Rotations & Articulated Objects

contents

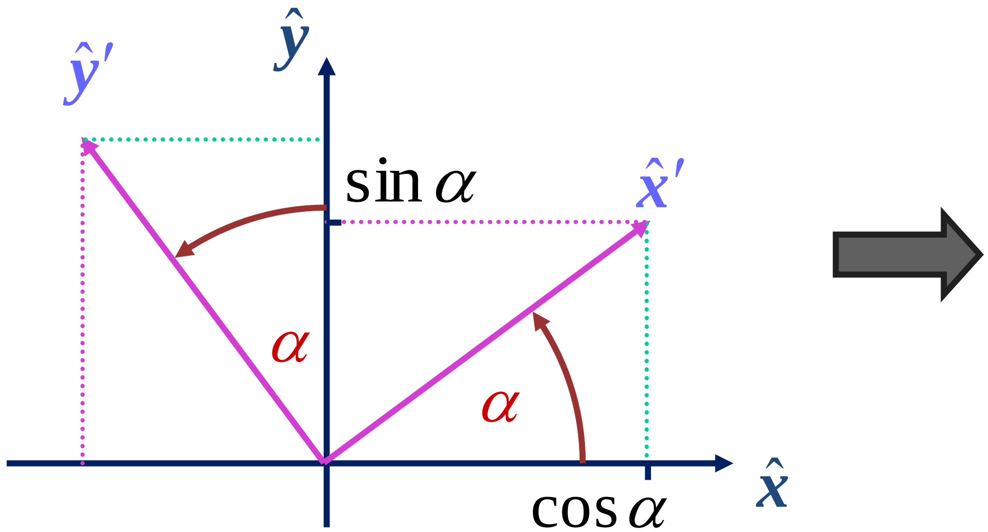
Computer Graphics and Visualization

- 2D rotations
- 3D rotations
- Quaternions
- Articulated Objects

2D ROTATIONS

2D Rotations

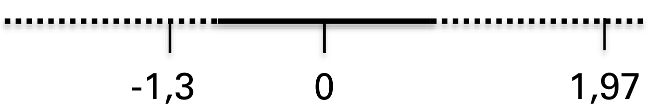
- Transformation matrices contain the new base vectors in their columns (or rows, depending on the convention)
- For example: counter clockwise rotation around angle α

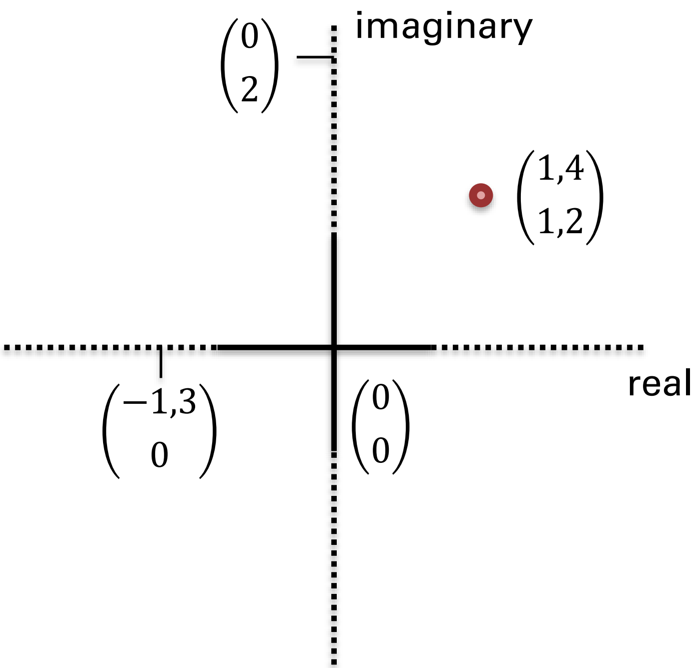


$$M = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

Complex Numbers

- The equation $x^2+1=0$ has no solution in $\boldsymbol{R}$
- With the definition $i=\sqrt{-1}$ we get the solutions $x=\pm i$
- Besides the "real" dimension, an imaginary dimension extends the 1d real space to the 2d complex space
- By definition, the real dimension and the imaginary dimension are orthogonal



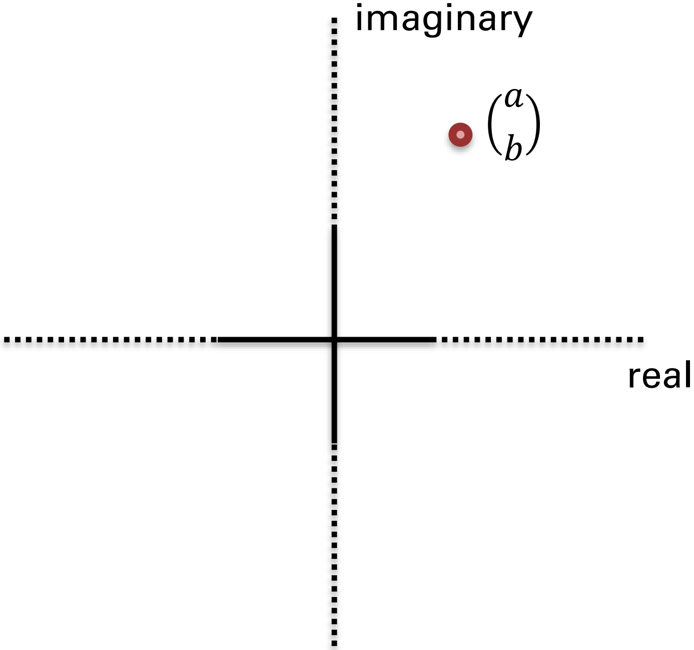


Complex Numbers

- Different equivalent notions exist to describe a point in the complex plane

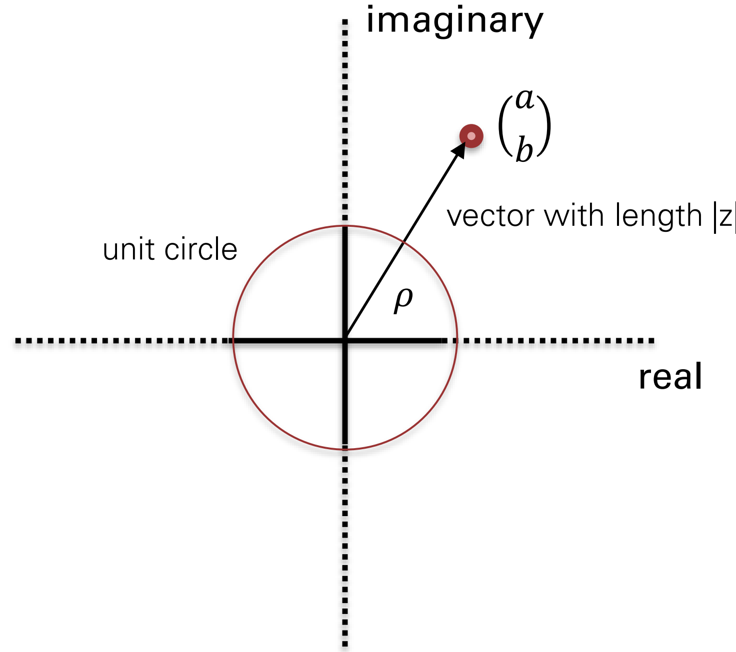
component form

$$z = a + i \cdot b$$



trigonometric form

$$z = |z| \cdot (\cos\rho + i \cdot \sin\rho)$$



Properties of Complex Numbers

- product is defined by distribution law and the basic products $1 \cdot 1 = 1$, $1 \cdot i = i$, $i \cdot 1 = i$, $i \cdot i = -1$:

$$(a + ib)(c + id) = ac - bd + i(bc + ad)$$

- complex product is commutative:

$$z_1 \cdot z_2 = z_2 \cdot z_1$$

- conjugation: $z^* = (a + ib)^* = a - ib$
- norm: $|z|^2 = z \cdot z^* = a^2 + b^2$
- inverse: $z^{-1} = z^*/|z|^2$
- any polynomial $\sum_{j=0\ldots n} a_j \cdot x^j$ has n zero crossings in $\mathbb{C}$

Euler's Formula

- Euler published in 1748 the formula
$$\exp(i\cdot\rho)=\cos\rho+i\cdot\sin\rho$$
that shows $\exp(i\cdot\rho)$ repeatedly traces out the unit circle.
- The proof is given by Taylor series expansions:
$$\exp(x)=1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\frac{1}{120}x^5+\cdots$$
- plugging in $x=i\alpha$ yields
$$\exp(i\alpha)=1+i\alpha-\frac{1}{2}\alpha^2-i\frac{1}{6}\alpha^3+\frac{1}{24}\alpha^4+i\frac{1}{120}\alpha^5+\cdots$$
$$=1-\frac{1}{2}\alpha^2+\frac{1}{24}\alpha^4+\cdots+i\left(\alpha-\frac{1}{6}\alpha^3+\frac{1}{120}\alpha^5+\cdots\right)$$
$$=\cos\alpha+i\cdot\sin\alpha$$
- the trigonometric form of a complex number becomes
$$z=|z|\cdot\exp(i\rho)=|z|\cdot e^{i\rho}$$

2D Rotation with complex numbers

- multiplication of a complex number $z = a + ib$ from left or right with $e^{i\alpha}$ corresponds to 2D rotation of z around origin in complex plane
- proof by reduction to matrix form:

$$
\begin{aligned}
z \cdot e^{i\alpha} &= (a + ib) \cdot e^{i\alpha} \\
&= (a + ib) \cdot (\cos\alpha + i \cdot \sin\alpha) \\
&= a \cdot \cos\alpha + i \cdot a \cdot \sin\alpha + i \cdot b \cdot \cos\alpha - b \cdot \sin\alpha \\
&= a \cdot \cos\alpha - b \cdot \sin\rho + i \cdot (a \cdot \sin\alpha + b \cdot \cos\alpha) \\
&= \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \\
&= (\cos\alpha + i \cdot \sin\alpha) \cdot (a + ib) \\
&= e^{i\alpha} \cdot z
\end{aligned}
$$

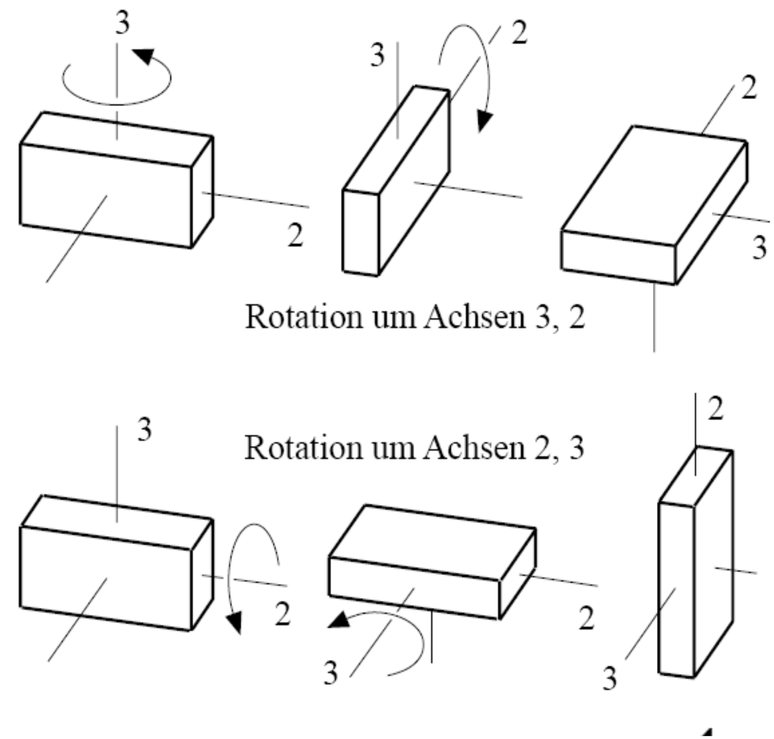
3D ROTATIONS

Properties of rotations in 3D

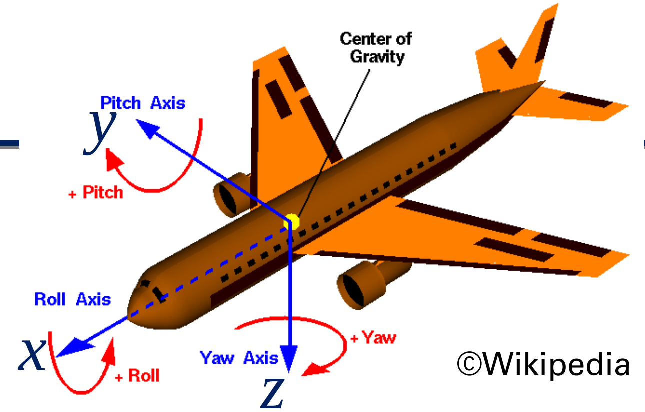
- Rotations retain the length of vectors, which means
  $$\langle\vec{v},\vec{v}\rangle=\langle\boldsymbol{R}\vec{v},\boldsymbol{R}\vec{v}\rangle$$
  $$=(\boldsymbol{R}\vec{v})^T\boldsymbol{R}\vec{v}$$
  $$=\vec{v}^T\boldsymbol{R}^T\boldsymbol{R}\vec{v}$$
- This is true for all $\vec{v}$, thus
  $$\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{1} \quad\Rightarrow\quad \boldsymbol{R}^{-1}=\boldsymbol{R}^T$$
  $$\Rightarrow\quad \det\boldsymbol{R}=\pm\boldsymbol{1}$$
- If $\det\boldsymbol{R}=-1$ the matrix contains a reflection. Rotations must have $\det\boldsymbol{R}=+1$
- Rotations form the special orthonormal group:

$$\forall \boldsymbol{R}_1,\boldsymbol{R}_2\in SO(3):\boldsymbol{R}_1\boldsymbol{R}_2\in SO(3)$$

- 3D rotations do not commute:



Forward Euler Angle



- A 3D rotation in the 123-convention of Euler angles as used in navigation for roll, pitch, yaw can be written in the form:

$$R_x(\theta_1)\,R_y(\theta_2)\,R_z(\theta_3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{pmatrix} \begin{pmatrix} c_2 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_2 \end{pmatrix} \begin{pmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} c_2 c_3 & c_2 s_3 & -s_2 \\ s_1 s_2 c_3 - c_1 s_3 & s_1 s_2 s_3 + c_1 c_3 & s_1 c_2 \\ c_1 s_2 c_3 + s_1 s_3 & c_1 s_2 s_3 - s_1 c_3 & c_1 c_2 \end{pmatrix}$$

with $c_1 = \cos\theta_1$, $s_1 = \sin\theta_1$, etc.

- The Euler angles can be computed from a given rotation matrix $\boldsymbol{R}$ according to the Pseudo code on next slide. We define the vector $\vec{\boldsymbol{\omega}}$ of Euler angles as

$$\vec{\boldsymbol{\omega}} = (\theta_1, \theta_2, \theta_3)^T = R_{123}^{-1}(\boldsymbol{R})$$

Euler Angles Inversion – 2 Approaches

Computer Graphics and Visualization

$$M = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix} = \begin{pmatrix} c_2c_3 & c_2s_3 & -s_2 \\ s_1s_2c_3 - c_1s_3 & s_1s_2s_3 + c_1c_3 & s_1c_2 \\ c_1s_2c_3 + s_1s_3 & c_1s_2s_3 - s_1c_3 & c_1c_2 \end{pmatrix}$$

Ken Shoemake. "Euler angle conversion." Graphics gems IV. 1994. 222-229.

- checks for gimble lock case where c_2 becomes very small with epsilon check on m00 and m01 and assumes $\theta_3 = 0$.

- Mike Day. "Extracting euler angles from a rotation matrix", 2012
- avoids check by accounting for instable θ_1 in computation of θ_3
- this allows for restriction to single precision floats

$$\theta_1 = \mathrm{atan2}(m_{12}, m_{22})$$

$$c_2 = \sqrt{{m_{00}}^2 + {m_{01}}^2}$$

$$\theta_2 = \mathrm{atan2}(-m_{02}, c_2)$$

$$\theta_3 = \mathrm{atan2}(m_{01}, m_{00})$$

$$= \mathrm{atan2}(c_2s_3, c_2c_3)$$

$$s_1 = \sin(\theta_1),\ \ c_1 = \cos(\theta_1)$$

$$\theta_3 = \mathrm{atan2}(s_1m_{20} - c_1m_{10},\ c_1m_{11} - s_1m_{21})$$

Rotation around Axis

- Every rotation can be described by an axis $\hat{\boldsymbol{n}}$ and an angle α
- Notion of this rotation matrix $\boldsymbol{R}(\hat{\boldsymbol{n}},\alpha)$
- For every rotation exist two axis-angle combinations:

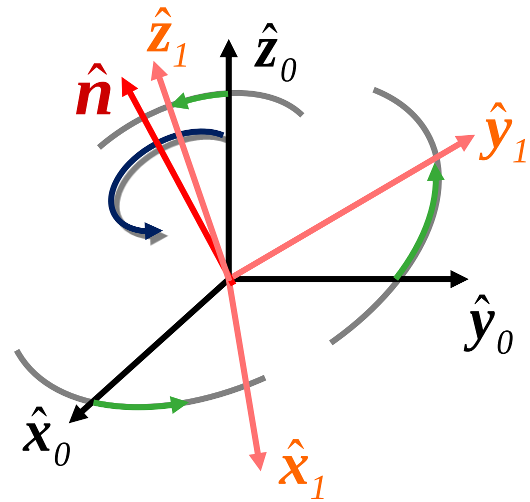
$$\boldsymbol{R}(\hat{\boldsymbol{n}},\alpha)=\boldsymbol{R}(-\hat{\boldsymbol{n}},-\alpha)$$

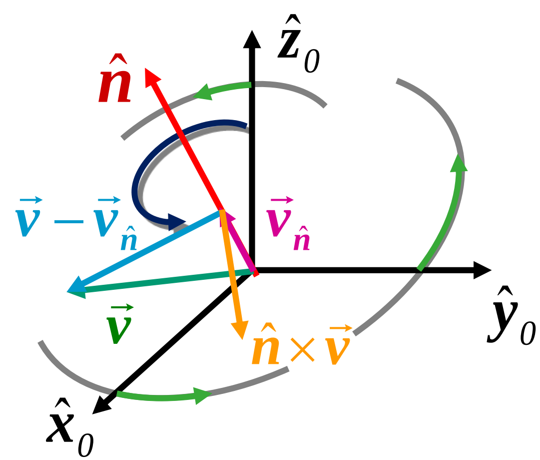
$$\vec{\boldsymbol{v}}'=\underbrace{(\hat{\boldsymbol{n}}^T\vec{\boldsymbol{v}})\hat{\boldsymbol{n}}}_{\text{Components along } \hat{n} \text{ are preserved}}+\underbrace{(\vec{\boldsymbol{v}}-(\hat{\boldsymbol{n}}^T\vec{\boldsymbol{v}})\hat{\boldsymbol{n}})}_{\text{x-component perpendicular to } \hat{n}}\cos\alpha+\underbrace{(\hat{\boldsymbol{n}}\times\vec{\boldsymbol{v}})}_{\text{y-component perpendicular to } \hat{n} \text{ and } \vec{v}}\sin\alpha$$

- Matrix notation

$$\boldsymbol{R}(\hat{\boldsymbol{n}},\alpha)=\boldsymbol{P}_{\hat{n}}+(\boldsymbol{1}-\boldsymbol{P}_{\hat{n}})\cos\alpha+\underbrace{\hat{\boldsymbol{n}}^*}_{\text{cross product matrix}}\sin\alpha$$

using $\mathbf{P}_{\hat{\mathrm{n}}}=\hat{\mathbf{n}}\hat{\mathbf{n}}^T$





QUATERNIONS

Derivation of Euler Parameters

Computer Graphics and Visualization

$$\vec{\mathbf{v}}' = (\vec{\mathbf{v}}\cdot\hat{\boldsymbol{n}})\hat{\boldsymbol{n}} + \left(\vec{\mathbf{v}} - (\vec{\mathbf{v}}\cdot\hat{\boldsymbol{n}})\hat{\boldsymbol{n}}\right)\cos\alpha + \left(\hat{\boldsymbol{n}}\times\vec{\mathbf{v}}\right)\sin\alpha$$

$$\vec{\mathbf{v}}' = \vec{\mathbf{v}}\cos\alpha + (\vec{\mathbf{v}}\cdot\hat{\boldsymbol{n}})\hat{\boldsymbol{n}}(1-\cos\alpha) + \left(\hat{\boldsymbol{n}}\times\vec{\mathbf{v}}\right)\sin\alpha$$

$\sin 2x = 2\sin x\cos x$

$\cos 2x = 1 - 2\sin^2 x$

$\cos 2x = \cos^2 x - \sin^2 x$

addition theorems

$$\vec{\mathbf{v}}' = \vec{\mathbf{v}}\left(\cos^2\tfrac{\alpha}{2} - \sin^2\tfrac{\alpha}{2}\right) + 2(\vec{\mathbf{v}}\cdot\hat{\boldsymbol{n}})\hat{\boldsymbol{n}}\sin^2\tfrac{\alpha}{2} + 2\left(\hat{\boldsymbol{n}}\times\vec{\mathbf{v}}\right)\sin\tfrac{\alpha}{2}\cos\tfrac{\alpha}{2}$$

$$\boxed{\vec{\mathbf{v}}' = \left(e_0^2 - \|\vec{\boldsymbol{e}}\|^2\right)\vec{\mathbf{v}} + 2(\vec{\mathbf{v}}\cdot\vec{\boldsymbol{e}})\vec{\boldsymbol{e}} + 2e_0\left(\vec{\boldsymbol{e}}\times\vec{\mathbf{v}}\right)}$$

Eulerparameter

$$e_0 = \cos\tfrac{\alpha}{2}$$

$$\vec{\boldsymbol{e}} = \hat{\boldsymbol{n}}\sin\tfrac{\alpha}{2} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

$$1 = \sum_{j=0}^{3} e_j^2$$

Quaternion Definition

- comparison with complex numbers suggests definition of quaternions:

$$z = \cos\alpha + i\sin\alpha \quad \Leftrightarrow \quad q = e_0 + ie_1 + je_2 + ke_3$$

- with complex roots i,j and k whose direct products define the rule of multiplying two quaternions
- when reducing to 2D rotations, the complex numbers should result. This implies

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

- This results in the following product table:
(*ij* = *ji* would result in a commutative product, but rotations in 3D don't commute)

$k = ij$

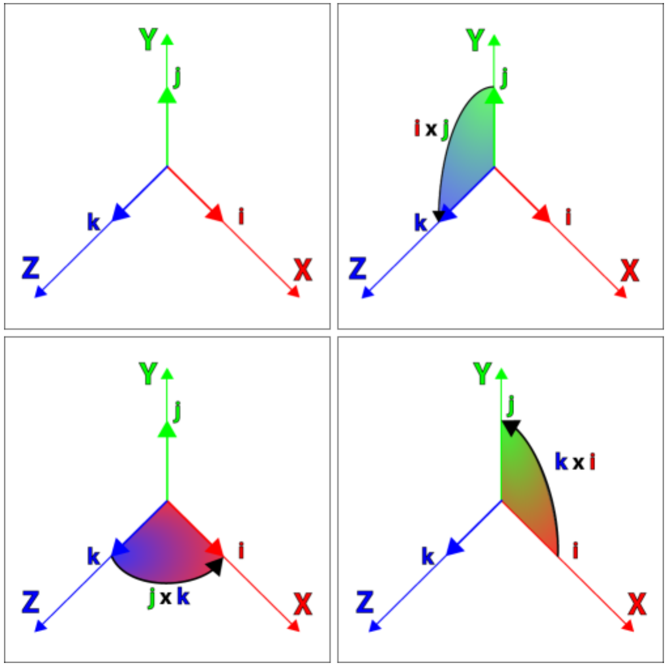
*	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Quaternion Definition

- Note the similarity between i, j, k and cross products of the orthogonal unit vectors $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{z}}$:

$*$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

$\times$	$\hat{\boldsymbol{x}}$	$\hat{\boldsymbol{y}}$	$\hat{\boldsymbol{z}}$
$\hat{\boldsymbol{x}}$	$\mathbf{0}$	$\hat{\boldsymbol{z}}$	$-\hat{\boldsymbol{y}}$
$\hat{\boldsymbol{y}}$	$-\hat{\boldsymbol{z}}$	$\mathbf{0}$	$\hat{\boldsymbol{x}}$
$\hat{\boldsymbol{z}}$	$\hat{\boldsymbol{y}}$	$-\hat{\boldsymbol{x}}$	$\mathbf{0}$



- detailed explanation of quaternions can be found at: http://3dgep.com/?p=1815

More on Quaternions

- conjugation: $q^* = e_0 - e_1 i - e_2 j - e_3 k$
- norm: $\|q\|^2 = qq^* = e_0^2 + e_1^2 + e_2^2 + e_3^2$
- inverse: $q^{-1} = \frac{1}{q} = \frac{q^*}{qq^*}$ for unit quaternions: $\hat{q}^{-1} = \hat{q}^*$
- scalar+vector-interpretation $q = (e_0, \vec{\boldsymbol{e}}) = (s, \vec{\boldsymbol{v}}) = (s, x, y, z)$
- multiplication $q_1 q_2 = (s_1 s_2 - \vec{\boldsymbol{v}}_1 \cdot \vec{\boldsymbol{v}}_2, s_1 \vec{\boldsymbol{v}}_2 + s_2 \vec{\boldsymbol{v}}_1 + \vec{\boldsymbol{v}}_1 \times \vec{\boldsymbol{v}}_2)$
- unit quaternions $\hat{q} = \left(\cos\frac{\alpha}{2}, \hat{\boldsymbol{n}} \sin\frac{\alpha}{2}\right)$ can be interpreted as rotation by α around $\hat{\boldsymbol{n}}$. To rotate a 3d-vector $\vec{\boldsymbol{p}}$ construct quaternion $p = (0, \vec{\boldsymbol{p}})$ and compute
$$p' = \hat{q} p \hat{q}^*$$
- efficient concatenation of rotations: $\hat{q}_{12} = \hat{q}_1 \hat{q}_2$
- When interpolating rotations with quaternions one has to ensure that result is a proper rotation. Then one can normalize the result quaternion or use SLERP.

Example

- concatenation of rotation by α around x followed by rotation by β around y
- $q_x = \left(\cos\frac{\alpha}{2} \quad \sin\frac{\alpha}{2} \quad 0 \quad 0\right)$
- $q_y = \left(\cos\frac{\beta}{2} \quad 0 \quad \sin\frac{\beta}{2} \quad 0\right)$
- quat. product: $q_1 q_2 = (s_1 s_2 - \underbrace{\vec{\boldsymbol{v}}_1 \cdot \vec{\boldsymbol{v}}_2}_{0}, s_1 \vec{\boldsymbol{v}}_2 + s_2 \vec{\boldsymbol{v}}_1 + \vec{\boldsymbol{v}}_1 \times \vec{\boldsymbol{v}}_2)$

$$q_y q_x = \left(\cos\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \sin\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \cos\frac{\alpha}{2}\sin\frac{\beta}{2} \quad -\sin\frac{\alpha}{2}\sin\frac{\beta}{2}\right)$$

- this corresponds to rotation around new axis $\hat{\boldsymbol{n}}$ and new angle γ with

$$\vec{\boldsymbol{n}} = \left(\sin\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \cos\frac{\alpha}{2}\sin\frac{\beta}{2} \quad -\sin\frac{\alpha}{2}\sin\frac{\beta}{2}\right),$$

Example

Computer Graphics and Visualization

- $q_y q_x = \left(\cos\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \sin\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \cos\frac{\alpha}{2}\sin\frac{\beta}{2} \quad -\sin\frac{\alpha}{2}\sin\frac{\beta}{2}\right)$
- this corresponds to rotation around new axis $\hat{\boldsymbol{n}}$ and new angle γ with

$$\vec{\boldsymbol{n}} = \left(\sin\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \cos\frac{\alpha}{2}\sin\frac{\beta}{2} \quad -\sin\frac{\alpha}{2}\sin\frac{\beta}{2}\right),$$

$$\|\vec{\boldsymbol{n}}\|^2 = \sin^2\frac{\alpha}{2}\cos^2\frac{\beta}{2} + \cos^2\frac{\alpha}{2}\sin^2\frac{\beta}{2} + \sin^2\frac{\alpha}{2}\sin^2\frac{\beta}{2} \quad = \sin^2\frac{\alpha}{2}\cos^2\frac{\beta}{2} + \sin^2\frac{\beta}{2},$$

$$= 1 - \cos^2\frac{\beta}{2} + \sin^2\frac{\alpha}{2}\cos^2\frac{\beta}{2} = 1 - \cos^2\frac{\beta}{2}\left(1 - \sin^2\frac{\alpha}{2}\right) = 1 - \cos^2\frac{\beta}{2}\cos^2\frac{\alpha}{2}$$

$$\hat{\boldsymbol{n}} = \frac{1}{\sqrt{1 - \cos^2\frac{\beta}{2}\cos^2\frac{\alpha}{2}}}\left(\sin\frac{\alpha}{2}\cos\frac{\beta}{2} \quad \cos\frac{\alpha}{2}\sin\frac{\beta}{2} \quad -\sin\frac{\alpha}{2}\sin\frac{\beta}{2}\right),$$

- and angle

$$\gamma = \operatorname{arctan2}\left(\sqrt{1 - \cos^2\frac{\beta}{2}\cos^2\frac{\alpha}{2}}, \cos\frac{\alpha}{2}\cos\frac{\beta}{2}\right)$$

Conversions from and to matrix

- conversion to rotation matrix can be derived by transforming the base vectors $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{z}}$ with the quaternion and writing result in columns of rotation matrix:

$$\boldsymbol{R}\left(q=\begin{pmatrix}s\\x\\y\\z\end{pmatrix}\right)=\begin{pmatrix}s^2+x^2-y^2-z^2 & 2(xy-sz) & 2(xz+sy)\\ 2(xy+sz) & s^2-x^2+y^2-z^2 & 2(yz-sx)\\ 2(xz-sy) & 2(yz+sx) & s^2-x^2-y^2+z^2\end{pmatrix}$$

- conversion back to quaternion from diagonal elements of $\boldsymbol{R}$ and normalization constraint:

$$\begin{pmatrix}1&1&-1&-1\\1&-1&1&-1\\1&-1&-1&1\\1&1&1&1\end{pmatrix}\begin{pmatrix}s^2\\x^2\\y^2\\z^2\end{pmatrix}=\begin{pmatrix}R_{xx}\\R_{yy}\\R_{zz}\\1\end{pmatrix}\Rightarrow\begin{pmatrix}s^2\\x^2\\y^2\\z^2\end{pmatrix}=\frac{1}{4}\begin{pmatrix}1&1&1&1\\1&-1&-1&1\\-1&1&-1&1\\-1&-1&1&1\end{pmatrix}\begin{pmatrix}R_{xx}\\R_{yy}\\R_{zz}\\1\end{pmatrix}$$

- The signs of the components can be derived from (see)
$s=1; x=\mathrm{sgn}(R_{zy}-R_{yz}); \mathrm{y}=\mathrm{sgn}(R_{xz}-R_{zx}); \mathrm{z}=\mathrm{sgn}(R_{yx}-R_{xy});$

Conversions from and to matrix

- Example of derivation of x-column of rotation matrix:

$$\hat{q} = \left(s, \begin{pmatrix} x \\ y \\ z \end{pmatrix}\right) \qquad q_1 q_2 = (s_1 s_2 - \vec{\boldsymbol{v}}_1 \cdot \vec{\boldsymbol{v}}_2, s_1 \vec{\boldsymbol{v}}_2 + s_2 \vec{\boldsymbol{v}}_1 + \vec{\boldsymbol{v}}_1 \times \vec{\boldsymbol{v}}_2) \qquad \hat{q}^* = \left(s, \begin{pmatrix} -x \\ -y \\ -z \end{pmatrix}\right)$$

$$\hat{q}\left(0, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\right)\hat{q}^* = \left(-x, \begin{pmatrix} s \\ z \\ -y \end{pmatrix}\right)\hat{q}^* = \left(\begin{array}{c} \overbrace{-xs - (-xs - zy + yz)}^{0}, \\ \underbrace{-x\begin{pmatrix} -x \\ -y \\ -z \end{pmatrix} + s\begin{pmatrix} s \\ z \\ -y \end{pmatrix}} + \underbrace{\begin{pmatrix} s \\ z \\ -y \end{pmatrix} \times \begin{pmatrix} -x \\ -y \\ -z \end{pmatrix}} \end{array} \right)$$

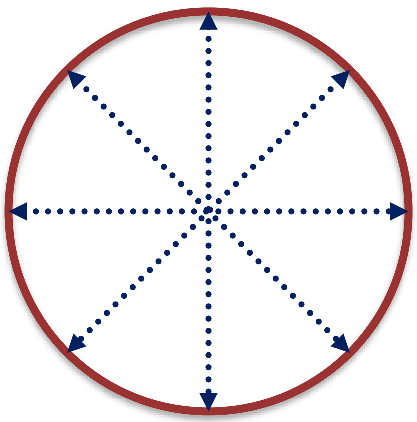
$$= \left(0, \begin{pmatrix} s^2 - x^2 \\ xy + sz \\ xz - sy \end{pmatrix} + \begin{pmatrix} y^2 - z^2 \\ xy + sz \\ xz - sy \end{pmatrix}\right)$$

$$= \left(0, \begin{pmatrix} s^2 - x^2 + y^2 - z^2 \\ 2(xy + sz) \\ 2(xz - sy) \end{pmatrix}\right)$$

Double Cover of Rotations

Computer Graphics and Visualization

- The axis-angle representation of rotations with $\alpha \in\ ]-\pi,\pi]$ is not unique, as
$$\boldsymbol{R}(\hat{\boldsymbol{n}},\alpha) = \boldsymbol{R}(-\hat{\boldsymbol{n}},-\alpha)$$
- Each rotation has two representations. We call this a double cover of the group of rotations
- The quaternions are also a double cover as
$$qpq^* = (-q)p(-q^*)$$
with
$$-q = \left(-\cos\frac{\alpha}{2} \quad -\sin\frac{\alpha}{2}\hat{\boldsymbol{n}}\right) =$$
$$\left(\cos\left(\pi-\frac{\alpha}{2}\right) \quad \sin\left(\pi-\frac{\alpha}{2}\right)(-\hat{\boldsymbol{n}})\right) =$$
$$\left(\cos\frac{2\pi-\alpha}{2} \quad \sin\frac{2\pi-\alpha}{2}(-\hat{\boldsymbol{n}})\right)$$
- On the 3-unit sphere in 4D space the unit quaternions that are related by point reflection at origin represent same rotation



2D

Pktrep.	Rotrep.	Transf.

- 2x2 Matrix

$\vec{\boldsymbol{p}} = \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$	$\boldsymbol{T}_{\text{rot}}\, \vec{\boldsymbol{p}}$

- komplexe Zahlen

$z = x + iy$	$e^{i\alpha} = \cos\alpha + i\sin\alpha$	$ze^{i\alpha} = e^{i\alpha}z$

3D

Pktrep.	Rotrep.	Transf.

- 3x3 Matrix

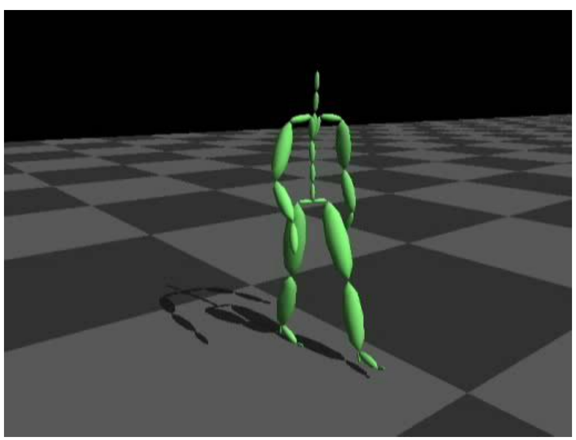
$\vec{\boldsymbol{p}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$	$\boldsymbol{T}_{\text{rot}}\, \vec{\boldsymbol{p}}$

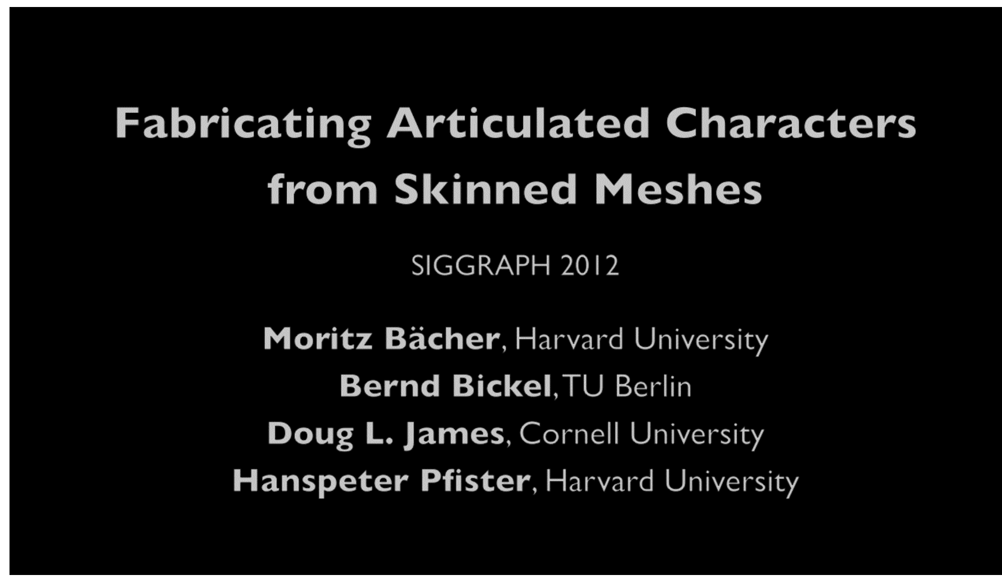
- Quaternionen

$p = (0, \vec{\boldsymbol{p}}) = xi + yj + zk$	$q = \left(\cos\tfrac{\alpha}{2}, \hat{\boldsymbol{n}}\sin\tfrac{\alpha}{2}\right) = \cos\tfrac{\alpha}{2} + \sin\tfrac{\alpha}{2}\left(n_x i + n_y j + n_z k\right)$	$qpq^{-1} = qpq^{*}$

ARTICULATED OBJECTS

Examples





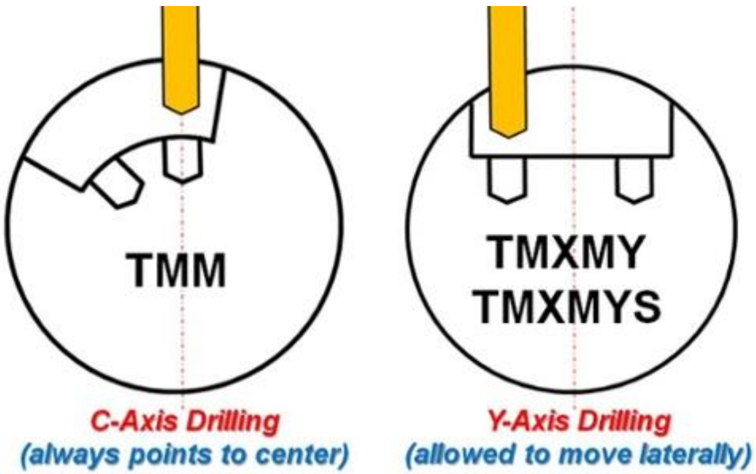
Fabricating Articulated Characters
using Skinned Meshes, Siggraph 2012

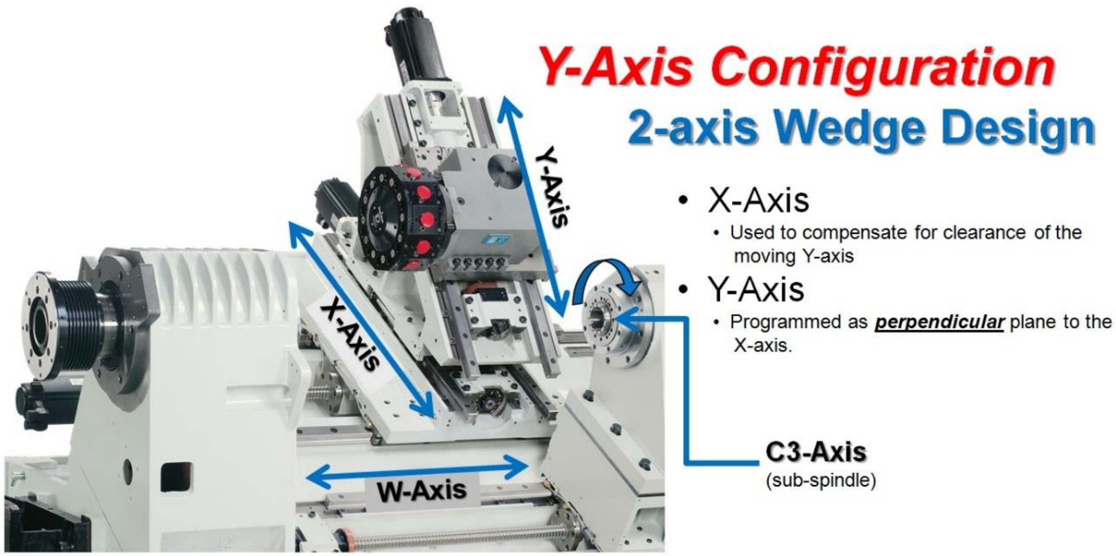
Motivation – CNC-Milling Machines

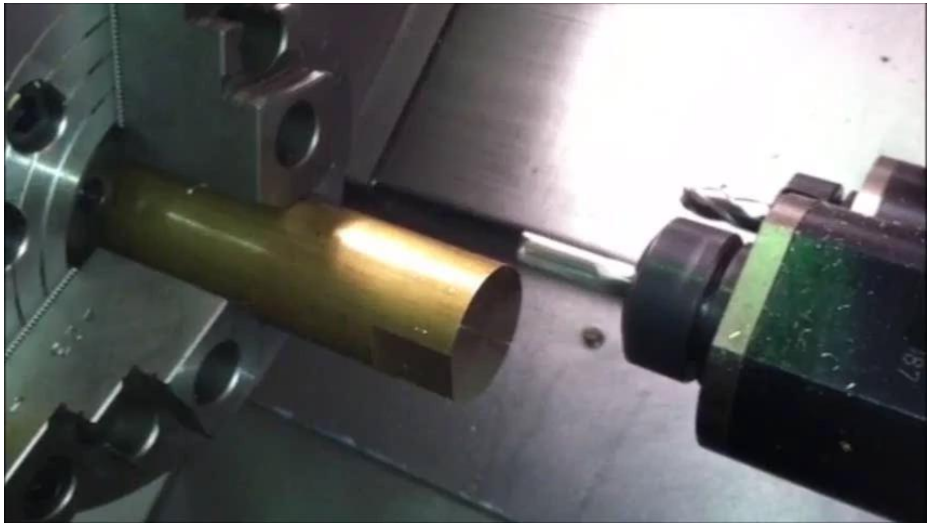
Computer Graphics and Visualization

X&C3-Axes only

Y-Axis needed







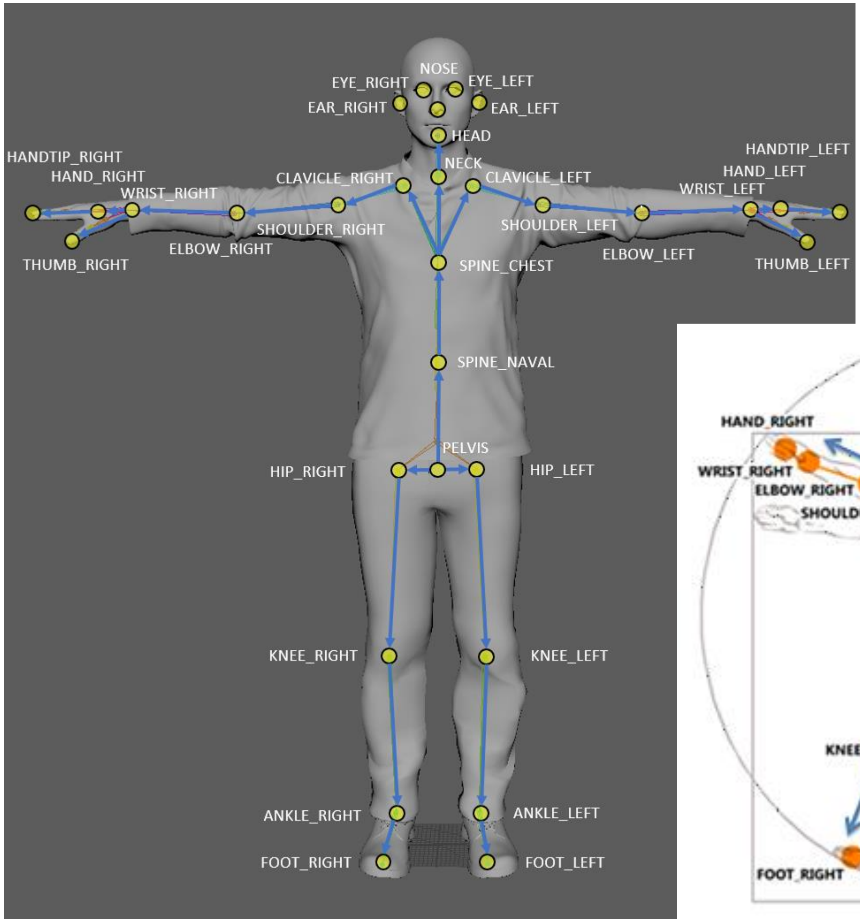
X&C3-Axes only flats

X&C3-Axes only circle

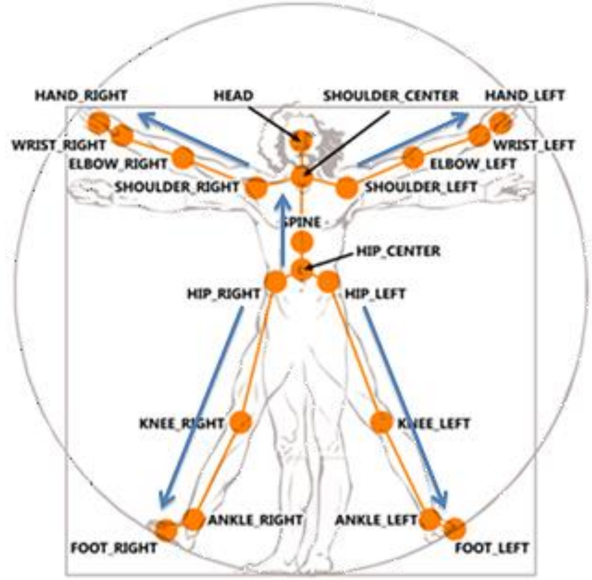
http://blog.hurco.com/blog/bid/281989/An-Introduction-to-Mill-Turn-Technology

Motivation – Skeletal Animation

biped body tracking



kinect azure skeleton



kinect 1.0 skeleton

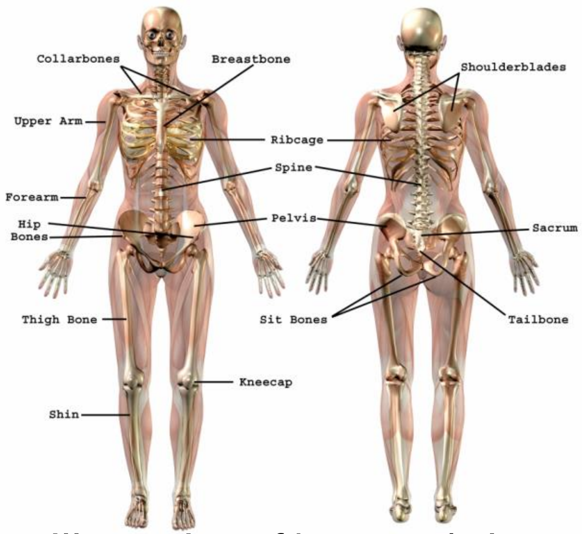
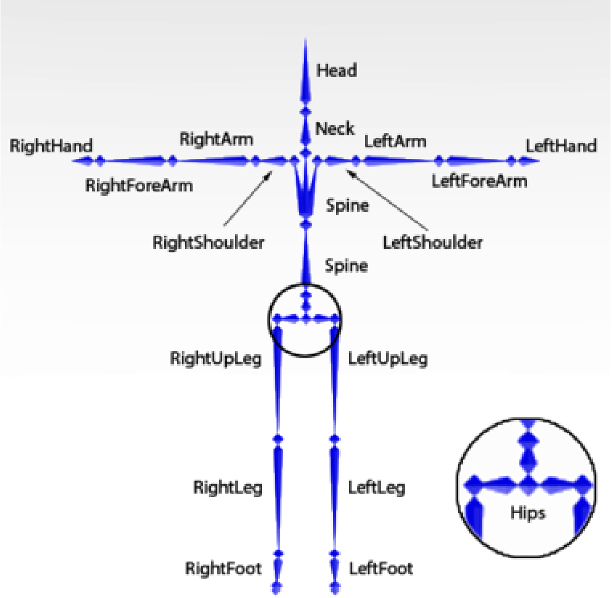


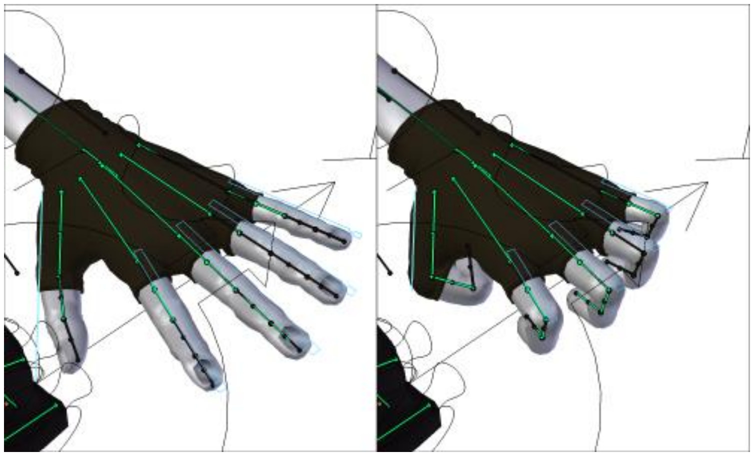
illustration of human skeleton



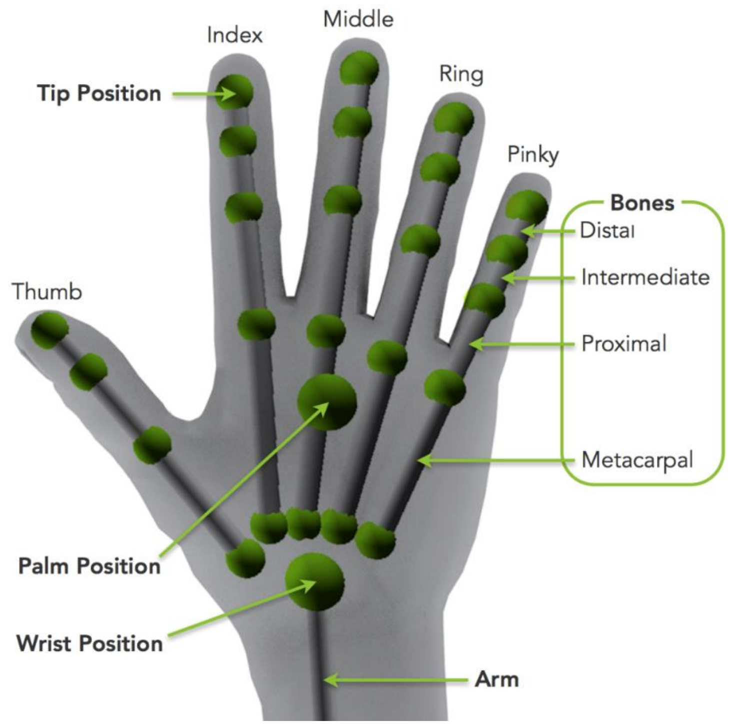
BVH skeleton
(mocap file format: Biovision hierarchical data)

Motivation – Skeletal Animation

hand tracking



© wikipedia

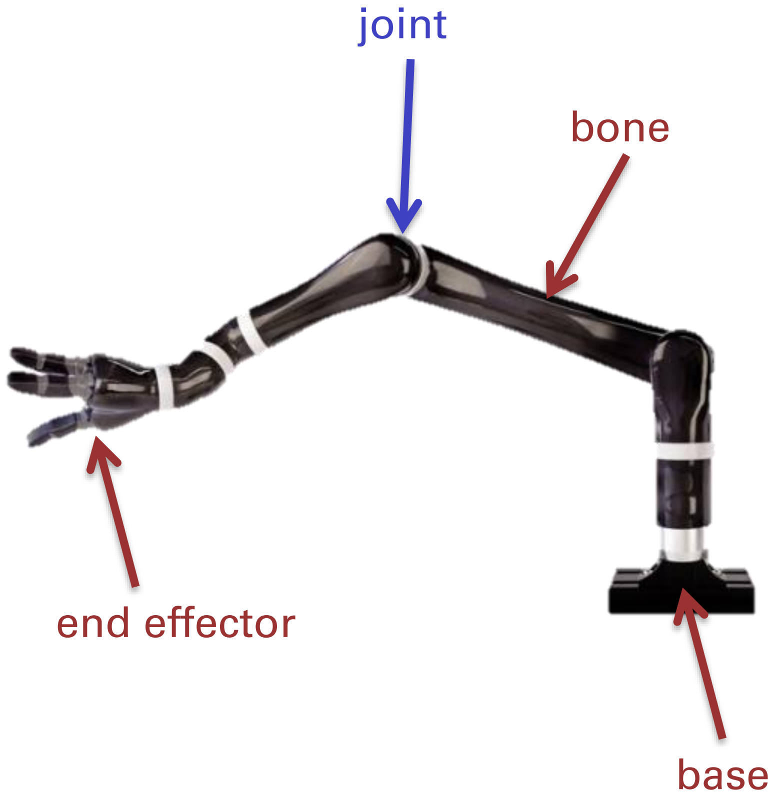


leap motion hand skeleton

Other applications: facial animations

Kinematic Chain – Definition

- bone/limb/link corresponds to a stiff part and a bone coordinate system
- the arm is fixed at the first bone, which is called base
- the last bone is also called end effector and used for example for grabbing
- joints connect two bones and often have an own coordinate system aligned with their rotation axis
- bones and joints form a kinematic chain



Robot arms with Bones and Joints

Kinematic Chain – Coordinate Systems

- In robotics and milling the most basic joint types are revolute and prismatic joints with one axis each
- per bone three coordinate systems are defined:
  - input joint (subscript I) is reference coordinate system of bone
  - bone (subscript B) is used to place bone geometry
  - output joint (subscript O) is used to connect next bone
- joint coordinate systems are aligned with joint axis

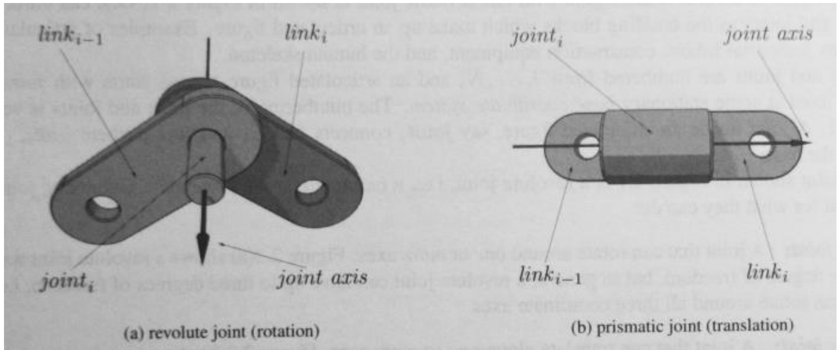


Figure 2.2: Examples of different joints used to construct articulated figures.

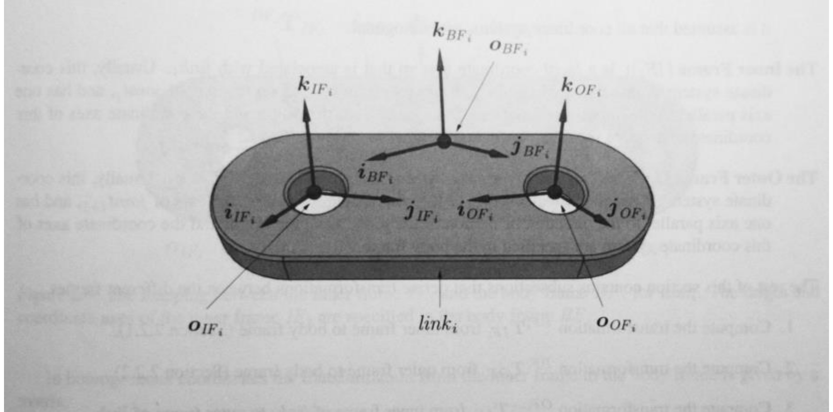


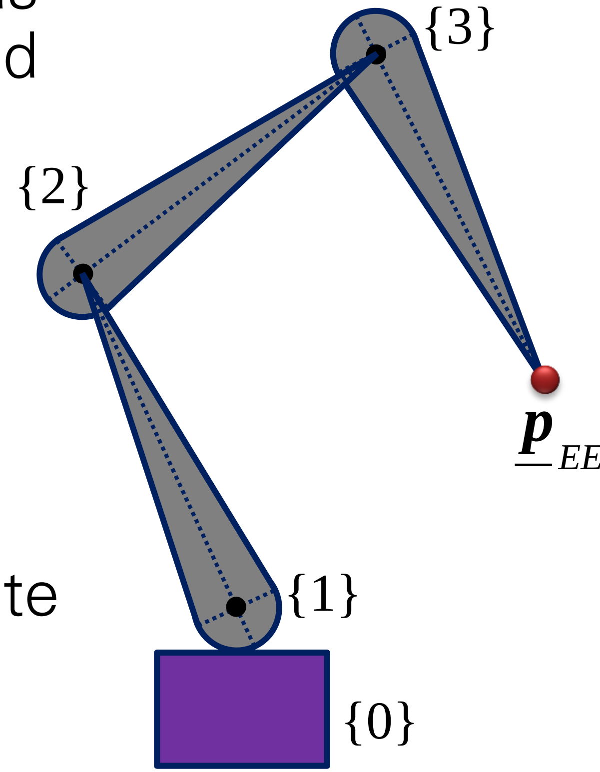
Figure 2.3: The three coordinate systems associated with each $link_i$: BF_i, IF_i, and OF_i. The origins and coordinate axes of the inner-frame IF_i, and the outer-frame OF_i are specified in the body-frame BF_i.

Kinematic Chain – Coordinate Systems

- input joint coordinate systems are used as reference for base / bone and enumerated from 0 (base/world) to N (end effector
- Transformations are composed along kinematic chain

$$ {}^{0}\boldsymbol{T}_N = {}^{0}\boldsymbol{T}_1 \cdot {}^{1}\boldsymbol{T}_2 \cdot \ldots \cdot {}^{N-1}\boldsymbol{T}_N $$

- *model transform view:* place bones from base to end effector
- *system transform view:* convert coordinate system from end effector to base
- This can be further refined into

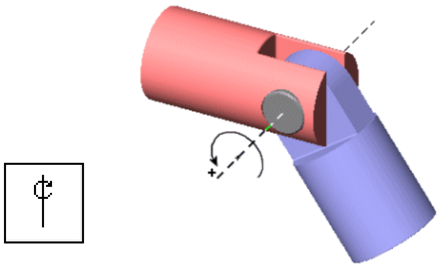


local joint transformations

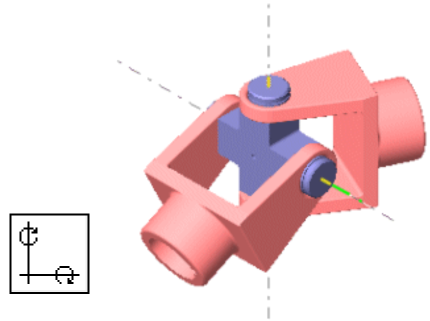
$$ \boldsymbol{T}_{\text{chain}} = \overbrace{{}^{\text{world}}\boldsymbol{T}_{OF_0} \cdot {}^{OF_0}\boldsymbol{T}_{IF_1}}^{{}^{0}\boldsymbol{T}_1} \cdot \overbrace{{}^{IF_1}\boldsymbol{T}_{BF_1} \cdot {}^{BF_1}\boldsymbol{T}_{OF_1} \cdot {}^{OF_1}\boldsymbol{T}_{IF_2}}^{{}^{1}\boldsymbol{T}_2} \cdot \overbrace{{}^{IF_2}\boldsymbol{T}_{BF_2} \cdot {}^{BF_2}\boldsymbol{T}_{OF_2} \cdot {}^{OF_2}\boldsymbol{T}_{IF_3}}^{{}^{2}\boldsymbol{T}_3} \cdots {}^{IF_{\text{end}}}\boldsymbol{T}_{BF_{\text{end}}} $$

dependent on joint parameters

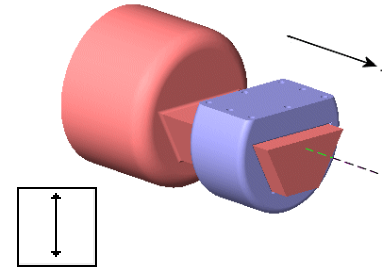
Basic Joint Types



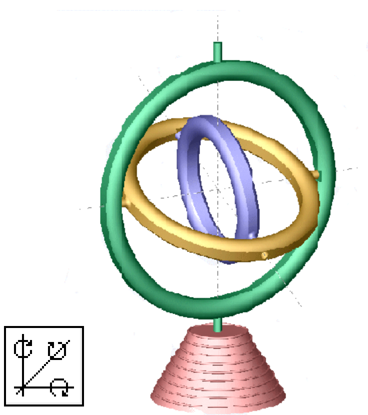
1R Revolute



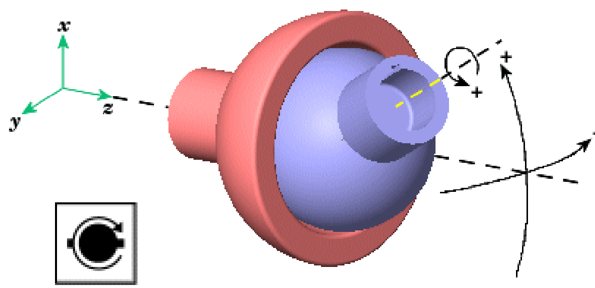
2R Universal



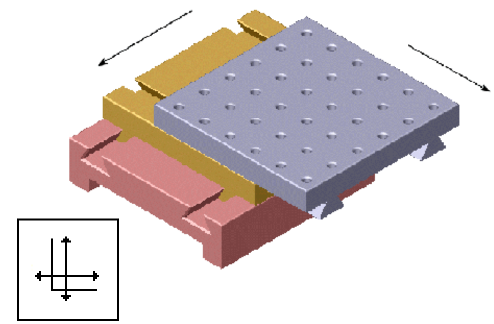
1T prismatic



3R Gimbal



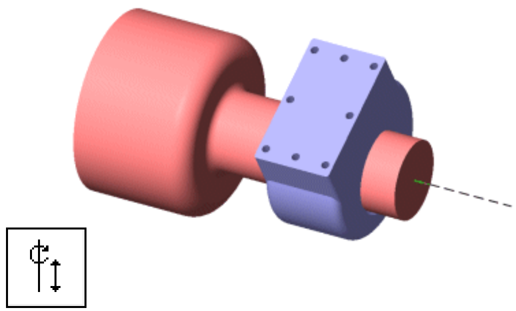
3R Spherical



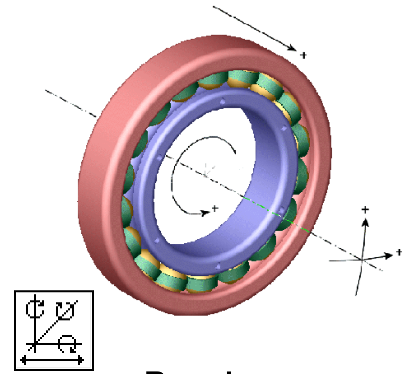
2T in-planar

http://www.mathworks.de/de/help/physmod/sm/assembled-joints.html

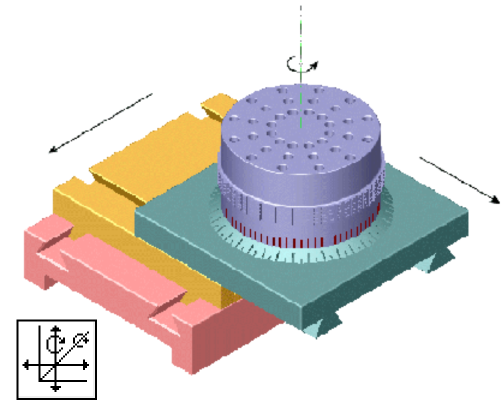
Special Joint Types



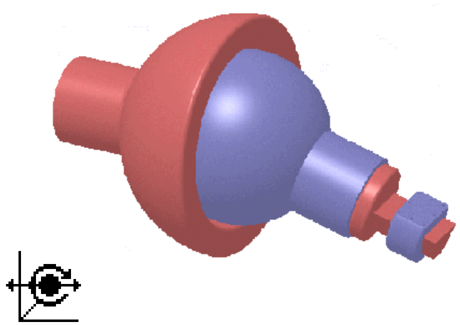
1R1T Cylindrical



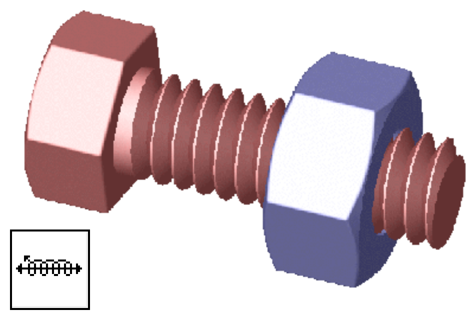
3R1T Bearing



1R2T planar

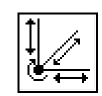


3R1T Telescoping



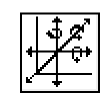
Screw

Six-DoF



3R3T

Bushing



3R3T

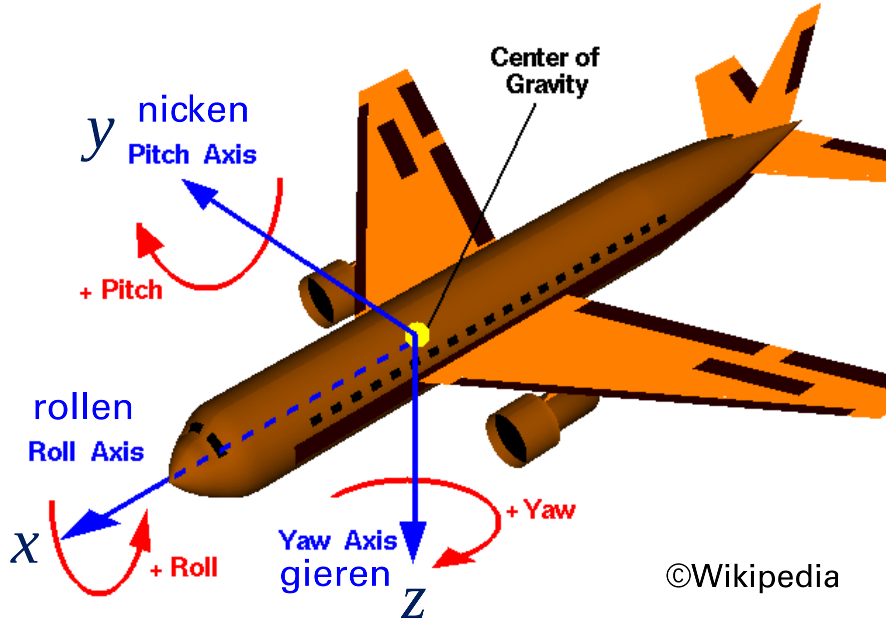
http://www.mathworks.de/de/help/physmod/sm/assembled-joints.html

Euler Angle Representation of Orientation

Roll-Pitch-Yaw

- An arbitrary rotation is defined by 3 free parameters
- They can be defined by 3 rotation angles which are called Euler angles
- Coming from aironautics, the terms roll (x), pitch (y) and yaw (z) are commonly used

$$\boldsymbol{R}_{\text{roll-pitch-yaw}} = \boldsymbol{R}_Z(\phi_{\text{yaw}})\boldsymbol{R}_Y(\phi_{\text{pitch}})\boldsymbol{R}_X(\phi_{\text{roll}})$$

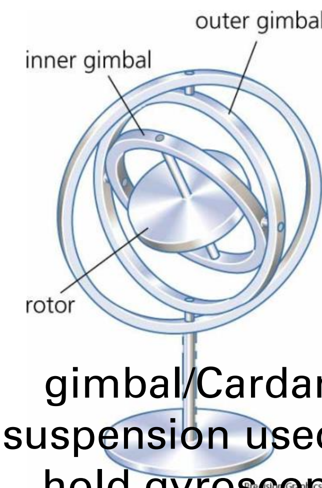


©Wikipedia

Navigation using gyroscopes

- Commonly used: 313-Convention
- The first and third axis can become parallel, thus reducing one degree of freedom. This is called "gimbal lock".

$$\boldsymbol{R}_{313}(\alpha,\beta,\gamma) = \boldsymbol{R}_Z(\alpha)\boldsymbol{R}_X(\beta)\boldsymbol{R}_Z(\gamma)$$



gimbal/Cardan suspension used to hold gyroscope


gimbal lock only 2R left

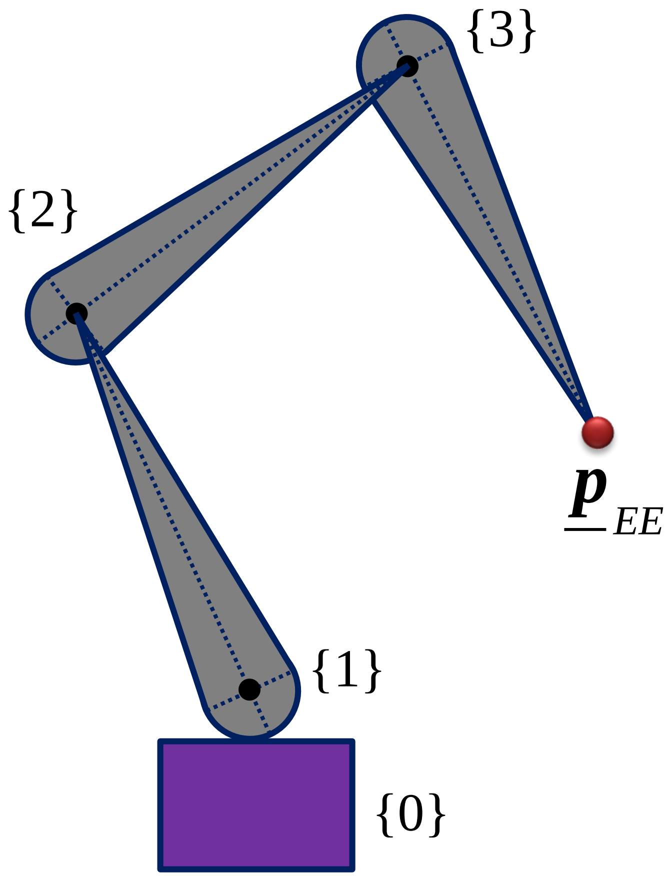
Forward Kinematics

- Given a kinematic chain (robot arm or path in skeleton) with relative transformations ${}^{(i-1)}T_i\,(q_{ik})$ depending on parameters q_{ik} location and orientation of the end effector in world coordinates are a function of the q_{ik} also:

$$\underline{\boldsymbol{p}}^{0}_{EE} = {}^{0}\boldsymbol{T}_N\,\underline{\boldsymbol{p}}^{N}_{EE} = \underline{\boldsymbol{f}}\left(q_{ik}\right)$$

$$\boldsymbol{\omega}^{0}_{EE} = R^{-1}_{313}\left({}^{0}\boldsymbol{T}_N \Big|_{\underline{xyz}} \right) = \boldsymbol{F}\left(q_{ik}\right)$$

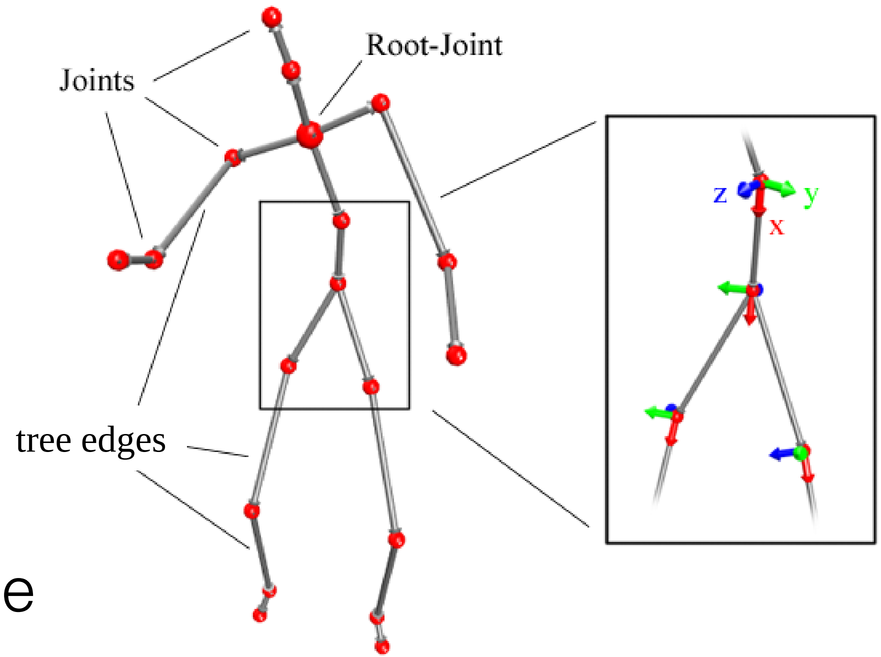
Orientation for example given as Euler angles and computed from 3x3-rotation matrix



$${}^{0}\boldsymbol{T}_N = {}^{0}\boldsymbol{T}_1 \cdot {}^{1}\boldsymbol{T}_2 \cdot \ldots \cdot {}^{N-1}\boldsymbol{T}_N$$

Kinematic Tree / Skeleton

- a skeleton is a kinematic tree structure with joints as nodes and bones along edges.
- it has a single root joint and several end effectors
- at each joint i a joint coordinate frame F_i is defined
- Local joint transformations ${}^{p(i)}\boldsymbol{T}_i$ map from parent frame $F_{p(i)}$ to F_i with a rigid body transformation
- together all local joint transforms define the pose of the skeleton



Representation of Local joint transformations

- In the Denavit-Hartenberg notation for each link there is one adjustible parameter q_{ik} corresponding to d_i or φ_i depending on the joint type (prismatic or revolution)

$$
{}^{i-1}\boldsymbol{T}_i(d_i \vee \varphi_i) = \begin{pmatrix} \cos\varphi_i & -\sin\varphi_i & 0 & a_{i-1} \\ \sin\varphi_i \cos\alpha_{i-1} & \cos\varphi_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\varphi_i \sin\alpha_{i-1} & \cos\varphi_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

- Using Euler angles one has 6 parameters

$$
{}^{i-1}\boldsymbol{T}_i(\alpha_i, \beta_i, \gamma_i, \vec{\boldsymbol{t}}_i) = \boldsymbol{T}(\vec{\boldsymbol{t}}_i)\boldsymbol{R}_Z(\gamma_i)\boldsymbol{R}_X(\beta_i)\boldsymbol{R}_Z(\alpha_i) =
$$

$$
\begin{bmatrix} \cos(\gamma_i)\cos(\alpha_i) - \sin(\gamma_i)\cos(\beta_i)\sin(\alpha_i) & -\cos(\gamma_i)\sin(\alpha_i) - \sin(\gamma_i)\cos(\beta_i)\cos(\alpha_i) & \sin(\gamma_i)\sin(\beta_i) & t_x \\ \sin(\gamma_i)\cos(\alpha_i) + \cos(\gamma_i)\cos(\beta_i)\sin(\alpha_i) & -\sin(\gamma_i)\sin(\alpha_i) + \cos(\gamma_i)\cos(\beta_i)\cos(\alpha_i) & -\cos(\gamma_i)\sin(\beta_i) & t_y \\ \sin(\beta_i)\sin(\alpha_i) & \sin(\beta_i)\cos(\alpha_i) & \cos(\beta_i) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

- Using quaternions one has 7 parameters plus one normalization constraint

$$
s^2 + x^2 + y^2 + z^2 = 1
$$

$$
{}^{i-1}\boldsymbol{T}_i(q_i = (s, x, y, z), \vec{\boldsymbol{t}}_i) =
$$

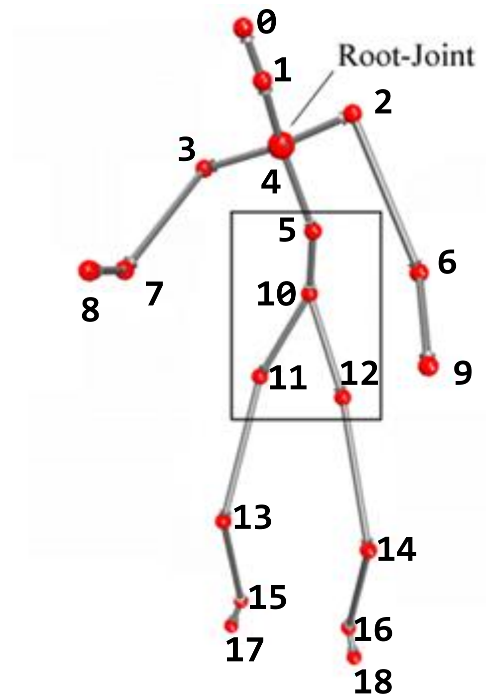
$$
\begin{pmatrix} 1-2y^2-2z^2 & 2xy-2sz & 2xz+2ys & t_x \\ 2xy+2sz & 1-2x^2-2z^2 & 2yz-2sx & t_y \\ 2xz-2sy & 2yz+2sx & 1-2x^2-2y^2 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Computing Joint to World Transforms

Computer Graphics and Visualization

joint index	4	3	2	1	5	7	6	0	10	8	9	11	12	13	14	15	16	17	18
p(i)	-1	0	0	0	0	1	2	3	4	5	6	8	8	11	12	13	14	15	16

- the skeleton tree can be linearized in breadth or depth first traversal
- for rendering we need for each joint the joint to world transformation ${}^{0}\boldsymbol{T}_i$
- these transformations can be stored linearly in breadth or depth first order
- Both orders guarantee that parent to world transformation is computed before joint to world transformation, allowing for sequential computation:



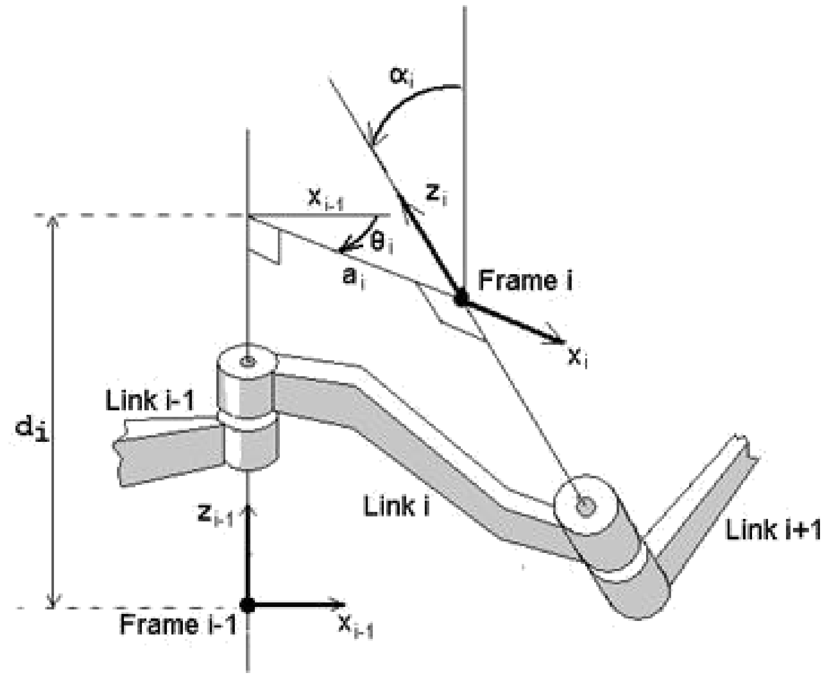

```
initialize ⁰T₁
for i from 1 to n do
```

$${}^{0}\boldsymbol{T}_i = {}^{0}\boldsymbol{T}_{p(i)}\,{}^{p(i)}\boldsymbol{T}_i(q_{ik})$$

DENAVIT-HARTENBERG NOTATION

Denavit-Hartenberg notation

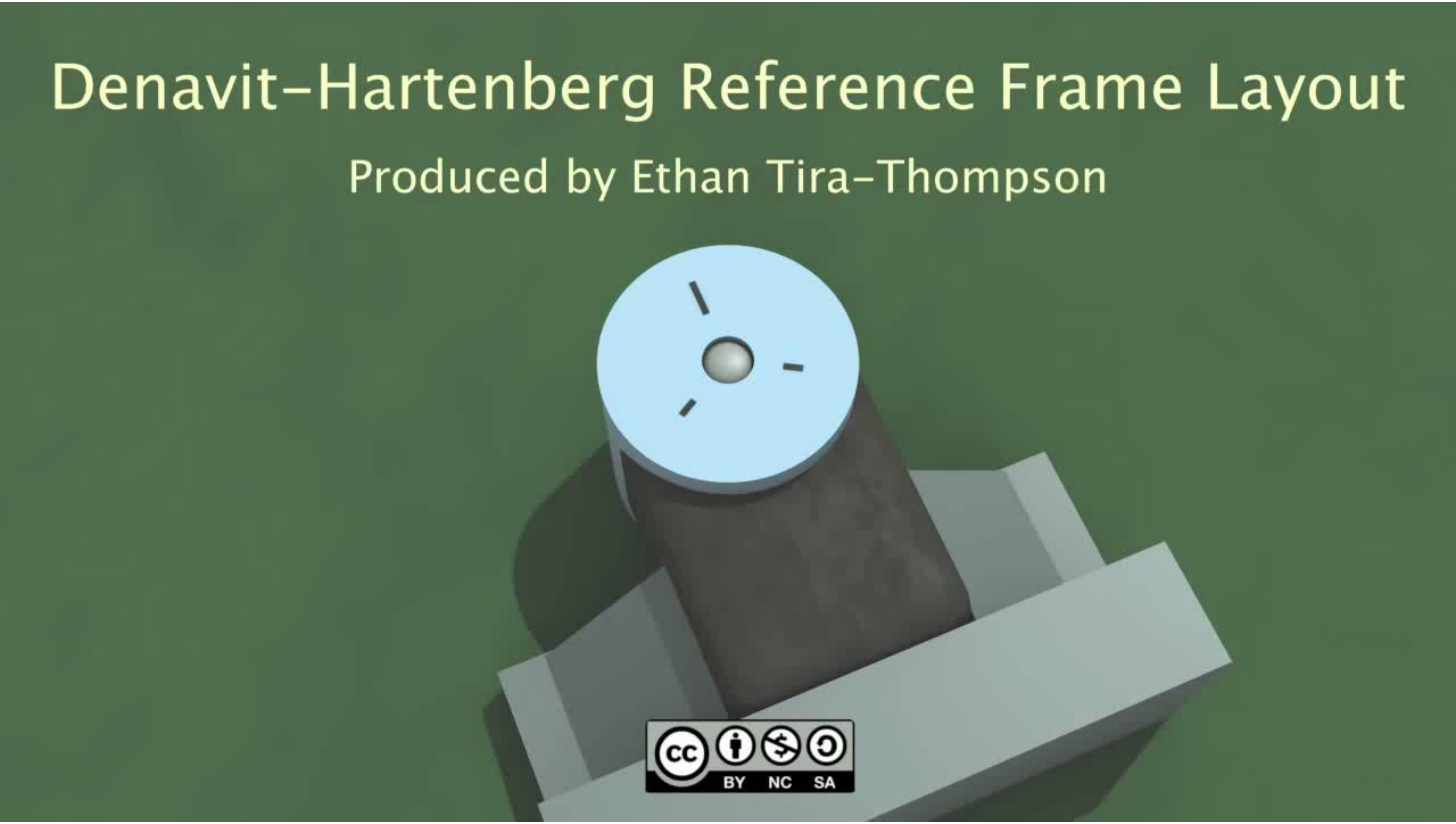
- **input:** joint axes $\underline{\boldsymbol{p}}_i + \lambda \cdot \hat{\boldsymbol{z}}_i$
- **output:** joint input coordinate frames $\underline{\boldsymbol{o}}_i, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i$ and four parameters d_i, θ_i, a_i and α_i per joint:
  - $d_i \ldots$ is the Euclidean distance along axis $\hat{\boldsymbol{z}}_{i-1}$ to the point where the common perpendicular intersects axis $\hat{\boldsymbol{z}}_{i-1}$. (parameter of prismatic variable)
  - $\theta_i \ldots$ joint angle / rotation angle around $\hat{\boldsymbol{z}}_{i-1}$ that rotates $\hat{\boldsymbol{x}}_{i-1}$ axis onto $\hat{\boldsymbol{x}}_i$ axis (parameter of revolute joint)
  - $a_i \ldots$ link length / perp. distance between joint axes
  - $\alpha_i \ldots$ link twist / rotation angle between joint axes (around $\hat{\boldsymbol{x}}_i$)



➔ $^{i-1}\boldsymbol{T}_i = \mathrm{Rot}_{\boldsymbol{z}}(\theta_i) \cdot \mathrm{Trans}_{\boldsymbol{z}}(d_i) \cdot \mathrm{Trans}_{\boldsymbol{x}}(a_i) \cdot \mathrm{Rot}_{\boldsymbol{x}}(\alpha_i)$

- x-axis of base can be chosen freely

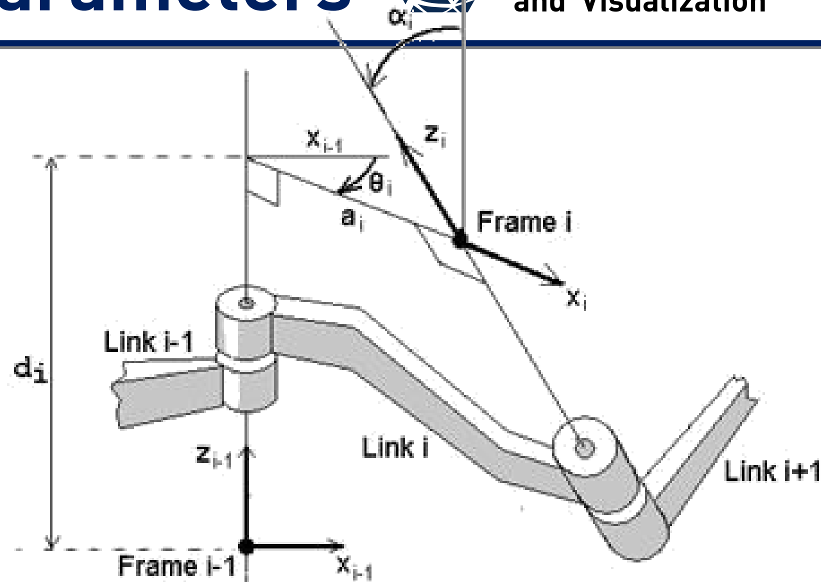
Short Illustration



here a_i, α_i, d_i, φ_i are denoted as r, α, d, θ_i

https://www.youtube.com/watch?v=rA9tm0gTln8

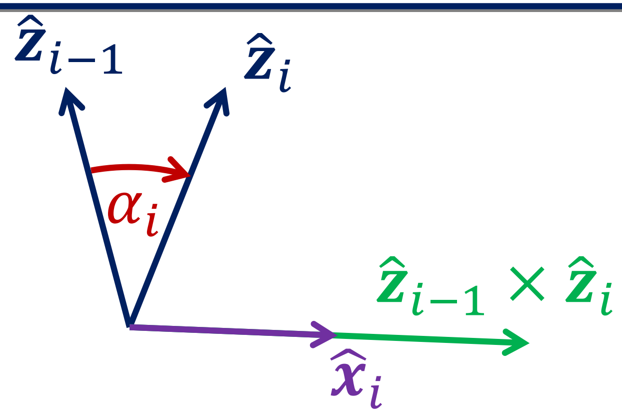
DH Frames and Translation Parameters

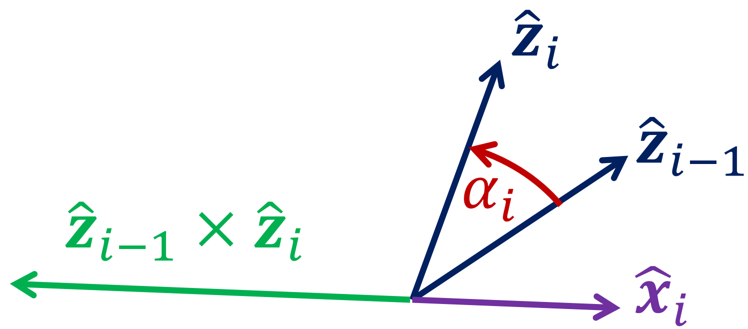


- new $\hat{\boldsymbol{x}}_i$ axis is perpendicular to both $\hat{\boldsymbol{z}}$ axes:
  $$\hat{\boldsymbol{x}}_i = \pm\text{normalize}(\hat{\boldsymbol{z}}_{i-1} \times \hat{\boldsymbol{z}}_i)$$
- sign of $\hat{\boldsymbol{x}}_i$ is determined from constraint that $a_i > 0$, where a_i is the projected distance from $\underline{\boldsymbol{p}}_i$ to $\underline{\boldsymbol{p}}_{i+1}$: $a_i = \langle \underline{\boldsymbol{p}}_{i+1} - \underline{\boldsymbol{p}}_i, \hat{\boldsymbol{x}}_i \rangle$
- we get from origin $\underline{\boldsymbol{o}}_{i-1}$ to $\underline{\boldsymbol{o}}_i$ along the path
  $$\underline{\boldsymbol{o}}_i = \underline{\boldsymbol{o}}_{i-1} + d_i \hat{\boldsymbol{z}}_{i-1} + a_i \hat{\boldsymbol{x}}_i$$
- $\underline{\boldsymbol{o}}_i$ is on axis i: $\underline{\boldsymbol{o}}_i = \underline{\boldsymbol{p}}_i + \lambda \cdot \hat{\boldsymbol{z}}_i = \underline{\boldsymbol{o}}_{i-1} + d_i \hat{\boldsymbol{z}}_{i-1} + a_i \hat{\boldsymbol{x}}_i$
- we can compute d_i and λ by forming triple products:
  $$d_i = \left\langle \underline{\boldsymbol{p}}_i - \underline{\boldsymbol{o}}_{i-1}, \hat{\boldsymbol{z}}_i \times \hat{\boldsymbol{x}}_i \right\rangle / \langle \hat{\boldsymbol{z}}_{i-1}, \hat{\boldsymbol{z}}_i \times \hat{\boldsymbol{x}}_i \rangle$$
  $$\lambda = \left\langle \underline{\boldsymbol{o}}_{i-1} - \underline{\boldsymbol{p}}_i, \hat{\boldsymbol{z}}_{i-1} \times \hat{\boldsymbol{x}}_i \right\rangle / \langle \hat{\boldsymbol{z}}_i, \hat{\boldsymbol{z}}_{i-1} \times \hat{\boldsymbol{x}}_i \rangle$$
- The frames are completed with $\hat{\boldsymbol{y}}_i = \hat{\boldsymbol{z}}_i \times \hat{\boldsymbol{x}}_i$

DH 360° Angle Computation

- Take care when computing angles via **`arctan2`** through $\sin\alpha_i = \|\hat{\mathbf{z}}_{i-1} \times \hat{\mathbf{z}}_i\|$ and $\cos\alpha_i = \langle \hat{\mathbf{z}}_{i-1}, \hat{\mathbf{z}}_i \rangle$
- As the sine is always positive, the range of α_i is $[0, \pi]$
- One needs to determine the sign of α_i from the sign of $\langle \hat{\mathbf{z}}_{i-1} \times \hat{\mathbf{z}}_i, \hat{\boldsymbol{x}}_i \rangle$, i.e.





$$\alpha_i = \operatorname{sgn}(\langle \hat{\mathbf{z}}_{i-1} \times \hat{\mathbf{z}}_i, \hat{\boldsymbol{x}}_i \rangle) \cdot \operatorname{arctan2}(\|\hat{\mathbf{z}}_{i-1} \times \hat{\mathbf{z}}_i\|, \langle \hat{\mathbf{z}}_{i-1}, \hat{\mathbf{z}}_i \rangle)$$

- Similarly one gets $\theta_i = \operatorname{sgn}(\langle \hat{\boldsymbol{x}}_{i-1} \times \hat{\boldsymbol{x}}_i, \hat{\mathbf{z}}_{i-1} \rangle) \cdot \operatorname{arctan2}(\|\hat{\boldsymbol{x}}_{i-1} \times \hat{\boldsymbol{x}}_i\|, \langle \hat{\boldsymbol{x}}_{i-1}, \hat{\boldsymbol{x}}_i \rangle)$

References

- [Spong] … Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control (2nd Edition), 2004, Chapter 3 – Forward Kinematics: DH Convention
- [Bächer] … Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. 2012. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph.* 31, 4, Article 47 (July 2012), 9 pages. DOI: https://doi.org/10.1145/2185520.2185543