_2 \vee v_0}^{\text{three remaining faces of tetrahedron}}\| \\ &= \|o \vee (v_0 \vee v_1 + v_1 \vee v_2 + v_2 \vee v_0)\| \\ &= \|\underbrace{v_0 \vee v_1 + v_1 \vee v_2 + v_2 \vee v_0}_{\text{boundary edges of triangle}}\|_\infty \end{aligned} \tag{4.3}$$

Hence, the area of the triangle $[v_0, v_1, v_2]$ can be computed on the face as $\|v_0 \vee v_1 \vee v_2\|$ or using its boundary $\partial[v_0, v_1, v_2] = [v_1, v_2] - [v_0, v_2] + [v_0, v_1]$ as $\|v_0 \vee v_1 + v_1 \vee v_2 + v_2 \vee v_0\|_\infty$. These two examples clearly illustrate that the magnitude of k-simplex can alternatively be computed from its boundary, leading to theorem 4.1.

Theorem 4.1 (Magnitude of a k-simplex σ_k from the boundary $\partial\sigma_k$)**.** *The magnitude (volume, area, ...) of a k-simplex σ_k can be computed over the $(k-1)$-dimensional facets (areas, lengths, ...) comprising its oriented boundary $\partial\sigma_k$:*

$$\begin{aligned}\operatorname{mag}\sigma_k = \frac{1}{k!}\|S_k\| &= \frac{1}{k!}\Big\|\sum_{\sigma_{k-1}\in\partial\sigma_k} S(\sigma_{k-1})\Big\|_\infty \\ &= \frac{1}{k!}\Big\|\sum_{i=0}^{k}(-1)^i\, v_0\vee\cdots\vee\cancel{v_i}\vee\cdots\vee v_k\Big\|_\infty\,,\end{aligned}$$

where $\cancel{v_i}$ means the vertex v_i is missing from the product. Note that the sign swaps are precisely those in the definition of the boundary $\partial\sigma_k$.

Proof. Any vertex $v_i = o + \vec{v}_i^*$, and hence

$$v_0\vee\cdots\vee v_k = (o+\vec{v}_0^*)\vee\cdots\vee(o+\vec{v}_k^*) \tag{4.4}$$

$$= o\vee\sum_{i=0}^{k}(-1)^i v_0\vee\cdots\vee\cancel{v_i}\vee\cdots\vee v_k \quad + \quad \underbrace{\vec{v}_0^*\vee\cdots\vee\vec{v}_k^*}_{\text{ideal term}}\,, \tag{4.5}$$

where we repeatedly use $o\vee\vec{v}_i^* = o\vee v_i$. It therefore follows that $\|S_k\| = \|o\vee\sum_{\partial\sigma_k} S_{k-1}\| = \|\sum_{\partial\sigma_k} S_{k-1}\|_\infty$. □

The relationship between the Euclidean and ideal terms have a strong Stokes' Theorem feeling to them, as it relates relates the magnitude of a k-simplex to the magnitude of the facets on its boundary. This is not a coincidence, as will become even more clear in section 5. Table 4 gives an overview of the relationship between the Euclidean and ideal norms for various simplices.

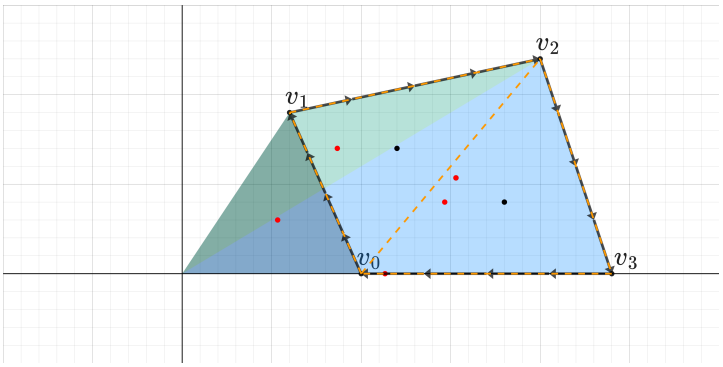
$\boldsymbol{x}$	**simplex/carrier**	$\|\boldsymbol{x}\|$	$\|\boldsymbol{x}\|_\infty = \|o\vee\boldsymbol{x}\|$
v	vertex/point	1	length of line to origin
$v_0\vee v_1$	edge/line	length of edge	2× area of triangle with origin
$v_0\vee v_1\vee v_2$	triangle/plane	2× triangle area	6× volume of tetra with origin
$v_0\vee v_1\vee v_2\vee v_3$	tetrahedron/volume	6× tetrahedron volume	0

Table 4: Norm and ideal norm for 3D PGA elements built from joining normalized vertices. The regular PGA pattern continues into d dimensions.

(c) A k-Complex and its Magnitude

Theorem 4.1 holds the key to extending from simplices to simplicial complexes, enabling us to easily calculate k-magnitudes of k-complexes $\mathcal{K}_k$. Recall that a k-complex $\mathcal{K}_k$ is a collection of k-simplices σ_k. For example, a triangle mesh is a collection of triangles (2-simplices) so it is a 2-complex, and a 3D discrete shape can be represented as a collection of tetrahedra (3-simplices) in its interior so it is a 3-complex. The common representation os a 3D mesh is however as a 2-complex representing its oriented, closed and manifold boundary. That boundary representation is independent of the actual volumetric simplices composing the complex. In practical terms, theorem 4.1 enables the direct computation of areas of arbitrary planar polygons from either the volumetric or boundary representations (without triangulation), and volumes of arbitrary triangle meshes without tetrahedralization.

To illustrate, consider the polygon shown in the figure below. It can be represented using either a 2-chain $\mathcal{K}_2 = [v_0, v_1, v_2] + [v_0, v_2, v_3]$, or using its 1-chain boundary $\mathcal{K}_1 = \partial\mathcal{K}_2 = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_0]$.



Extending the lessons of section (b) from simplices to a complexes, the signed area A of the polygon can be computed using either $\mathcal{K}_2$, or from the boundary $\partial\mathcal{K}_2$ by joining in an arbitrary apex point (taken to be the origin o):

$$A = \tfrac{1}{2}\sum_{\sigma_2\in\mathcal{K}_2} S(\sigma_2) = \tfrac{1}{2}(v_0\vee v_1\vee v_2 + v_0\vee v_2\vee v_3) \tag{4.6}$$

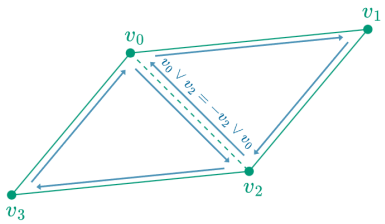
$$= \tfrac{1}{2}\, o\vee\sum_{\sigma_1\in\partial\mathcal{K}_2} S(\sigma_1) = \tfrac{1}{2}\, o\vee(v_0\vee v_1 + v_1\vee v_2 + v_2\vee v_3 + v_3\vee v_0) \tag{4.7}$$

Note that if the origin o (which can be freely chosen) coincides with any of the v_i, eq. (4.7) reduces to eq. (4.6), which shows that the two are indeed equal. But if only the unsigned area is required, eq. (4.7) simplifies to $\|\sum_{\sigma_1\in\partial\mathcal{K}_2} S(\sigma_1)\|_\infty$, see theorem 4.1. Note that to correctly compute the area it is important to first sum and then take the ideal norm, because we need to use signed areas in order for overlapping regions to cancel one another. Hence the (unsigned) k-magnitude of a k-complex $\mathcal{K}_k$ is given by the following theorem.

Theorem 4.2 (Magnitude of a k-complex $\mathcal{K}_k$)**.** *The k-dimensional magnitude of a k-complex $\mathcal{K}_k$ is a function of the k-simplices in its interior, or of the $(k-1)$-simplices on its oriented boundary $\partial\mathcal{K}_k$:*

$$\operatorname{mag}\mathcal{K}_k \coloneqq \sum_{\mathcal{K}_k}\operatorname{mag}\sigma_k = \frac{1}{k!}\sum_{\mathcal{K}_k}\|S(\sigma_k)\| = \frac{1}{k!}\|\sum_{\partial\mathcal{K}_k} S(\sigma_{k-1})\|_\infty \tag{4.8}$$

Before we provide the proof to this theorem, it is important to stress that the sum runs directly over the boundary of the k-complex, and not over the boundary of every individual k-simplex in the complex. This is due to the fact that contributions from internal boundaries of k-simplexes in the complex cancel, as the following figure shows, where the 1-chains $[v_0, v_2] = -[v_2, v_0]$, a property reflected in the anti-symmetry of the join product $v_0\vee v_2 = -v_2\vee v_0$.



We now proceed with the proof to theorem 4.2.

Proof. Recall from the proof of theorem 4.1 that $S(\sigma_k) = o \vee \sum_{\partial\sigma_k} S(\sigma_{k-1}) + \text{ideal}$, so by direct computation we find

$$\sum_{\sigma_k \in \mathcal{K}_k} S(\sigma_k) = o \vee \sum_{\sigma_k \in \mathcal{K}_k} \sum_{\sigma_{k-1} \in \partial\sigma_k} S(\sigma_{k-1}) + \text{ideal term} \,,$$

where we have been verbose to clarify every sum. However, this sum includes all internal simplices σ_{k-1}, which cancel due to the anti-symmetry of the join product if the k-complex $\mathcal{K}_k$ is consistently oriented, leaving only the boundary vertices to contribute to the sum, and thus

$$\sum_{\mathcal{K}_k} \sum_{\partial\sigma_k} S_{k-1} = \sum_{\partial\mathcal{K}_k} S_{k-1} \,.$$

Hence, $\|\sum_{\mathcal{K}_k} S_k\| = \|o \vee \sum_{\partial\mathcal{K}_k} S_{k-1}\| = \|\sum_{\partial\mathcal{K}_k} S_{k-1}\|_\infty$. □

In conclusion, we can compute the magnitude of a complex $\mathcal{K}_k$, which was originally defined in terms of a simplex decomposition, entirely in terms of the sum of the boundary simplices $\sum_{\partial\mathcal{K}_k} S_{k-1}$, by taking the ideal norm of this sum. We will now investigate what the significance of the Euclidean norm of this sum over the boundary simplices is. The sum is zero if the original complex is closed; but if the complex is not, then the Euclidean norm of this sum is actually the norm of the sum of the $(k-1)$-dimensional simplex facets missing from the boundary. This leads to the following corollary:

Corollary 4.1 (magnitude of complex gap)**.** *Given a k-complex $\mathcal{K}_k$ with a non-closed boundary $\partial\mathcal{K}_k$, i.e. $\partial\partial\mathcal{K}_k \neq 0$, the $(k-1)$-magnitude of the complex gap $\not\partial\mathcal{K}_k$ is identical to that of $\partial\mathcal{K}_k$, and hence given by*

$$\operatorname{mag} \not\partial\mathcal{K}_k = \operatorname{mag} \partial\mathcal{K}_k = \frac{1}{(k-1)!} \|\sum_{\partial\mathcal{K}_k} S_{k-1}\| \tag{4.9}$$

Hence, the magnitude of a complex gap can be measured directly, without needing to explicitly reconstruct this gap first. Note again that it is the norm of the sum, and not the sum of the norm. This magnitude will equal exactly the area of the missing boundary simplices iff those simplices share the same carrier. (i.e., if they are edges on the same line or triangles in the same plane, etc.). We shall now give a hands-on example that demonstrates the power of corollary 4.1.

(d) Example: Volume of a Capped Mesh

Let us use the formulas from table 1 on page 2, to solve a real world 3D engineering problem. Given a closed manifold mesh of an airplane fuel tank, and a plane representing the fuel's top surface, determine the remaining amount of fuel, as well as track the evolving center of mass[6]. The setup is shown in the figure on the first page.

After embedding our vertices, edges and faces as their respective PGA carriers using the unnormalized join of 1, 2 and 3 points respectively, our solution has two parts. First we determine which triangles are under the fuel plane, subdividing triangles that cross the level plane. Next, we calculate the volume and center of mass of the, no longer closed, mesh formed by the triangles under the plane.

PGA's tools will turn out to be well suited to handle both tasks. Determining which triangles are under the plane requires finding the side of the fuel plane a given vertex is on. The PGA join $\vee$ provides the perfect tool for this, as we can directly produce a signed volume by joining the fuel plane with the tested point. Worked out in coefficients, $p \vee v_i$ produces the familiar efficient expression:

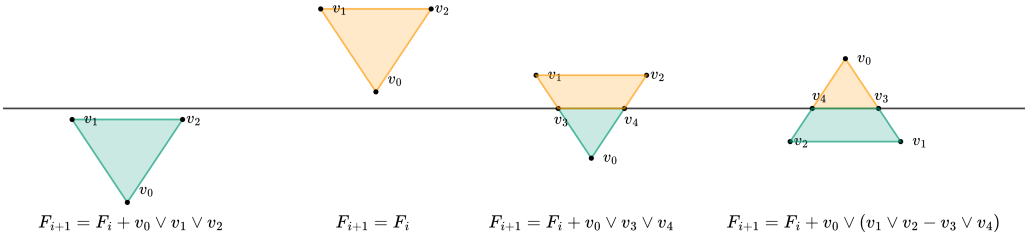
[6]This problem was first posed via bivector.net by Dr Todd Ell of Collins Aerospace. The solution presented is in use today.

$$\begin{aligned} p \vee v_i &= (a\mathbf{e}_1 + b\mathbf{e}_2 + c\mathbf{e}_3 + d\mathbf{e}_0) \vee (x\mathbf{e}_{032} + y\mathbf{e}_{013} + z\mathbf{e}_{021} + \mathbf{e}_{123}) \\ &= ax + by + cz + d. \end{aligned} \tag{4.10}$$

Armed with eq. (4.10), we iterate over all triangles in our mesh, discarding those that are entirely above the plane based on the sign of $p \vee v_i$, while retaining those entirely below. The triangles we retain are not stored, instead their associated (and precalculated) carrier plane is accumulated in a running sum $F = \sum v_0 \vee v_1 \vee v_2$. The triangles that intersect the plane must be split into three new triangles. Here the PGA meet $\wedge$ operation is the perfect tool. The vertex created when an edge E intersects a plane p is found as

$$\begin{aligned} E \wedge p &= (d_x\mathbf{e}_{01} + d_y\mathbf{e}_{02} + d_z\mathbf{e}_{03} + z\mathbf{e}_{12} + y\mathbf{e}_{31} + x\mathbf{e}_{23}) \wedge (a\mathbf{e}_1 + b\mathbf{e}_2 + c\mathbf{e}_3 + d\mathbf{e}_0) \\ &= (bd_z - cd_y - dx)\mathbf{e}_{032} + (cd_x - ad_z - dy)\mathbf{e}_{013} + \\ &\quad (ad_y - bd_x - dz)\mathbf{e}_{021} + (ax + by + cz)\mathbf{e}_{123}. \end{aligned} \tag{4.11}$$

For triangles that have a single vertex under the fuel plane, we construct and normalize the new vertices, join them with the vertex under the plane into a new triangle and add the resulting vector to the running sum. For triangles with a single vertex above the plane, we can avoid having to construct the two new triangles under the plane by realizing that all three subdivided triangles must sum up to our (precalculated) starting triangle. So in this scenario we can construct the single triangle above the plane and subtract it from our precalculated value for the entire triangle.



At the end of this iteration we have a sum F, a vector that is simply the sum of all triangles under the intersection plane. As we saw before, this is all that is needed to calculate the volume under the plane, even though this mesh is not closed. By starting with a manifold closed mesh, we know that all missing triangles lie in the same carrier, namely the intersection plane itself. Because of that, $a = \frac{1}{2}\|F\|$ is exactly the area of the gap. We also know the signed distance of the intersection plane $h = \overline{p} \vee o$ to the origin, hence we can calculate the signed volume of the simplicial extension of the gap as $\frac{1}{3}ah = \frac{1}{6}\|F\|(\overline{p} \vee o)$ enabling us to express the total volume as

$$V_{\text{fuel}} = \frac{1}{6}\big(F \vee o + \|F\|(\overline{p} \vee o)\big) \tag{4.12}$$

Understanding the arbitrary nature of the origin in this equation allows us to simplify further as for any point $o' \in p$, we have $\overline{p} \vee o' = 0$ so the second term vanishes. Hence we pick a point $o' = (o \cdot p)p^{-1}$ and find

$$V_{\text{fuel}} = \frac{1}{6}F \vee o' \tag{4.13}$$

Recalling eq. (4.10) we note that if o' is indeed chosen as the origin ($x = y = z = 0$), the expression $F \vee o'$ for a vector F is simply the extraction of the $\mathbf{e}_0$ or d coefficient of its associated linear equation $ax + by + cz + d = 0$. The other coefficients of F are then unused, and do not need to be accumulated or calculated for the intersecting triangles, reducing the accumulator to a sum of these d values. For the full triangles they are precalculated and for a new intersecting triangle, in

function of the coordinates x_i, y_i, z_i of its three points works out to just

$$d = x_3(y_2z_1 - y_1z_2) + x_2(y_1z_3 - y_3z_1) + x_1(y_3z_2 - y_2z_3)$$

When the intersection plane does not pass through the origin, we can still pick an o' with two zero coefficients, by intersecting one of the coordinate axis with p. In this case only two coefficients of F need to be calculated and accumulated, and from eq. (4.10) we note that the join with o' now involves only a single multiply-and-add.

Finally, we can halve the average runtime once more by noting that the computational cost is dominated by the number of triangles included in the sum. Since the total volume of the fuel tank is fixed, we dont always need to integrate the part below the plane. When the tank is nearly full, it is more efficient to calculate the (smaller) volume above the plane and subtract that from the total volume.

5. Moments of Simplices and Complexes

Now that we have seen how to compute k-magnitudes for simplices and complexes of arbitrary k, we turn our attention to the computation of dynamical quantities such as center of mass and inertia. Before we consider these specific quantities however, we briefly discuss arbitrary functions over a k-complex to establish some basic concepts and notation.

(a) Integrating Multivector-valued Functions over a Complex

Let $f(\sigma_k)$ be a multivector-valued function defined on a complex $\mathcal{K}_k$. How the function processes the input σ_k is up to the function, e.g. $f(\sigma_k) = \frac{1}{k+1}\sum_{i=0}^{k} v_i$ would return the center of mass of σ_k. Then the integral of $f(\sigma_k)$ over a complex $\mathcal{K}_k$ is given by

$$F(\mathcal{K}_k) = \sum_{\sigma_k \in \mathcal{K}_k} f(\sigma_k) S(\sigma_k)\,, \tag{5.1}$$

where $S(\sigma_k)$ ensures that $f(\sigma_k)$ is weighted by the signed k-magnitude of σ_k. Using the same technique as in theorems 4.1 and 4.2 the integration can also be performed over the boundary of the k-complex $\mathcal{K}_k$:

$$F(\mathcal{K}_k) = \sum_{\sigma_k \in \mathcal{K}_k} f(\sigma_k) S(\sigma_k) = \sum_{\sigma_{k-1} \in \partial\mathcal{K}_k} f([o, \sigma_{k-1}])(o \vee S(\sigma_{k-1})) + \text{ideal term}\,. \tag{5.2}$$

We have already seen that when $f(\sigma_k) = 1$ we obtain the signed k-magnitude, and in the following sections we will see how the integral of $f(\sigma_k) = \frac{1}{k+1}\sum_{i=0}^{k} v_i$ leads to the center of mass (c.o.m.), and there is even a function that leads to the inertia tensor.

(b) Center of Mass of a Mesh/Complex

Theorem 5.1 (Center of mass). *In k-D PGA, a k-simplicial complex $\mathcal{K}_k$, defined by a boundary of $(k-1)$-simplices $\partial\mathcal{K}_k$, with a uniform mass distribution, has a center of mass C_{com} given by*

$$\begin{aligned} C_{com} = \frac{1}{\|C\|}\int_C \vec{x}\, dV &= \frac{1}{\|C\|(k+1)!}\sum_{\partial\mathcal{K}_k}(\sum_{i=0}^{k-1} v_i + o)(S_{k-1} \vee o) \\ &= \frac{1}{\|C\|(k+1)!}\sum_{\partial\mathcal{K}_k}(v_0 + \ldots + v_{k-1} + o)(v_0 \vee \cdots \vee v_{k-1} \vee o) \end{aligned} \tag{5.3}$$

Where in practice, we prefer to leave out the factor $\frac{1}{\|C\|}$, and arrive with a homogeneous point at the center of mass scaled with the volume. In this form several such quantities can be added to find the composite center of mass.

We illustrate this in fig. 2 on the next page, where the choice of extra point produces the same area and center of mass for our pentagon.

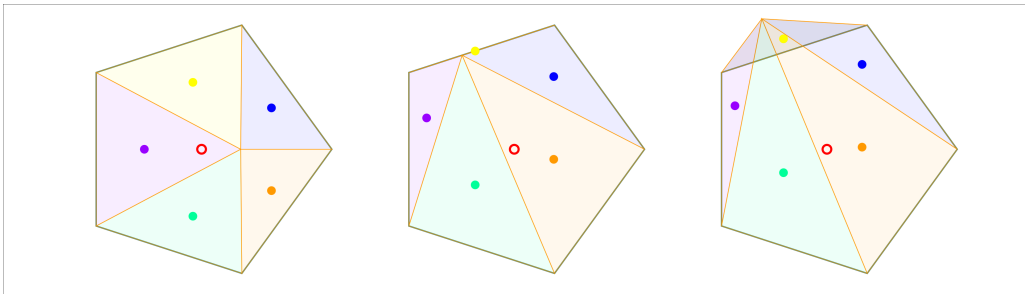


Figure 2: The area of the pentagon is the sum of the signed areas of the triangulation w.r.t. to any point. The c.o.m. of the pentagon, drawn as red circle, is the area-weighted-average of the c.o.m. of the individual triangles, drawn as solid dots. In d-D, the triangles become pyramidal cones from the arbitrary apex point to the boundary facets, still contributing their hypervolume-weighted centroids to the centroid of the complex. But the polygon centroids are then $(d-k)$-blades, whose intersection with the k-blade representing the carrier plane gives a centroid point d-blade.

(c) Example Continued

We continue the example above, to compute the center of mass of the sliced fuel tank. We can again proceed without knowing the exact location of the missing triangles by picking our $o' \in p$. We now find

$$F_{\text{com}} = \frac{1}{24} \sum (v_0 + v_1 + v_2 + o')(v_0 \vee v_1 \vee v_2 \vee o'). \tag{5.4}$$

The resulting homogeneous point is then located at the center of mass, and scaled with the total volume. As before a proper choice of o' minimizes the computations needed, and as before the homogeneous scale ensures that the sum of the center of mass below the plane and that above equals the total center of mass.

The total volume of the fuel can be computed from this center of mass as its Euclidean norm, hence no separate computation is required. An interactive demonstration is available [**?**].

(d) Inertia of a Mesh/Complex

In an earlier article on kinematics and dynamics in PGA [8] we assumed an arbitrary rigid body was aligned to its principal axis and that the principal moments were known. Using those, one can then construct an inertial duality map. Finding that inertial frame for an arbitrary mesh was left as an exercise to the reader. Now that we have calculated the 0th moment (volume) and 1st moment (center of mass) for an arbitrary mesh, it is a natural question if we can do the same for the 2nd moment - the inertia tensor.

While the scalar volume and vector center of mass have natural representations in PGA, the inertia tensor is a rank 2 tensor that has no direct PGA equivalent. We can however still use the same techniques once we realize that this rank 2 tensor can be represented as a set of three vectors: the same vectors that would make up its matrix representation.

Using this representation we can then accumulate inertia over the boundary of our mesh as before, and directly construct the $2k$-reflection (aka *rotor*) required for diagonalization, enabling us to align our arbitrary mesh with its inertial frame. This allows us to avoid using matrix representations where we would need to convert the aligning matrix back to a motor.

Definition 5.1 (Tetrahedron Inertial Frame)**.** *The canonical inertia around the origin for a solid tetrahedron defined by three points at positions* $[x_i, y_i, z_i]$ *and the origin can be described by a frame of*

three vectors I_1, I_2, I_3, converted from a scalar version in the literature (e.g. [16]), can be written as

$$\begin{aligned} I_1 &= (I_Y + I_Z)\mathbf{e}_1 + I_{XY}\mathbf{e}_2 + I_{XZ}\mathbf{e}_3 \\ I_2 &= I_{XY}\mathbf{e}_1 + (I_Z + I_X)\mathbf{e}_2 + I_{YZ}\mathbf{e}_3 \\ I_3 &= I_{XZ}\mathbf{e}_1 + I_{YZ}\mathbf{e}_2 + (I_X + I_Y)\mathbf{e}_3 \\ I_X &= X \cdot (X + X') \\ I_{XY} &= -X \cdot Y - \frac{1}{2}(X \cdot Y' + Y \cdot X') \\ X = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3, &\qquad X' = x_2\mathbf{e}_1 + x_3\mathbf{e}_2 + x_1\mathbf{e}_3 \end{aligned} \tag{5.5}$$

with the definitions of $\{Y, Z\}$, $\{Y', Z'\}$, $\{I_Y, I_Z\}$ and $\{I_{XZ}, I_{YZ}\}$ following those for X, X', I_X and I_{XY} respectively. This inertial frame I_i can now be weighted with the signed volume and accumulated for all triangles in the mesh - similar to our calculation of the center of mass.

$$I_{\text{tot}i} = \frac{1}{10\|C\|}\sum_{\partial C} I_i(o \vee v_0 \vee \cdots \vee v_{k-1}). \tag{5.6}$$

Such a set of vectors I_{tot} (corresponding to a symmetric matrix) can always be diagonalized, which we will do in our framework using the Jacobi algorithm. For this we first need to define the GA equivalent of a matrix similarity transformation ($OAO^\top$) by an orthogonal matrix (O); in our case a *rotor* (normalized $2k$-reflection) applied to such a frame.

Definition 5.2 (Similarity Transformation of a frame I_i)**.** *Given a rotor R, and a frame of vectors I_i, the similarity transformation which in matrix language would be written $\mathbf{I}' = \mathbf{RIR^{-1}}$ can be calculated in PGA as*

$$I'_i = \sum_{j=1}^{3} R(\mathbf{e}_j \cdot (\tilde{R}\mathbf{e}_i R)I_j)\tilde{R} \tag{5.7}$$

This operation plays a central role in the Jacobi algorithm that we will use to diagonalize this frame, and find the eigenframe of a simplicial complex.

Definition 5.3 (Diagonalized Eigenframe)**.** *Given a set of three symmetric vectors I_i, there exists a rotor R so that I'_i, the similarity transformation of I by R, is a set of eigenvalue scaled basis vectors.*

$$I'_i = \sum_{j=1}^{3} R(\mathbf{e}_j \cdot (\tilde{R}\mathbf{e}_i R)I_j)\tilde{R} = \lambda_i \mathbf{e}_i \tag{5.8}$$

To find the $R = R_1 R_2 \cdots R_k$, we perform a set of consecutive Givens rotations, following the standard implementation of the Jacobi eigenvalue algorithm. We iterate over the principal bivectors $\mathbf{e}_{pq}$, generating a rotor R_k in the k-th iteration step as

$$\begin{aligned} R_k &= e^{-\frac{1}{2}\phi\mathbf{e}_{pq}} \\ \phi &= \tan^{-1}\left(\frac{\operatorname{sign}\tau}{|\tau| + \sqrt{1+\tau^2}}\right) \\ \tau &= \frac{I_q \cdot \mathbf{e}_q - I_p \cdot \mathbf{e}_p}{2I_p \cdot \mathbf{e}_q} \end{aligned} \tag{5.9}$$

Updating our frame I_i each step with R_k using eq. (5.7). Steps where the denominator of τ are very small are skipped as the I_p vector is sufficiently orthogonal to $\mathbf{e}_q$. The algorithm terminates when this denominator is sufficiently small for all basis planes. The reverse of the resulting rotor R

can then be used to align a mesh with its eigenframe, while the resulting frame satisfies $I'_i = \lambda_i \mathbf{e}_i$. The matching eigenvectors are $R\mathbf{e}_i\tilde{R}$. This technique avoids the otherwise needed conversion from matrices to motors, and allows us to calculate the inertial bivector used in [8], within the PGA framework.

6. Conclusion

The join operation in 3D PGA uses the vertices of a mesh to build an algebraic representation that compactly contains the relevant geometric information to compute various magnitudes: area, volume, center of mass, and inertia. This construction and the subsequent formulas are coordinate-free, equivariant under Euclidean transformations, and can be computed efficiently. The Euclidean split was our tool in developing these equations, and the resulting formulas took the form of 'norm of sums' or 'sum of norms' of the Euclidean or Ideal terms of the geometrical elements.

PGA is also very well suited to compute the physical motion of objects, and the computations of center of mass and inertia of a mesh-given object displayed in this paper tie in perfectly with the PGA-bivector-based treatment of Newtonian mechanics in [8]. That framework unifies the translational and rotational motion equations in a manner that makes them integrable, by actually purposely ignoring the Euclidean split. We recommend it as your next read.

In this paper, we have specialized to 3D PGA $\mathbb{R}_{3,0,1}$. The dimensional recursion we employed in the treatment of meshes can actually be generalized to arbitrary dimensions, and the formulas can be written in a manner that makes them dimension-agnostic (by using $\mathbf{e}_0^*$ rather than o for the origin, which is then automatically the d-dimensional Euclidean pseudoscalar). Coding in $\mathbb{R}_{d,0,1}$, we can thus use exactly the same 3D software to treat 2D polygon meshes and their properties, merely setting $d = 2$ in the first line of code. The algebra always generates the required entities allowed in the required dimension for each operation, no more and no less. No other framework can do this.

The present paper is a harbinger; you may have noticed glimpses of the connection to Stokes' theorem in this paper which need to be developed in more detail. We intend to produce a series of papers spelling out the advantages of PGA for Euclidean geometry and Newtonian physics, tying it to applications in CGI and robotics, with accompanying tutorials on bivector.net. We are convinced that PGA $\mathbb{R}_{d,0,1}$ is the preferred framework in those fields, encompassing all that went before, in any dimensionality, in a structural and efficient manner.

References

1. Clifford, W.K.: Problem in probability. Educational Times (January 1866), problem 1878, proposed by the author. Solution published November 1866. Reprinted in *Mathematical Questions with Solutions*, vol. VI, London, 1866, pp. 83–87, and in *Mathematical Papers*, London, 1882, pp. 601–607
2. Crane, K., de Goesa nd Mathieu Desbrun, F., Schröder, P.: Digital geometry processing with discrete exterior calculus. In: ACM SIGGRAPH 2013 courses. SIGGRAPH '13, ACM, New York, NY, USA (2013)
3. De Keninck, S.: Gamphetamine.js, https://github.com/enkimute/GAmphetamine.js
4. De Keninck, S.: ganja.js. Zenodo. https://doi.org/10.5281/ZENODO.3635774 (2020). https://doi.org/10.5281/ZENODO.3635774, https://zenodo.org/record/3635774
5. De Keninck, S., Roelfs, M.: Normalization, square roots, and the exponential and logarithmic maps in geometric algebras of less than 6D **47**(3) (2024). https://doi.org/10.1002/mma.8639
6. Doran, C., Lasenby, A.: Geometric Algebra for Physicists. Cambridge University Press, Cambridge (2003). https://doi.org/10.1017/CBO9780511807497
7. Dorst, L., De Keninck, S.: A guided tour to the plane-based geometric algebra pga. version 2.0 (2022). URL: bivector. net/PGA4CS. html
8. Dorst, L., De Keninck, S.: Physical geometry by Plane-Based Geometric Algebra. Springer (1 2024). https://doi.org/10.1007/978-3-031-55985-3_2, https://doi.org/10.1007/978-3-031-55985-3_2

9. Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science (2007)
10. Gunn, C.G., De Keninck, S.: Geometric algebra and computer graphics (2019). https://doi.org/10.1145/3305366.3328099
11. Hadfield, H., Wieser, E., Arsenovic, A., Kern, R., The Pygae Team: pygae/clifford. https://doi.org/10.5281/zenodo.1453978, https://doi.org/10.5281/zenodo.1453978
12. Munkres, J.R.: Elements of algebraic topology. Addison-Wesley (1984)
13. Roelfs, M.: The willing kingdon clifford algebra library (2025), https://arxiv.org/abs/2503.10451
14. Roelfs, M., De Keninck, S.: Graded symmetry groups: Plane and simple. Advances in Applied Clifford Algebras **33**(3), 30 (May 2023). https://doi.org/10.1007/s00006-023-01269-9
15. Ruhe, D., Gupta, J.K., De Keninck, S., Welling, M., Brandstetter, J.: Geometric clifford algebra networks. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 29306–29337. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/ruhe23a.html
16. Tonon, F.: Explicit Exact Formulas for the 3-D Tetrahedron Inertia Tensor in Terms of its Vertex Coordinates. Journal of Mathematics and Statistics **1**(1), 8–11 (1 2005). https://doi.org/10.3844/jmssp.2005.8.11, https://doi.org/10.3844/jmssp.2005.8.11
17. Zhdanov, M.: Flash clifford: Hardware-efficient implementation of clifford algebra neural networks (2025), https://github.com/maxxxzdn/flash-clifford