Rigid Body Motions in Geometric Algebra

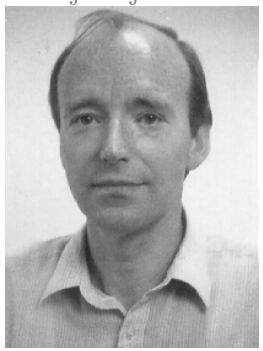
Leo Dorst (L.Dorst@uva.nl)
Intelligent Systems Lab, Informatics Institute,
University of Amsterdam, The Netherlands

Henk Pijls' farewell, June 20, 2008

1 How do I know Henk?

- Teaching Linear Algebra for bachelor Artificial Intelligence
- Common interest in algebraization of geometry.
- Specific Geometric algebra projects:

Henkje Pijls



  - Ph.D. co-advisor on Tim Bouma (2000-∞), AIO sponsored by IvI/KdV.
    Geometric Modeling and the Fundamental Theorem of Projection Operators (working title).
  - Wout Hammerstein (M.Sc. Mathematics 2006)
    Cliffords Geometric Algebra: Clifford-algebraic tools for differential geometry and physics.
  - Anton Zeef (Ba β/γ 2006)
    Conforme afbeeldingen & Cliffordalgebras
  - Sebastiaan Eliëns (M.Sc. Mathematics 2008?) *Subject TBA*

- Few mathematicians were prepared to look beyond the uncoventional formulation of Hestenes' geometric algebra. Henk did, and wanted to straighten things out.

Henk is known as "*Leo's Mathematician*" in the GA community; few of us have been able to interest any. Yet we have many non-mathematically phrased essentially mathematical questions. The above begins to provide an embedding.

2 Still a Need for a Good Representation of Rigid Body Motions?

- *Classical physics:* explicit classical parametrizations often good enough; very 3D, specification not very computational (too much natural language).
- *Satellite navigation:* quaternions most efficient for 3D rotations.
- *Robotics:* homogeneous coordinates for kinematic transformations, specific screw representation efficient for dynamics computations.
- *Computer Graphics:* homogeneous coordinates used generatively on kinematics of huge collections of points. Interpolation of motions not trivial.
- *Computer Vision:* analysis of image sequences. Homogeneous coordinates for projective geometry, but not sufficiently 'metrical'. Data noise requires estimation, tracking requires extrapolation.

State of the art in Computer Science:

> Homogeneous coordinates for point transformations are standard, all else (Grassmann-Cayley, Plücker coordinates, quaternions) are more specialist and exotic.

Focus on higher-level algorithms, not low-level representations and data structures. But lowest level is *not structure-preserving*, therefore intermediate level is *ad hoc*, and coordinate-based.

3 Structural Aspects of a Language for Geometry

- *primitives*: points, lines, planes, circles, spheres, tangents
- *constructions*: connections, intersections, orthogonal complement, duality
- *motions*: translations, rotations, reflection, projection
- *properties*: size, location, orientation, cross ratio
- *numerics*: approximation, estimation, linearization

Motions are at the basis of geometry (Klein), structure preservation:

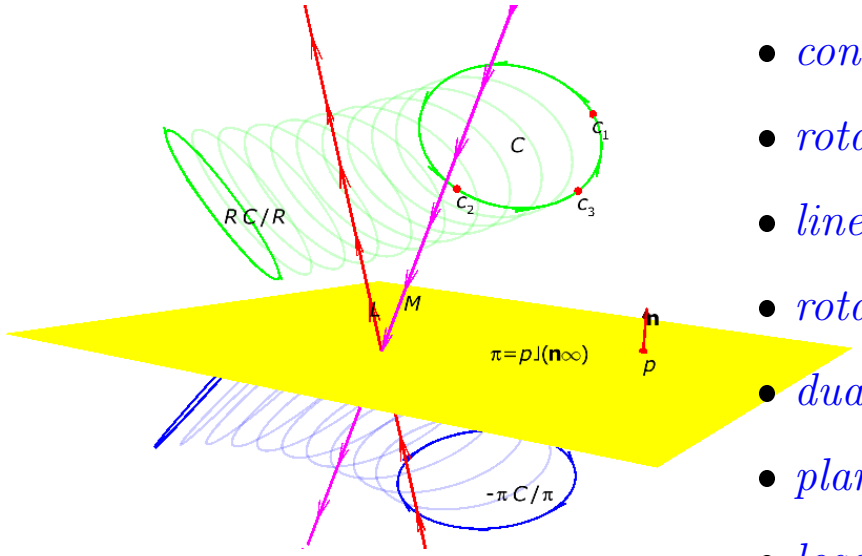
Constructions and properties of primitives should be covariant under motions.

Example: covariant intersection.

$$\begin{array}{ccc} \text{line } \Lambda, \text{ plane } \Pi & \xrightarrow{\text{intersect}} & \text{point } \Lambda \cap \Pi \\ \Big\downarrow{\scriptstyle \text{motion}} & & \Big\downarrow{\scriptstyle \text{motion}} \\ \text{line } \Lambda', \text{ plane } \Pi' & \xrightarrow{\text{intersect}} & \Lambda' \cap \Pi' = (\Lambda \cap \Pi)' \end{array}$$

Not automatic in standard linear algebra!

4 An Example: Reflection of a Rotating Circle



FIG(1,1)

- *construction of a circle*: $C = c_1 \wedge c_2 \wedge c_3$
- *rotation*: $C \mapsto RC/R$
- *line representation*: $L = a_1 \wedge a_2 \wedge \infty = a_1 \wedge \mathbf{u} \wedge \infty$
- *rotation around line*: $R = \exp(\phi\, L^*/2)$
- *dual plane representation*: $\pi = p\rfloor(\mathbf{n}\infty)$
- *plane reflection*: $X \mapsto -\pi\, X/\pi$
- *logarithms of motions*: $R^{1/n} = \exp(\log(R)/n)$

Note that all is specified directly in terms of the geometric elements, and some algebraic operations. No coordinates at all in the language (just in the data).

5 Gaigen2 Implementation Matches the Algebra

```
// l1, l2, c1, c2, c3, p1 are points, n a normal vector
line L; circle C; dualPlane p; vector n;

L = unit_r(l1 ^ l2 ^ ni);
C = c1 ^ c2 ^ c3;
p = p1 << (n^ni);

draw(L); draw(C); draw(p);

draw( - p * L * inverse(p) );                    // draw reflected line (magenta)
draw( - p * C * inverse(p) );                    // draw reflected circle (blue)

// compute rotation versor:
const float phi = (float)(M_PI / 2.0);
TRversor R;
R = exp(0.5f * phi * dual(L));

draw(R * C * inverse(R));                        // draw rotated cicle (green)
draw(-p * R * C * inverse(R) * inverse(p));      // draw reflected, rotated circle (blue)

// draw interpolated circles
pointPair LR = log(R);                           // get log of R
for (float alpha = 0; alpha < 1.0; alpha += 0.1f)
{
  TRversor iR;
  iR = exp(alpha * LR);                          // compute interpolated rotor

  draw(iR * C * inverse(iR));                    // draw rotated circle (light green)
  draw(-p * iR * C * inverse(iR) * inverse(p)); // draw reflected, rotated circle (light blue)
}
```

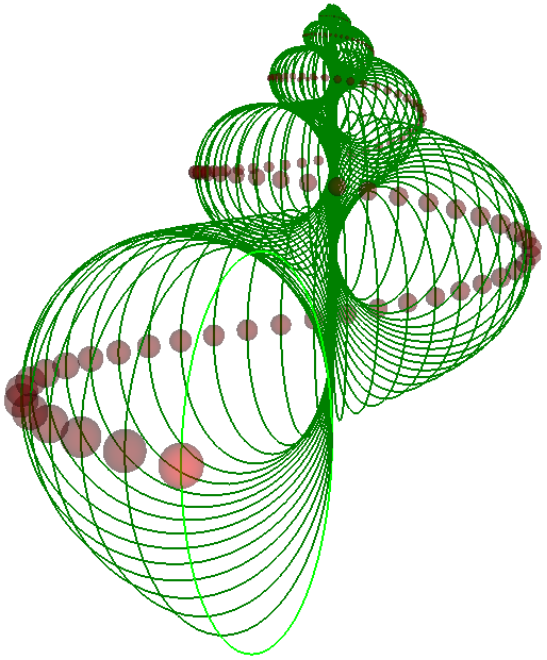
6 By Contrast, the Example in Linear Algebra

- *construction of a circle*: none, treat the points separately.
- *rotation*: by 4×4 *homogeneous coordinate matrix* $\begin{bmatrix}\mathsf{R} & (\mathsf{I}-\mathsf{R})\mathbf{t}\\ 0^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix}$ acting on points $(\mathbf{x},1)^T$.
- *line representation*:
  - as (position vector, direction vector)-pair $(\mathbf{p},\mathbf{u})$; each component moves differently.
  - as the kernel of two homogeneous plane equations: $[\![\ \pi_1,\pi_2\ ]\!]^T$
  - using 6D Plücker coordinates: $\{\mathbf{u},\mathbf{p}\times\mathbf{u}\}$.
- *rotation around line*: $[\![\mathsf{R}]\!]=\mathbf{u}\mathbf{u}^T+\cos\phi\,([\![1]\!]-\mathbf{u}\mathbf{u}^T)+\sin\phi\,[\![\mathbf{u}^\times]\!]$, then move into place.
- *dual plane representation*: $\pi=[\mathbf{n},-\mathbf{p}\cdot\mathbf{n}]$
- *plane reflection*: Use point reflection $[\![\mathsf{P}]\!]=\begin{bmatrix}\mathsf{I}-2\mathbf{n}\mathbf{n}^T & 2\delta\mathbf{n}\\ 0^T & 1\end{bmatrix}$. On planes as $[\![\mathsf{P}]\!]^{-T}$, on Plücker lines as more involved 6×6 matrix.
- *interpolation of general rotation*: non-elementary (but can be done by specialized logarithm of matrix).

Linear algebra code typically consists of such coordinate tricks, applied to the points.
No direct circle rotation, or line reflection, or rotation generation.

7 Conformal Compactification Combined with Clifford Algebra

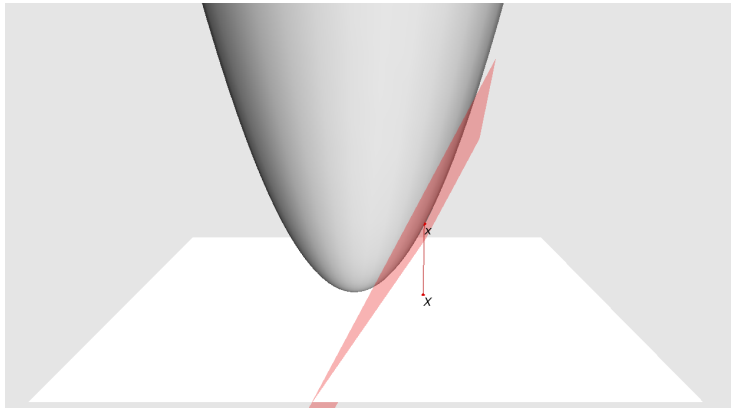
Consider Euclidean geometry not as a specific projective geometry, but as conformal geometry. Embed $\mathbb{R}^n$ isometrically into $\mathbb{R}^{n+1,1}$. Then we get a unification of techniques:



FIG(16,3)

- Conformal transformations of $\mathbb{R}^n$ are represented as orthogonal transformations of $\mathbb{R}^{n+1,1}$.
- We will represent orthogonal transformations as multiple reflections, following Cartan-Dieudonné.
- We represent multiple reflections using the geometric product of Clifford algebra as spinors, which preserve structure.
- We automatically get a non-metric exterior product to have Grassmannians represent geometric primitives (lines, spheres, tangent vectors etc). This gives structure.
- We use its Grassmann-Cayley algebra to do intersections (though more quantitatively, with better metric products).
- Spinors as exponentials of bivectors give the Lie algebra. Logarithms then permit interpolation.

8 Euclidean Point Representation



FIG(14,3): point

FIG(14,4): circle

FIG(14,6): circle meet

Represent point with 3D Euclidean position vector $\mathbf{x}$ in *5D Minkowski space* as a ray vector:

$$\boxed{x \sim o + \mathbf{x} + \tfrac{1}{2}\|\mathbf{x}\|^2 \infty}$$

where o is the standard point at the origin, $\mathbf{x}$ the Euclidean 'position vector', ∞ point at infinity.
Minkowski space: $o \cdot o = 0$, $\infty \cdot \infty = 0$, $o \cdot \infty = -1$.

Basically, like *two extra homogeneous coordinates*:

$$x \sim (1, \mathbf{x}, \tfrac{1}{2}\|\mathbf{x}\|^2)^T$$

on the 5D basis $\{o, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \infty\}$.

9 Inner Product Represents Squared Euclidean Distance

Metric of the representation space $\mathbb{R}^{n+1,1}$ is Minkowski. In preferred basis:

$\cdot$	o	$\mathbf{x}$	∞
o	0	0	-1
$\mathbf{x}$	0	$\|\mathbf{x}\|^2$	0
∞	-1	0	0

Now look what happens between two unit-weight points:

$$\begin{aligned} x \cdot y &= (o + \mathbf{x} + \tfrac{1}{2}\|\mathbf{x}\|^2 \infty) \cdot (o + \mathbf{y} + \tfrac{1}{2}\|\mathbf{y}\|^2 \infty) \\ &= (0 + 0 - \tfrac{1}{2}\|\mathbf{y}\|^2) + (0 + \mathbf{x} \cdot \mathbf{y} + 0) + (-\tfrac{1}{2}\|\mathbf{x}\|^2 + 0 + 0) \\ &= -\tfrac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2 \end{aligned}$$

Weird metric, nice trick: linearization of a squared distance.

The inner product in the representation space gives the squared Euclidean distance!

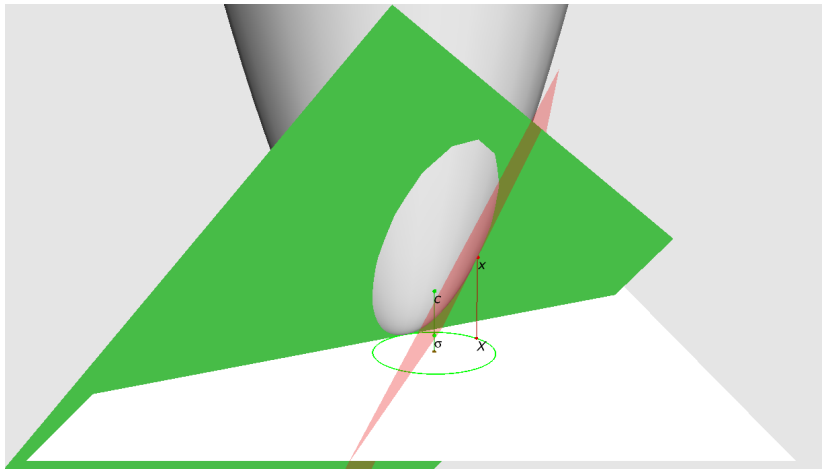
Therefore, Euclidean motions are represented by orthogonal transformations.

10 Vectors in $\mathbb{R}^{n+1,1}$ Represent Spheres and Planes in $\mathbb{R}^n$

- *general vectors of* $\mathbb{R}^{n+1,1}$ *are oriented, weighted (dual) spheres in* $\mathbb{R}^n$:

$$d_E^2(X,C) = \rho^2 \quad \Leftrightarrow \quad x \cdot c = -\tfrac{1}{2}\rho^2 \quad \Leftrightarrow \quad x \cdot (c - \tfrac{1}{2}\rho^2 \infty) = 0$$

  Points are merely (dual) speres of zero radius.



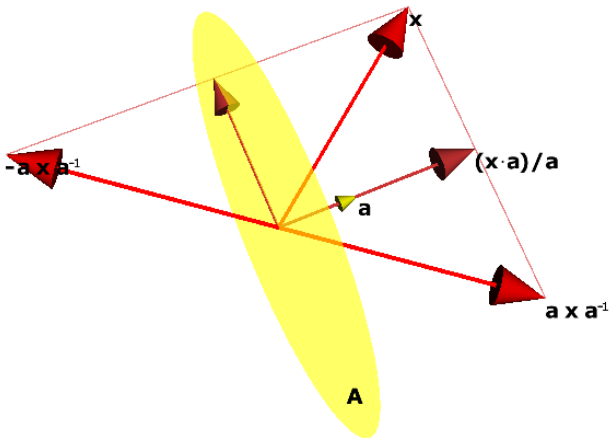
FIG(14,4)

- *oriented, weighted (dual) planes of* $\mathbb{R}^n$ *are vectors in* $\mathbb{R}^{n+1,1}$ without o-component:

$$d_E^2(X,A) = d_E^2(X,B) \quad \Leftrightarrow \quad x \cdot a = x \cdot b \quad \Leftrightarrow \quad x \cdot (a - b) = 0$$

  Note that the o-component satsifies $\infty \cdot (a - b) = 0$. (Geometrically, the point at infinity is on all planes.) General dual plane is of the form $\mathbf{n} + \delta\,\infty$, with $\mathbf{n} \in \mathbb{R}^n$.

11 Geometric Reflections as Algebraic Sandwiching



FIG(7,1)

Reflection in an origin plane with unit normal $\mathbf{a}$

$$\mathbf{x} \;\mapsto\; \mathbf{x} - 2(\mathbf{x}\cdot\mathbf{a})\,\mathbf{a}/\|\mathbf{a}\|^2 \quad \textit{(classic LA)}$$

Now consider the dot product as the symmetric part of a more fundamental geometric product:

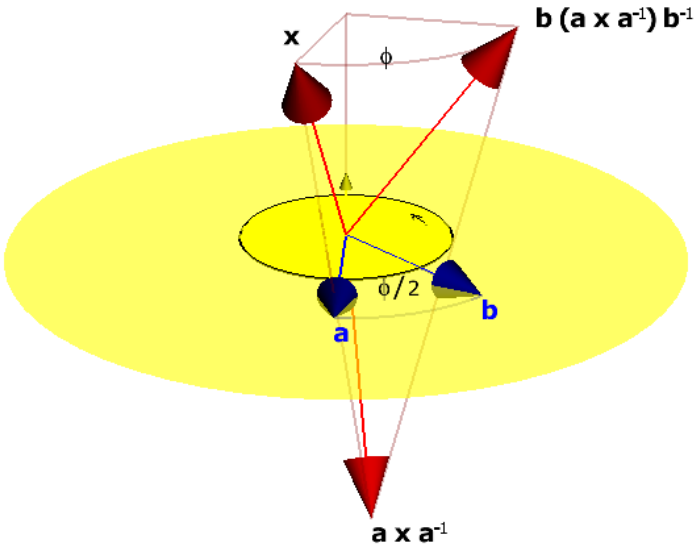
$$\mathbf{x}\cdot\mathbf{a} = \tfrac{1}{2}(\mathbf{x}\,\mathbf{a} + \mathbf{a}\,\mathbf{x})$$

Then rewrite:

$$\begin{aligned}\mathbf{x} \;&\mapsto\; \mathbf{x} - (\mathbf{x}\,\mathbf{a} + \mathbf{a}\,\mathbf{x})\,\mathbf{a}/\|\mathbf{a}\|^2 \quad \textit{(GA product)}\\ &=\; -\mathbf{a}\,\mathbf{x}\,\mathbf{a}^{-1}.\end{aligned}$$

with *the geometric inverse of a vector*: $\mathbf{a}^{-1} = \mathbf{a}/\|\mathbf{a}\|^2$.

12 Orthogonal Transformations as Versors



FIG(7,2)

A reflection in two successive origin planes:

$$\begin{aligned} \mathbf{x} &\mapsto -\mathbf{b}\,(-\mathbf{a}\,\mathbf{x}\,\mathbf{a}^{-1})\,\mathbf{b}^{-1} \\ &= (\mathbf{b}\,\mathbf{a})\,\mathbf{x}\,(\mathbf{b}\,\mathbf{a})^{-1} \end{aligned}$$

So a rotation is represented by the geometric product of two vectors. (Actually, in 3D these are quaternions.)

Multiple reflections are the fundamental representation:

The geometric product of vectors is a versor.
It acts as an orthogonal transformation.
(And it generates the Lipschitz spinor group.)

As we will see, versors are structure-preserving.

13 The Fundamental Geometric Product

Take a real metric vector space $\mathbb{R}^n$ with inner product $\cdot : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$.

Define the *geometric product* through:

- *associative*: $X\,(Y\,Z) = (X\,Y)\,Z$.
- *linear*: $X\,(\alpha Y + \beta Z) = \alpha\,X\,Y + \beta\,X\,Z$.
- *vectors have scalar square*: $\mathbf{x}\,\mathbf{x} = \mathbf{x}\cdot\mathbf{x}$.

That's all, mathematically. Not necessarily commutative.

In coordinate form, for $\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$, and $\mathbf{y} = y_1\mathbf{e}_1 + y_2\mathbf{e}_2 + y_3\mathbf{e}_3$:

$$\begin{aligned}\mathbf{x}\,\mathbf{y} &= x_1y_1 + x_2y_2 + x_3y_3 + (x_2y_3 - y_2x_3)\,\mathbf{e}_2\,\mathbf{e}_3 + (x_3y_1 - y_3x_1)\,\mathbf{e}_3\,\mathbf{e}_1 + (x_1y_2 - y_1x_2)\,\mathbf{e}_1\,\mathbf{e}_2\\ &= \mathbf{x}\cdot\mathbf{y} + (\mathbf{x}\times\mathbf{y})/\mathbf{I}_3.\end{aligned}$$

In 3D, recognizable classical products. But mixed grades!

Starting from vectors in $\mathbb{R}^n$, one generates the geometric algebra of a 2^n-dimensional dimensional multivector space, sometimes denoted $\mathbb{R}_n$.

Alternatively, define the Clifford algebra as the quotient of a tensor algebra on the metric space (V, Q) with the two-sided ideal of $\mathbf{x}\otimes\mathbf{x} - Q(\mathbf{x},\mathbf{x})$. Or talk to Henk.

14 Spanning subspaces

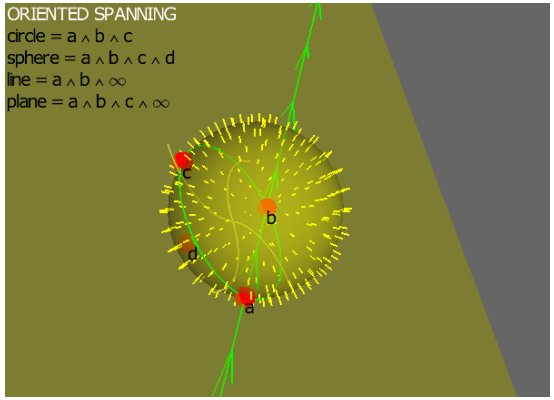
The skew-symmetric part of the geometric product gives the exterior product of Grassmann algebra:

$$\mathbf{x} \wedge \mathbf{a} = \tfrac{1}{2}(\mathbf{x}\,\mathbf{a} - \mathbf{a}\,\mathbf{x})$$

It is bilinear and associative. Use it to span oriented subspaces as Grassmannians:

$$\mathbf{x} \wedge (\mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_k) = 0 \iff \mathbf{x} \text{ in } \operatorname{span}(\mathbf{a}_1, \cdots, \mathbf{a}_k)$$

The elements of Euclidean geometry are all represented as $\wedge$-factorizable multivectors ('blades'):



DEMO by hand

- *Rounds: spheres, circles, point pairs*
  $a \wedge b \wedge c \wedge d$ is the (oriented & weighted) sphere through 4 points, etc.
- *Flats: planes, lines, flat points*
  $a \wedge b \wedge c \wedge \infty$ is (oriented & weighted) plane through 3 points, etc.
- *k-dimensional direction elements*
  $\mathbf{B} \wedge \infty$ is a pure Euclidean k-space $\mathbf{B}$, at infinity.
- *k-dimensional tangents*
  $p \rfloor (p \wedge \mathbf{B} \wedge \infty)$ is the tangent k-space $\mathbf{B}$ at p.

15 Contraction, Duality, Meet and Join, Projection

Introduce a scalar product $*$ as the scalar part of the geometric product, and define the (left) contraction via adjoint relationship to outer product:

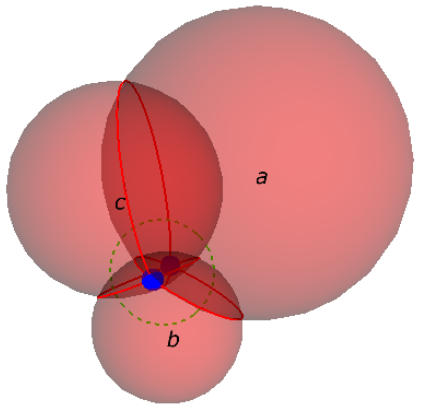
$$X * (A \rfloor B) = (X \wedge A) * B$$

Contraction with the unit element I_n^{-1} of grade n in V^n gives duality. (Actually the Hodge dual.)

In metric spaces, the algebra of $\wedge$ and $\rfloor$ is more convenient than the Grassmann-Cayley algebra of meet and join (which it includes). $A \cap B = B^* \rfloor A$.

Extended algebraic vocabulary very handy for explicit specification of geometric elements:

- dual sphere with center c, point p on it: $\sigma = p \rfloor (c \wedge \infty)$
- dual plane with normal $\mathbf{n}$, point p on it: $\pi = p \rfloor (\mathbf{n} \wedge \infty)$
- dual intersection of two dual spheres: $\kappa = \sigma_1 \wedge \sigma_2$
- circle orthogonal through 3 spheres: $C = \sigma_1 \wedge \sigma_2 \wedge \sigma_3$
- tangent vector with direction $\mathbf{u}$ at p: $T = p \rfloor (p \wedge \mathbf{u} \wedge \infty)$
- general projection of X onto A: $(X \rfloor A)/A$.



FIG(15,1)

16 Structure Preservation

Reflection of vector $\mathbf{x}$ in a plane with normal $\mathbf{a}$ is:

$$\mathbf{x} \quad \mapsto \quad -\mathbf{a}\,\mathbf{x}\,\mathbf{a}^{-1}$$

If X is a product of vectors $\mathbf{x}_i$, this extends to:

$$X \quad \mapsto \quad \mathbf{a}\,\widehat{X}\,\mathbf{a}^{-1}$$

with $\widehat{\mathbf{x}} = (-1)^{\dim(\mathbf{x})}\,\mathbf{X}$, the main involution.

This then distributes over the 'constructive' products $\wedge$ and $\rfloor$ which are effectively weighted sums of geometric products:

The sandwiching product is structure preserving.

So now all motions are universally applicable to all geometric elements using their versors:

$$\boxed{\begin{array}{rl} V\text{even}: & X \quad \mapsto V\,X\,V^{-1} \\ V\text{odd}: & X \quad \mapsto V\,\widehat{X}\,V^{-1} \end{array}}$$

This universality is very unlike the homogeneous coordinate approach, and enormously simplifies software.

17 Perturbations as Bivectors; Lie Algebra

Versors were introduced through products of invertible vectors. An even versor V can be written as the *exponential of a bivector* B:

$$V = e^B.$$

Bivectors are linear and can be averaged, interpolated, etc. Conversely:

$$B = \log(V)$$

Example: rotation in plane through the origin as exponential:

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

with the plane satisfying $\mathbf{I}^2 = -1$. (These are 'complex numbers' and 'quaternions', but fully real.)
Better still, rotation around a general line L is $e^{L^*\phi/2}$.

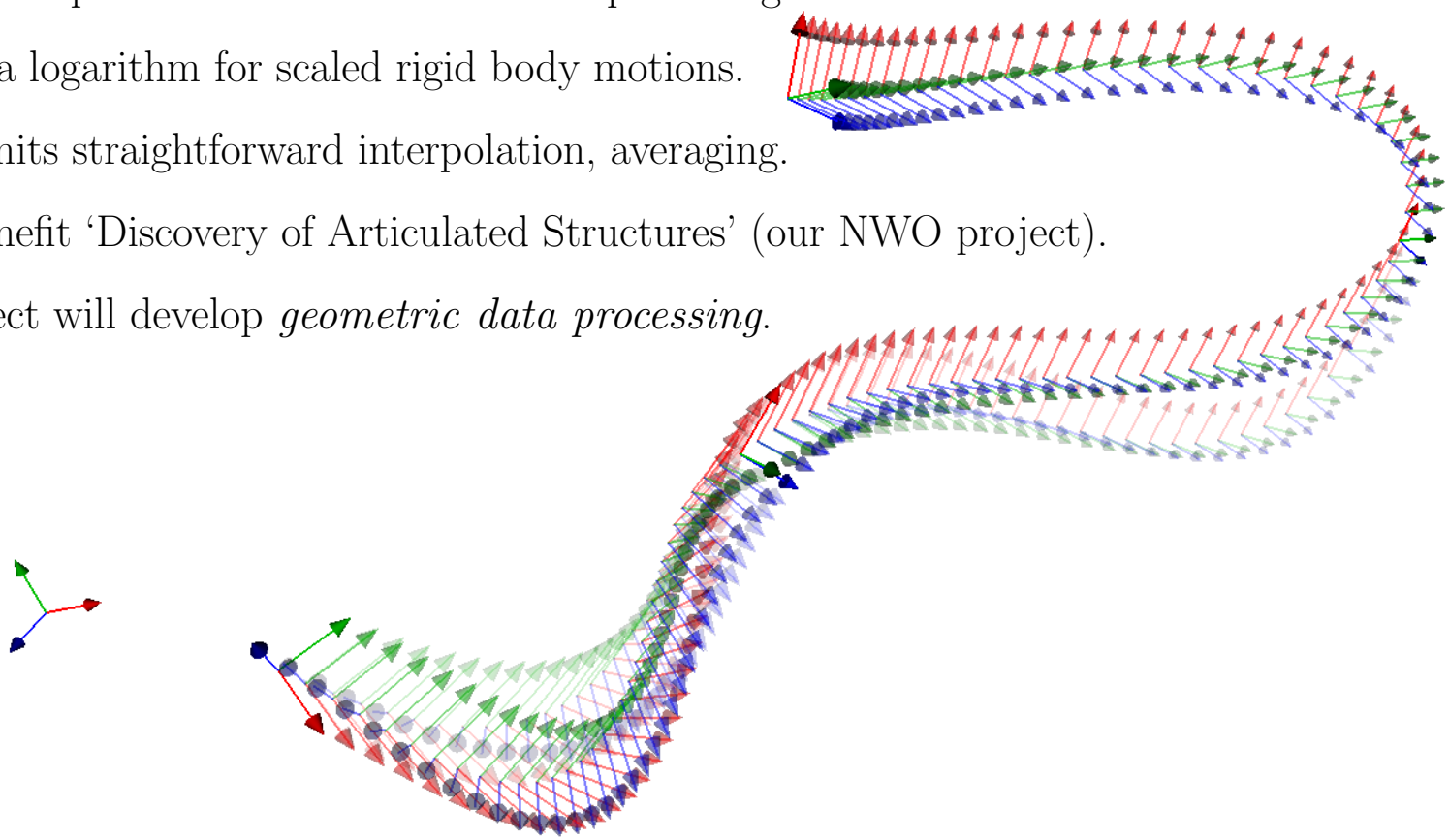
We get a *geometric calculus* using these principles. Perturbations are simple, and linear in B:

$$e^{-B/2}\,X\,e^{B/2} \approx X + \tfrac{1}{2}(X\,B - B\,X) \equiv X + X \times B$$

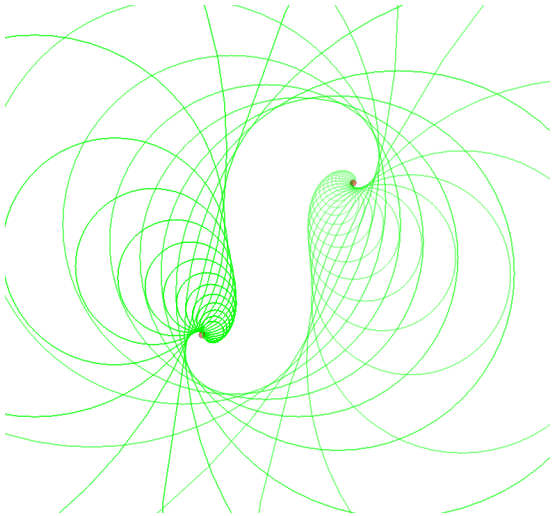
Gives structure-preserving differential geometry. First order treatment of second-order motions! Exact linearity, so apply linear data processing methods with greatly extended functionality!

18 Rigid Body Motion Processing

- The bivector parametrization of motions is promising.
- We have a logarithm for scaled rigid body motions.
- This permits straightforward interpolation, averaging.
- It will benefit 'Discovery of Articulated Structures' (our NWO project).
- The project will develop *geometric data processing*.



19 Bonus: n-D Conformal Geometry



FIG(16,6) loxodrome FIG(16,12) dupin

Reflections in planes give the Euclidean transformations.
Represented as versors by products of vectors $\mathbf{n} - \delta\infty$ (dual planes).

Reflections in spheres give the full conformal group.
Prototypical vector versor $o - \infty/2$ (dual unit sphere at origin).

This gives uniform scaling and transversion.

But we are still looking for the logarithm of a general versor in the conformal model...

motion	versor	exp form	Möbius matrix
translation over $\mathbf{t}$	$1 - \mathbf{t}\,\infty/2$	$\exp(-\mathbf{t} \wedge \infty/2)$	$\begin{bmatrix} 1 & \mathbf{t} \\ 0 & 1 \end{bmatrix}$
rotation over $\mathbf{I}\phi$	$\cos(\phi/2) - \mathbf{I}\,\sin(\phi/2)$	$\exp(-\mathbf{I}\,\phi/2)$	$\begin{bmatrix} e^{-\mathbf{I}\phi/2} & 0 \\ 0 & e^{-\mathbf{I}\phi/2} \end{bmatrix}$
scaling by e^{γ}	$\cosh(\gamma/2) + o \wedge \infty\,\sinh(\gamma/2)$	$\exp(o \wedge \infty\,\gamma/2)$	$\begin{bmatrix} e^{\gamma/2} & 0 \\ 0 & e^{-\gamma/2} \end{bmatrix}$
transversion over $\mathbf{v}$	$1 + o\,\mathbf{v}$	$\exp(o \wedge \mathbf{v})$	$\begin{bmatrix} 1 & 0 \\ \mathbf{v} & 1 \end{bmatrix}$

20 Linear Algebra is Not Good Enough for Geometry

- *primitives*: only vectors and covectors (hyperplanes)
- *constructions*: hardly any at algebraic level, some as matrix manipulation
- *motions*: linear transformations too general; orthogonal transformations too cumbersome
- *properties*: involved non-linear parametrizations of geometric transformations
- *numerics*: strongly developed techniques for linearized estimation

The lack of algebraic constructions leads to the bad habit of *specification by coordinates*. The limited number of primitives and corresponding data structures, combined with lack of covariance, then produces *confusion and errors*.

Linear algebra is the assembly language of geometry.

Structure preservation needs to be carefully and explicitly designed and enforced.

21 Geometric Algebra Is Tailored To Geometry

- *primitives*: general subspaces (which model points, lines, planes, spheres, tangents, etc.)
- *constructions*: algebraic spanning, intersection, orthogonality, duality
- *motions*: motions are automatically structure preserving
- *properties*: parametrization immediately in terms of geometric primitives
- *numerics*: geometric differentiation, more extended linearization, estimation (but immature)

Now everything can be specified using the geometry directly:

Geometric algebra is the high-level language of geometry.

Structure preservation is automatic.

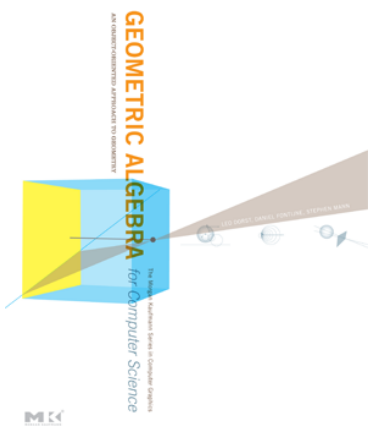
22 Implementation: Size Matters, But Is Not Prohibitive

- GA can represent all 2^m subspaces of an m-D vector space as elements of computation.
- To represent Euclidean motions in $\mathbb{R}^n$ as versors, you need GA of $\mathbb{R}^{n+1,1}$-D space.
- That is a 32-dimensional representation for 3D Euclidean gometry!
- Efficient implementation is therefore an issue.
- Solved by using the strucure of GA in an automatic code generator.
  (Fontijne 2007 PhD: *Efficient Implementation of Geometric Algebra*, NWO/UvA/VU)
- Result: high-level programming in GA available (subspace products, sandwiching).
- The actual algebraic computation takes care of the type administration; the implementation performs this at compile time. The program does hardly more than linear algebra (but only at the assembly level).

Executive summary:

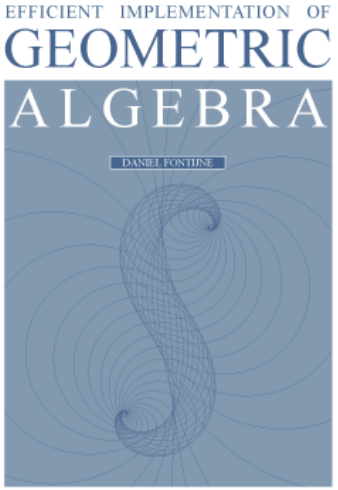
> *Your people can now use a high-level language (GA) to specify Euclidean geometry, at code generation level, for competitive efficiency with classical approach, but with more maintainable and less error-prone code.*

23 Perhaps Inspirational (Though Not Written for Mathematicians)



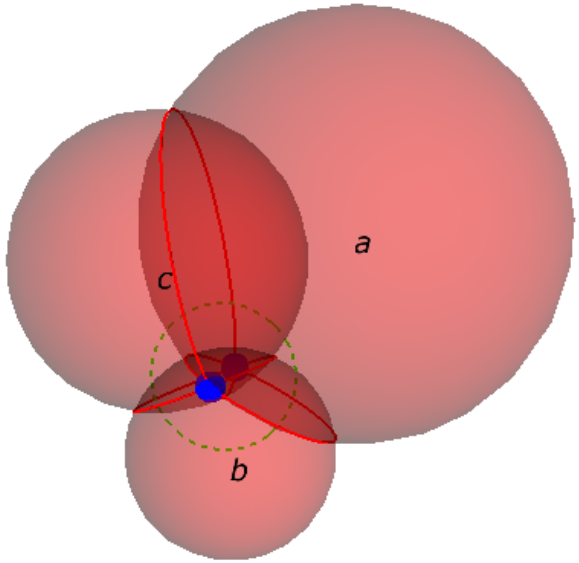
Geometric Algebra for Computer Science
An Object-Oriented Approach to Geometry
Leo Dorst, Daniel Fontijne, Stephen Mann
(Morgan-Kaufmann Publishers 2007, ISBN 0-12-369465-5)

- 22 chapters, 4 appendices, 650 pages, 150+ full color figures, free software.
- Available everywhere, price € 45-90.
- Book website, freely downloadable software and demos:
  `www.geometricalgebra.net`



Efficient Implementation of Geometric Algebra
Daniel Fontijne
Ph.D. thesis UvA, 2007, ISBN-13: 978-90-889-10-142, available at
`www.science.uva.nl/~fontijne/phd.html`

24 Euclid's Elements in the Conformal Model



FIG(15,1)

- *primitives as subspaces*:
  points, lines, planes, circles, spheres, tangents
- *constructions as subspace products*:
  connections, intersections, orthogonal complement, duality
- *motions as versors*:
  translations, rotations, reflection
  (actually, any conformal transformation)
- *properties parametrized*:
  size, weight, location, orientation, direction, carrier
- *numerics exactly linearized*:
  linear (bivector) parametrization of motions