

A summary of Projective Geometric Algebra

PATROLIN

September 29, 2022

Abstract

This is an explanation of how to calculate PGA, but not necessarily why it works - for that you can try [sudgylacmoe](#) on youtube. I would also point you to [bivector.net](#), but their dual calculations are whack, the [ganja.js](#) ones are presumably correct, since their demos work.

I. GEOMETRIC NUMBERS

$$1 + x^2 = 0 \\ x = ?$$

x is not a real number; but if it's not real, why should the other numbers be real?

$$(e_1)^2 + (e_2)^2 = (e_0)^2 \\ (e_1)^2 = 1; (e_2)^2 = -1; (e_0)^2 = 0$$

In fact, we can define as many of these as we want, the simplest examples being:

$$a + be_1 \text{ // hyperbolic numbers} \\ a + be_2 \text{ // complex numbers} \\ a + be_0 \text{ // dual numbers}$$

We can multiply these numbers together using the geometric product:

$$e_i e_i = \{1, -1, 0\} \\ e_i e_j = -e_j e_i$$

This product is neither commutative nor anticommutative, but it is distributive and associative:

$$AB \neq BA \\ AB \neq -BA \\ A(B + C) = AB + AC \\ (AB)C = A(BC) \\ aB = Ba; a \in \mathbb{R}$$

Thus the product of two complex numbers:

$$(A_1 + A_2 e_2)(B_1 + B_2 e_2) \\ = A_1 B_1 + A_1 B_2 e_2 + A_2 e_2 B_1 + A_2 e_2 B_2 e_2 \\ = A_1 B_1 + A_1 B_2 e_2 + A_2 B_1 e_2 + A_2 B_2 \\ = (A_1 B_1 + A_2 B_2) + (A_1 B_2 + A_2 B_1) e_2$$

II. ROTATIONS

A multivector with n basis vectors consists of 2^n blades:

- scalar = 0-vector = 1
- vector = 1-vector
- bivector = 2-vector
- trivector = 3-vector
- ...
- (n-1)-vector = pseudovector
- n-vector = pseudoscalar = 1

Where a k-vector has $\binom{n}{k}$ blades, for example:

$$A = A_1 \\ + A_2 e_0 + A_3 e_1 + A_4 e_2 \\ + A_5 e_{01} + A_6 e_{02} + A_7 e_{12} \\ + A_8 e_{012}$$

We can abbreviate blades like $e_1 e_2$ as e_{12} .

Multiplying two multivectors gives you another multivector, we can use the taylor series expansion of the exponential function to find a

rotation e^A :

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

$$e^{ae_2} = 1 + ae_2 - \frac{a^2}{2} - \frac{a^3}{3}e_2 + \dots$$

$$= (1 - \frac{a^2}{2} + \dots) + (a - \frac{a^3}{3} + \dots)e_2$$

$$= \cos(a) + \sin(a)e_2$$

Similarly you can find

$$e^{e_i} = \begin{cases} \cos(a) + \sin(a)e_i & ((e_i)^2 = -1) \\ \cosh(a) + \sinh(a)e_i & ((e_i)^2 = 1) \\ 1 + e_i & ((e_i)^2 = 0) \end{cases}$$

This gives us rotations, hyperbolic rotations and translations (rotations through infinity) respectively.

Then for a multivector we would have:

$$e^A = e^{A_1} e^{A_2 e_0} e^{A_3 e_1} \dots e^{A_n e_I}$$

III. UNARY OPERATORS

For the i -th blade in a multivector

$$X_i \in \text{k-vector}$$

we can define some operations, like reversing the order of basis vectors in the blade, that amount to flipping some signs:

$$\tilde{X}_i = (-1)^{\lfloor k/2 \rfloor} X_i \text{ // reverse}$$

$$X_i^\dagger = (-1)^{\lfloor k \rfloor} X_i \text{ // involute}$$

$$\bar{X}_i = (-1)^{\lfloor k+k/2 \rfloor} X_i \text{ // conjugate}$$

$$f(A) = \sum_i f(X_i)$$

Poincaré duality states that maps between k -vectors and $(n-k)$ -vectors exist.

$$X_i \text{ dual}(X_i) = \pm 1$$

$$\text{dual}(X_i) = \pm X_{2^n - i + 1}$$

$$\text{dual}(A) = \sum_i \text{dual}(X_i)$$

For example:

$$\underline{X}_i X_i = 1 \text{ // left complement}$$

$$X_i \bar{X}_i = 1 \text{ // right complement}$$

$$X_i X_i^* = \text{sign}(X_i^{ND} \widetilde{X_i^{ND}}) 1 \text{ // hodge dual}$$

Where X_i^{ND} is X_i without degenerate basis vectors, e.g.

$$X_i = e_{012}; X_i^{ND} = e_{12}$$

Let $\mathbb{G}_{a,b,c}$ be a geometric algebra with a positive, b negative and c zero basis vectors.

Then for $\mathbb{G}_{a,0,c}$:

$$\bar{X}_i = X_i^*$$

And if all that wasn't confusing enough, applying a dual twice changes the signs, so we also want the inverses of these duals:

$$(X_i^*)^{*-1} = X_i$$

$$\bar{\bar{X}}_i = X_i$$

$$\underline{\underline{X}}_i = X_i$$

IV. SHAPES AND SIZES

Let $\mathbb{G}_{d+1,0,1}$ be a d -dimensional PGA with $(e_0)^2 = 0$ and $(e_i)^2 = 1$.

It turns out, in 2D and 3D, we can simplify implementations by swapping two basis vectors in some blades such that \bar{X}_i does not flip signs, e.g.

$$A = A_1 + A_2 e_0 + A_3 e_1 + A_4 e_2$$

$$+ A_5 e_{01} + A_6 e_{20} + A_7 e_{12} + A_8 e_{012}$$

$$\bar{A} = A_8 + A_7 e_0 + A_6 e_1 + A_5 e_2$$

$$+ A_4 e_{01} + A_3 e_{20} + A_2 e_{12} + A_1 e_{012}$$

Points in PGA are represented by $(n-1)$ -vectors:

$$\overline{e_0 + x e_1 + y e_2 + \dots}$$

This is a kind of a lie, you can make an equivalent dual algebra by letting points be vectors, in fact many operations make use of computing in this dual algebra via $\bar{\bar{A}} \text{ op } \bar{\bar{B}}$

Let $\langle A \rangle_k$ be the grade selection operator:

$$\langle A \rangle_k = \sum_i \langle X_i \rangle_k$$

$$\langle X_i \rangle_k = \begin{cases} X_i & (X_i \in \text{k-vector}) \\ 0 & (X_i \notin \text{k-vector}) \end{cases}$$

Then we can start defining binary operators:

$$A \cdot B = \sum_{j,k} \langle \langle A \rangle_j \langle B \rangle_k \rangle_0 \quad // \text{ dot product}$$

$$A \wedge B = \sum_{j,k} \langle \langle A \rangle_j \langle B \rangle_k \rangle_{j+k} \quad // \text{ wedge product}$$

$$A \vee B = \overline{\overline{A}} \wedge \overline{\overline{B}} \quad // \text{ antiwedge product}$$

In other words the wedge product is equal to the geometric product if the grades add together:
 $j\text{-vector} \wedge k\text{-vector} = (j+k)\text{-vector}$, otherwise it's 0

All of these operators retain distributivity and associativity

The antiwedge product allows you to join points into lines, planes, ...

$$line = point_1 \vee point_2$$

$$plane = point_1 \vee point_2 \vee point_3$$

In fact this works with any two geometric objects, operators that don't have this property aren't really worth your time.

And the wedge product allows you to meet two objects:

$$point = line_1 \wedge line_2$$

$$line = plane_1 \wedge plane_2$$

If you meet two parallel lines, you get an infinite point = a point at infinity:

$$line_1 \wedge line_2 = \overline{xe_1 + ye_2 + \dots}$$

All objects in GA have an orientation, so if you flip the direction of one of the lines, you would get an infinite point in the other direction

TODO: something about sizes of line segments / volumes

TODO: projection to camera plane + depth buffer

V. MOTORS

TODO: something about bivector blades being rotations

$$rotor = e^{bivector^{ND}}$$

$$motor = e^{bivector}$$

In 3D, rotors are quaternions.

TODO: applying motors and line forces