Aalto CS-C3100 Computer Graphics

6.2 Quaternions

...or, adventures on the 4D unit sphere

Jaakko Lehtinen

with lots of slides from Frédo Durand

Wikipedia user Bluffink

Video on YouTube

- Watch the fantastic video by Grant Sanderson (3Blue1Brown)

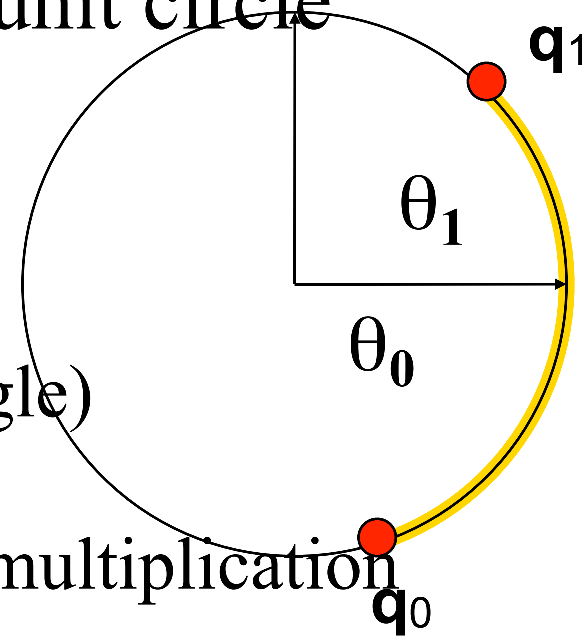
- These slides are only for your reference!

In These Slides

- Quaternions
  - Warmup: 2D rotations and complex numbers
  - Spherical linear interpolation (slerp)
  - Representing rotations using quaternions

1D Sphere and Complex Plane

- Represent 2D rotation by point on unit circle
  - 2 coordinates but only 1 DOF
- Let's take the 2D plane to be the complex plane
  - Orientation = complex argument (angle)
  - Unit circle = complex magnitude is 1
    composition of rotation ⇔ complex multiplication
  - Trivial with exponential notation $re^{i\theta}$



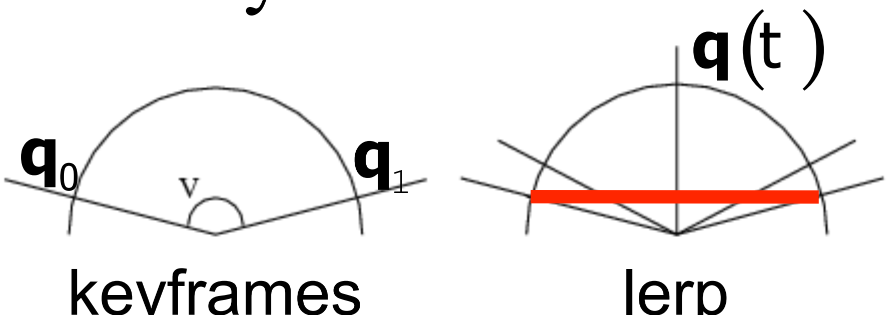
- Remember homogeneous coordinates? Adding a dimension can make life easier.
- Interpolation of angle is easy: Just slide the point along the circle.

Velocity Issue: lerp vs. slerp

- Linear Interpolation (lerp) between the 2D points interpolates the straight line between the two orientations

$$\mathrm{lerp}\left(\mathbf{q}_0, \mathbf{q}_1, t\right) = \mathbf{q}(t) = \mathbf{q}_0\left(1 - t\right) + \mathbf{q}_1 t$$

- Renormalize q(t) to lie on the circle again
- → lerp motion does not have uniform angular velocity

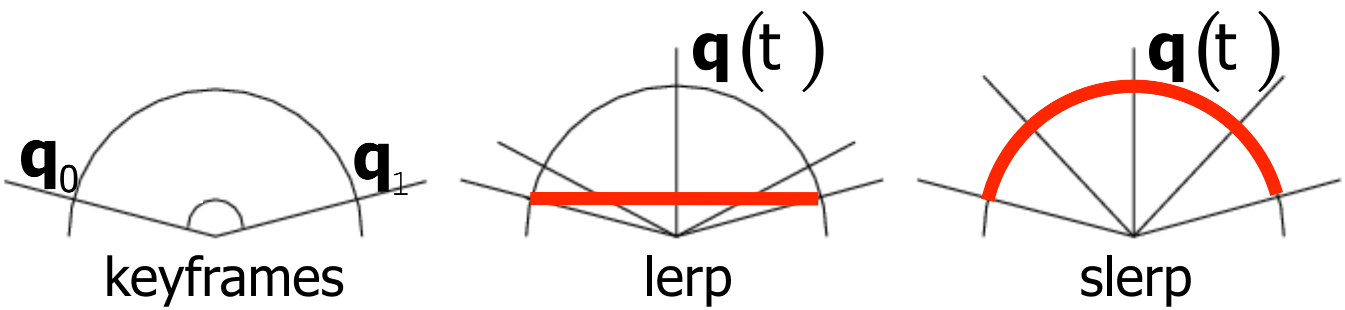


Velocity Issue: lerp vs. slerp

- Spherical Linear Interpolation (slerp) interpolates along the arc lines by adding a sine term:

$$\mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin((1-t)\omega) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)},$$

  where ω is the angle between q0 and q1
- We still interpolate in 2D plane, but along an arc
- Silly to make things so complex in 2D, but will be critical in 3D



interpolate along arc line rather than secant

Velocity Issue: lerp vs. slerp

Brain teasers

Can you prove that...

1) slerp produces a constant-speed curve?

2) the result is always a unit vector when $\mathbf{q}_0$ and $\mathbf{q}_1$ are unit vectors?

(Hint for 1: Differentiate w.r.t. *t,* take magnitude, trig identities
General hints: trig identities, $\mathbf{q}_0$ and $\mathbf{q}_1$ are unit, definition of ω)

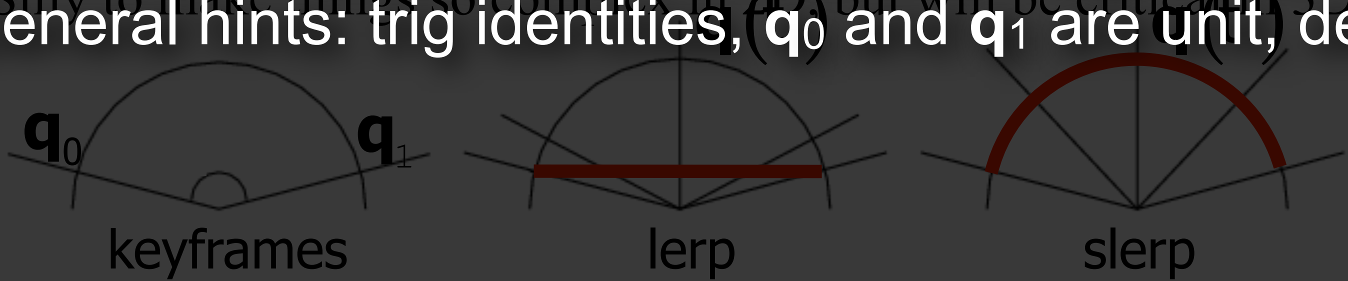
- Linear Interpolation (lerp) ... interpolates the straight line between the two orientations

$$\mathrm{lerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \mathbf{q}_0(1-t) + \mathbf{q}_1 t$$

- → lerp motion does not have uniform angular velocity
- Spherical Linear interpolation (slerp) interpolates along the arc lines by adding a sine term

$$\mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin((1-t)\omega) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)}$$

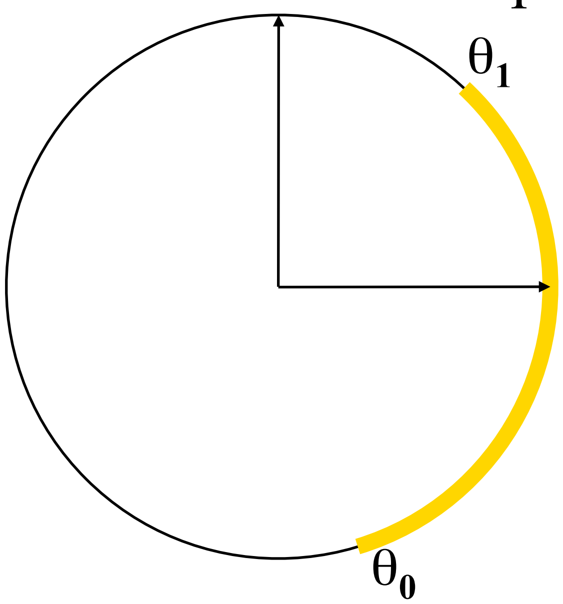
- where ω is the angle between q_0 and q_1
- We still interpolate in 2D plane at unit speed, but along an arc
- Silly to make things so complex in 2D, but will be critical in 3D



interpolate along arc line rather than secant

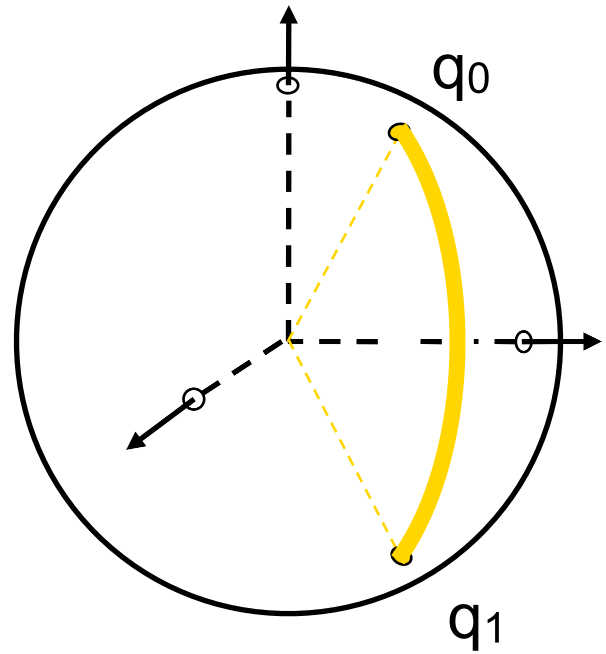
Questions?

- Recap
  - plane rotation in 2D: a point on unit circle
    - complex number interpretation
  - use slerp for uniform speed
    - works on the sphere in any dimension



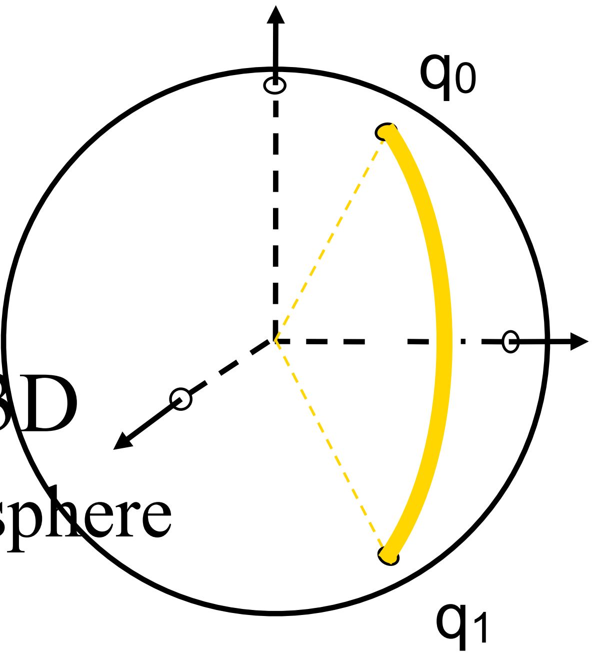
2-DOF Orientation

- Can represent by 2 angles
  - But this is messy because modulo 2π and pole...



(2-DOF Orientation)

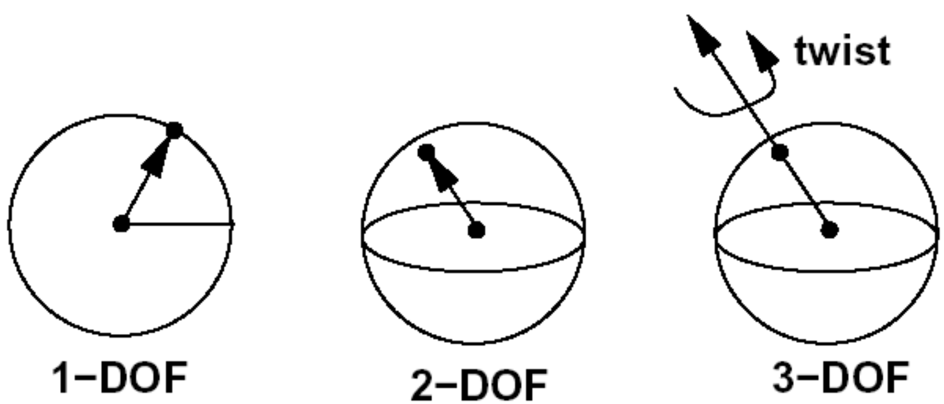
- Can represent by 2 angles
  - But this is messy because modulo 2π and pole...
- Solution: Embed 2-sphere in 3D
  - Interpolate 3D points on the 2-sphere along great circles
  - When done interpolating, convert the point back to angles



- Use slerp for uniform velocity & to stay on sphere
  - Note that it's still a 1D problem along the great circle
  - $\mathbf{q}_0$ and $\mathbf{q}_1$ are now 3D points

3 DOF – Quaternions!

- Use the same principle
  - interpolate on higher-dimensional sphere
  - use slerp formula to get uniform angular velocity, stay on 3-sphere
- 3-sphere embedded in 4D
  - More complex, harder to visualize
  - A point on 3-sphere corresponds to an 3D orientation



Quaternions: Hypercomplex Numbers

- Due to Hamilton (1843)
- Can be defined like complex numbers but with 4 coordinates
  - d+ai+bj+ck
  - One real part (d), three imaginary ones.

- Based on three different roots of -1:
  - $i^2 = j^2 = k^2 = -1$
  - and weird multiplication rules
    - $ij = k = -ji$
    - $jk = i = -kj$
    - $ki = j = -ik$

http://en.wikipedia.org/wiki/William_Rowan_Hamilton

Quaternions: Hypercomplex Numbers

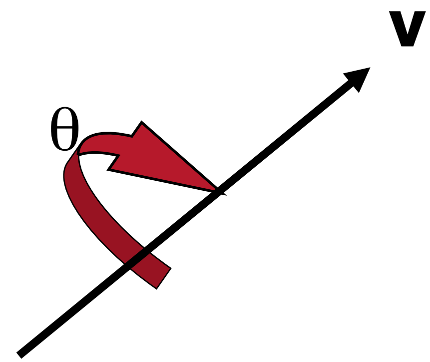
- Due to Hamilton (1843)
- Can be defined like complex numbers but with 4 coordinates
  - d+ai+bj+ck
  - One real part (d), three imaginary ones.

- Or defined with an imaginary part **v** that is a 3D vector:
  - (s, **v**)
  - simpler notation

http://en.wikipedia.org/wiki/William_Rowan_Hamilton

Quaternions: Rotation

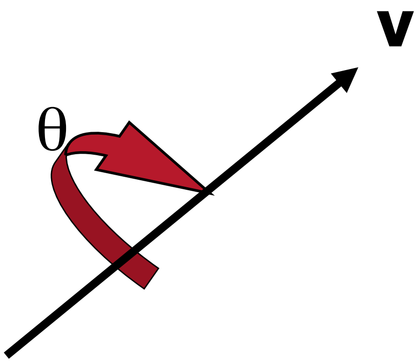
- Rotations represented by **unit** vectors in 4D
  - Right-hand rotation of θ radians about **v**:
    $\mathbf{q} = (\cos(\theta/2); \mathbf{v} \sin(\theta/2))$,



- Notes
  - **unit** quaternions are restricted to the unit 3-sphere in 4D (by definition of the unit sphere)
  - **q** & **-q** represent the same orientation
    - Why? (Hint: Graphs of sine and cosine, what happens to angle when axis flips if rotation is to remain same?)
  - Resembles axis-angle, but with the sines and cosines

Quaternions: Identity

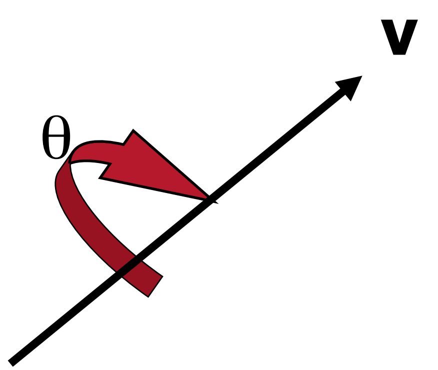
- Rotations represented by **unit** vectors in 4D
  - Right-hand rotation of θ radians about $\mathbf{v}$:
    $\mathbf{q} = (\cos(\theta/2);\ \mathbf{v}\sin(\theta/2))$,



- Identity orientation?

Quaternions: Identity

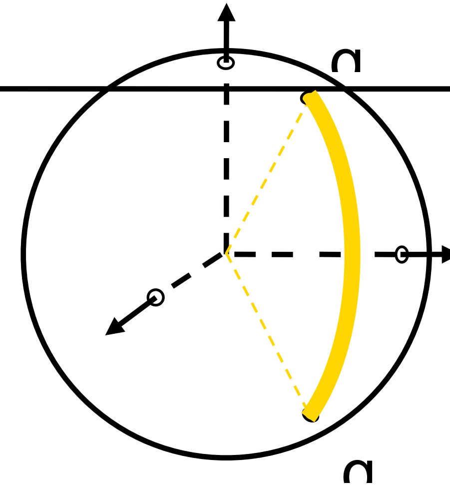
- Rotations represented by **unit** vectors in 4D
  - Right-hand rotation of θ radians about **v**:
    $\mathbf{q} = (\cos(\theta/2); \mathbf{v} \sin(\theta/2))$,

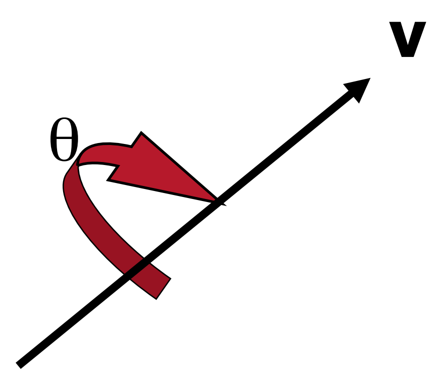


- Identity orientation?
  - θ is zero => scalar part = 1
  - Axis can be arbitrary, but since we want a unit quaternion => $\mathbf{q} = (1, \mathbf{0})$
  - BUT: Can also take $\mathbf{q} = (-1, \mathbf{0})$
    - **q** & **-q** represent the same rotation, remember

Question?

- Recap:
  - Rotation in 2D embedded on unit circle
    - complex number interpretation
    - slerp for uniform speed
      - works on the sphere in any dimension
  - Quaternions
    - 4D extension of complex numbers
    - rotations = unit quaternions (on 3-sphere)
    - $(\cos(\theta/2); \mathbf{v}\sin(\theta/2))$: rotation of θ around $\mathbf{v}$





Interpolating Rotations

- Given two unit quaternions, we want to interpolate
- Use slerp!
  - Works on the sphere in any dimension

$$\mathrm{slerp}\left(\mathbf{q}_0,\mathbf{q}_1,t\right)=\mathbf{q}(t)=\frac{\mathbf{q}_0\sin\left((1-t)\omega\right)+\mathbf{q}_1\sin\left(t\omega\right)}{\sin\left(\omega\right)},$$

  - Where ω is still the angle between $\mathbf{q}_0$ and $\mathbf{q}_1$ like in 2D
  - Note: This is again a linear combination of $\mathbf{q}_0$ and $\mathbf{q}_1$

Linear Combination of

- Just like vectors, just like complex numbers!

- Addition: Componentwise
  - (s, **v**) + (s', **v**') = (s+s', **v**+**v**')

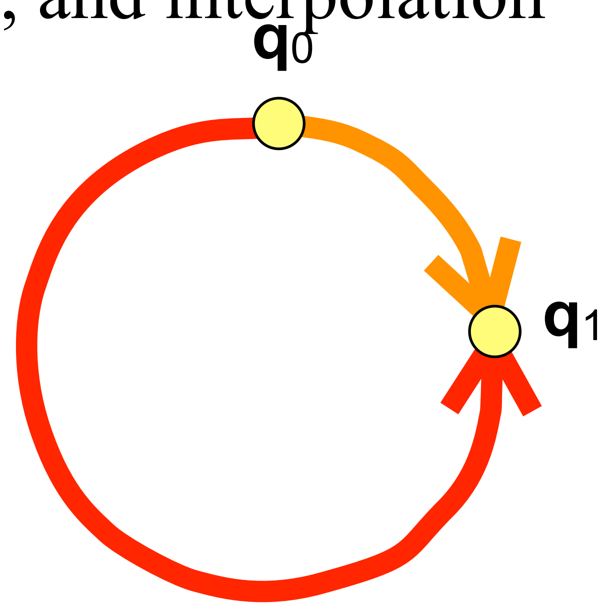
- Multiplication by scalar
  - t(s,**v**)=(ts, t**v**)

$$\mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin((1-t)\omega) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)},$$

You Might Need To Invert q

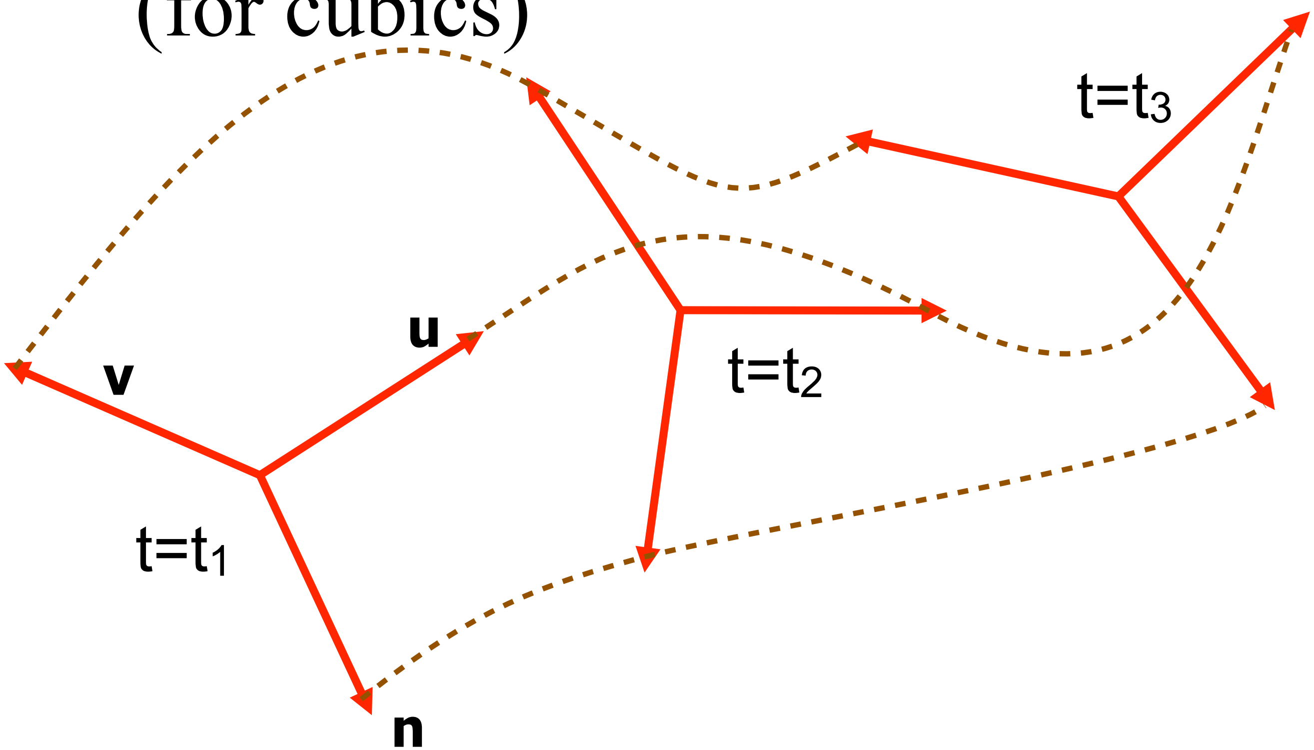
- Recall: **q** & **-q** represent the same rotation

- Given $\mathbf{q}_0$ and $\mathbf{q}_1$, test the angle (in 4D!)
  - If dot product of $\mathbf{q}_0$ and $\mathbf{q}_1$ is negative, they are on opposite sides of the hypersphere, and interpolation will take the longer route (red)
  - If this is the case, just use $-\mathbf{q}_1$ instead of $\mathbf{q}_1$



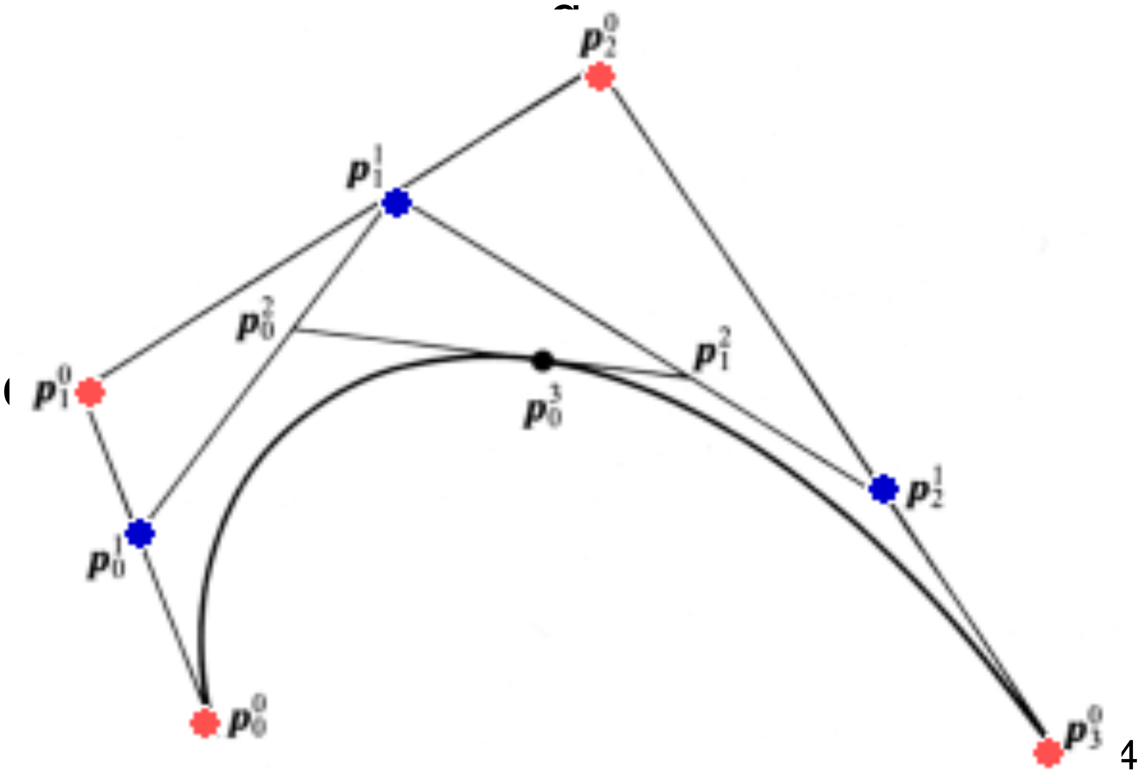
Problem with Splines

- Slerp only works to interpolate between **two** positions
- For splines, we need to blend more, typically 4 (for cubics)



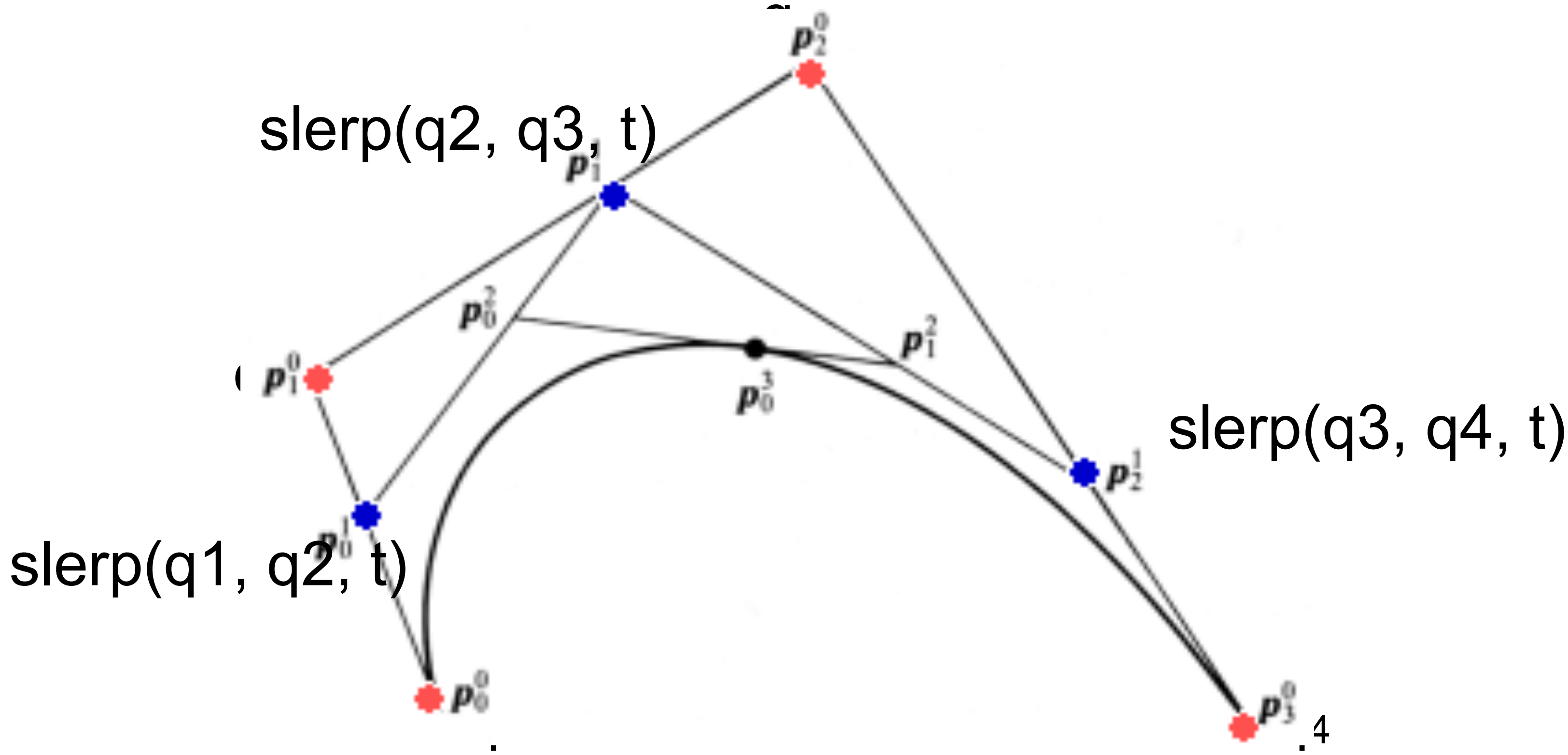
De Casteljau Construction w/ Slerp

- Remember what we did with cubic Bézier curves!
- Works to construct a point at any *t*
  - Only requires interpolation between pairs of points



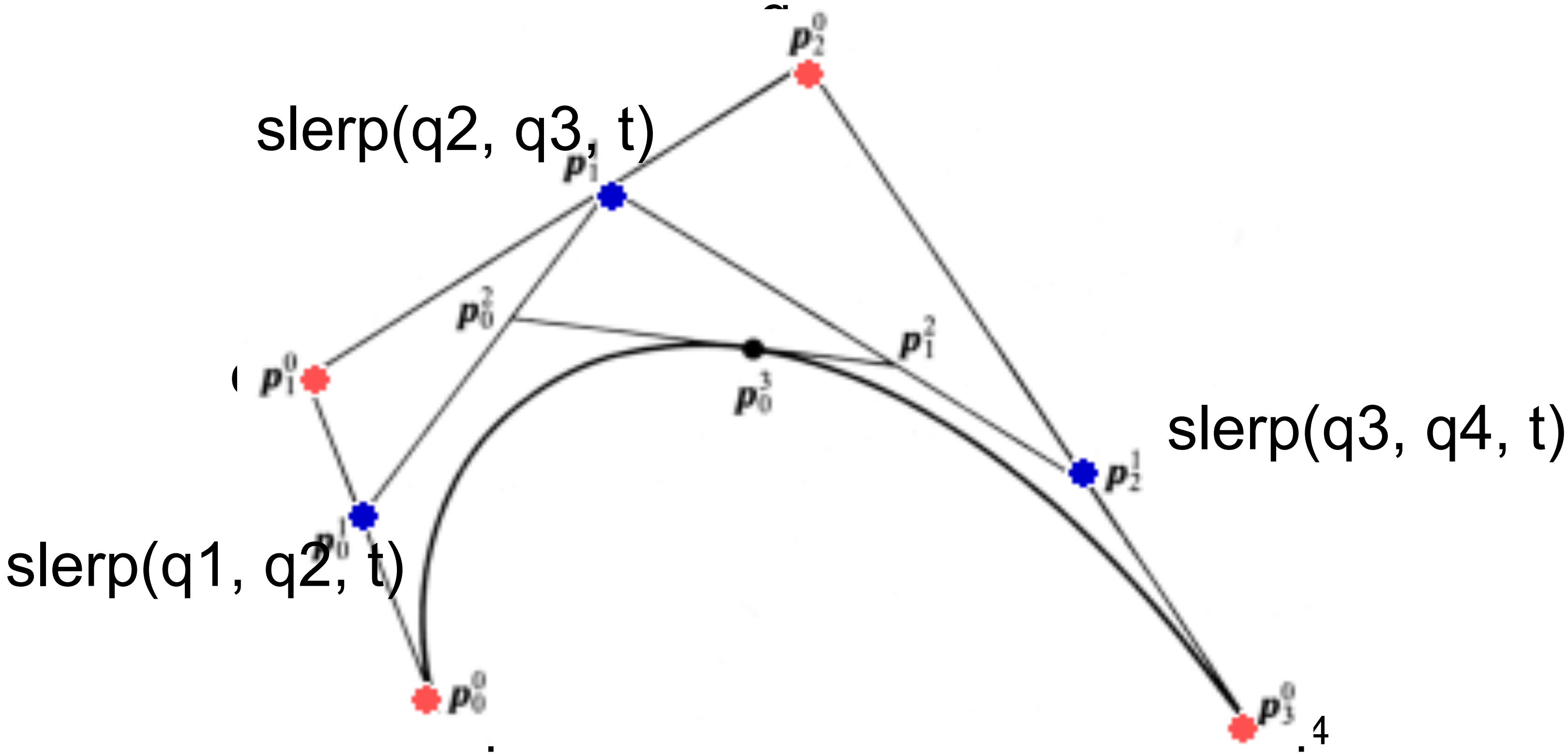
De Casteljau Construction w/ Slerp

- Remember what we did with cubic Bézier curves!
- Works to construct a point at any *t*
  - Only requires interpolation between pairs of points



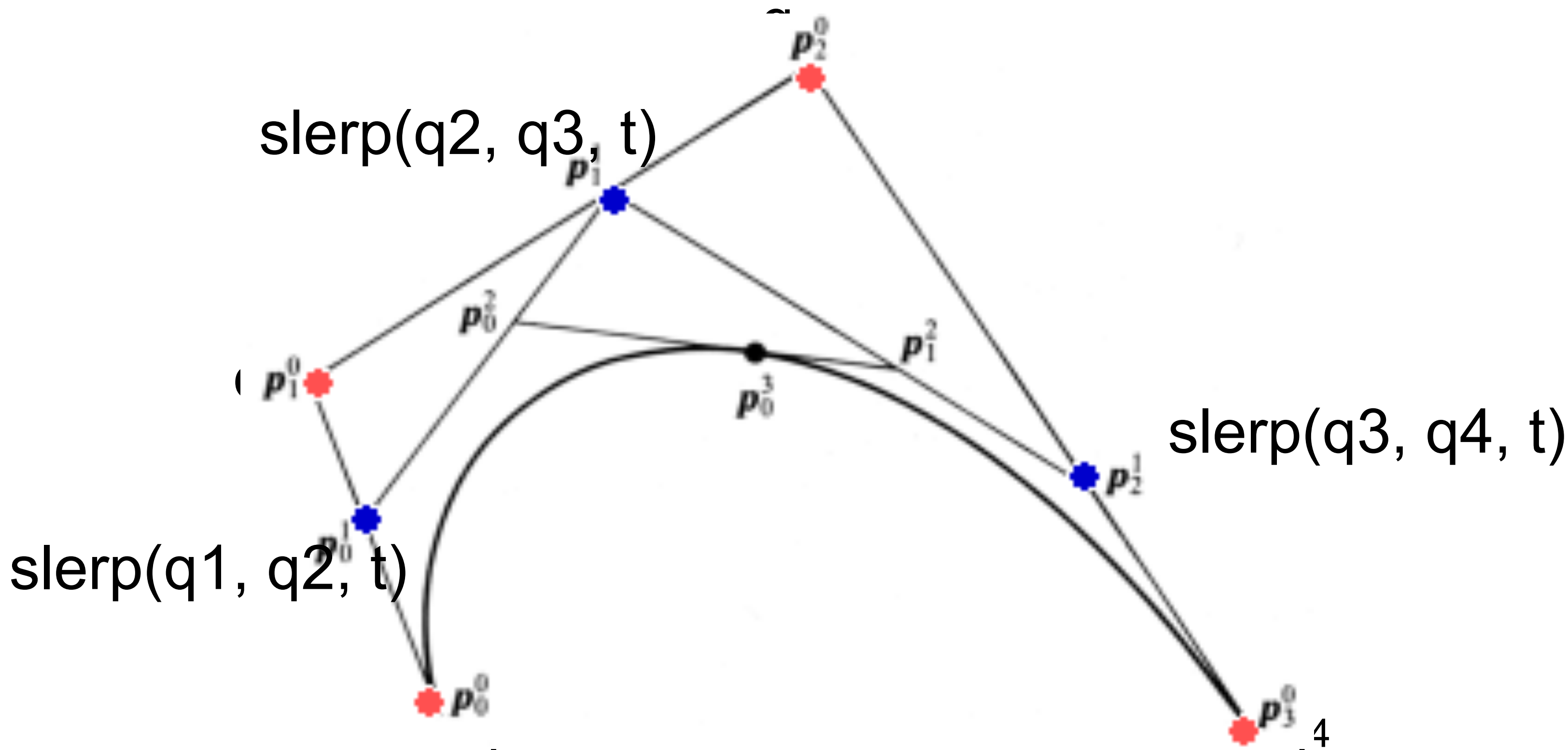
De Casteljau Construction w/ Slerp

- Remember what we did with cubic Bézier curves!
- Works to construct a point at any *t*
  - Only requires interpolation between pairs of points



De Casteljau Construction w/ Slerp

- Remember what we did with cubic Bézier curves!
- Works to construct a point at any *t*
  - Only requires interpolation between pairs of points



This is an easy-ish extra in Assignment 2!

Extensions

- Better interpolation
  - E.g. minimize acceleration, velocity constraint
  - http://www.gg.caltech.edu/STC/rr_sig97.html
  - http://portal.acm.org/citation.cfm?id=218486&dl=ACM&coll=portal&CFID=1729050&CFTOKEN=74418864
  - http://portal.acm.org/citation.cfm?id=134086&dl=ACM&coll=portal&CFID=1729050&CFTOKEN=74418864

From Kim et al. 1995

Cookbook Recipe

- You need matrices to draw (e.g. OpenGL)
- General approach for 3 DOF rotations
  - Store keyframe orientations as quaternions
  - Interpolate between them using slerp (pairwise) or slerp + De Casteljau (splines)
  - Convert to quaternion to matrix
  - Profit.
  - (Or, store matrices, convert to quaternions for interpolation, then convert back.)

Cookbook Recipe

- You need matrices to draw (e.g. OpenGL)

- General approach for 3 DOF rotations
  - Store keyframe orientations as quaternions
  - Interpolate between them using slerp (pairwise) or slerp + De Casteljau (splines)
  - Convert to quaternion to matrix
  - Profit.
- Often need to convert from matrix to quaternion.
  - **Next**: Conversion to/from matrices.

Quaternion to Rotation Matrix

- Quaternion (q0, q1, q2, q3) corresponds to matrix

$$\begin{pmatrix} 1-2q_2^2-2q_3^2 & 2(q_1q_2-q_3q_0) & 2(q_1q_3+q_2q_0) \\ 2(q_1q_2+q_3q_0) & 1-2q_1^2-2q_3^2 & 2(q_2q_3-q_1q_0) \\ 2(q_1q_3-q_2q_0) & 2(q_1q_0+q_2q_3) & 1-2q_1^2-2q_2^2 \end{pmatrix}$$

- Similar to Rodrigues' rotation formula
  - but recall that quaternions use $\theta/2$
- After conversion, you can combine rotations and other affine/projective transforms!

3x3 Orthonormal Matrix to Quaternion

- More challenging (e.g., not all **M**s are rotations)
- if **M** is a rotation, trace(**M**)>0
  then you get quaternion (s, x, y, z) through:
  - $s = \mathrm{sqrt}(1 + M_{11} + M_{22} + M_{33}) / 2$
  - $x = (M_{23} - M_{32}) / (4 * s)$
  - $y = (M_{31} - M_{13}) / (4 * s)$
  - $z = (M_{12} - M_{21}) / (4 * s)$
- if trace(**M**)<0, need permutations/sign changes

General Conversion Resource

- http://en.wikipedia.org/wiki/Rotation_representation_%28mathematics%29

What about other transforms?

- What to do if the matrix to be interpolated does not only rotation, but scale, shear, etc.?

Non-orthonormal 3x3 matrix

- “Polar decomposition” breaks arbitrary matrix **M** into
  - Rotation **Q** (+potential reflection)
  - Symmetric positive definite **S** (anisotropic scale)

Figure 4. Direct Shear Interpolation

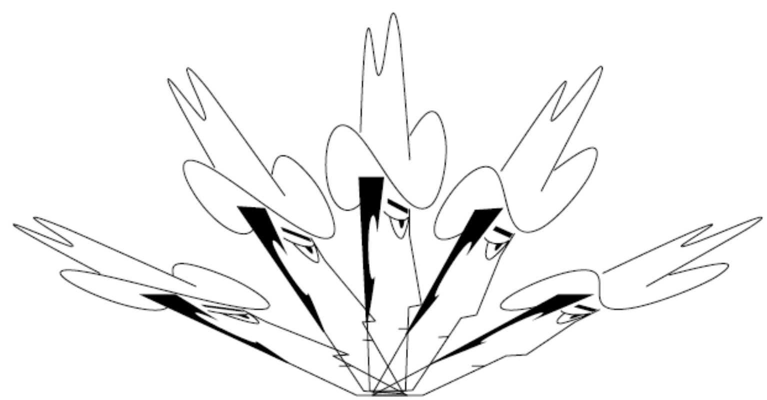


Figure 5. Decomposed Shear Interpolation

Polar Decomposition Algorithm

- Given 3x3 Matrix $\mathbf{M}$
- Compute the rotation factor $\mathbf{Q}$ by averaging the matrix with its inverse transpose until convergence:
  - Set $\mathbf{Q}_0 = \mathbf{M}$,
  - then $\mathbf{Q}_{i+1} = 1/2(\mathbf{Q}_i + \mathbf{Q}_i^{-T})$ until $\mathbf{Q}_{i+1} - \mathbf{Q}_i \approx 0$.

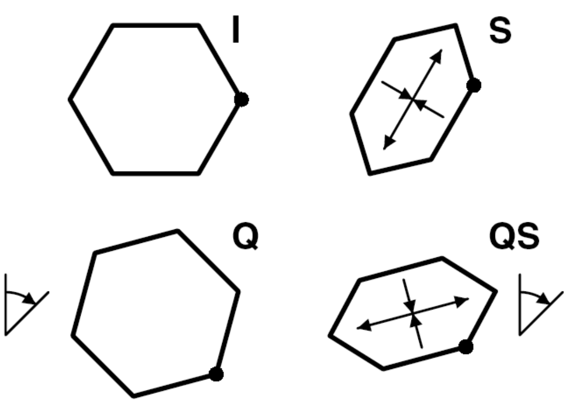


Figure 3. Physical View of Polar Decomposition

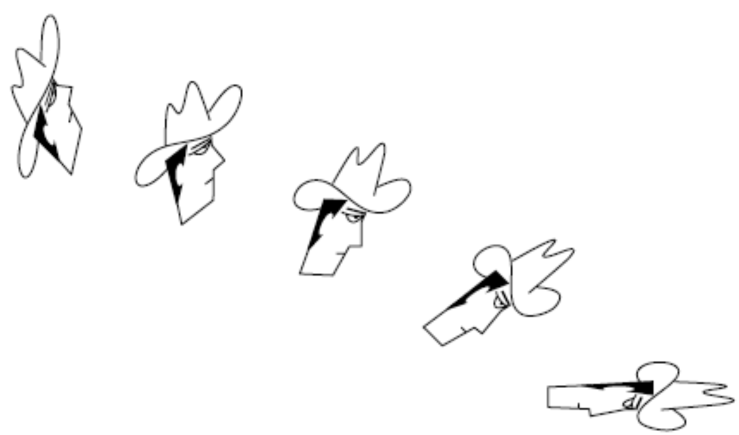


Figure 6. Polar Decomposed Matrix Interpolation

More Quaternion Magic: Multiplication

- Turns out that quaternion multiplication corresponds to composing rotations
  - $\mathbf{q}_2 = \mathbf{q}_1\mathbf{q}_0$ is equivalent to first rotating by $\mathbf{q}_0$, then $\mathbf{q}_1$.

$$(\theta; \boldsymbol{v})(\theta'; \boldsymbol{v}') =$$
$$(\theta\theta' - \boldsymbol{v} \cdot \boldsymbol{v}';\ \theta\boldsymbol{v}' + \theta'\boldsymbol{v} + \boldsymbol{v} \times \boldsymbol{v}')$$

- Multiplication is **not commutative** (why? cross product)
  - $\mathbf{q}_1\mathbf{q}_0$ does not equal $\mathbf{q}_0\mathbf{q}_1$ except in special cases
  - Makes sense, rotations are not commutative either

Even More Quaternion Magic

- Let's define a conjugate $\mathbf{q}^* = (\theta, -\mathbf{v})$
  - Remember complex conjugate? $a = x + iy$, $a^* = x - iy$
- Is there an inverse quaternion $\mathbf{q}^{-1}$ such that $\mathbf{q}\mathbf{q}^{-1}=(1; \mathbf{0})$ for unit $\mathbf{q}$? Let's try the conjugate...
  - Again, compare to complex: $aa^* = x^2+y^2 = 1$ when a is unit length.

$$
(\theta; \boldsymbol{v})(\theta'; \boldsymbol{v}') = \\
(\theta\theta' - \boldsymbol{v} \cdot \boldsymbol{v}';\ \theta\boldsymbol{v}' + \theta'\boldsymbol{v} + \boldsymbol{v} \times \boldsymbol{v}')
$$

Conjugate = Inverse for Unit Q's

- Let's define a conjugate **q*** = (θ, -**v**)
  - Remember complex conjugate? a = x + iy, a* = x - iy
- Let's see:

$$\boldsymbol{q}\boldsymbol{q}^* =$$

$$(\theta^2 + \boldsymbol{v}\cdot\boldsymbol{v};\ \theta\boldsymbol{v} - \theta\boldsymbol{v} + \boldsymbol{v}\times\boldsymbol{v}) = (1;\mathbf{0})$$

  - Note that this only works for unit **q**. If not unit, need normalization factor.

Conjugate = Inverse for Unit Q's

- Let's define a conjugate **q*** = (θ, **-v**)
  - Remember complex conjugate? a = x + iy, a* = x - iy
- Let's see:

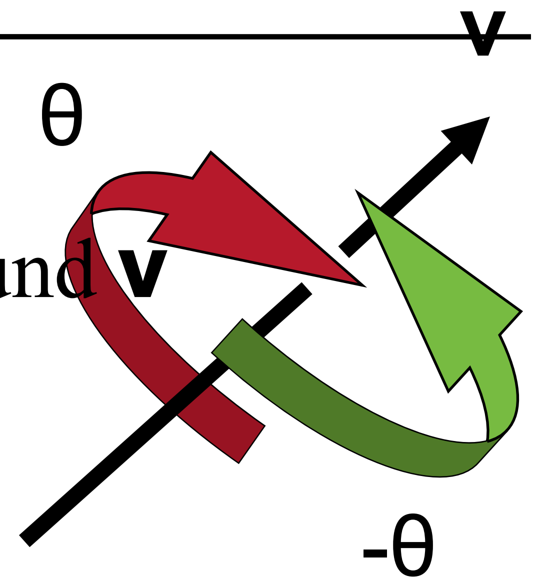
$$\boldsymbol{q}\boldsymbol{q}^* =$$

q* = q^{-1} for unit quaternions

$$(\theta^2 + \boldsymbol{v} \cdot \boldsymbol{v};\ \theta\boldsymbol{v} - \theta\boldsymbol{v} + \boldsymbol{v} \times \boldsymbol{v}) = (1; \mathbf{0})$$

  - Note that this only works for unit **q**. If not unit, need normalization factor.

Inverse & Conjugate: Geometry



- $\mathbf{q} = (\cos \theta/2; \sin\theta/2\ \mathbf{v})$
  represents a rotation of angle θ around $\mathbf{v}$

- Inverse rotation $\mathbf{q}^{-1}$:
  - Angle -θ around $\mathbf{v}$
  - Angle θ around $-\mathbf{v}$

- In both cases, leads to $(\cos \theta/2; -\sin\theta/2\ \mathbf{v})$
  - $\mathbf{q}^* = (\theta, -\mathbf{v})$, remember

Inverse & Conjugates: Matrices

- What is the inverse of a rotation matrix?

Inverse & Conjugates: Matrices

- What is the inverse of a rotation matrix?
- The conjugate/transpose matrix!
  - For a rotation (or any orthonormal matrix) $\mathbf{M}^T\mathbf{M}=\mathbf{I}$
    - (Formally, to get the conjugate of a complex-valued matrix, take the transpose and the conjugate of each coefficient. But we don't care here.)
- The notion of conjugation is related between matrices & quaternions
  - Isn't that cool?

Even More 4D Magic: Rotating a Point

- 3D vector **p** is represented by quaternion (0, **p**)
- To rotate 3D point/vector **p** by rotation/quaternion **q**, compute

$$\mathbf{q}\mathbf{p}\mathbf{q}^{-1} = \mathbf{q}(0; \mathbf{p})\mathbf{q}^{-1}$$

- (In practice, better convert the quaternion to a matrix first.)

That’s All Folks!