Visualizing Quaternions

Andrew J. Hanson

Computer Science Department

Indiana University

IUPUI CS Colloquium, 10 Feb 2006

OUTLINE

I: Hamilton's strange number system

II: Twisting Belts, Rolling Balls, and Locking Gimbals:

Explaining Rotation Sequences with Quaternions

III: Introduction to Quaternion Fields:

Curves, Surfaces, and Volumes

Strange number systems: Where Did *Quaternions* Come From?

. . . from the discovery of *Complex Numbers:*

- $\boxed{z = x + iy}$ Complex numbers = realization that $z^2 + 1 = 0$ cannot be solved unless you have an "imaginary" number with $i^2 = -1$.
- **Euler's formula**: $e^{i\theta} = \cos\theta + i\sin\theta$ allows you to do most of 2D geometry.

Hamilton

The first to ask *"If you can do 2D geometry with complex numbers, how might you do 3D geometry?"* was William Rowan Hamilton, circa 1840.

Sir William Rowan Hamilton
4 August 1805 — 2 September 1865

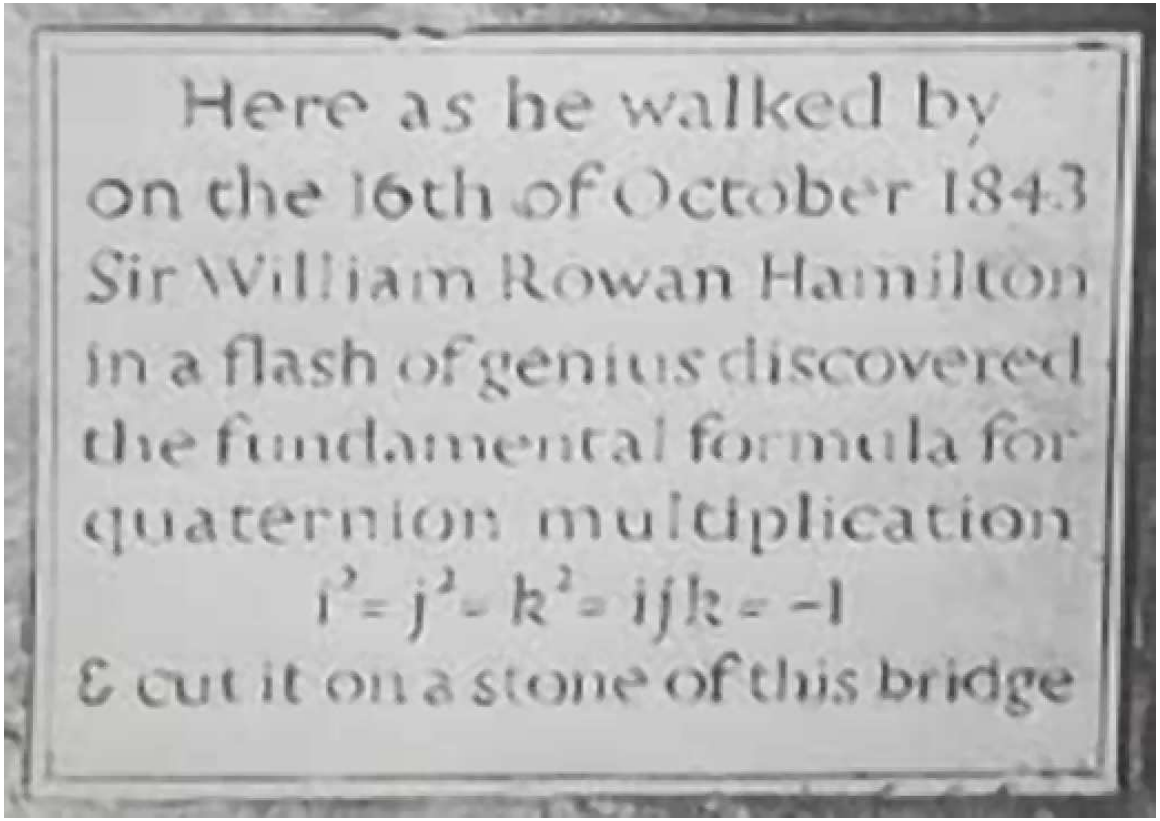
Hamilton's epiphany: 16 October 1843

"An electric circuit seemed to close; and a spark flashed forth . . . Nor could I resist the impulse – unphilosophical as it may have been – to cut with a knife on a stone of Brougham Bridge, as we passed it, the fundamental formula with the symbols, i, j, k; namely,

$$i^2 = j^2 = k^2 = ijk = -1$$

which contains the Solution of the Problem..."

...at the site of Hamilton's carving



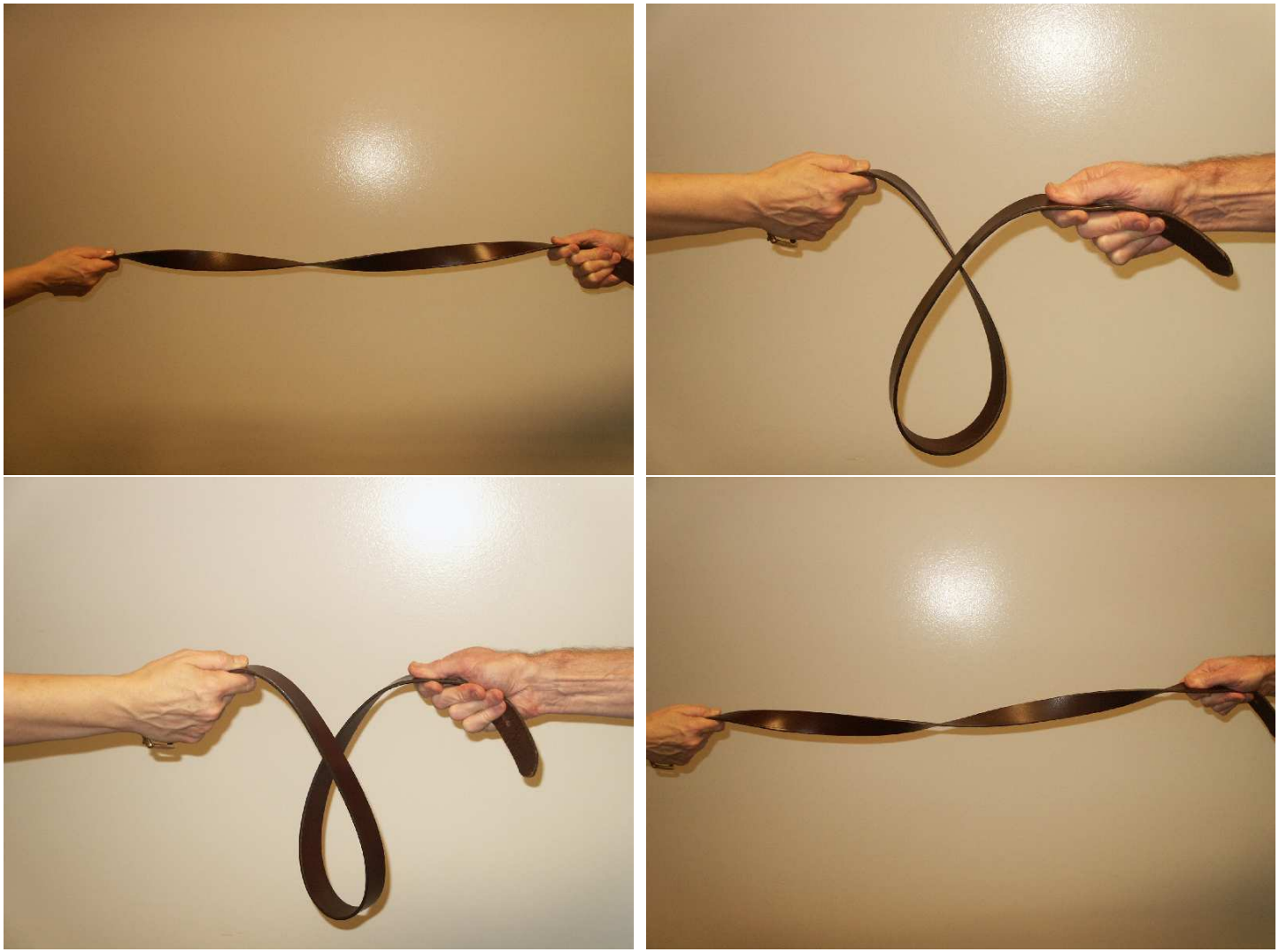
The plaque on Broome Bridge in Dublin, Ireland, commemorating the legendary location where Hamilton conceived of the idea of quaternions. (Hamilton apparently misspelled it as "Brougham Bridge" in his letter.)

The Belt Trick

Quaternion Geometry in our daily lives

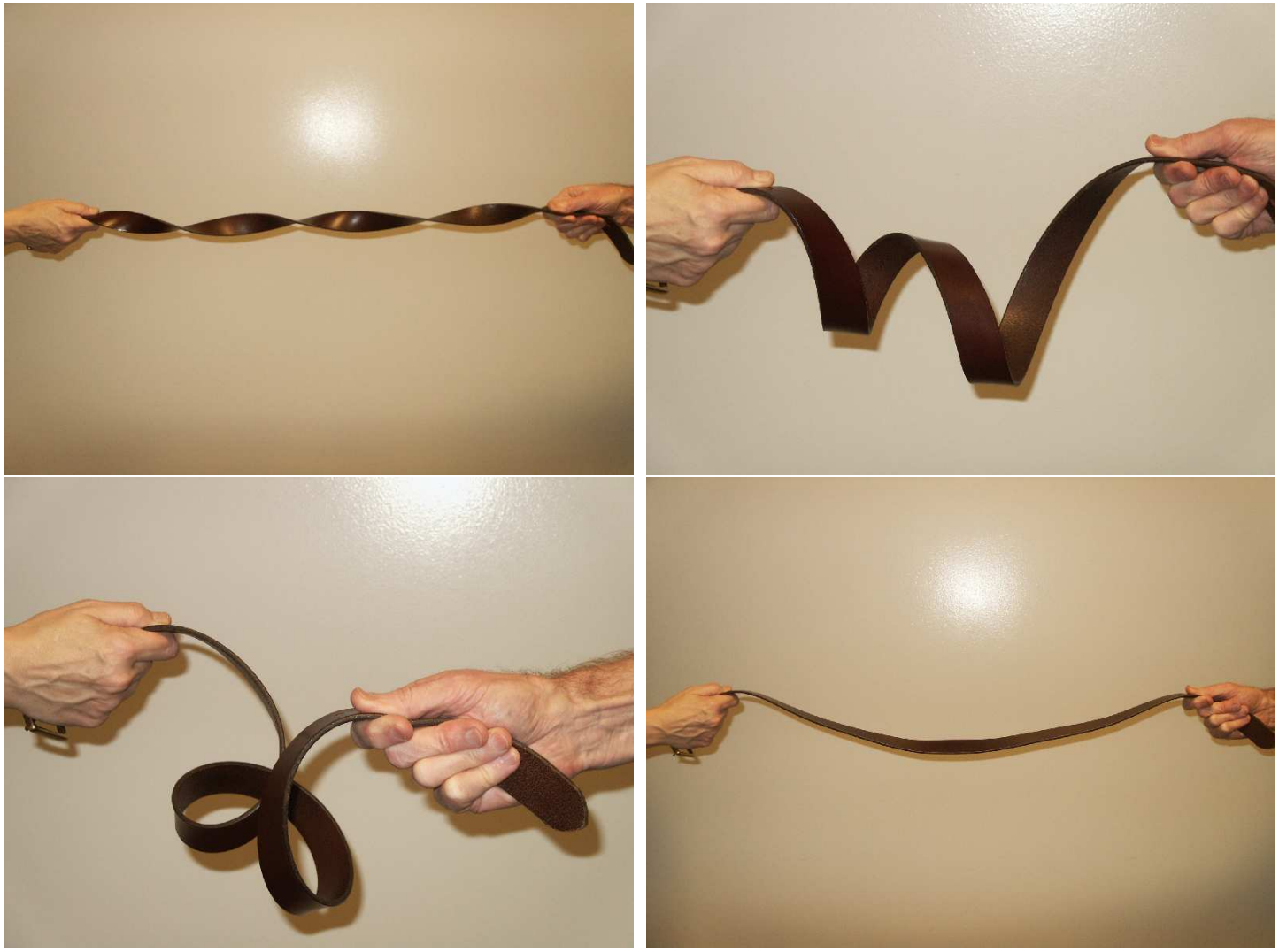
- Two people hold ends of a belt.
- Twist the belt either 360 degrees or 720 degrees.
- **Rule:** *Move belt ends any way you like but do not change orientation of either end.*
- Try to straighten out the belt.

360 Degree Belt



360 twist: stays twisted, can change DIRECTION!

720 Degree Belt

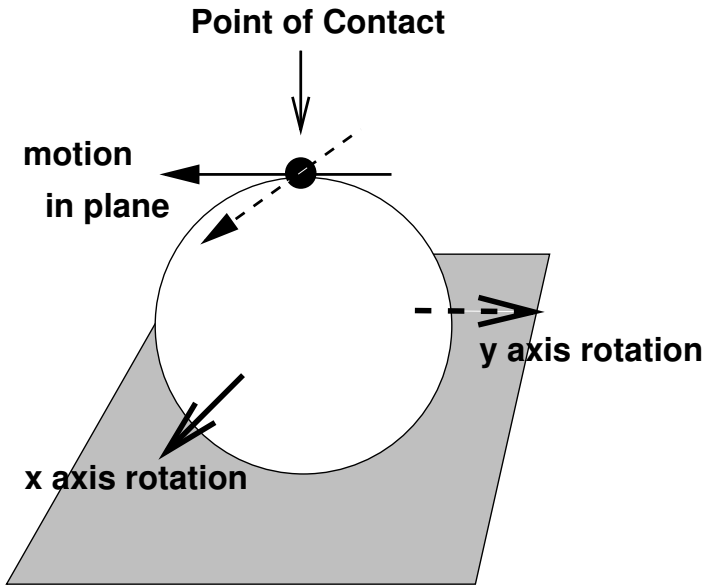


720 twist: CAN FLATTEN OUT WHOLE BELT!

Rolling Ball Puzzle

1. Put a ball on a flat table.
2. Place hand flat on top of the ball
3. Make circular rubbing motion, as though polishing the tabletop.
4. Watch a point on the equator of the ball.
5. *small clockwise circles* $\longrightarrow$ **equator goes counterclockwise**
6. *small counterclockwise circles* $\longrightarrow$ **equator goes clockwise**

Rolling Ball Scenario



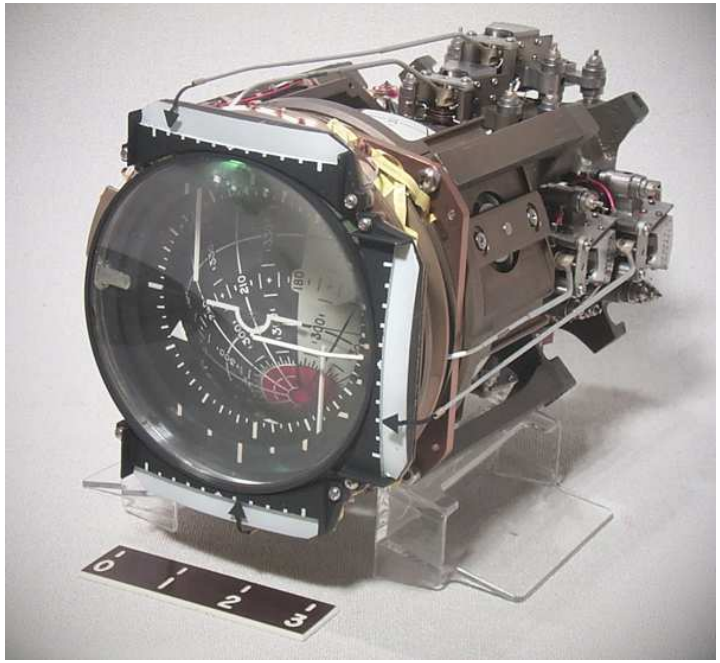
Gimbal Lock

Gimbal Lock occurs when a mechanical or computer system experiences an anomaly due to an (x, y, z)-based orientation control sequence.

- *Mechanical systems cannot avoid all possible gimbal lock situations .*

- *Computer orientation interpolation systems can avoid gimbal-lock-related glitches* **by using quaternion interpolation.**

Gimbal Lock — Apollo Systems



Red-painted area = Danger of real Gimbal Lock

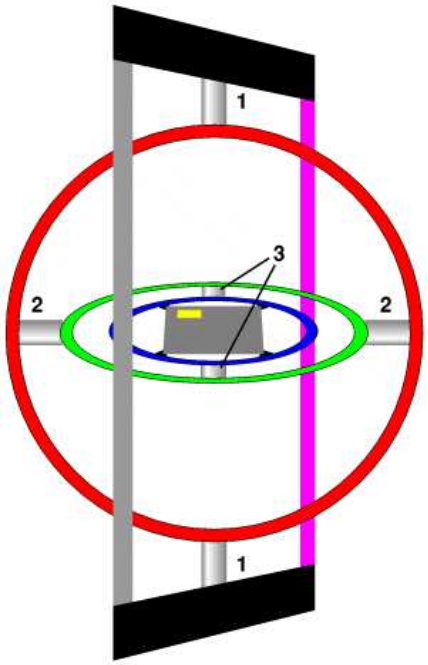


FIGURE 2

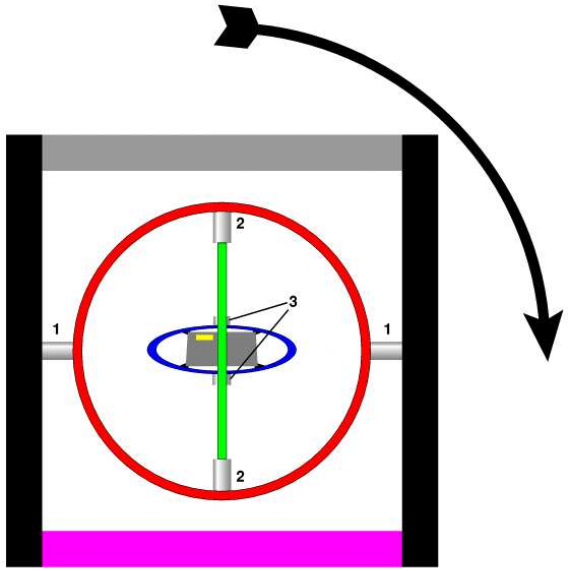


FIGURE 3

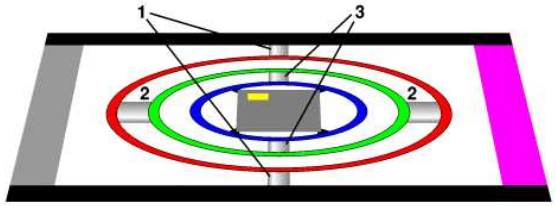


FIGURE 4

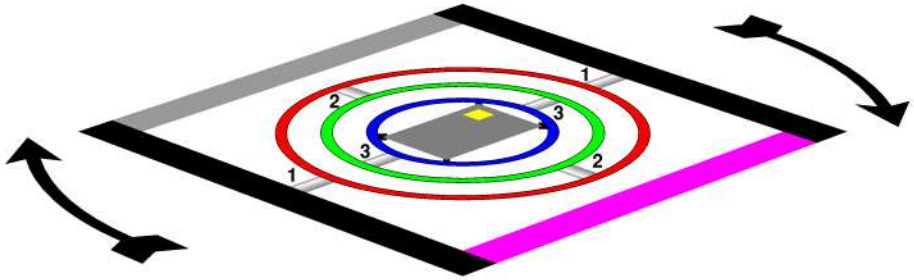


FIGURE 5

Mechanical Gimbal Lock: **Using x, y, z axes to encode orientation gives singular situations.**

2D Rotations

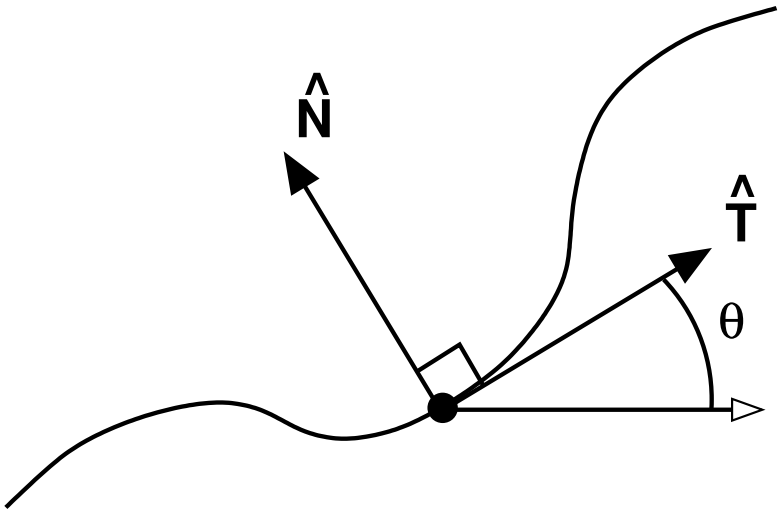
- 2D rotations $\leftrightarrow$ *complex numbers.*
- Why? $e^{i\theta}\,(x+iy) = (x'+iy')$

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}$$

- **Complex numbers** are a subspace of quaternions — so exploit 2D rotations to **introduce us to quaternions** and their geometric meaning.

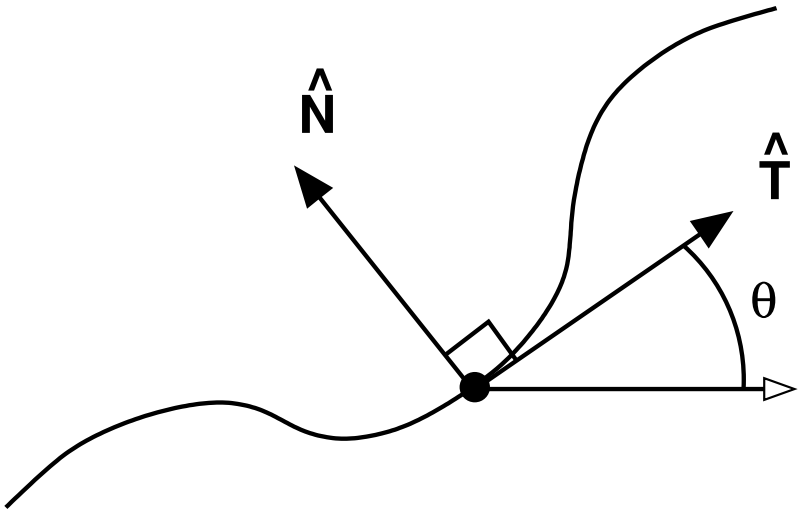
Frames in 2D

The tangent and normal to 2D curve move continuously along the curve:



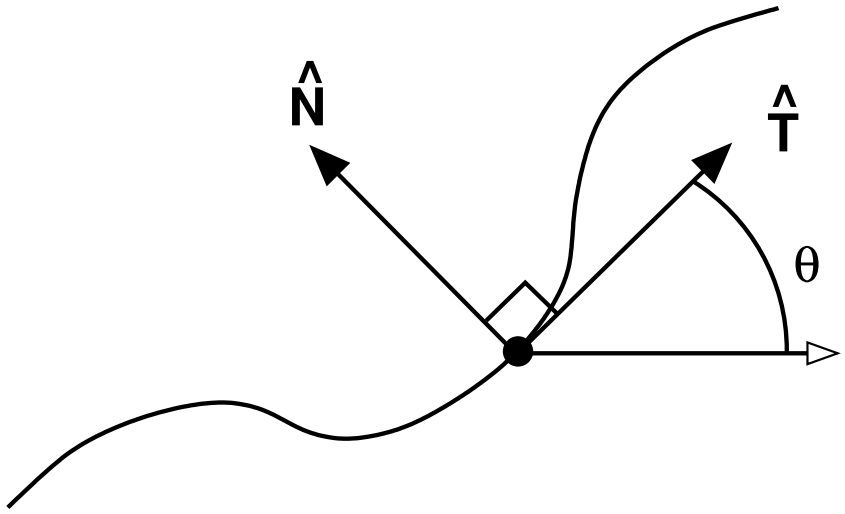
Frames in 2D

The tangent and normal to 2D curve move continuously along the curve:



Frames in 2D

The tangent and normal to 2D curve move continuously along the curve:



Frame Matrix in 2D

This motion is described at each point (or time) by the matrix:

$$R_2(\theta) = \left[\hat{\mathbf{T}} \;\; \hat{\mathbf{N}} \right]$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

The Belt Trick Says:

There is a Problem...at least in 3D

How do you get $\cos\theta$ to know about 720 degrees?

The Belt Trick Says:

There is a Problem...at least in 3D

How do you get $\cos\theta$ to know about 720 degrees?

Hmmmmm. $\cos(\theta/2)$ knows about 720 degrees, right?

Half-Angle Transform:

A Fix for the Problem?

Let $a = \cos(\theta/2)$, $b = \sin(\theta/2)$,

(i.e., $\cos\theta = a^2 - b^2$, $\sin\theta = 2ab$),

and parameterize 2D rotations as:

$$R_2(a,b) \;=\; \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix} .$$

where orthonormality implies

$$(a^2 + b^2)^2 = 1$$

which reduces back to $a^2 + b^2 = 1$.

Frame Evolution in 2D

Examine time-evolution of 2D frame (on our way to 3D): First in $\theta(t)$ coordinates:

$$\left[\begin{array}{cc} \hat{\mathbf{T}} & \hat{\mathbf{N}} \end{array}\right] = \left[\begin{array}{cc} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array}\right] .$$

Differentiate to find frame equations:

$$\begin{aligned} \dot{\hat{\mathbf{T}}}(t) &= +\kappa\hat{\mathbf{N}} \\ \dot{\hat{\mathbf{N}}}(t) &= -\kappa\hat{\mathbf{T}} , \end{aligned}$$

where $\kappa(t) = d\theta/dt$ is the **curvature**.

2D Quaternion Frames!

Rearranging terms, *both* $\dot{\hat{\mathrm{T}}}$ and $\dot{\hat{\mathrm{N}}}$ eqns reduce to

$$\begin{bmatrix} \dot{a} \\ \dot{b} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\kappa \\ +\kappa & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix}$$

This is the square root *of frame equations.*

Rotation as Complex Multiplication

If we let $(a + ib) = \exp(i\,\theta/2)$ we see that

rotation is complex multiplication!

"Quaternion Frames" in 2D are just complex numbers, with

Evolution Eqns = derivative of $\exp(i\,\theta/2)$!

Rotation with no matrices!

Due to an extremely deep reason in Clifford Algebras,

$$a + ib = e^{i\theta/2}$$

represents rotations "more nicely" than the matrices $R(\theta)$.

$$(a' + ib')(a + ib) = e^{i(\theta'+\theta)/2} = A + iB$$

where if we *really want* the matrix, we write:

$$R(\theta')R(\theta) = R(\theta' + \theta) = \begin{bmatrix} A^2 - B^2 & -2AB \\ 2AB & A^2 - B^2 \end{bmatrix}$$

The Algebra of 2D Rotations

The algebra corresponding to 2D rotations is easy: just complex multiplication!!

$$\begin{aligned}(a', b') * (a, b) &\cong (a' + ib')(a + ib) \\ &= a'a - b'b + i(a'b + ab') \\ &\cong (a'a - b'b,\ a'b + ab') \\ &= (A,\ B)\end{aligned}$$

2D Rotations are just **complex multiplication**, and take you around the unit circle!

Quaternion Frames

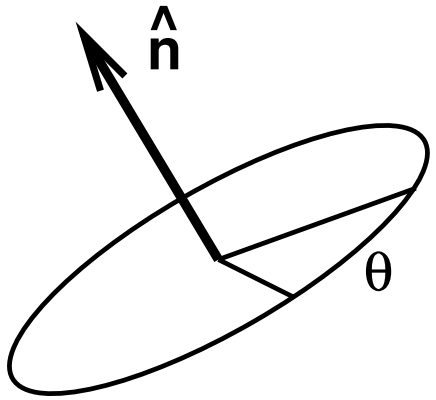
In 3D, repeat our trick: take square root of the frame, but now use *quaternions*:

- **Write down the 3D frame.**
- **Write as double-valued quadratic form.**
- **Rewrite frame evolution equations linearly in the new variables.**

The Geometry of 3D Rotations

We begin with a basic fact:

Euler theorem: *every* 3D frame can be written as a spinning by θ about a fixed axis $\hat{\mathbf{n}}$, the eigenvector of the rotation matrix:



Quaternion Frames . . .

The Matrix $R_3(\theta, \hat{\mathbf{n}})$ giving 3D rotation by θ about axis $\hat{\mathbf{n}}$ is :

$$\begin{bmatrix} c + (n_1)^2(1-c) & n_1 n_2(1-c) - s n_3 & n_3 n_1(1-c) + s n_2 \\ n_1 n_2(1-c) + s n_3 & c + (n_2)^2(1-c) & n_3 n_2(1-c) - s n_1 \\ n_1 n_3(1-c) - s n_2 & n_2 n_3(1-c) + s n_1 & c + (n_3)^2(1-c) \end{bmatrix}$$

where $c = \cos\theta$, $s = \sin\theta$, and $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$.

Can we find a 720-degree form?

Remember 2D: $a^2 + b^2 = 1$
$a = \cos\theta/2, \quad b = \sin\theta/2$
then substitute $1 - \cos\theta = (a^2 + b^2) - (a^2 - b^2) = 2b^2$
to find the remarkable expression for $\mathbf{R}(\theta, \hat{\mathbf{n}})$:

$$\begin{bmatrix} a^2 - b^2 + (n_1)^2(2b^2) & 2b^2 n_1 n_2 - 2abn_3 & 2b^2 n_3 n_1 + 2abn_2 \\ 2b^2 n_1 n_2 + 2abn_3 & a^2 - b^2 + (n_2)^2(2b^2) & 2b^2 n_2 n_3 - 2abn_1 \\ 2b^2 n_3 n_1 - 2abn_2 & 2b^2 n_2 n_3 + 2abn_1 & a^2 - b^2 + (n_3)^2(2b^2) \end{bmatrix}$$

Rotations and Quadratic Polynomials

Remember $(n_1)^2 + (n_2)^2 + (n_3)^2 = 1$ and $a^2 + b^2 = 1$; letting

$$q_0 = a = \cos(\theta/2) \qquad \mathbf{q} = b\hat{\mathbf{n}} = \hat{\mathbf{n}}\sin(\theta/2)$$

We find a matrix $R_3(q)$

$$\begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Quaternions and Rotations . . .

HOW does $q = (q_0, \mathbf{q})$ represent rotations?

LOOK at

$$R_3(\theta, \hat{\mathbf{n}}) \overset{?}{=} R_3(q_0, q_1, q_2, q_3)$$

THEN we can verify that choosing

$$q(\theta, \hat{\mathbf{n}}) = (\cos\frac{\theta}{2}, \ \hat{\mathbf{n}}\sin\frac{\theta}{2})$$

makes the R_3 equation an *IDENTITY*.

Quaternions and Rotations . . .

WHAT happens if you do **TWO** rotations?

EXAMINE the action of two rotations

$$R(q')R(q) = R(Q)$$

EXPRESS in quadratic forms in q and LOOK FOR an analog of complex multiplication:

Quaternions and Rotations . . .

RESULT: the following multiplication rule

$\boxed{q' * q = Q}$ yields **exactly** the correct 3×3 rotation matrix $R(Q)$:

$$\begin{bmatrix} Q_0 = [q' * q]_0 \\ Q_1 = [q' * q]_1 \\ Q_2 = [q' * q]_2 \\ Q_3 = [q' * q]_3 \end{bmatrix} = \begin{bmatrix} q'_0 q_0 - q'_1 q_1 - q'_2 q_2 - q'_3 q_3 \\ q'_0 q_1 + q'_1 q_0 + q'_2 q_3 - q'_3 q_2 \\ q'_0 q_2 + q'_2 q_0 + q'_3 q_1 - q'_1 q_3 \\ q'_0 q_3 + q'_3 q_0 + q'_1 q_2 - q'_2 q_1 \end{bmatrix}$$

This is Quaternion Multiplication.

Algebra of Quaternions

= 3D Rotations!

2D rotation matrices are represented

by **complex multiplication**

3D rotation matrices are represented

by **quaternion multiplication**

Algebraic 2D/3D Rotations

Therefore in 3D, the 2D complex multiplication

$$(a', b') * (a, b) = (a'a - b'b,\ a'b + ab')$$

is replaced by 4D quaternion multiplication:

$$\begin{aligned} q' * q \ = \ & (q'_0 q_0 - q'_1 q_1 - q'_2 q_2 - q'_3 q_3, \\ & q'_0 q_1 + q'_1 q_0 + q'_2 q_3 - q'_3 q_2, \\ & q'_0 q_2 + q'_2 q_0 + q'_3 q_1 - q'_1 q_3, \\ & q'_0 q_3 + q'_3 q_0 + q'_1 q_2 - q'_2 q_1) \end{aligned}$$

Algebra of Quaternions . . .

The is easier to remember by dividing it into the *scalar* piece q_0 and the *vector* piece $\vec{\mathbf{q}}$:

$$q' * q = (q'_0 q_0 - \vec{\mathbf{q}'} \cdot \vec{\mathbf{q}}, \\ q'_0 \vec{\mathbf{q}} + q_0 \vec{\mathbf{q}'} + \vec{\mathbf{q}'} \times \vec{\mathbf{q}})$$

Now we can SEE quaternions!

Since $(q_0)^2 + \mathbf{q} \cdot \mathbf{q} = 1$ then

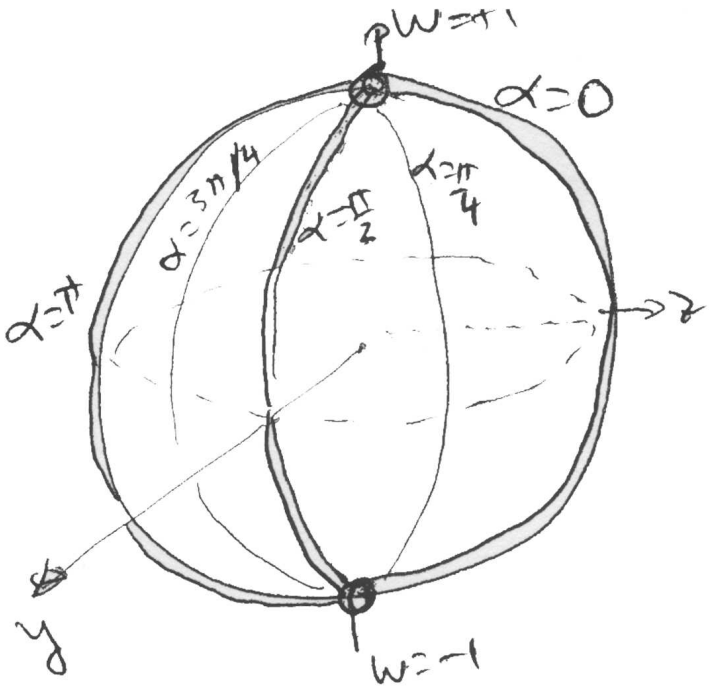
$$q_0 = \sqrt{1 - \mathbf{q} \cdot \mathbf{q}}$$

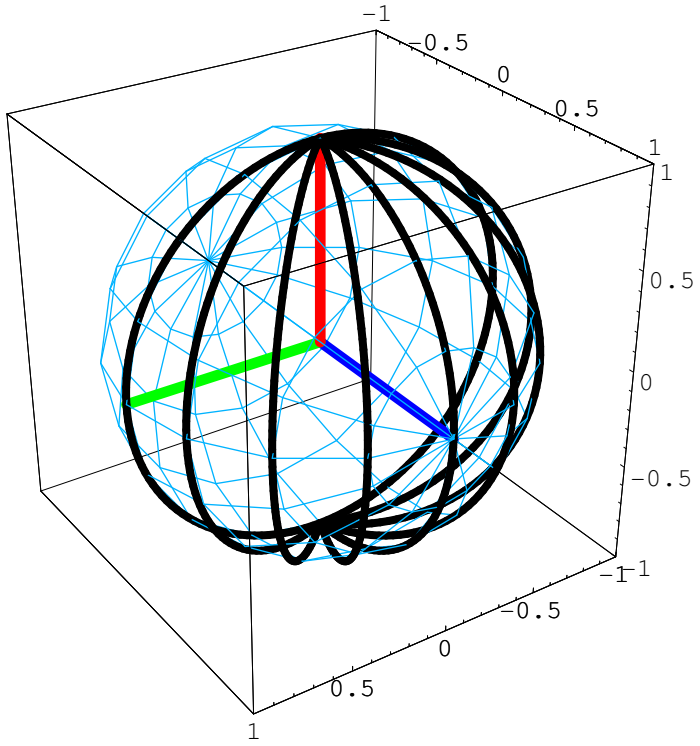
Plot just the 3D vector: $\mathbf{q} = (q_x, q_y, q_z)$

q_0 is KNOWN! Can also use any other triple: the fourth component is dependent.

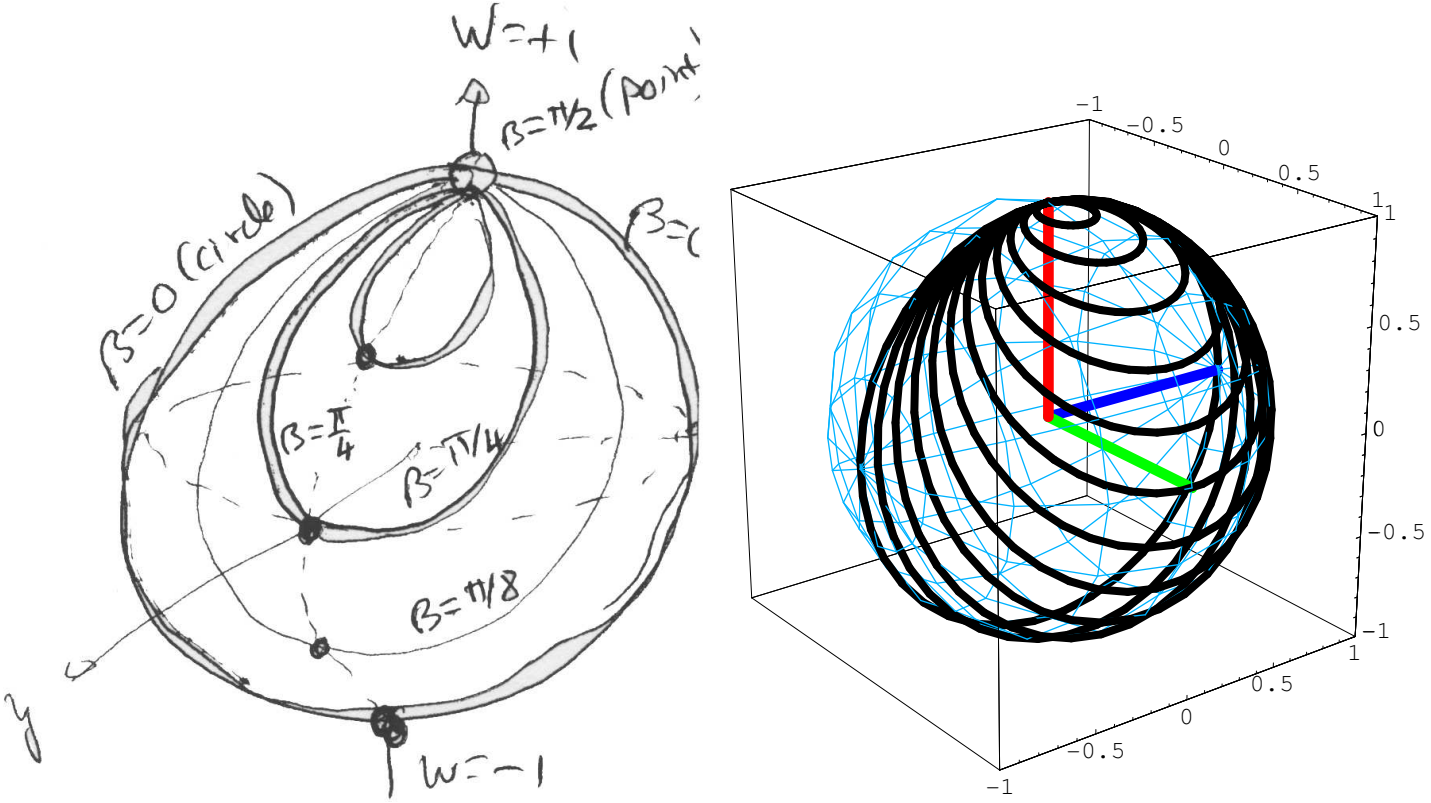
DEMO

360° **Belt Trick in Quaternion Form**

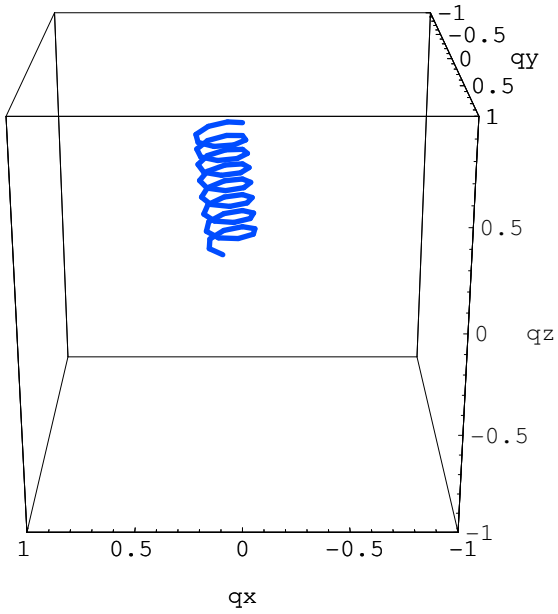


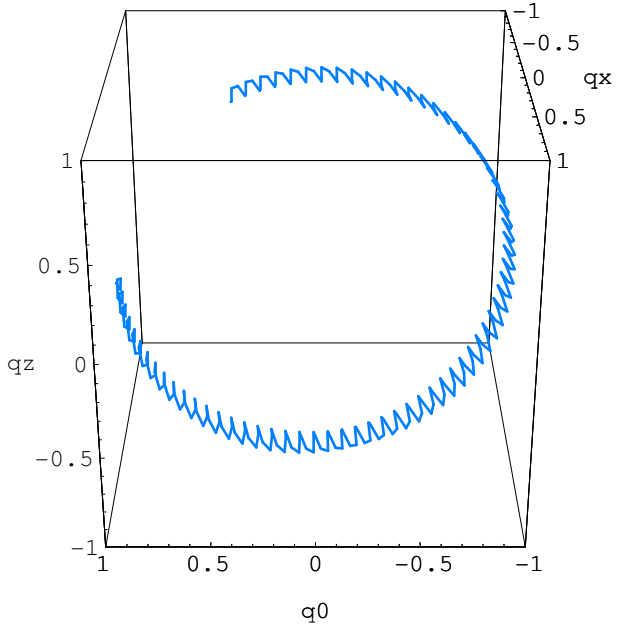


720° **Belt Trick in Quaternion Form**



Rolling Ball in Quaternion Form

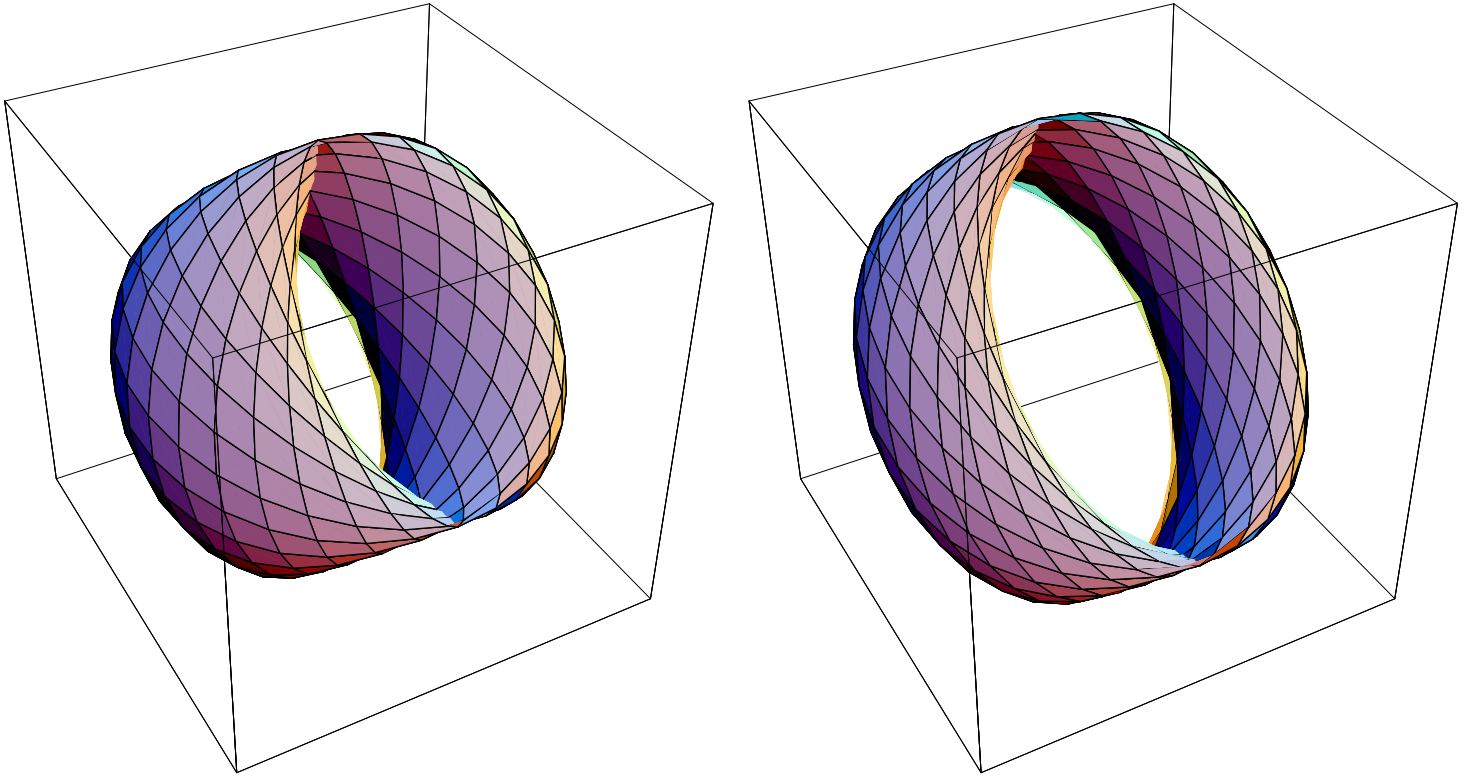




$\mathbf{q}$ vector-only plot.

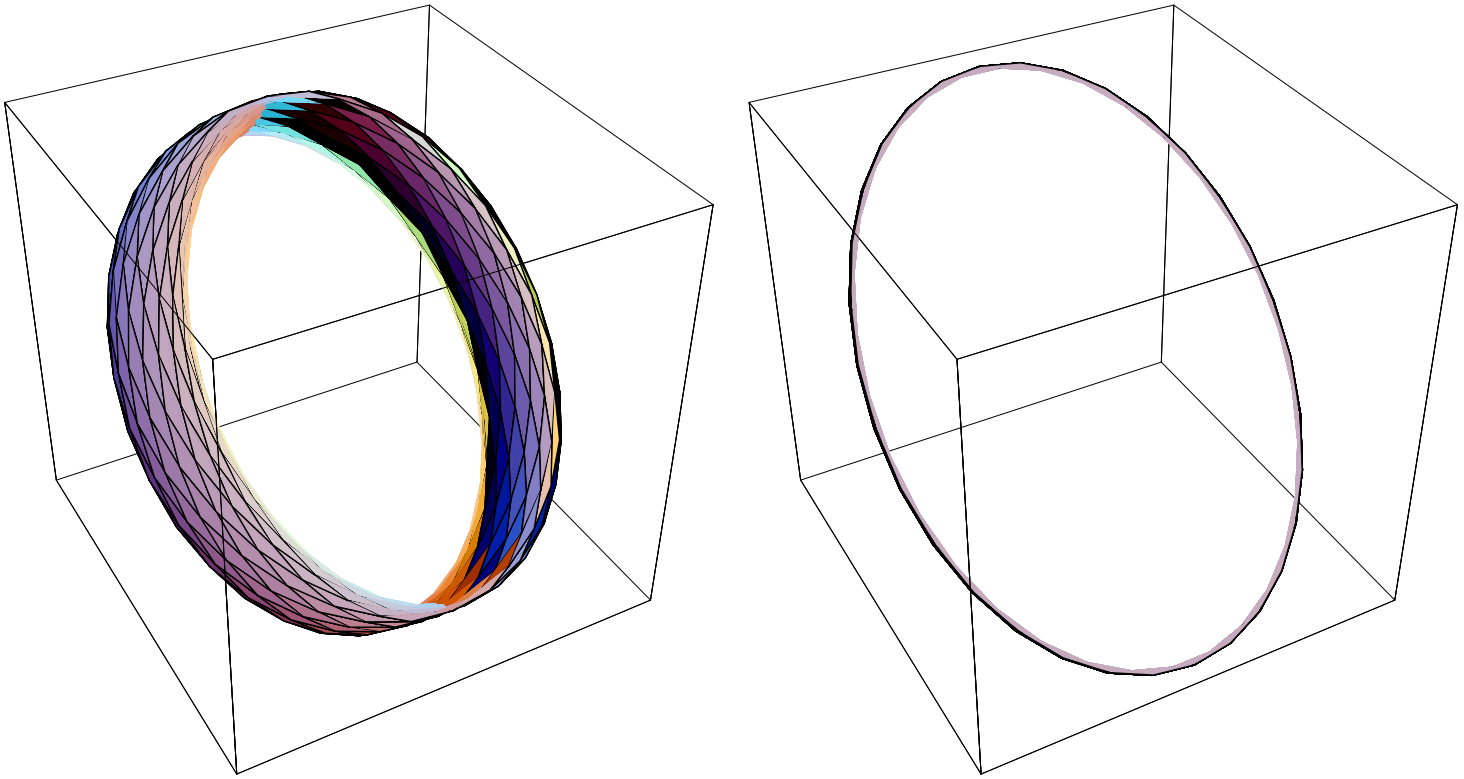
(q_0, q_x, q_z) plot

Gimbal Lock in Quaternion Form



Quaternion Plot of the *remaining* orientation degrees of freedom of $\mathbf{R}(\theta, \hat{\mathbf{x}}) \cdot \mathbf{R}(\phi, \hat{\mathbf{y}}) \cdot \mathbf{R}(\psi, \hat{\mathbf{z}})$ at $\phi = 0$ and $\phi = \pi/6$

Gimbal Lock in Quaternion Form, contd



Choosing ϕ and plotting the *remaining* orientation degrees in the rotation sequence $\mathbf{R}(\theta, \hat{\mathbf{x}}) \cdot \mathbf{R}(\phi, \hat{\mathbf{y}}) \cdot \mathbf{R}(\psi, \hat{\mathbf{z}})$, we see degrees of freedom **decrease from TWO to ONE as** $\phi \rightarrow \pi/2$

Quaternion Interpolations

- Shoemake (Siggraph '85) proposed using quaternions instead of Euler angles to get smooth frame interpolations without **Gimbal Lock**:

  BEST CHOICE: Animate objects and cameras using rotations represented on S^3 by quaternions

Interpolating on Spheres

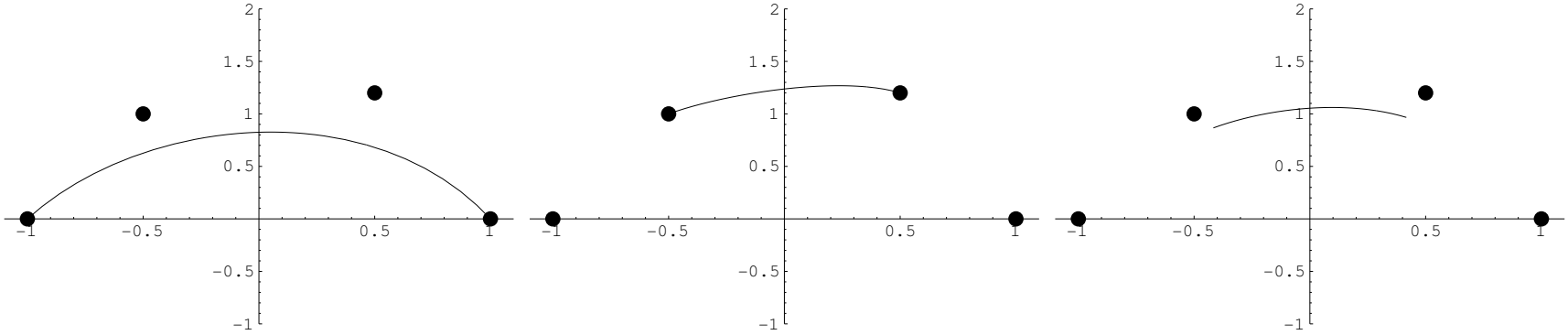
General quaternion spherical interpolation employs the "SLERP," a constant angular velocity transition between two directions, $\hat{\mathbf{q}}_1$ and $\hat{\mathbf{q}}_2$:

$$\begin{aligned} \hat{\mathbf{q}}_{12}(t) &= \mathrm{Slerp}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, t) \\ &= \hat{\mathbf{q}}_1 \frac{\sin((1-t)\theta)}{\sin(\theta)} + \hat{\mathbf{q}}_2 \frac{\sin(t\theta)}{\sin(\theta)} \end{aligned}$$

where $\cos\theta = \hat{\mathbf{q}}_1 \cdot \hat{\mathbf{q}}_2$.

Plane Interpolations

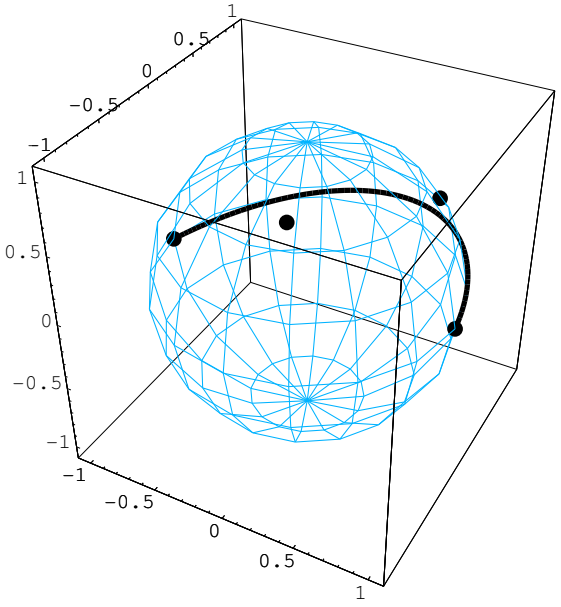
In Euclidean space, these three basic cubic splines look like this:



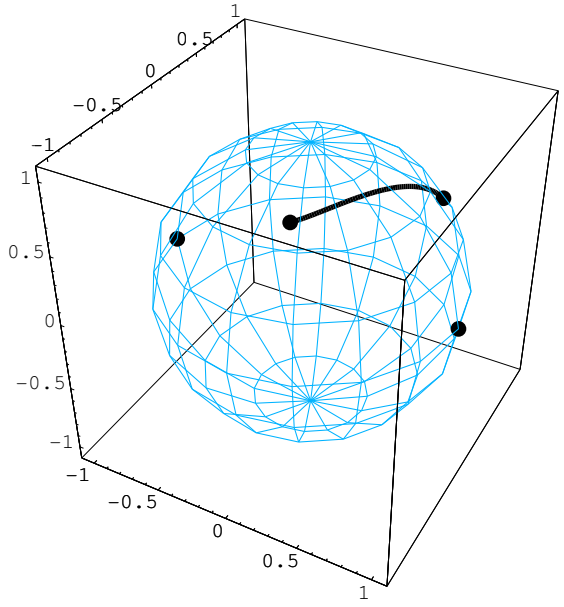
Bezier Catmull-Rom Uniform B

The differences are in the derivatives: Bezier has to start matching all over at every fourth point; Catmull-Rom matches the first derivative; and B-spline is the cadillac, matching all derivatives but no *control points*.

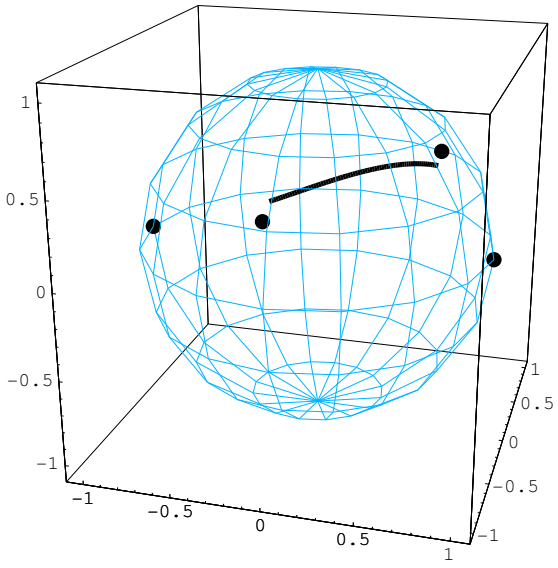
Spherical Interpolations



Bezier

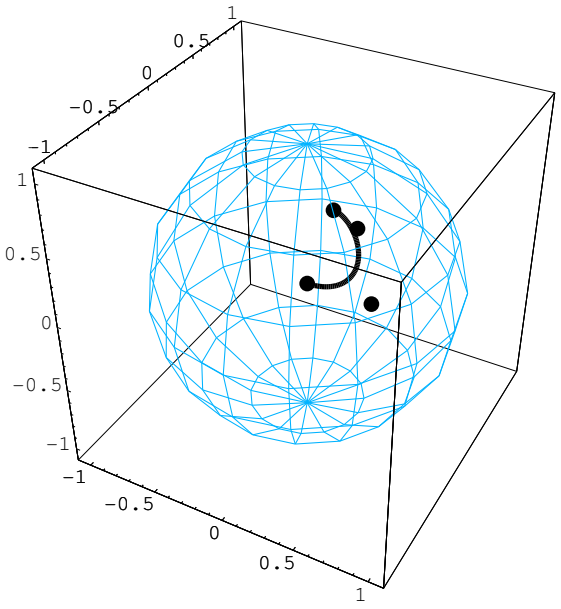


Catmull-Rom

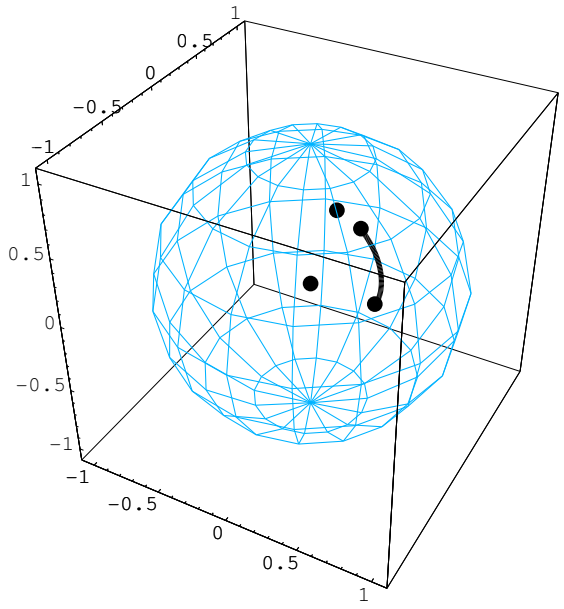


Uniform B

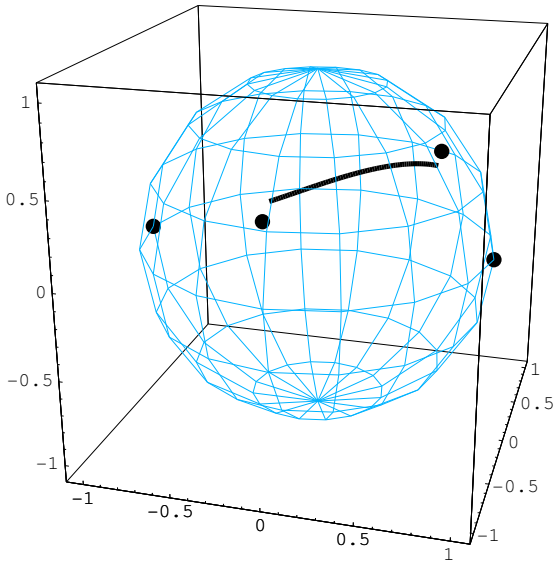
Quaternion Interpolations



Bezier



Catmull-Rom



Uniform B

Key to Quaternion Intuition

Fundamental Intuition: We know

$$q_0 = \cos(\theta/2), \quad \vec{\mathbf{q}} = \hat{\mathbf{n}} \sin(\theta/2)$$

We also know that *any coordinate frame* M can be written as $M = R(\theta, \hat{\mathbf{n}})$.

Therefore

$\vec{\mathbf{q}}$ points exactly along the axis we have to rotate around to go from identity I to M, and the length of $\vec{\mathbf{q}}$ tells us how much to rotate.

Summarize Quaternion Properties

- **Unit four-vector.** Take $q = (q_0, q_1, q_2, q_3) = (q_0, \vec{\mathbf{q}})$ to obey constraint $q \cdot q = 1$.

- **Multiplication rule.** The quaternion product q and p is

  $q * p = (q_0 p_0 - \vec{\mathbf{q}} \cdot \vec{\mathbf{p}},\ q_0 \vec{\mathbf{p}} + p_0 \vec{\mathbf{q}} + \vec{\mathbf{q}} \times \vec{\mathbf{p}})$,

  or, alternatively,

$$\begin{bmatrix} [q*p]_0 \\ [q*p]_1 \\ [q*p]_2 \\ [q*p]_3 \end{bmatrix} = \begin{bmatrix} q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3 \\ q_0 p_1 + q_1 p_0 + q_2 p_3 - q_3 p_2 \\ q_0 p_2 + q_2 p_0 + q_3 p_1 - q_1 p_3 \\ q_0 p_3 + q_3 p_0 + q_1 p_2 - q_2 p_1 \end{bmatrix}$$

Quaternion Summary . . .

- **Rotation Correspondence.** The unit quaternions q and $-q$ correspond to a single 3D rotation R_3:

$$\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

If

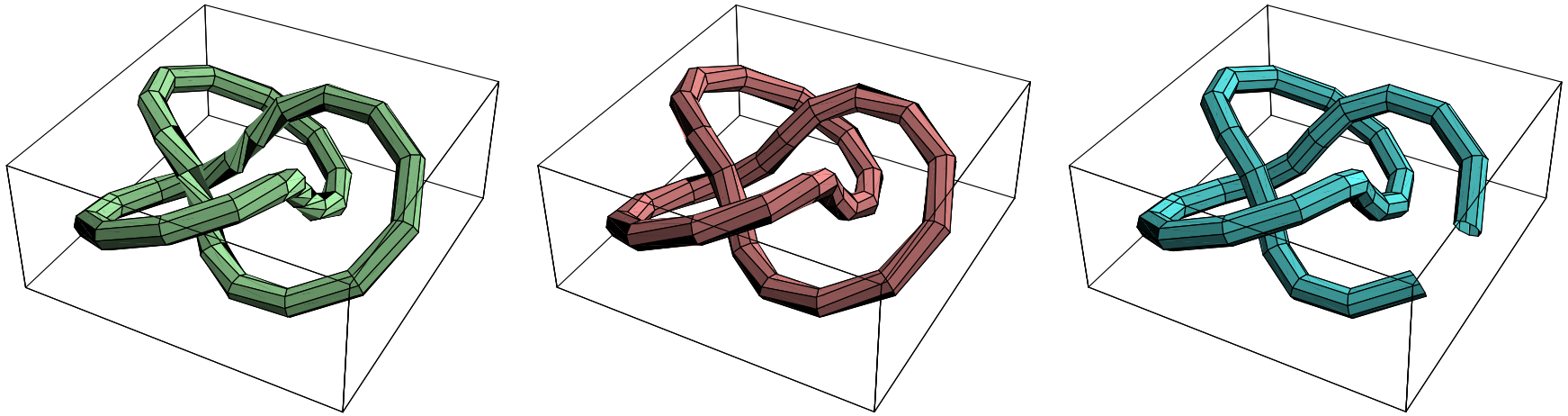
$$q = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2}) \ ,$$

with $\hat{\mathbf{n}}$ a unit 3-vector, $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$. Then $R(\theta, \hat{\mathbf{n}})$ is usual 3D rotation by θ in the plane $\perp$ to $\hat{\mathbf{n}}$.

Quaternion Fields

- **Quaternion Curves:** generalize the Frenet Frame, optimize in quaternion space
- **Quaternion Surfaces:** generalize Gauss map, optimize in quaternion space
- **Quaternion Volumes:** visualize degrees of freedom of a joint

Sample Curve Tubings and their Frames

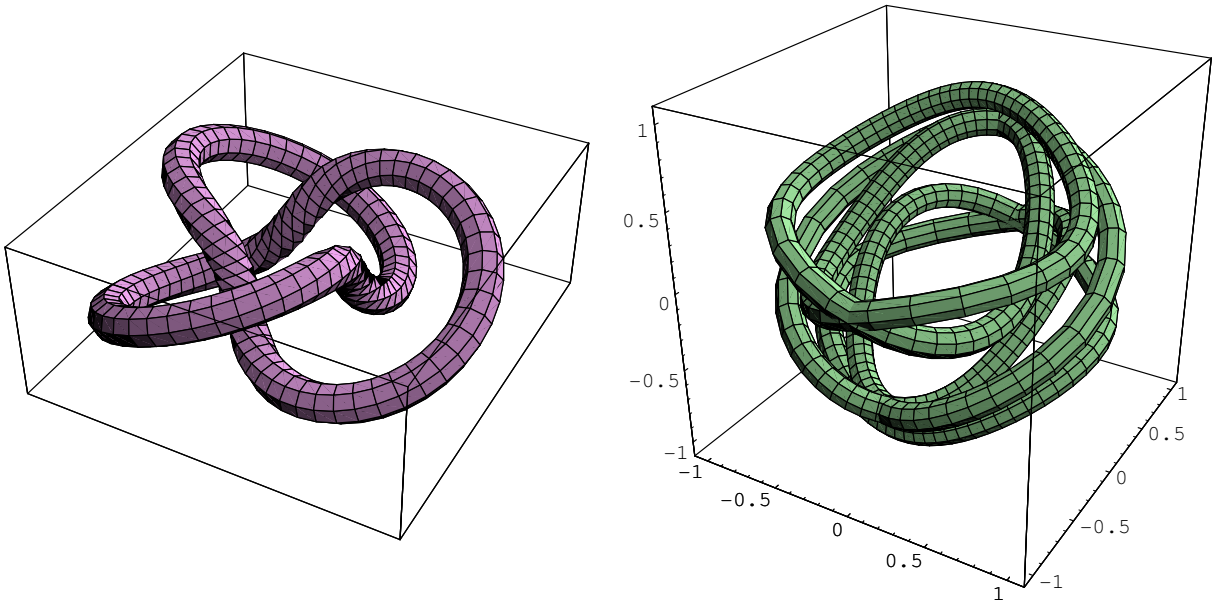


Frenet Geodesic Reference Parallel Transport

Easily see PT has least “Twist,” but lacks periodicity.

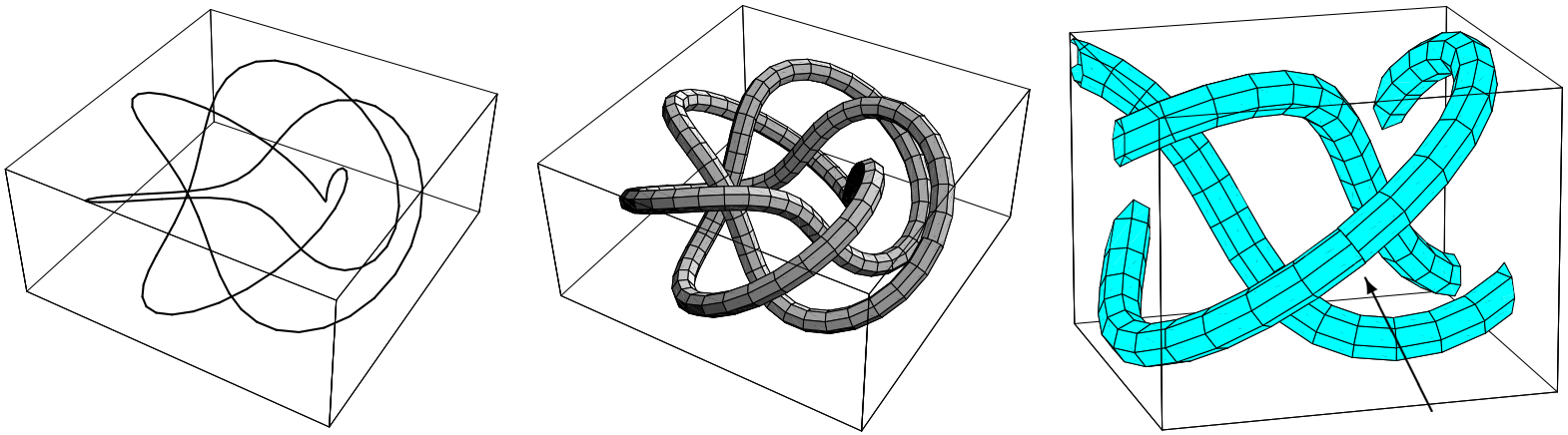
Example of a Quaternion Frame Curve

Left Curve = torus knot tubed with Frenet frame; Right Curve is projection from 4D of (twice around) quaternion Frenet frames:



see Notes: Hanson and Ma, "Quaternion Frame Approach to Streamline Visualization," *IEEE Trans. on Visualiz. and Comp. Graphics*, **1**, No. 2, pp. 164–174 (June, 1995).

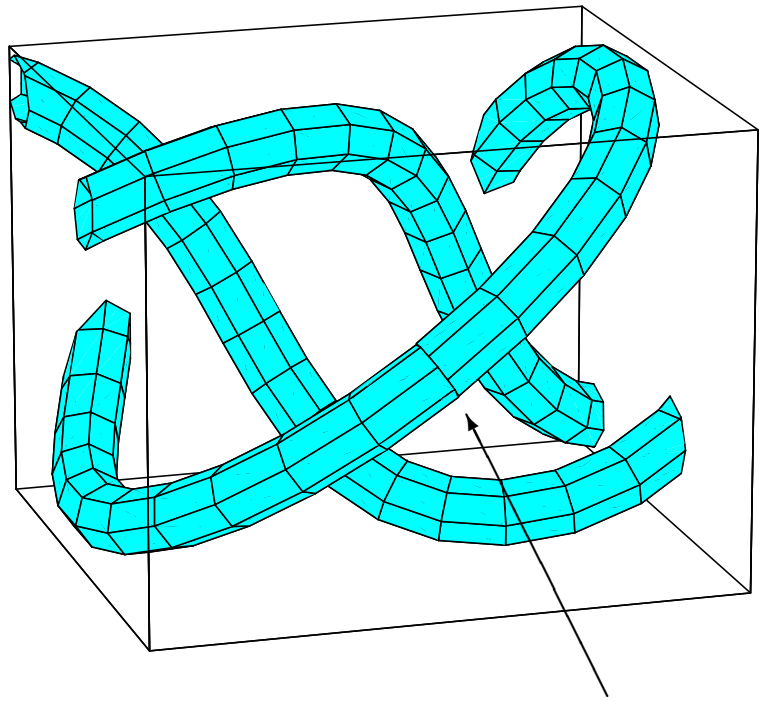
Motivating Problem: Framing Curves



The (3,5) torus knot.

- Line drawing $\approx$ useless.
- Tubing based on parallel transport, **not periodic.**
- Closeup of the non-periodic mismatch.

Motivating Problems: Curves

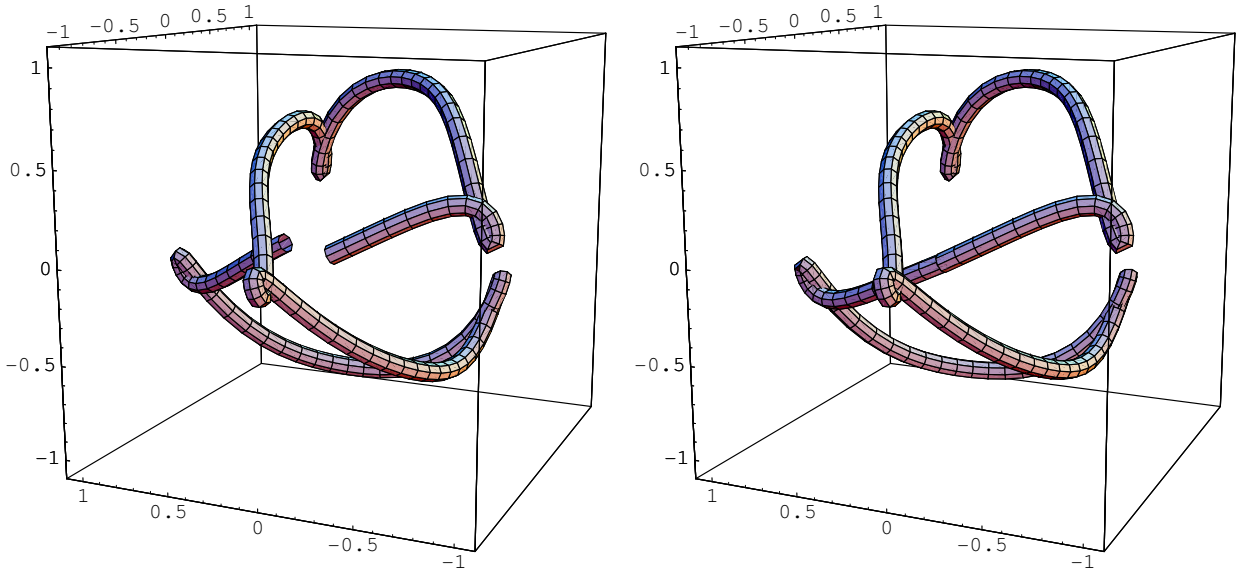


Closeup of the non-periodic mismatch.

Can't apply texture.

Minimizing Quaternion Length Solves Periodic Tube

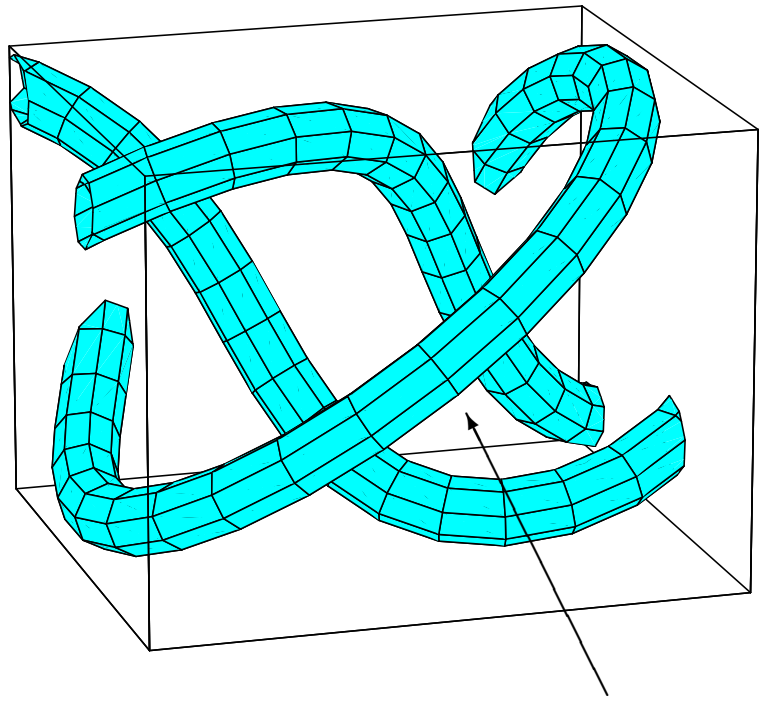
Quaternion space optimization of the non-periodic parallel transport frame of the (3,5) torus knot.



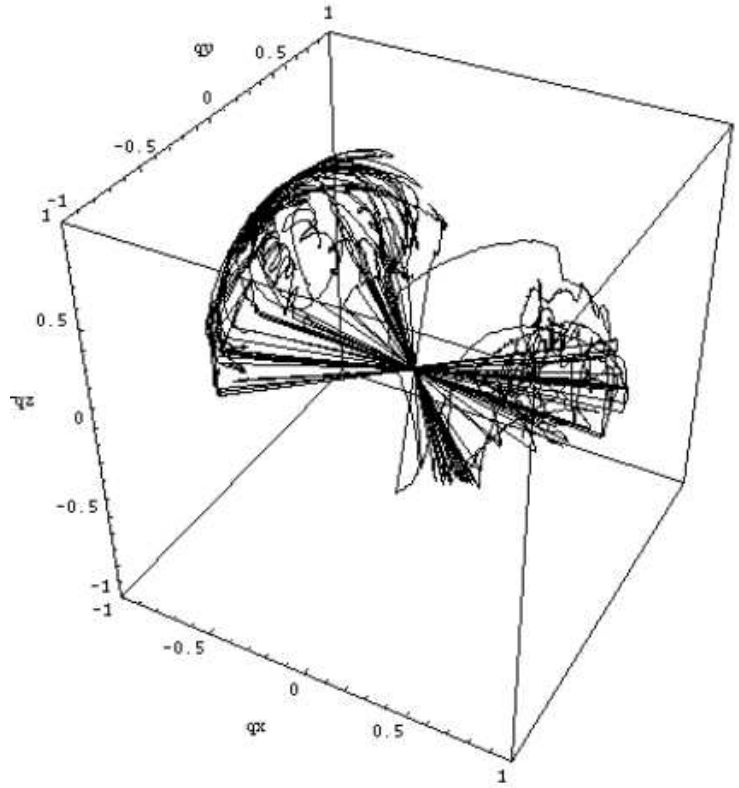
see Notes: "Constrained Optimal Framings of Curves and Surfaces using Quaternion Gauss Maps," *Proceedings of Visualization '98*, pp. 375–382 (1998).

Minimizing Quaternion Length Works

Result of Quaternion space optimization of the (3,5) torus knot frame.

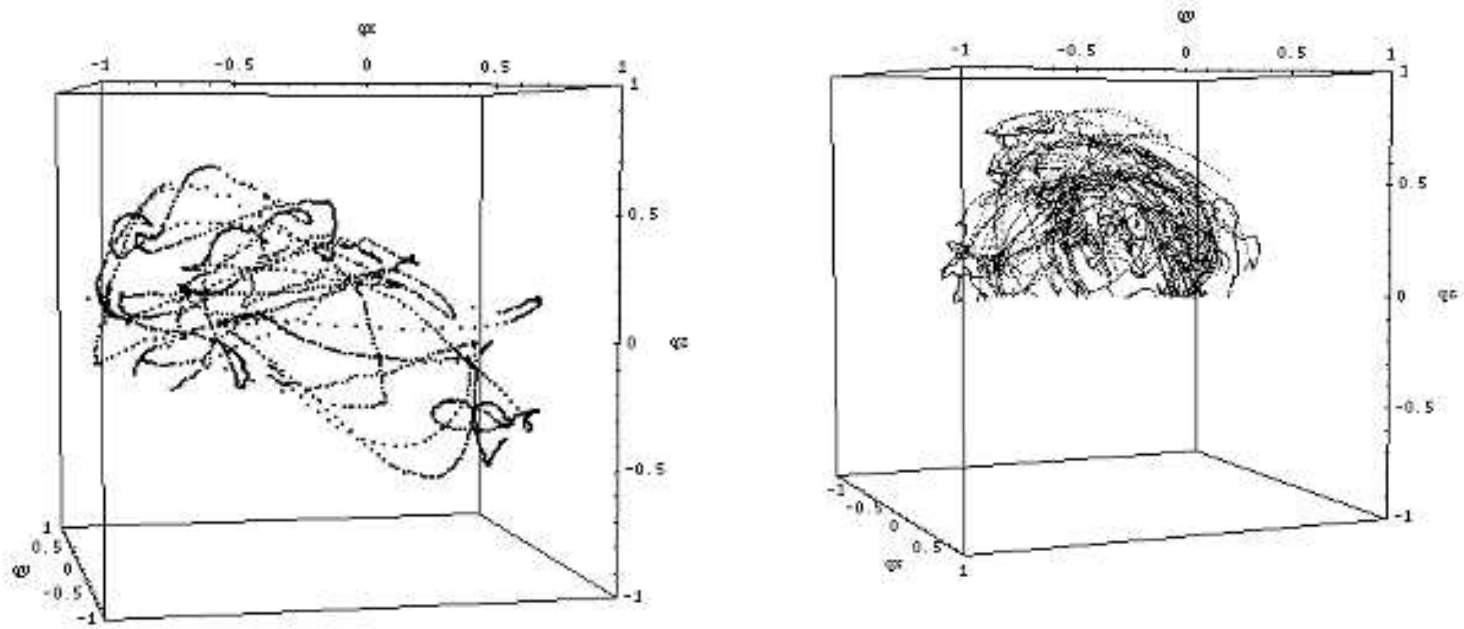


Quaternion volumes: Shoulder data



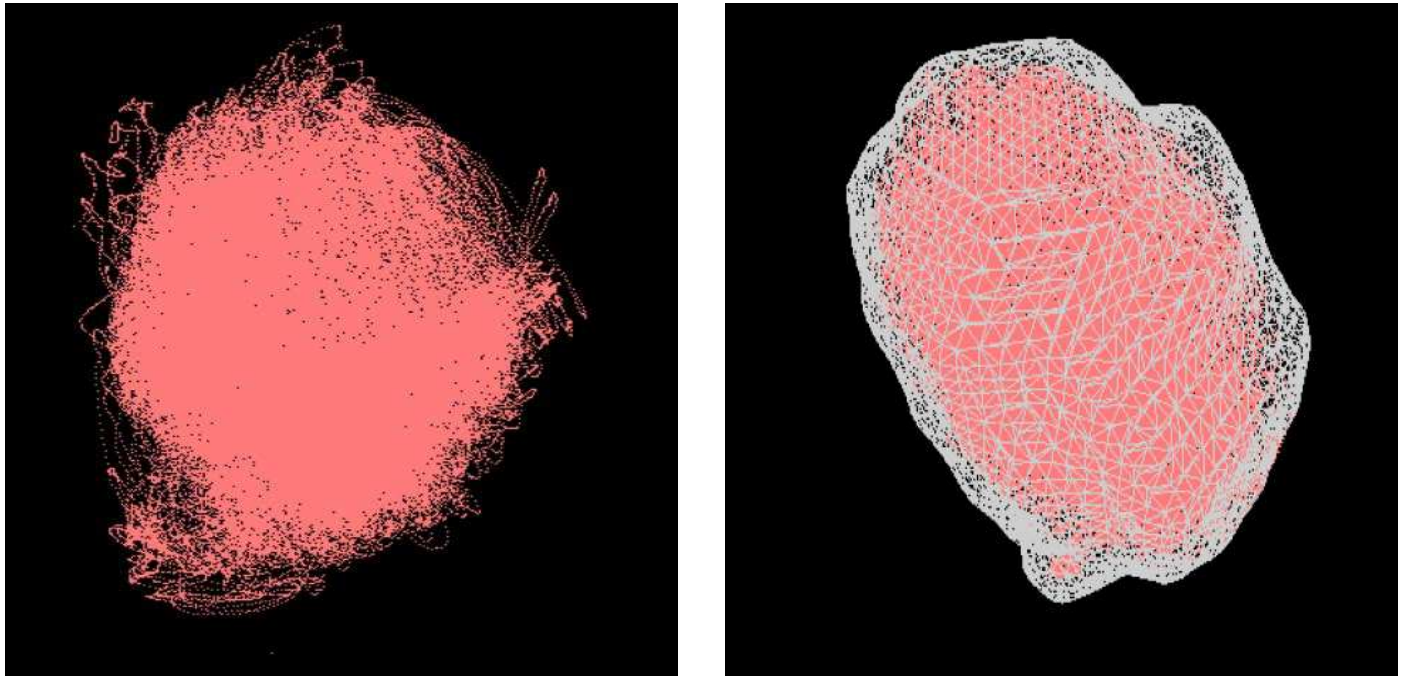
Quaternion shoulder joint data before correction for doubling.

Quaternion volumes: Shoulder data



Shoulder data with neighbors forced to be in same hemisphere of quaternion space as their predecessors.

Quaternion volumes: Shoulder data



(a) (b)

(a) A dense sample of shoulder orientation data in quaternion space.

(b) Implicit surface model fitted to the data. (Herda et al.)

SUMMARY

- Complex numbers represent 2D frames, 2D rotation
- Quaternions represent 3D frames, 3D rotation
- Quaternions are points on a hypersphere
- Quaternions paths can be visualized with 3D display
- Belt Trick, Rolling Ball, and Gimbal Lock can be understood as Quaternion Paths.
- Any collection of 3D frames can be *visualized* as a quaternion field.