Applications of Geometric Algebra I

Chris Doran
Cavendish Laboratory
Cambridge University

C.Doran@mrao.cam.ac.uk
www.mrao.cam.ac.uk/~clifford
3D Algebra

- 3D basis consists of 8 elements
- Represent lines, planes and volumes, from a common origin

Grade 0
Scalar
1

Grade 1
Vector
$e_1, e_2, e_3$

Grade 2
Bivector
$e_1 e_2, e_2 e_3, e_3 e_1$

Grade 3
Trivector
$I$
Algebraic Relations

- Generators anticommute: $e_1 e_2 = -e_2 e_1$
- Geometric product: $ab = a \cdot b + a \wedge b$
- Inner product: $a \cdot b = \frac{1}{2} (ab + ba)$
- Outer product: $a \wedge b = \frac{1}{2} (ab - ba)$
- Bivector norm: $(e_1 \wedge e_2)^2 = -1$
- Trivector: $I = e_1 e_2 e_3$
- Trivector norm: $I^2 = -1$
- Trivectors commute with all other elements
Lines and Planes

• Pseudoscalar gives a map between lines and planes
  \[ B = Ia \]
  \[ a = -IB \]

• Allows us to recover the vector (cross) product
  \[ a \times b = -I a \wedge b \]

• But lines and planes are different
• Far better to keep them as distinct entities
Quaternions

• For the bivectors set

\[ i = e_2 e_3, \quad j = -e_3 e_1, \quad k = e_1 e_2 \]

• These satisfy the quaternion relations

\[ i^2 = j^2 = k^2 = ijk = -1 \]

• So quaternions embedded in 3D GA

• Do not lose anything, but
  – Vectors and planes now separated
  – Note the minus sign!
  – GA generalises
Reflections

- Build rotations from reflections
- Good example of geometric product – arises in *operations*

\[
\begin{align*}
    a_\parallel &= (a \cdot n)n \\
    a_\perp &= a - (a \cdot n)n
\end{align*}
\]

- Image of reflection is

\[
b = a_\perp - a_\parallel = a - 2(a \cdot n)n
\]

\[
= a - (an + na)n = -nan
\]
Rotations

- 2 successive reflections give a rotation

Initial vector in red
Reflection in green
Rotated in blue
Rotations

• Direction perpendicular to the two reflection vectors is unchanged
• So far, will only talk about rotations in a plane with a fixed origin (more general treatment later)
Algebraic Formulation

• Now look at the algebraic expression for a pair of reflections

\[ a \rightarrow -m(-nan)m = mnanm \]

• Define the rotor \( R = mn \)

• Rotation encoded algebraically by

\[ a \rightarrow RaR^\dagger \quad R^\dagger = nm \]

• Dagger symbol used for the reverse
Rotors

- Rotor is a geometric product of 2 unit vectors
  \[ R = mn = \cos(\theta) + m \wedge n \]
- Bivector has square
  \[ (m \wedge n)^2 = (mn - \cos \theta)(-nm + \cos \theta) = -\sin^2 \theta \]
- Used to the negative square by now!
- Introduce unit bivector \( \hat{B} = \frac{m \wedge n}{\sin \theta} \)
- Rotor now written
  \[ R = \cos(\theta) + \sin(\theta)\hat{B} \]
Exponential Form

• Can now write \( R = \exp(\theta \hat{B}) \)
• But:
  – rotation was through \textbf{twice} the angle between the vectors
  – Rotation went with orientation \( n \mapsto m \)
• Correct these, get double-sided, half-angle formula
  \[
  a \rightarrow RaR^\dagger \quad R = \exp(-\theta \hat{B}/2)
  \]
• Completely general!
Rotors in 3D

- Can rewrite in terms of an axis via
  \[ R = \exp(-\theta I_n/2) \]
- Rotors even grade (scalar + bivector in 3D)
- Normalised: \( RR^\dagger = mnnm = 1 \)
- Reduces d.o.f. from 4 to 3 – enough for a rotation
- In 3D a rotor is a normalised, even element
- The same as a unit quaternion
Group Manifold

- Rotors are elements of a 4D space, normalised to 1
- They lie on a 3-sphere
- This is the group manifold
- Tangent space is 3D
- Natural linear structure for rotors
- Rotors $R$ and $-R$ define the same rotation
- Rotation group manifold is more complicated
Comparison

• Euler angles give a standard parameterisation of rotations

\[
\begin{pmatrix}
\cos \psi \cos \phi - \cos \theta \sin \phi \sin \psi & -\sin \psi \cos \phi - \cos \theta \sin \phi \cos \psi & \sin \theta \sin \phi \\
\cos \psi \sin \phi + \cos \theta \cos \phi \sin \psi & -\sin \psi \sin \phi + \cos \theta \cos \phi \cos \psi & -\sin \theta \cos \phi \\
\sin \theta \sin \psi & \sin \theta \cos \psi & \cos \theta
\end{pmatrix}
\]

• Rotor form far easier

\[R = \exp(-e_1 e_2 \phi/2) \exp(-e_2 e_3 \theta/2) \exp(-e_1 e_2 \psi/2)\]

• But can do better than this anyway – work directly with the rotor element
Composition

• Form the compound rotation from a pair of successive rotations
  \[ a \mapsto R_2(R_1 a R_1^\dagger) R_2^\dagger \]

• Compound rotor given by group combination law \( R = R_2 R_1 \)

• Far more efficient than multiplying matrices

• More robust to numerical error

• In many applications can safely ignore the normalisation until the final step
Oriented Rotations

- Rotate through 2 different orientations
- Positive Orientation
  \[ R = \exp(-\lambda e_1 e_2/2) = \exp(-e_1 e_2 \pi/4) \]
- Negative Orientation
  \[ S = \exp(\lambda e_1 e_2/2) = \exp(e_1 e_2 3\pi/4) = -R \]
- So \( R \) and \( -R \) encode the same absolute rotation, but with different orientations
Lie Groups

• Every rotor can be written as $\exp(-B/2)$
• Rotors form a continuous (Lie) group
• Bivectors form a Lie algebra under the commutator product
• All finite Lie groups are rotor groups
• All finite Lie algebras are bivector algebras
• (Infinite case not fully clear, yet)
• In conformal case (later) starting point of screw theory (Clifford, 1870s)!
Interpolation

• How do we interpolate between 2 rotations?
• Form path between rotors

\[
\begin{align*}
R(0) &= R_0 \\
R(1) &= R_1
\end{align*}
\]

• Find \( B \) from \( \exp(B) = R_0^\dagger R_1 \)
• This path is invariant. If points transformed, path transforms the same way
• Midpoint simply \( R(1/2) = R_0 \exp(-B/2) \)
• Works for \textit{all} Lie groups
Interpolation - SLERP

- For rotors in 3D can do even better!
- View rotors as unit vectors in 4D
- Path is a circle in a plane
- Use simple trig’ to get SLERP

\[ R(\lambda) = \frac{1}{\sin(\theta)} \left( \sin((1 - \lambda)\theta)R_0 + \sin(\lambda\theta)R_1 \right) \]

- For midpoint add the rotors and normalise!

\[ R(1/2) = \frac{\sin(\theta/2)}{\sin(\theta)} (R_0 + R_1) \]
Applications

- Use SLERP with spline constructions for general interpolation
- Interpolate between series of rigid-body orientations
- Elasticity
- Framing a curve
- Extend to general transformations
Linearisation

• Common theme is that rotors can **linearise** the rotation group, without approximating!
• Relax the norm constraint on the rotor and write
  \[ RA R^\dagger = \psi A \psi^{-1} \]
• \( \psi \) belongs to a linear space. Has a natural calculus.
• Very powerful in optimisation problems involving rotations
• Employed in computer vision algorithms
Recovering a Rotor

- Given two sets of vectors related by a rotation, how do we recover the rotor?
- Suppose $b_i = Ra_iR^\dagger$
- In general, assume not orthogonal.
- Need reciprocal frame

$$a^1 = \frac{a_2 \wedge a_3 I}{(a_1 \wedge a_2 \wedge a_3 I)}$$

- Satisfies $a^i \cdot a_j = \delta^i_j$
Recovering a Rotor II

• Now form even-grade object
  \[ b_i a^i = Ra_i(\alpha + B)a^i = R(3\alpha - B) = -1 + 4\alpha R \]

• Define un-normalised rotor
  \[ \psi = b_i a^i + 1 \]

• Recover the rotor immediately now as
  \[ R = \frac{\psi}{|\psi|} \]

• Very efficient, but
  – May have to check the sign
  – Careful with 180° rotations
Rotor Equations

• Suppose we take a path in rotor space \( R(\lambda) \)
• Differentiating the constraint tells us that
  \[
  \frac{d}{d\lambda} (RR^\dagger) = R'R^\dagger + RR'' = 0
  \]
• Re-arranging, see that
  \[
  R'R^\dagger = -(R'R^\dagger)^\dagger = \text{Bivector}
  \]
• Arrive at rotor equation
  \[
  R' = -\frac{1}{2} B R
  \]
• This is totally general. Underlies the theory of Lie groups
Example

• As an example, return to framing a curve.
• Define Frenet frame
• Relate to fixed frame

\[ \{t, n, b\} = R e_i R^i \]

• Rotor equation

\[ R' = -\frac{1}{2} R \Omega \quad \Omega = \kappa_1 e_2 e_1 + \kappa_2 e_3 e_2 \]

• Rotor equation in terms of curvature and torsion
Linearisation II

- Rotor equations can be awkward (due to manifold structure)
- Linearisation idea works again
- Replace rotor with general element and write
  \[ \psi' = -\frac{1}{2} B \psi \]
- Standard ODE tools can now be applied (Runge-Kutta, etc.)
- Normalisation of \( \psi \) gives useful check on errors
Elasticity

• Some basics of elasticity (solid mechanics):
  – When an object is placed under a stress (by stretching or through pressure) it responds by changing its shape.
  – This creates strains in the body.
  – In the linear theory stress and strain are related by the elastic constants.
  – An example is Hooke’s law $F=-kx$, where $k$ is the spring constant.
  – Just the beginning!
Bulk Modulus

- Place an object under uniform pressure $P$
- Volume changes by

$$-P = B \frac{\delta V}{V}$$

- $B$ is the bulk modulus
- Definition applies for small pressures (linear regime)
Shear Modulus

• Sheers produced by combination of tension and compression
• Sheer modulus $G$ is
  Shear stress / angle

\[ G = \frac{\tau}{2\theta} \]
LIH Media

- The simplest elastic systems to consider are **linear, isotropic** and **homogeneous** media.
- For these, \( B \) and \( G \) contain all the relevant information.
- There are many ways to extend this:
  - Go beyond the linearised theory and treat large deflections
  - Find simplified models for rods and shells
Foundations

• Key idea is to relate the spatial configuration to a ‘reference’ copy.

• \( y = f(x) \) is the displacement field. In general, this will be time-dependent as well.
Paths

• From \( f(x) \) we want to extract information about the strains. Consider a path

\[
\Delta f(x + \epsilon a) = \epsilon a \cdot \nabla f(x) = \epsilon F(a)
\]

• Tangent vectors map to

• \( F(a) = F(a; x) \) is a linear function of \( a \). Tells us about local distortions.
Path Lengths

• Path length in the reference body is

\[ \int \left( \frac{dx}{d\lambda} \cdot \frac{dx}{d\lambda} \right)^{1/2} d\lambda \]

• This transforms to

\[ \int (F(x') \cdot F(x'))^{1/2} d\lambda \]

• Define the function \( G(a) \), acting entirely in the reference body, by

\[ G(a) = \bar{F}F(a) \]
The Strain Tensor

• For elasticity, usually best to ‘pull’ everything back to the reference copy
• Use same idea for rigid body mechanics
• Define the strain tensor from $G(a)$
  – Most natural is
    $$E(a) = \frac{1}{2} (G(a) - a)$$
  – An alternative (rarely seen) is
    $$E(a) = \frac{1}{2} \ln G(a)$$
The Stress Tensor

• Contact force between 2 surfaces is a linear function of the normal (Cauchy)

\[ \tau(n) = \tau(n;x) \]

returns a vector in the material body. ‘Pull back’ to reference copy to define

\[ T(n) = F^{-1}(\tau(n)) \]
Constitutive Relations

• Relate the stress and the strain tensors in the reference configuration
• Considerable freedom in the choice here
• The simplest, LIH media have
  \[ T(a) = 2GE(a) + (B - \frac{2}{3} G)\text{tr}(E) a \]
• Can build up into large deflections
• Combined with balance equations, get full set of dynamical equations
• Can get equations from an action principle
Problems

• Complicated, and difficult numerically
• In need of some powerful advanced mathematics for the full nonlinear theory (FEM…)
• Geometric algebra helps because it
  – is coordinate free
  – integrates linear algebra and calculus smoothly
• But need simpler models
• Look at models for rods and beams
Deformable Rod

- Reference configuration is a cylinder

Configuration encoded in a rotor \( y = x(\lambda, t) + R\sigma R^\dagger \)

Line of centre of mass

\[\lambda\]

\[\sigma\]

\[x\]

\[y\]
Technical Part

- Spare details, but:
  - Write down an action integral
  - Integrate out the coordinates over each disk
  - Get (variable) bending moments along the centre line
- Carry out variational principle
- Get set of equations for the rotor field
- Can apply to static or dynamic configurations
Simplest Equations

• Static configuration, and ignore stretching
• Have rotor equation

\[ \frac{dR}{d\lambda} = -\frac{1}{2} R \Omega_B \]

• Find bivector from applied couple and elastic constants. \( I(B) \) is a known linear function of these mapping bivectors to bivectors

\[ \Omega_B = I^{-1}(R^\dagger CR) \]

• Integrate to recover curve

\[ x' = Re_1 R^\dagger \]
Example

- Even this simple set of equations can give highly complex configurations!

Small, linear deflections build up to give large deformations.
Summary

- Rotors are a general purpose tool for handling rotations in arbitrary dimensions
- Computationally more efficient than matrices
- Can be associated with a linear space
- Easy to interpolate
- Have a natural associated calculus
- Form basis for algorithms in elasticity and computer vision
- All this extends to general groups!
Further Information

• All papers on Cambridge GA group website: www.mrao.cam.ac.uk/~clifford

• Applications of GA to computer science and engineering are discussed in the proceedings of the AGACSE 2001 conference. www.mrao.cam.ac.uk/agacse2001

• IMA Conference in Cambridge, 9th Sept 2002

• ‘Geometric Algebra for Physicists’ (Doran + Lasenby). Published by CUP, soon.
Revised Timetable

- 8.30 – 9.15 Rockwood
  *Introduction and outline of geometric algebra*
- 9.15 – 10.00 Mann
  *Illustrating the algebra I*
- 10.00 – 10.15 Break
- 10.15 – 11.15 Doran
  *Applications I*
- 11.15 – 12.00 Lasenby
  *Applications II*
- 1.30 – 2.00 Doran
  *Beyond Euclidean Geometry*
- 2.00 – 3.00 Hestenes
  *Computational Geometry*
- 3.00 – 3.15 Break
- 3.15 – 4.00 Dorst
  *Illustrating the algebra II*
- 4.00 – 4.30 Lasenby
  *Applications III*
- 4.30 Panel