3D Kinematics

- Consists of two parts
  - 3D rotation
  - 3D translation
    - The same as 2D

- 3D rotation is more complicated than 2D rotation (restricted to z-axis)
- Next, we will discuss the treatment for spatial (3D) rotation

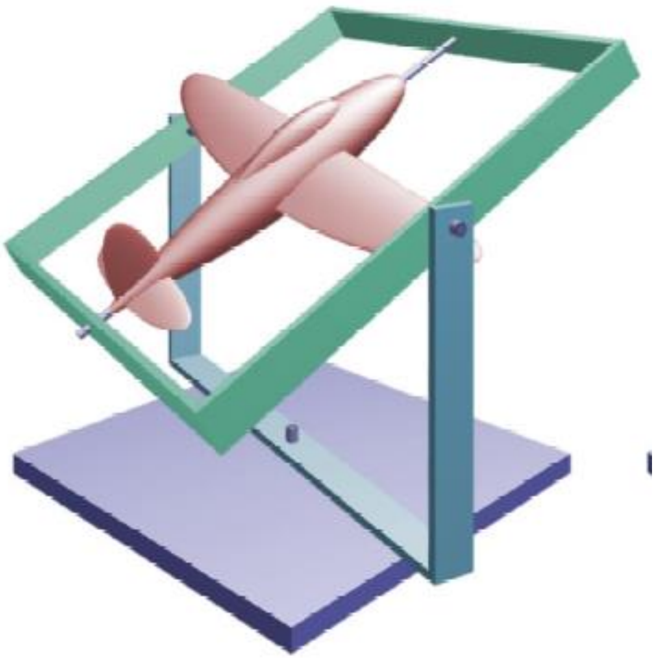
3D Rotation Representations

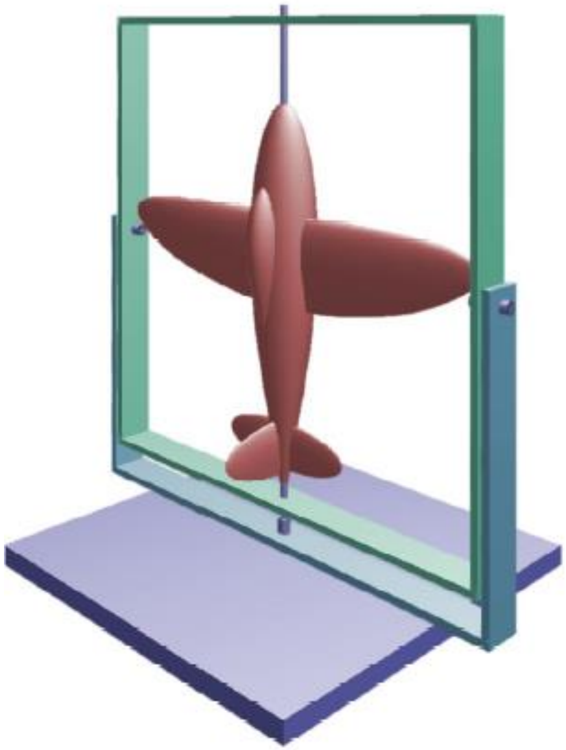
- Euler angles
- Axis-angle
- 3X3 rotation matrix
- Unit quaternion

- Learning Objectives
  - Representation (uniqueness)
  - Perform rotation
  - Composition
  - Interpolation
  - Conversion among representations
  - …

Euler Angles and **GIMBAL LOCK**

- Roll, pitch, yaw
- Gimbal lock: reduced DOF due to overlapping axes





Ref: http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

Axis-Angle Representation

Axis-Angle Representation

- $Rot(n,\theta)$
  - n: rotation axis (global)
  - θ: rotation angle (rad. or deg.)
  - follow right-handed rule
- $Rot(n,\theta)=Rot(-n,-\theta)$
- Problem with null rotation: rot(***n***,0), any ***n***
- Perform rotation
  - Rodrigues formula
- Interpolation/Composition: ***poor***
  - $Rot(n_2,\theta_2)Rot(n_1,\theta_1)$ =?= $Rot(n_3,\theta_3)$

We create matrix R for rotation

Quaternions

Quaternion - Brief History

- Invented in 1843 by Irish mathematician Sir William Rowan Hamilton
- Founded when attempting to extend complex numbers to the 3rd dimension
- Discovered on October 16 in the form of the equation:

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

From: http://en.wikipedia.org/wiki/Quaternion

Quaternion – Brief History

William Rowan Hamilton

Quaternion

- Definition

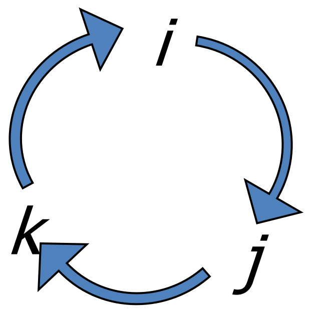
$$q = q_0 + q_1 i + q_2 j + q_3 k = q_0 + \vec{q}$$

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

$$ij = -ji = k$$

$$jk = -kj = i$$

$$ki = -ik = j$$



Applications of Quaternions

- Used to represent rotations and orientations of objects in three-dimensional space in:
  - Computer graphics
  - Control theory
  - Signal processing
  - Attitude controls
  - Physics
  - Orbital mechanics
  - Quantum Computing, quantum circuit design

From: http://en.wikipedia.org/wiki/Quaternion

Advantages of Quaternions

- Avoids *Gimbal Lock*
- Faster multiplication algorithms to combine successive rotations than using rotation matrices
- Easier to normalize than rotation matrices
- Interpolation
- Mathematically stable – suitable for statistics

Operators on Quaternions

- Operators
  - Addition

$$p = p_0 + p_1 i + p_2 j + p_3 k$$

$$q = q_0 + q_1 i + q_2 j + q_3 k$$

$$p + q \equiv (p_0 + q_0) + (p_1 + q_1)i + (p_2 + q_2)j + (p_3 + q_3)k$$

  - Multiplication

$$pq \equiv p_0 q_0 + p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q} - \vec{p} \cdot \vec{q}$$

  - Conjugate

$$q^* \equiv q_0 - \vec{q} \qquad (pq)^* \equiv q^* p^*$$

  - Length

$$|q| = \sqrt{q^* q} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

Unit Quaternion

- Define unit quaternion as follows to represent rotation

$$\mathrm{Rot}(\hat{n},\theta) \Rightarrow q = \cos\tfrac{\theta}{2} + \sin\tfrac{\theta}{2}\,\hat{n} \qquad |q| = 1$$

Why "unit"? DOF point of view!

- Example
  - Rot(z,90°)$\Rightarrow$ $\quad q = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}$
- q and –q represent the same rotation

Quaternion – **scalar** and **vector** parts

$$q = w + x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$$

- w, x, y, z are real numbers
- w is scalar part
- x, y, z are vector parts
- Thus it can also be represented as:

$$q = (w, \mathbf{v}(x,y,z)) \text{ or}$$

$$q = w + \mathbf{v}$$

Quaternion – Dimension and **Transformation**

- Scalar & Vector
- 4 dimensions of a quaternion:
  - 3-dimensional space (vector)
  - Angle of rotation (scalar)
- Quaternion can be transformed to other **geometric** algorithm:
  - Rotation matrix ↔ quaternion
  - Rotation axis and angle ↔ quaternion
  - Spherical rotation angles ↔ quaternion
  - Euler rotation angles ↔ quaternion

What are relations of quaternions to other topics in kinematics?

Details of Quaternion Operations

Quaternion Operations

- Addition/subtraction
- Multiplication
- Division
- Conjugate
- Magnitude
- Normalization
- Transformations
- Concatenation

Quaternion Operations

- Addition:
  - Given two quaternions:
    - $q_1 = q_1w + q_1xi + q_1yj + q_1zk$
    - $q_2 = q_2w + q_2xi + q_2yj + q_2zk$
  - The result quaternion q_3 is:

$$q_3 = q_1 + q_2$$

$$q_3 = (q_1w + q_2w) + (q_1x + q_2x)i + (q_1y + q_2y)j + (q_1z + q_2z)k$$

Quaternion Operations

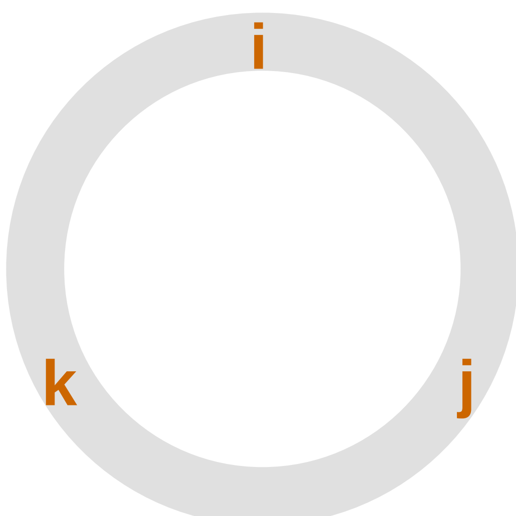
- Subtraction:
  - Given two quaternions:
    - $q_1 = q_1w + q_1xi + q_1yj + q_1zk$
    - $q_2 = q_2w + q_2xi + q_2yj + q_2zk$
  - The result quaternion q_3 is:

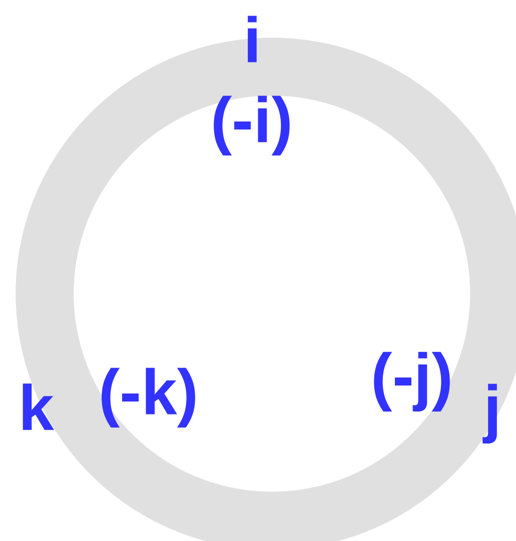
$$q_3 = q_1 - q_2$$

$$q_3 = (q_1w - q_2w) + (q_1x - q_2x)i + (q_1y - q_2y)j + (q_1z - q_2z)k$$

Quaternion Operations

- Multiplication
  - Distributive
  - Associative
  - Not commutative because of the $i^2 = j^2 = k^2 = -1$





Quaternion Operations

- Multiplication
  - Given two quaternions:
    - $q_1 = q_1w + q_1xi + q_1yj + q_1zk$
    - $q_2 = q_2w + q_2xi + q_2yj + q_2zk$
  - The result quaternion q_3 is:

$$q_3 = q_1 * q_2$$

$$q_3 = q_1w*q_2w + q_1w*q_2xi + q_1w*q_2yj + q_1w*q_2zk + q_1xi*q_2w + q_1xi*q_2xi + \text{etc...}$$

Quaternion Operations

- Multiplication
  - Resulting quaternion q_3 is:

$$
\begin{aligned}
q_3 = \; & (q_1w q_2w + q_1x q_2x + q_1y q_2y + q_1z q_2z) + \\
& (q_1w q_2x + q_1x q_2w + q_1y q_2z - q_1z q_2y)i + \\
& (q_1w q_2y + q_1y q_2w + q_1z q_2x - q_1x q_2z)j + \\
& (q_1w q_2z + q_1z q_2w + q_1x q_2y - q_1y q_2x)k
\end{aligned}
$$

Quaternion Operations

- Multiplication
  - Or, in scalar-vector format:

$$q_3 = q_1 q_2 = (q_1w, \mathbf{v}_1)(q_2w, \mathbf{v}_2)$$

$$= (q_1wq_2w - \mathbf{v}_1.\mathbf{v}_2,\ q_1w\mathbf{v}_2 + q_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$$

scalar — vector

Dot product — Cross product

or

$$q_3 = q_1wq_2w - \mathbf{v}_1.\mathbf{v}_2 + q_1w\mathbf{v}_2 + q_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2$$

Quaternion Operations

- Magnitude
  - Also called *modulus*
  - Is the length of the quaternion from the origin
  - Given a quaternion:
    - q = w + xi+ yj + zk
  - The magnitude of quaternion q is |q|, where:

$$|q| = \sqrt{qq^*} = \sqrt{w^2 + x^2 + y^2 + z^2}$$

Quaternion Operations

$$q$$

- Normalization
  - Normalization results in a *unit quaternion* where:

    $w^2 + x^2 + y^2 + z^2 = 1$
  - Given a quaternion:
    - $q = w + xi + yj + zk$
  - To normalize quaternion q, divide it by its magnitude (|q|):

$$\hat{q} = \frac{q}{|q|}$$

  - Also referred to as *quaternion sign*: sgn(q)

Quaternion Operations

- Conjugate:
  - Given a quaternion:
    - q = w + xi+ yj + zk
  - The conjugate of quaternion q is q*, where:
    - q* = w – xi – yj – zk

Quaternion Operations

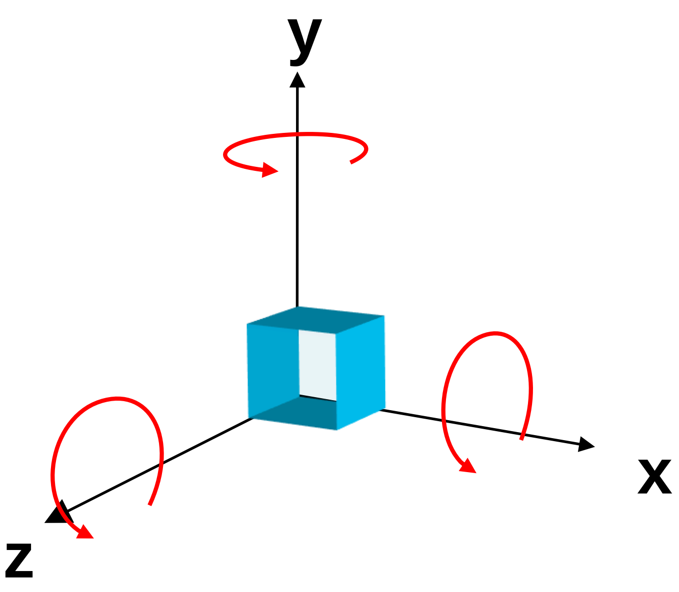
- Inverse
  - Can be used for division
  - Given a quaternion:
    - q = w + xi+ yj + zk
  - The inverse of quaternion q is q^{-1}, where:

$$q^{-1} = \frac{q^*}{|q|^2}$$

Quaternion Rotations

Matrix Rotation

- Matrix Rotation is based on 3 rotations:
  - On axes: x, y, z
  - Or yaw, pitch, roll (which one corresponds to which axis, depends on the orientation to the axes)
  - Sequence matters (x-y may not equal y-x)



Matrix Rotation

$x - axis\ rotation:$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}$$

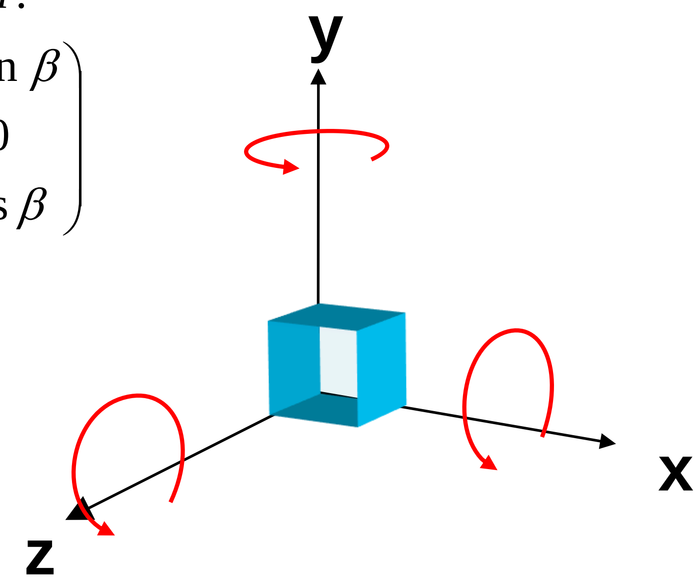
$x = 1$

$y = \cos\alpha + \sin\alpha$

$z = -\sin\alpha + \cos\alpha$

$y - axis\ rotation:$

$$\begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix}$$



$z - axis\ rotation:$

$$\begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$Final\ rotation\ matrix:$

$$\begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & -\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & -\sin\alpha\cos\beta \\ -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma + \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{pmatrix}$$

Quaternion Rotation

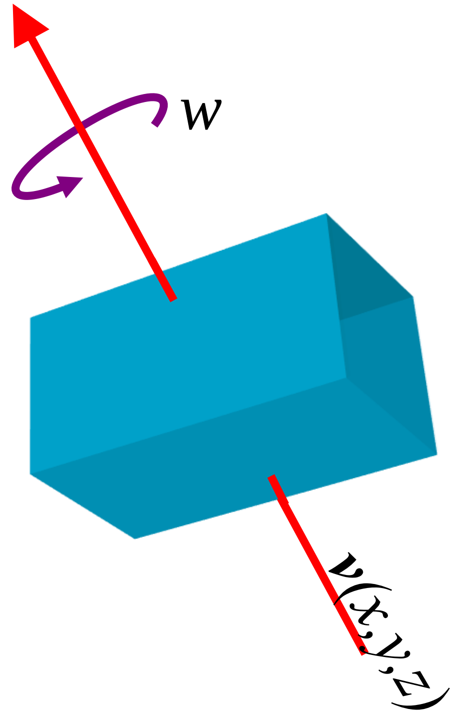
Parts of quaternion

- $w = \cos(\theta/2)$
- $v = \sin(\theta/2)\hat{\mathbf{u}}$
- Where $\hat{\mathbf{u}}$ is a unit/normalized vector $\boldsymbol{u}$ (i, j, k)
- Quaternion can be represented as:

$$q=\cos(\theta/2)+\sin(\theta/2)(xi+yj+zk)$$

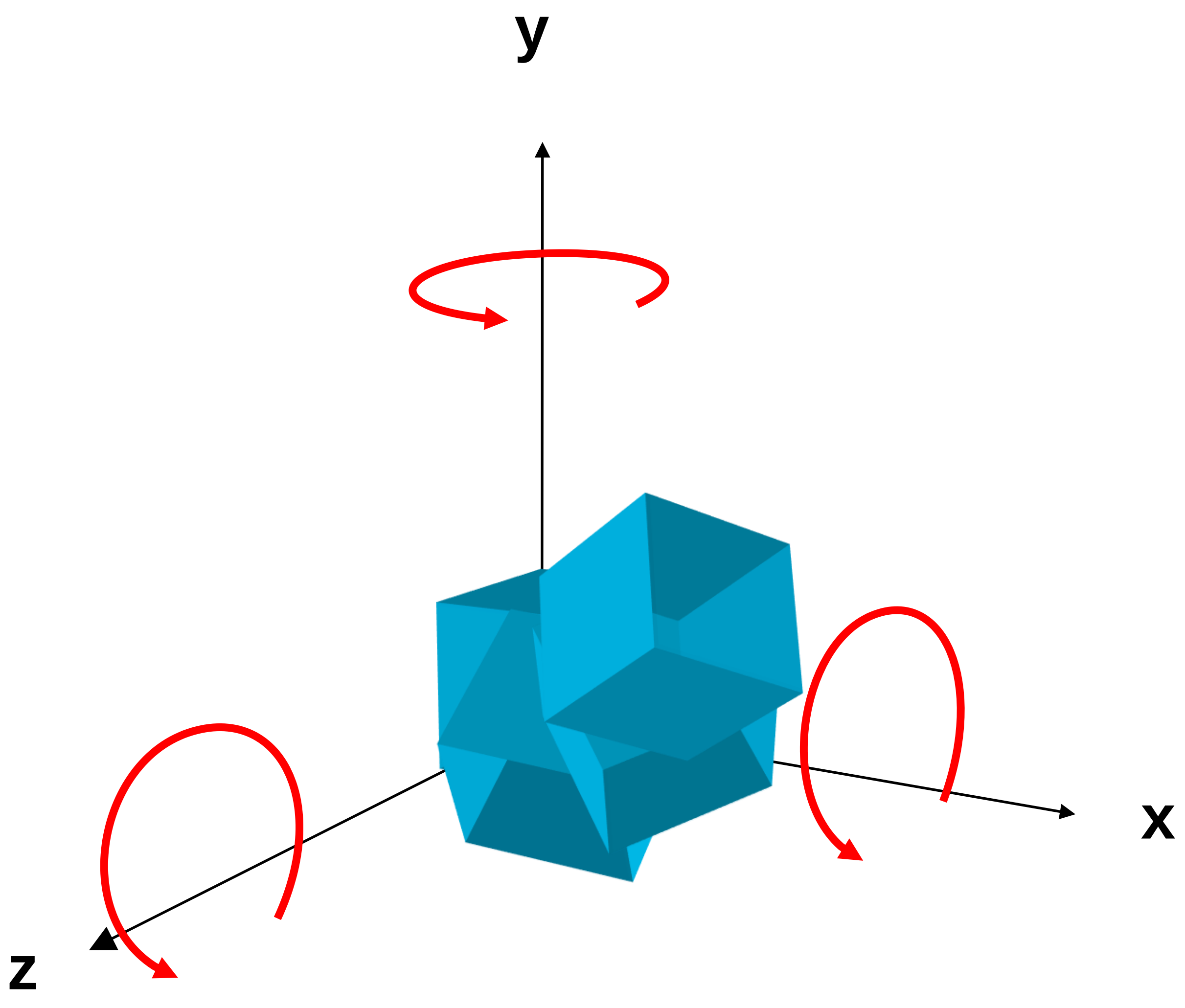
or

$$q=\cos(\theta/2)+\sin(\theta/2)\ \hat{\mathbf{u}}$$

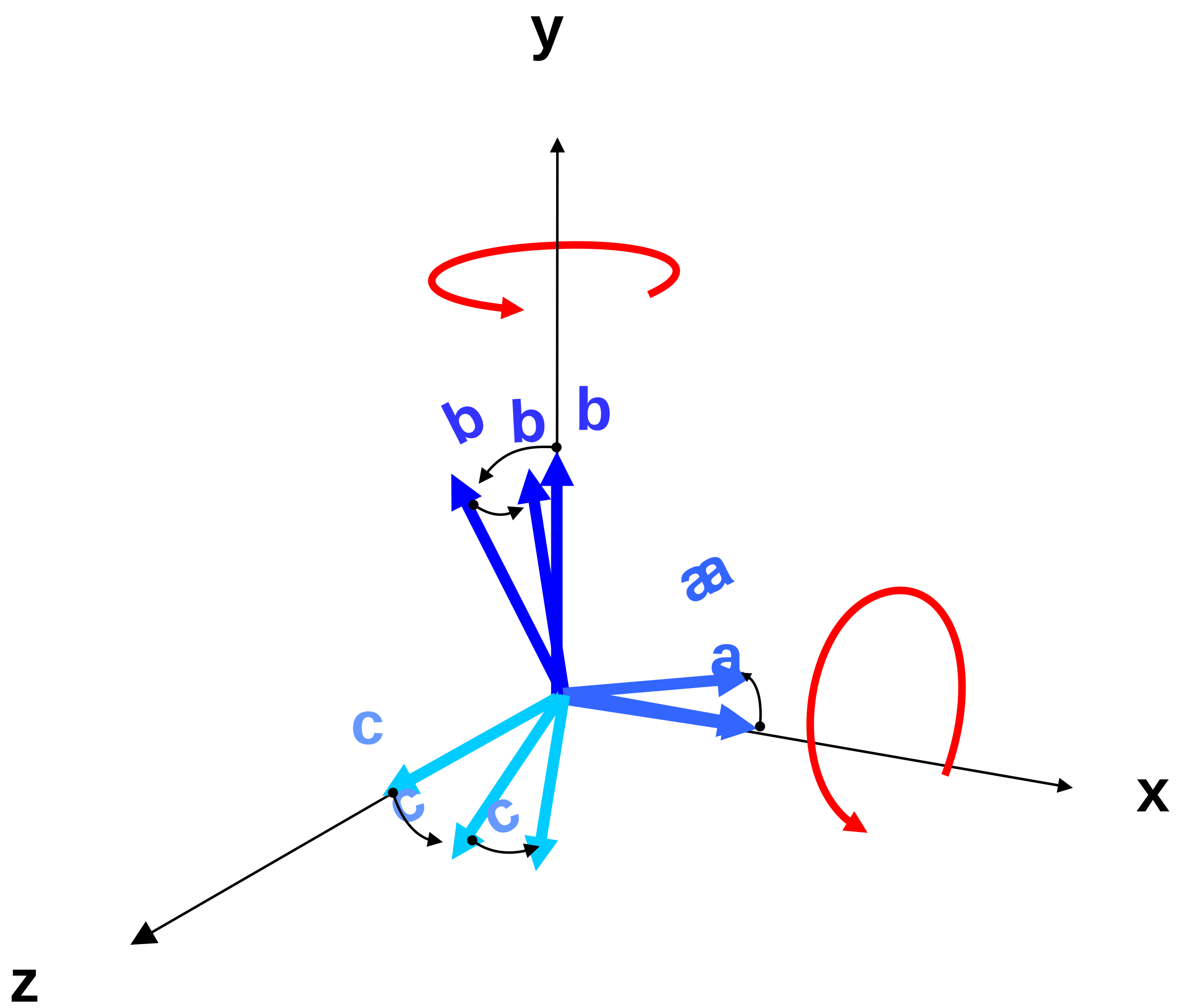


Matrix rotation versus quaternion rotation

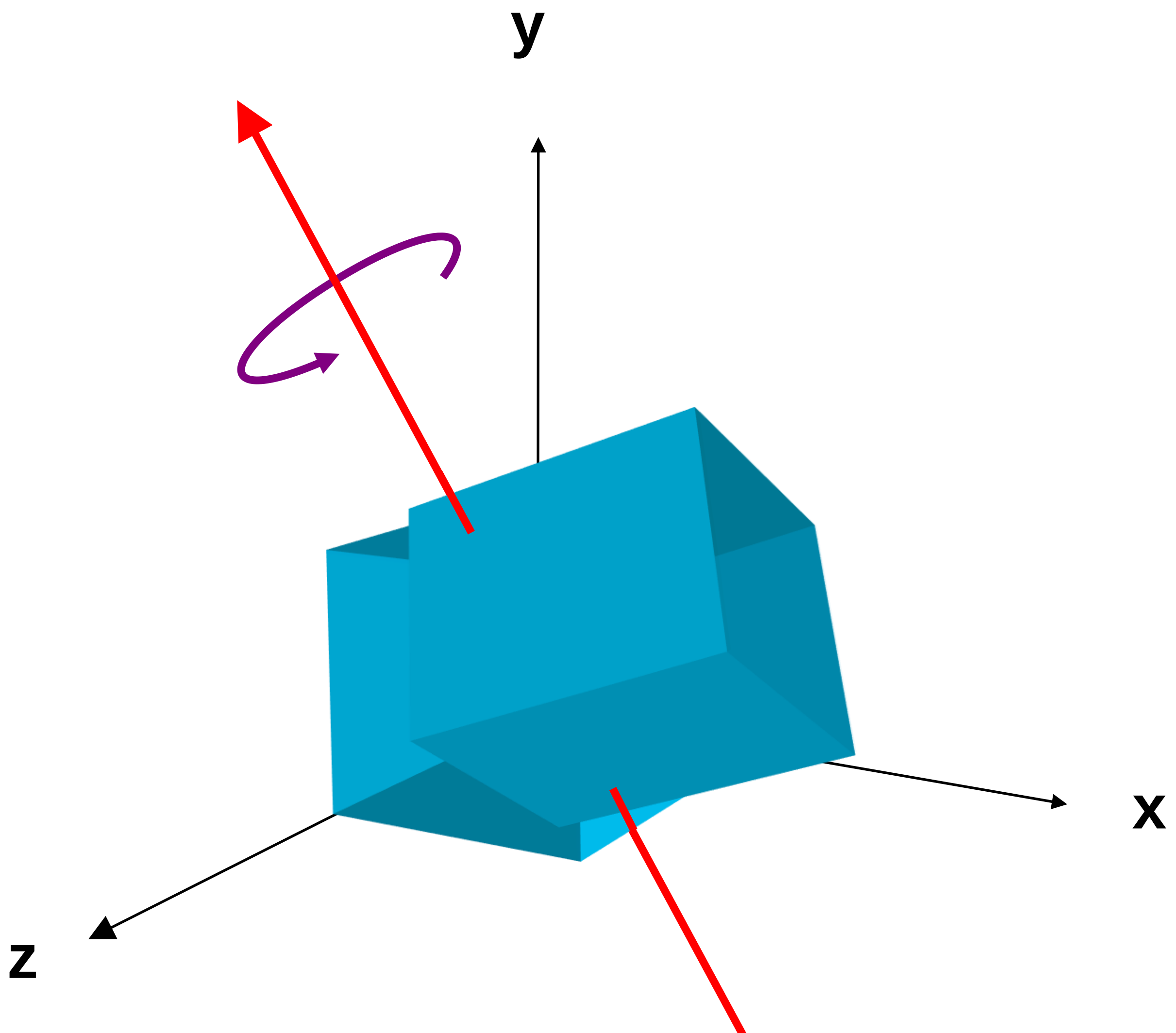
Let’s do rotation!



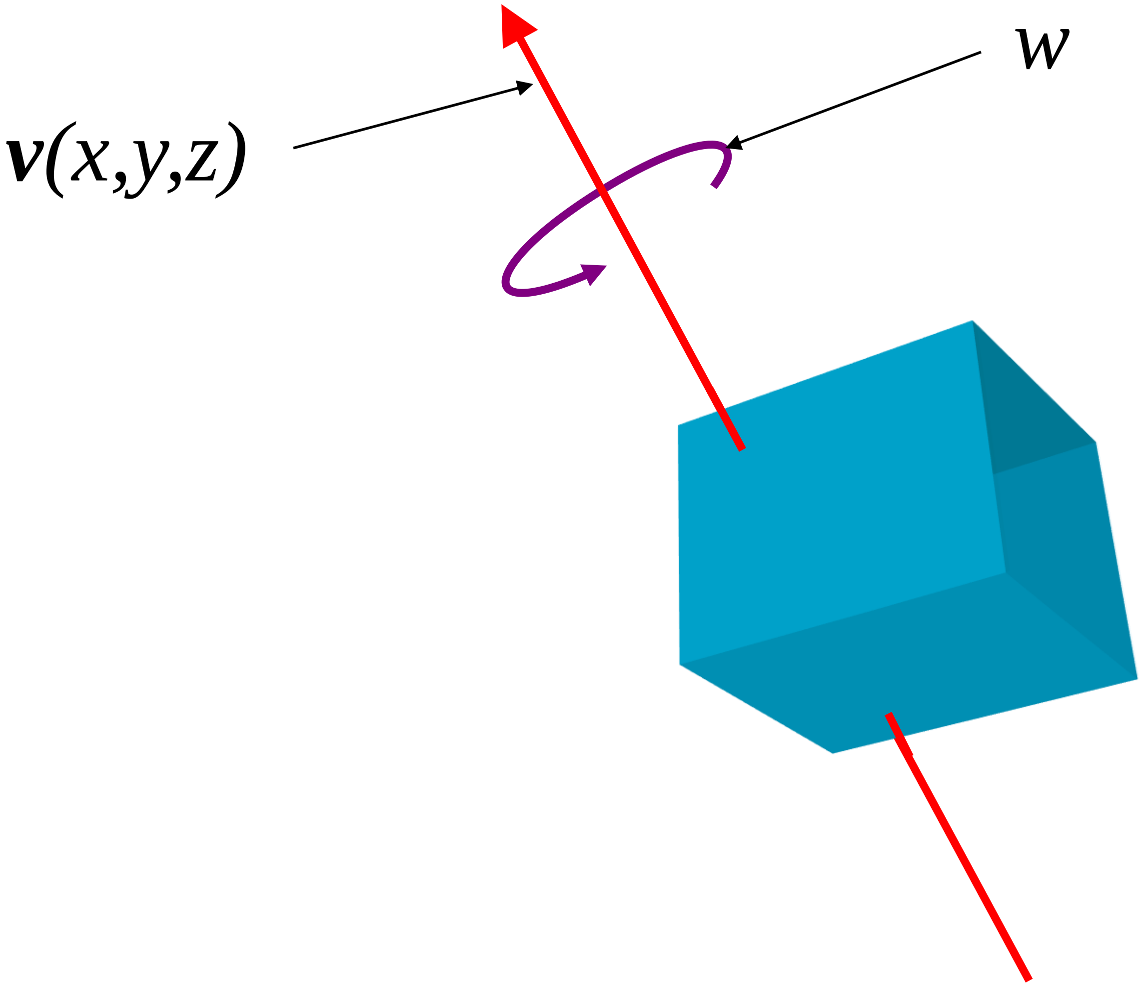
Let’s do rotation!



Let's do another one!



Quaternion?

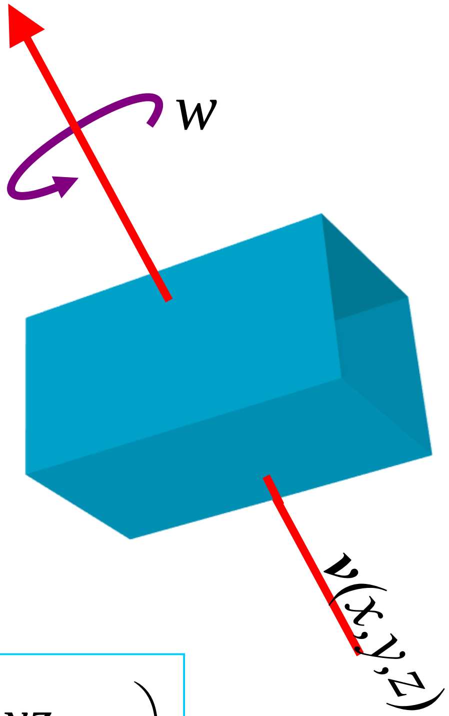


Can create rotation by using arbitrary axis ($\boldsymbol{v}(x,y,z)$) and rotate the object by w amount.

Rotation of a Quaternion

- Calculation is still done in matrix form
- Given a quaternion:

  q = w + xi+ yj + zk
- The matrix form of quaternion q is:

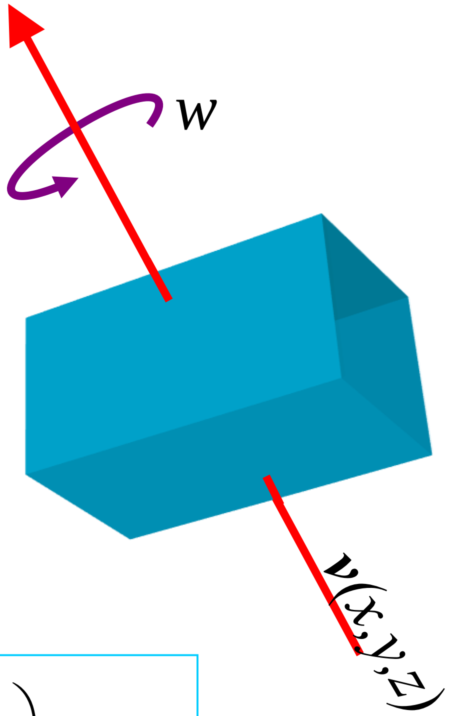


$$\begin{pmatrix} w^2+x^2-y^2-z^2 & 2xy-2wz & 2wy+2xz \\ 2wz+2xy & w^2-x^2+y^2-z^2 & 2yz-2wx \\ 2xz-2wy & 2wx+2yz & w^2-x^2-y^2+z^2 \end{pmatrix}$$

Matrix entries are taken all from quaternion

Quaternion Rotation

- When it is a unit quaternion:



$$\begin{pmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2wy+2xz \\ 2wz+2xy & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2wx+2yz & 1^2-2x^2-2y^2 \end{pmatrix}$$

Quaternion matrix for unit quaternion: w=1

Example of unit quaternion

Our notation:

quaternion

- Rotation of vector ***v*** is ***v'***
  - Where ***v*** is: $\boldsymbol{v}=ai+bj+ck$
- By quaternion $q=(w,\boldsymbol{u})$
  - Where w = 1
- Vector (axis): $\mathbf{u} = i + j + k$
- Rotation angle: 120° = (2π)/3 radian (θ)
- Length of $\mathbf{u} = \sqrt{3}$
- If we rotate a vector, the result should be a vector.

$$q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\hat{u}$$

$$q = \cos\frac{2\pi/3}{2} + \sin\frac{2\pi/3}{2}\hat{u}$$

$$q = \cos\frac{\pi}{3} + \sin\frac{\pi}{3}\hat{u}$$

$$q = \cos 60^\circ + \sin 60^\circ\hat{u}$$

$$q = \frac{1}{2} + \frac{\sqrt{3}}{2}\hat{u}$$

$$q = \frac{1}{2} + \frac{\sqrt{3}}{2}.\frac{(i+j+k)}{\sqrt{3}}$$

$$q = \frac{1+i+j+k}{2}$$

Example of quaternion rotation (cont'd)

- So to rotate $\boldsymbol{v}$:

$$\boldsymbol{v}' = q\,\boldsymbol{v}\,q^{*}$$

- Where q^* is conjugate of q:

$$q^* = \frac{(1 - i - j - k)}{2}$$

So now we can substitute $q\,\boldsymbol{v}\,q^{*}$ to matrix form of quaternion

Example of quaternion rotation (cont'd)

Quaternion matrix

$$qvq^* = \begin{pmatrix} w^2+x^2-y^2-z^2 & 2xy-2wz & 2wy+2xz \\ 2wz+2xy & w^2-x^2+y^2-z^2 & 2yz-2wx \\ 2xz-2wy & 2wx+2yz & w^2-x^2-y^2+z^2 \end{pmatrix} v$$

$$qvq^* = \begin{pmatrix} 1+1-1-1 & 2-2 & 2+2 \\ 2+2 & 1-1+1-1 & 2-2 \\ 2-2 & 2+2 & 1-1-1+1 \end{pmatrix} v$$

$$qvq^* = \begin{pmatrix} 0 & 0 & 4 \\ 4 & 0 & 0 \\ 0 & 4 & 0 \end{pmatrix} v$$

$$q = q_0 + q_1 i + q_2 j + q_3 k = q_0 + \vec{q}$$

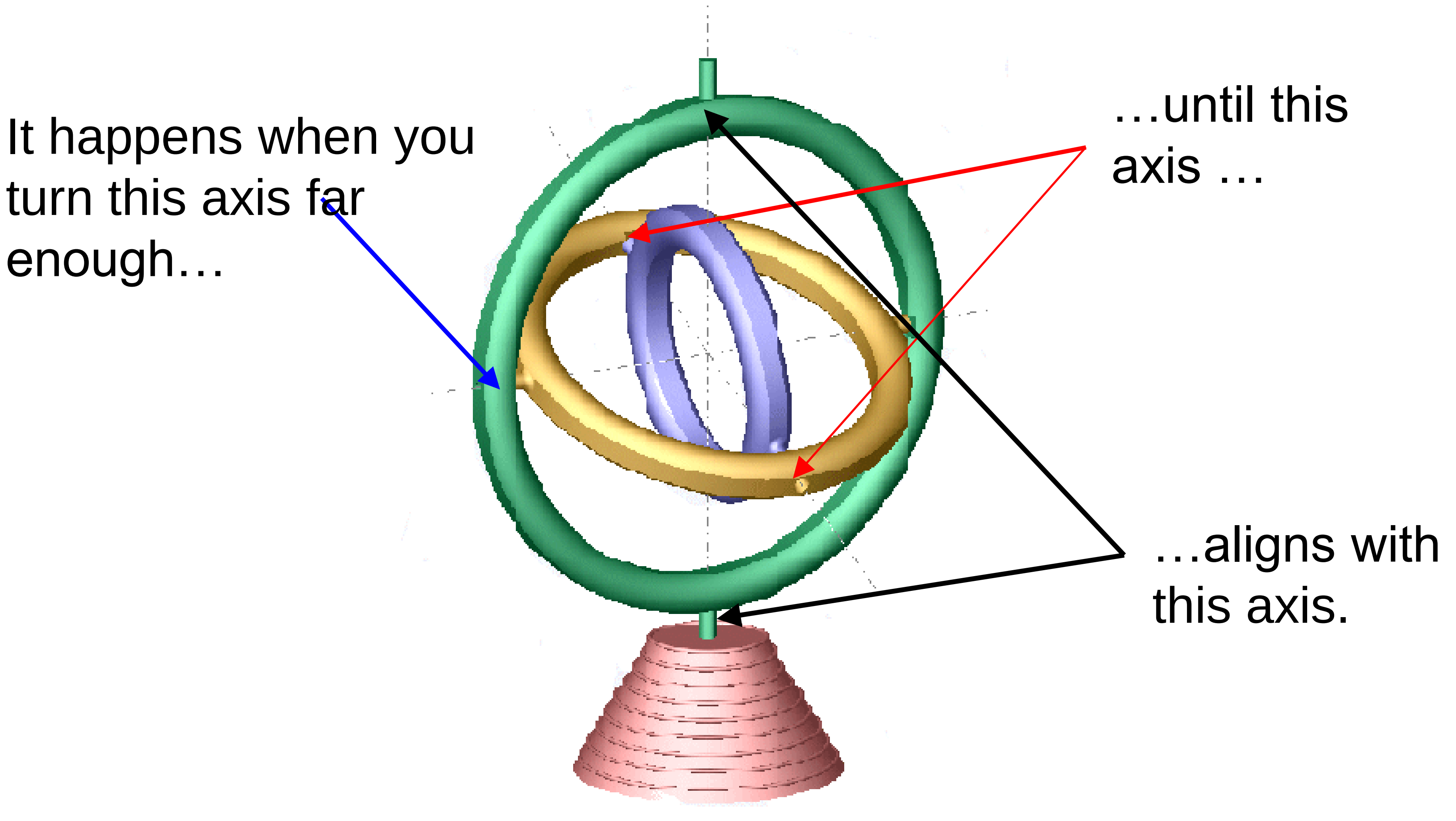
$$i^2 = j^2 = k^2 = -1$$

$$ij = -ji = k$$

$$jk = -kj = i$$

$$ki = -ik = j$$

Gimbal Lock



It happens when you turn this axis far enough…
…until this axis …
…aligns with this axis.

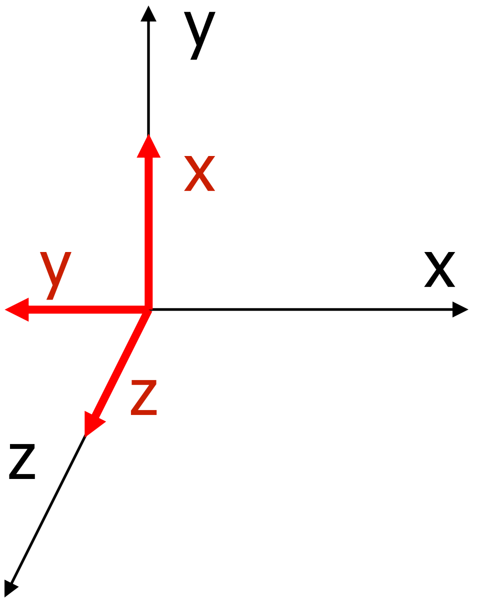
Gimbal Lock

Final rotation matrix :

$$\begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma+\cos\alpha\sin\gamma & -\sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & -\sin\alpha\cos\beta \\ -\cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma+\sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{pmatrix}$$

- Gimbal lock occurs when **rotated -90° or 90° on y-axis**
- **Remember that:**
  - α = rotation on x-axis
  - β = rotation on y-axis
  - γ = rotation on z-axis

Example 1 of using quaternions in robotics: *quaternion representing a rotation*



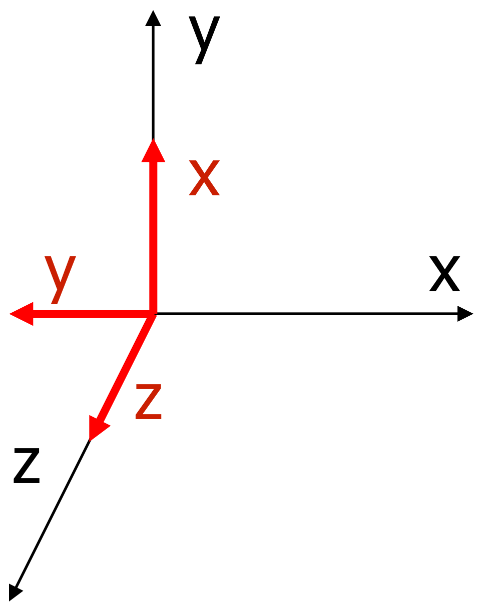
Rot(z,90°)

$$q = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rot (90, 0,0,1) OR Rot (-90,0,0,-1)

Example of using quaternions in robotics



Rot(z,90°)

How to represent rotation?

$$q = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}$$

Represented as quaternion

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Represented as matrix

Rot (90, 0,0,1) OR Rot (-90,0,0,-1)

Operations on Unit Quaternions

- Perform Rotation $x' = qxq^* = \ldots$

$$= (q_0^2 - \vec{q} \cdot \vec{q})x + 2q_0\vec{q} \times x + 2\vec{q}(\vec{q} \cdot x)$$

- Composition of rotations

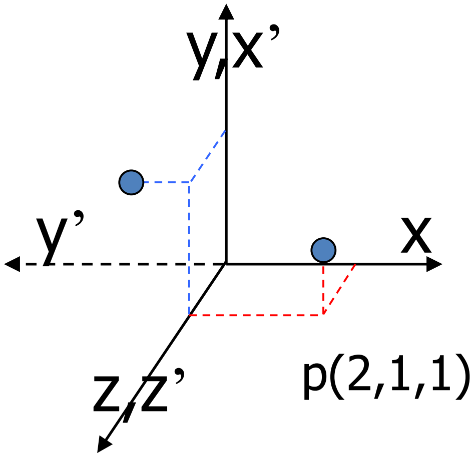
$$x' = pxp^*$$

$$x'' = qx'q^* = q\left(pxp^*\right)q^* = (qp)x(qp)^*$$

- Interpolation
  - Linear
  - Spherical linear (more later)

$$p(t) = (1-t)p^1 + tp^2, \; p = \frac{p(t)}{\left|p(t)\right|}$$

Example of rotation using unit quaternions



Rot(z,90°)

For comparison we first use matrices

$$R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p' = Rp = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

Example (cont)

$$p = \begin{pmatrix} 0 & 2 & 1 & 1 \end{pmatrix}$$
$$q = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}$$

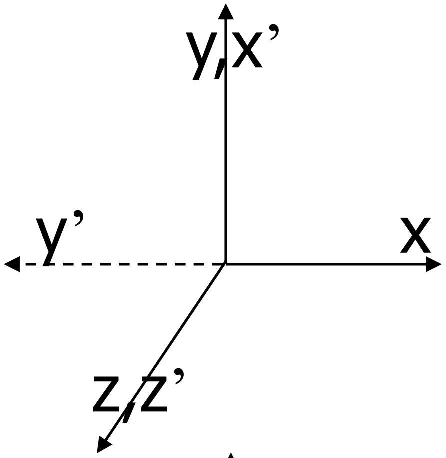
For comparison we use quaternions

$$p' = (q_0^2 - \vec{q}\cdot\vec{q})p + 2q_0\vec{q}\times p + 2\vec{q}(\vec{q}\cdot p)$$
$$= (\tfrac{1}{2} - \tfrac{1}{2})\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + 2\frac{\sqrt{2}}{2}\begin{bmatrix} i & j & k \\ 0 & 0 & \frac{\sqrt{2}}{2} \\ 2 & 1 & 1 \end{bmatrix} + 2\begin{bmatrix} 0 \\ 0 \\ \frac{\sqrt{2}}{2} \end{bmatrix}(\tfrac{\sqrt{2}}{2})$$
$$= \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

Next we convert to matrices

We get the same result

New Example: **multiplication of quaternions**



$$q_1 = \cos\tfrac{90}{2} + \sin\tfrac{90}{2}(0 \quad 0 \quad 1) = (\tfrac{\sqrt{2}}{2} \quad 0 \quad 0 \quad \tfrac{\sqrt{2}}{2})$$

$$q_2 = \cos\tfrac{90}{2} + \sin\tfrac{90}{2}(1 \quad 0 \quad 0) = (\tfrac{\sqrt{2}}{2} \quad \tfrac{\sqrt{2}}{2} \quad 0 \quad 0)$$

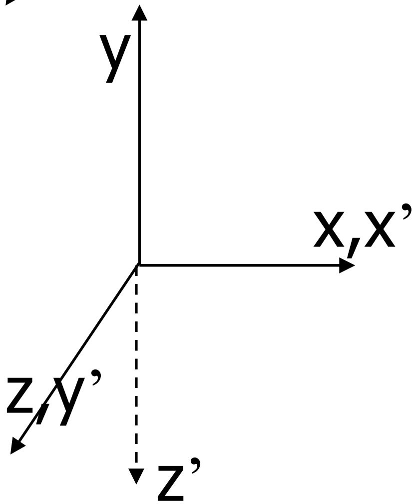
Compositio n :

$$q_2 q_1 = (\tfrac{\sqrt{2}}{2} \quad \tfrac{\sqrt{2}}{2} \quad 0 \quad 0)(\tfrac{\sqrt{2}}{2} \quad 0 \quad 0 \quad \tfrac{\sqrt{2}}{2})$$

$$= \tfrac{\sqrt{2}}{2}\tfrac{\sqrt{2}}{2} + \tfrac{\sqrt{2}}{2}(\tfrac{\sqrt{2}}{2} \quad 0 \quad 0) + \tfrac{\sqrt{2}}{2}(0 \quad 0 \quad \tfrac{\sqrt{2}}{2})$$

$$+ \begin{bmatrix} i & j & k \\ \tfrac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & \tfrac{\sqrt{2}}{2} \end{bmatrix} - (\tfrac{\sqrt{2}}{2} \quad 0 \quad 0) \cdot (0 \quad 0 \quad \tfrac{\sqrt{2}}{2})$$

$$= (\tfrac{1}{2} \quad \tfrac{1}{2} \quad \tfrac{-1}{2} \quad \tfrac{1}{2})$$



Example: Conversion of quaternion matrix to rotation matrix R

Matrix R represented with quaternions

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

We substitute values of q →

$$q = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}$$

And we get R →

$$R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Conversion Formulas

Relations between q_i and r_{ij}

$$q_0^2 = \frac{1}{4}(1 + r_{11} + r_{22} + r_{33})$$

$$q_1^2 = \frac{1}{4}(1 + r_{11} - r_{22} - r_{33})$$

$$q_2^2 = \frac{1}{4}(1 - r_{11} + r_{22} - r_{33})$$

$$q_3^2 = \frac{1}{4}(1 - r_{11} - r_{22} + r_{33})$$

$$q_0 q_1 = \tfrac{1}{4}(r_{32} - r_{23})$$

$$q_0 q_2 = \tfrac{1}{4}(r_{13} - r_{31})$$

$$q_0 q_3 = \tfrac{1}{4}(r_{21} - r_{12})$$

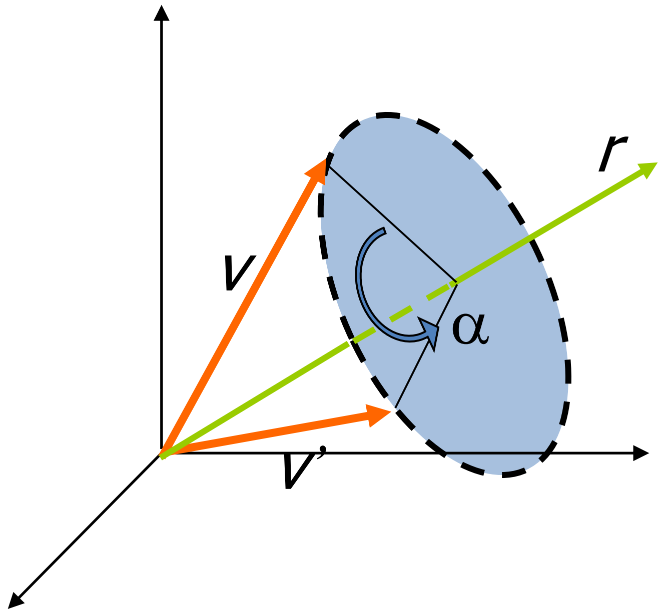
$$q_1 q_2 = \tfrac{1}{4}(r_{12} + r_{21})$$

$$q_1 q_3 = \tfrac{1}{4}(r_{13} + r_{31})$$

$$q_2 q_3 = \tfrac{1}{4}(r_{23} + r_{32})$$

Rodrigues Formula

$$R = c\,I + (1-c)\,rr^t + s\,r^\Lambda$$



$$c = \cos(\alpha),\, s = \sin(\alpha)$$

$$r^\Lambda = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix}$$

v' = *R* v

References:

http://mesh.caltech.edu/ee148/notes/rotations.pdf

http://www.cs.berkeley.edu/~ug/slide/pipeline/assignments/as5/rotation.html

Rotation Matrix

- Meaning of three columns
- Perform rotation: linear algebra
- Composition: trivial
  - orthogonalization might be required due to floating point errors
- Interpolation: ?

$$A = \left[a_{ij}\right] = \left[\begin{array}{c:c:c} \hat{u}_1' & \hat{u}_2' & \hat{u}_3' \end{array}\right]$$

$$\begin{aligned} x' &= x_1\hat{u}_1' + x_2\hat{u}_2' + x_3\hat{u}_3' \\ &= x_1 A\hat{u}_1 + x_2 A\hat{u}_2 + x_3 A\hat{u}_3 \\ &= Ax \end{aligned}$$

$$x' = R_1 x$$

$$x'' = R_2 x' = R_2 R_1 x = \left(R_2 R_1\right) x$$

Gram-Schmidt Orthogonalization

- If 3x3 rotation matrix no longer orthonormal, metric properties might change!

$$\begin{bmatrix} \hat{u}_1 & \vdots & \hat{u}_2 & \vdots & \hat{u}_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \hat{v}_1 & \vdots & \hat{v}_2 & \vdots & \hat{v}_3 \end{bmatrix}$$

$$\hat{v}_1 = \hat{u}_1$$

$$\hat{v}_2 = \hat{u}_2 - \frac{\hat{u}_2 \cdot \hat{v}_1}{\hat{v}_1 \cdot \hat{v}_1} \hat{v}_1$$

$$\hat{v}_3 = \hat{u}_3 - \frac{\hat{u}_3 \cdot \hat{v}_1}{\hat{v}_1 \cdot \hat{v}_1} \hat{v}_1 - \frac{\hat{u}_3 \cdot \hat{v}_2}{\hat{v}_2 \cdot \hat{v}_2} \hat{v}_2$$

Verify!

Spatial Displacement

- Any displacement can be decomposed into a rotation followed by a translation
- Matrix

$$x' = Rx + d$$

$$x = \begin{pmatrix} x \\ y \\ z \\ \hline 1 \end{pmatrix}, T = \left[\begin{array}{c:c} R & d \\ \hdashline 0 & 1 \end{array}\right] \Rightarrow x' = Tx$$

- Quaternion

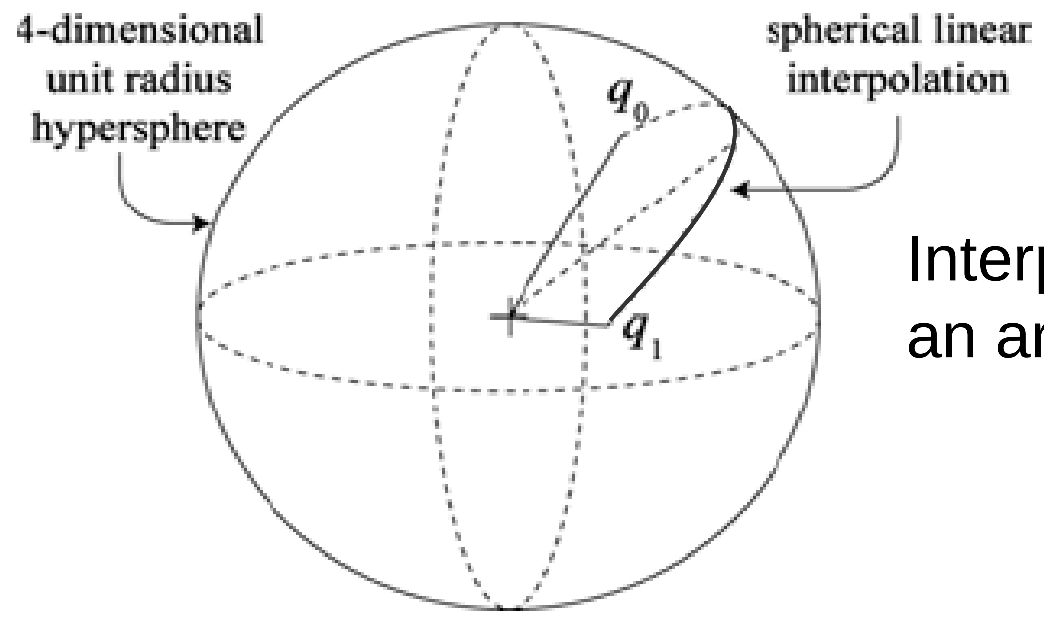
$$x' = qxq^{*} + d$$

Spherical Linear Interpolation

Interpolation

City University London

Spherical Linear Interpolation



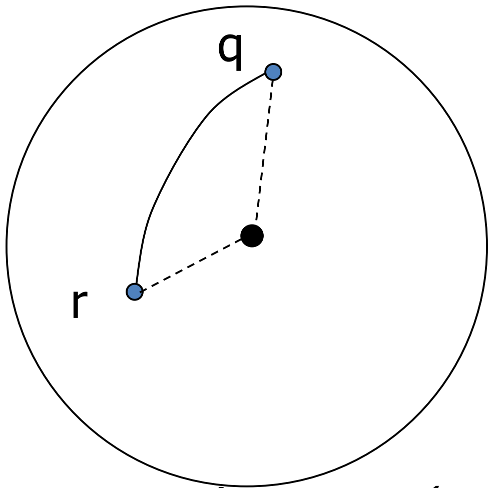
Interpolation produces an arc instead of a line

Spherical Linear Interpolation

$$s(t) = \frac{\sin \phi(1-t)}{\sin \phi} q + \frac{\sin \phi t}{\sin \phi} r$$

$$\cos \phi \equiv q_0 r_0 + q_1 r_1 + q_2 r_2 + q_3 r_3$$

quaternion



unit sphere in R^4

The computed rotation quaternion rotates about a fixed axis at constant speed

References:

http://www.gamedev.net/reference/articles/article1095.asp
http://www.diku.dk/research-groups/image/teaching/Studentprojects/Quaternion/
http://www.sjbrown.co.uk/quaternions.html
http://www.theory.org/software/qfa/writeup/node12.html

Spherical Linear Interpolation – Slerp for unit-length quaternions

$$slerp(t; q_0, q_1) = \frac{\sin((1-t)\theta)q_0 + \sin(t\theta)q_1}{\sin\theta}$$

- q_0 and q_1 are unit-length quaternions
- θ = angle between q_0 and q_1
- t is interval [0, 1]
- "*if q_0 and q_1 are the same quaternion, then θ = 0, but in this case, $q(t) = q_0$ for all t.*"

Spherical Linear Interpolation - Slerp

$$slerp(t; q_0, q_1) = \frac{\sin((1-t)\theta)q_0 + \sin(t\theta)q_1}{\sin\theta}$$

$$slerp(t; q_0, q_1) = \begin{cases} \sin(\pi(\tfrac{1}{2} - t))q_0 + \sin(\pi t)p & t \in [0, \tfrac{1}{2}] \\ \sin(\pi(1-t))p + \sin(\pi(t - \tfrac{1}{2}))q_1 & t \in [\tfrac{1}{2}, 0] \end{cases}$$

- if $q_1 = -q_0$ then $\theta = \pi$
- use a third quaternion p perpendicular to q_0 (which could be infinite number of vectors)
- interpolation is done from q_0 to p for $t \in [0, \tfrac{1}{2}]$ and from p to q_1 for $t \in [\tfrac{1}{2}, 1]$

Spherical Linear Interpolation - Slerp

1. Used in joint animation by storing starting and ending joint position as quaternions.

2. Allows smooth rotations in keyframe animations.

3. *"Spherical Linear interpolation supports the animation of joints when starting and ending joint positions are stored as quaternions that represent the joint rotations from canonical positions" – Rob Saunders, Advanced Games Design, Theory and Practice, March 2005, City University London*

1. The concept of canonical (or conjugate) variables is of major importance.

2. They always occur in complementary pairs, such as spatial location **x** and linear momentum **p**, angle φ and angular momentum L, and energy E and time t.

3. They can be defined as any coordinates whose Poisson brackets give a Kronecker delta (or a Dirac delta incase of discrete variables).

Quaternion in Multi-Sensor Robot Navigation System

(by S. Persa, P. Jonker, Technical University Delft, Netherlands)

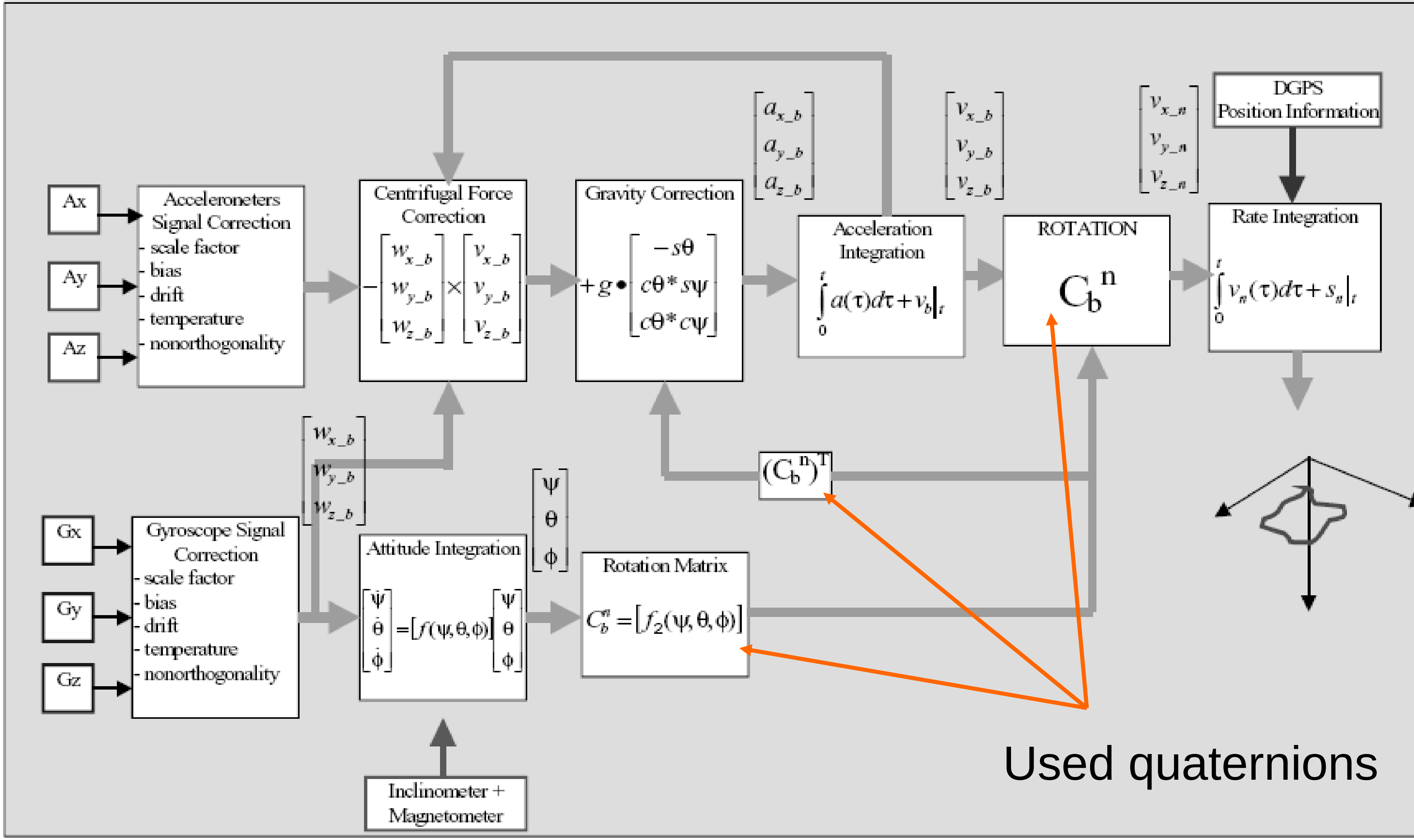


Figure 2. Flow-chart of the strapdown mechanization

Dimensional Synthesis of Spatial RR Robots

(A. Perez, J.M. McCarthy, University of California, Irvine)

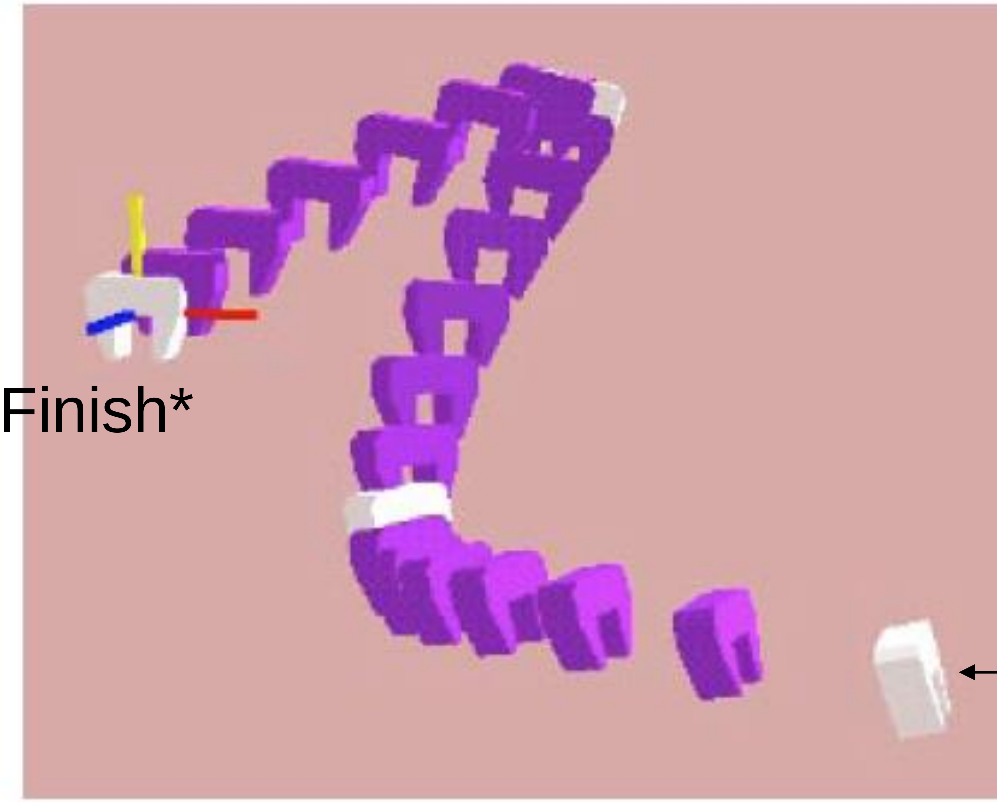
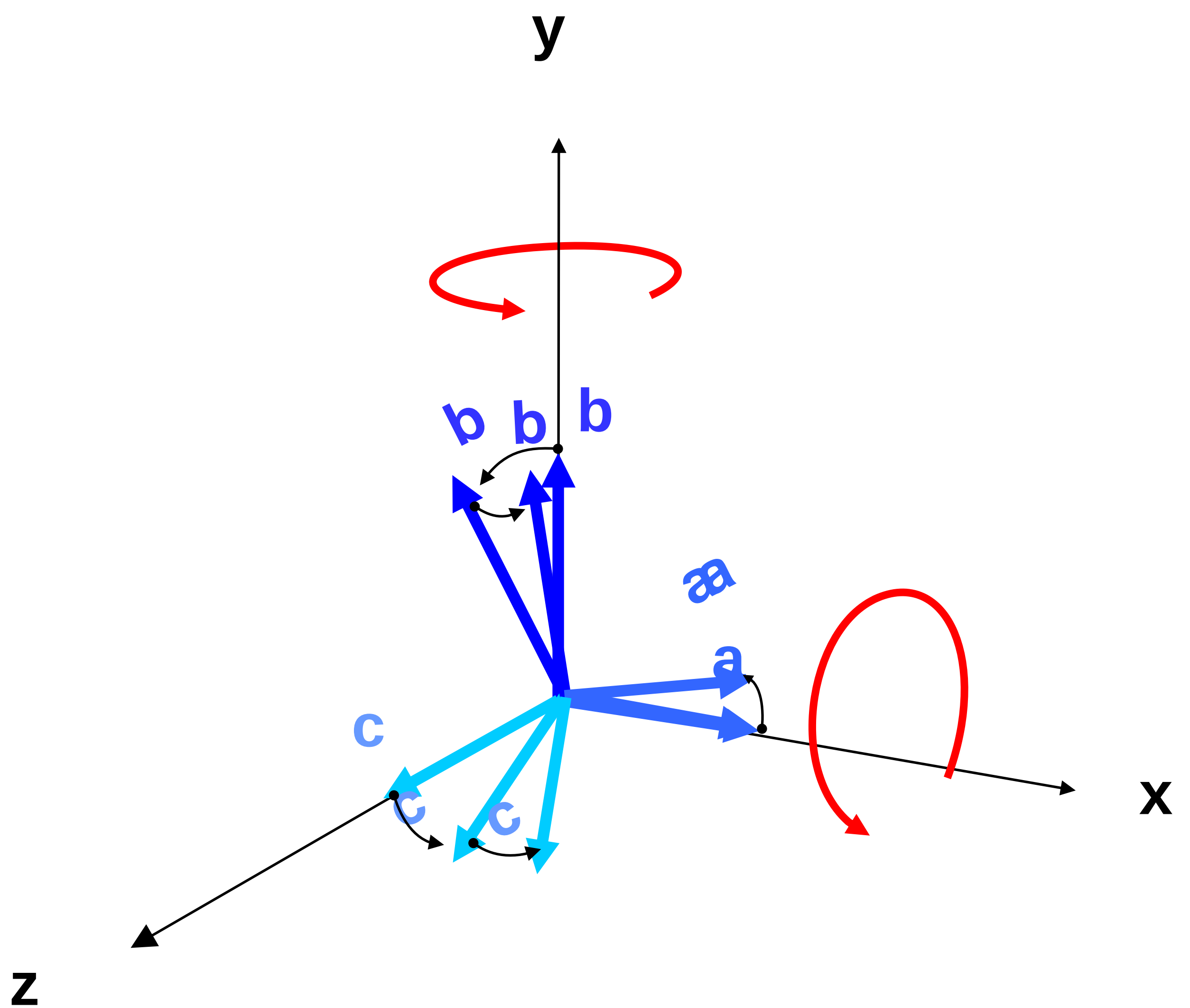


Figure 4. Double quaternion interpolated path

*assumption

Let’s do rotation!



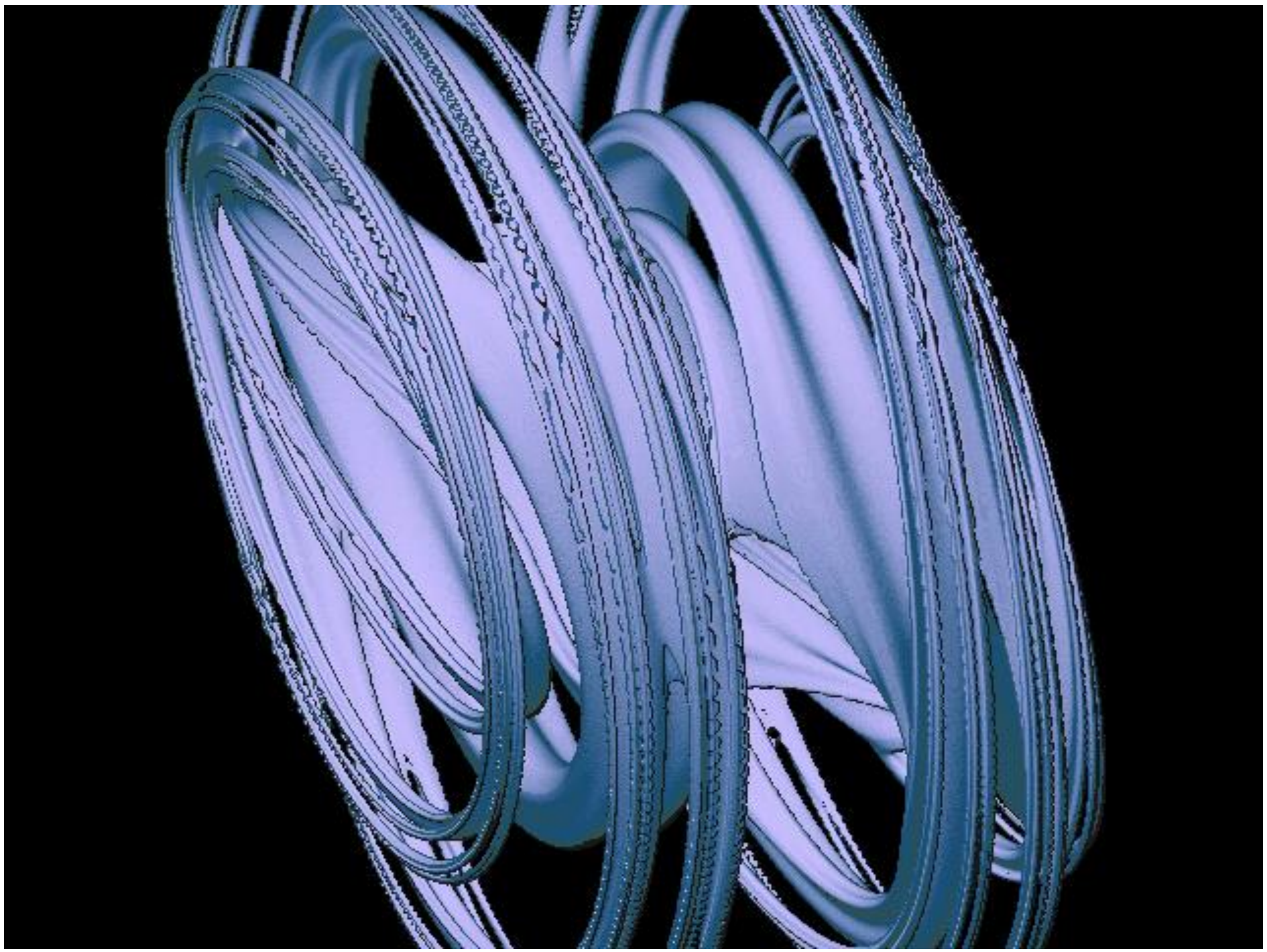
Visualization of quaternions

- Difficult to visualize
- Not for the weak-on-math

Visualizing Quaternion Rotation

Removing 720 degree
twist without moving either end

(from: J.C. Hart, G.K. Francis, L.H. Kauffman, Visualizing Quaternion Rotation, ACM Transactions on Graphics, Vol. 13, No. 3 July 1994, p.267)



Quaternion Explained!

By
Mathias Sunardi
for
Quantum Research Group Seminar
June 15, 2006