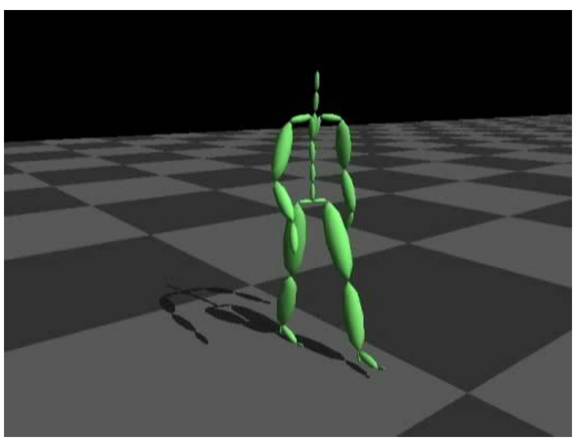
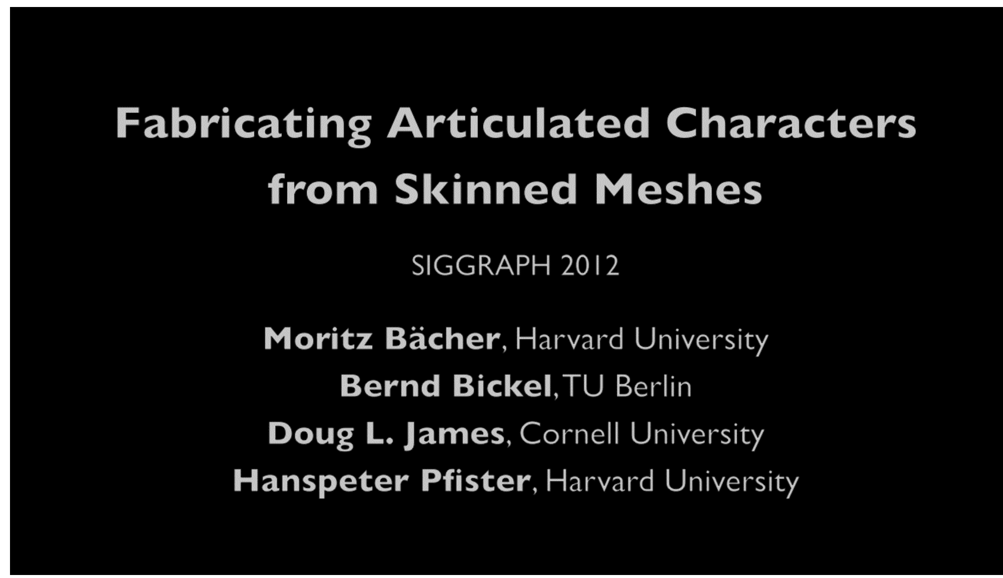
Articulated Objects

Examples

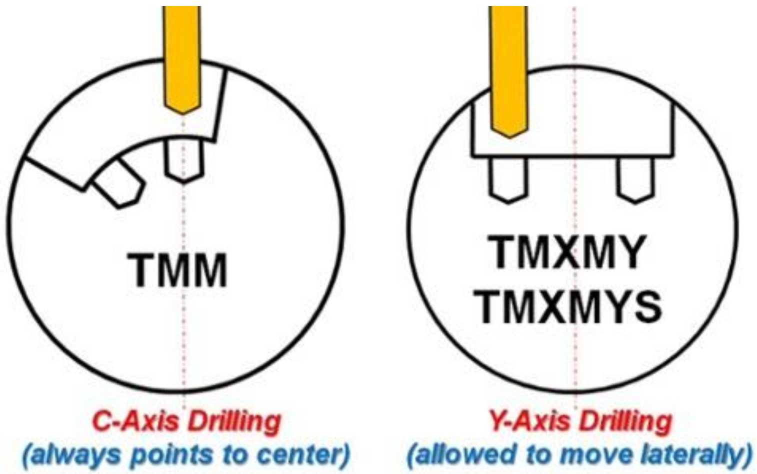


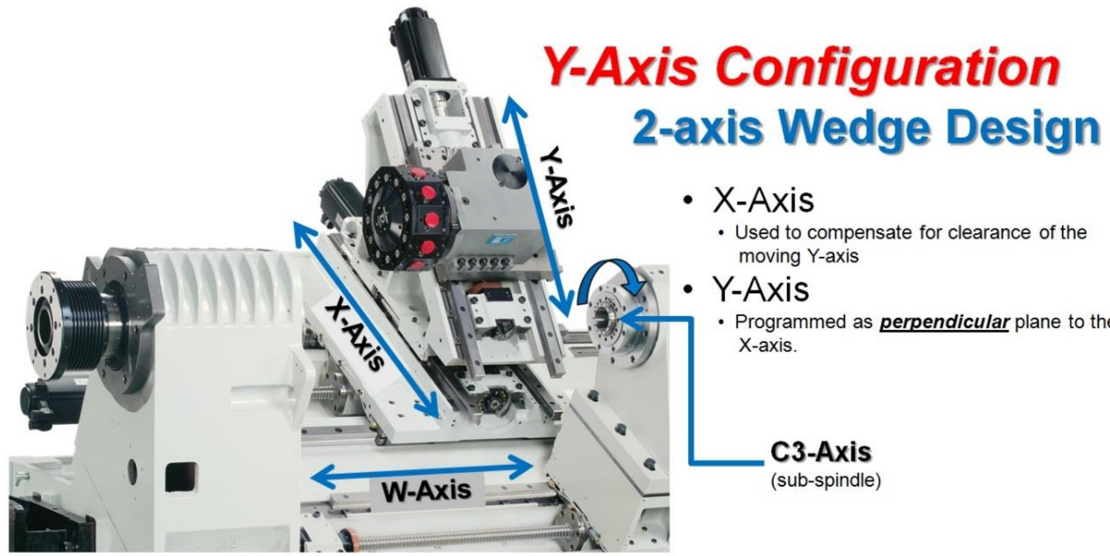


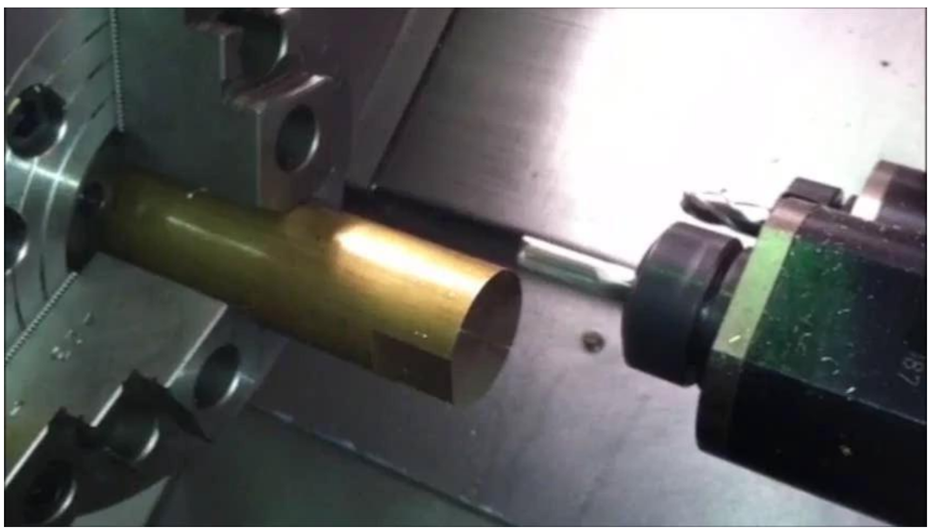
Fabricating Articulated Characters
using Skinned Meshes, Siggraph 2012

Motivation – CNC-Milling Machines

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

Computer Graphics and Visualization

biped body tracking

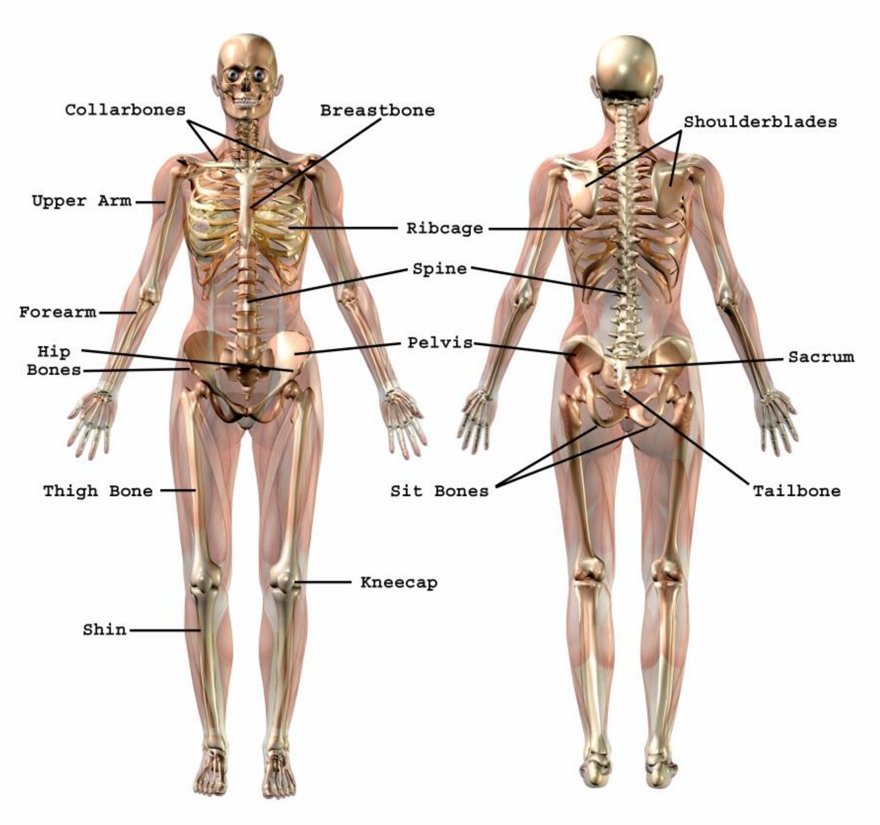
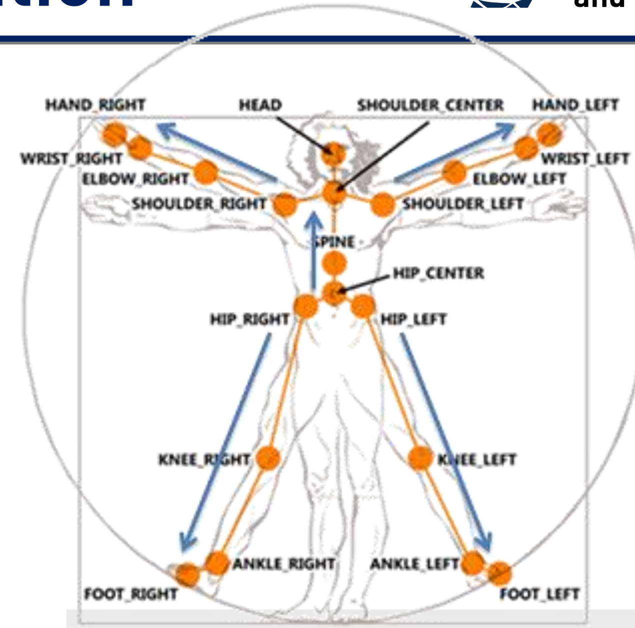
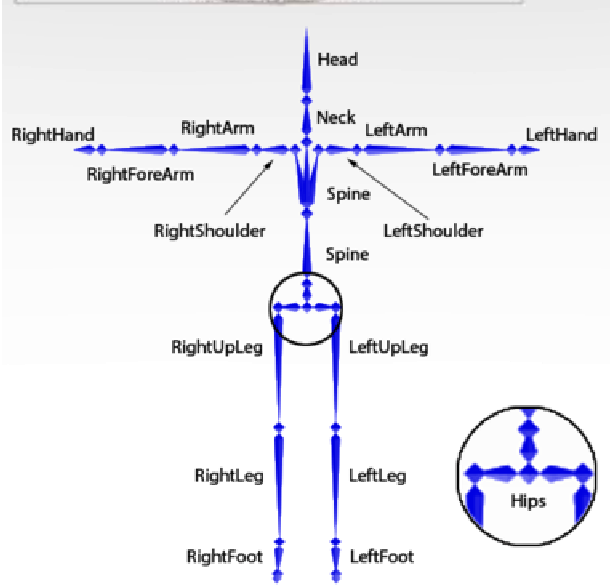


illustration of human skeleton



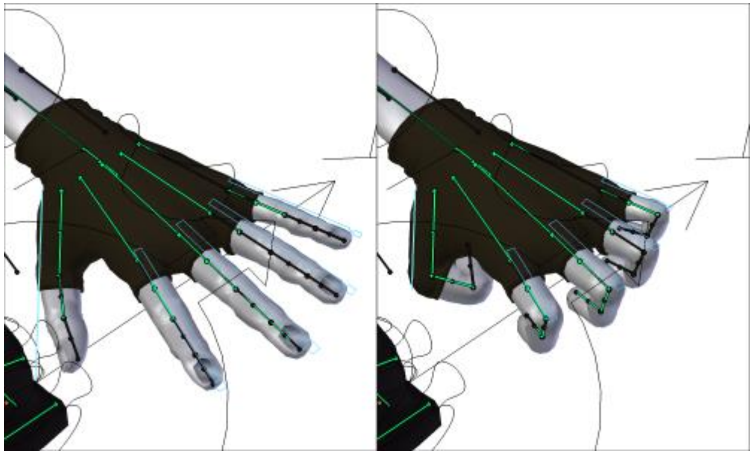
kinect skeleton



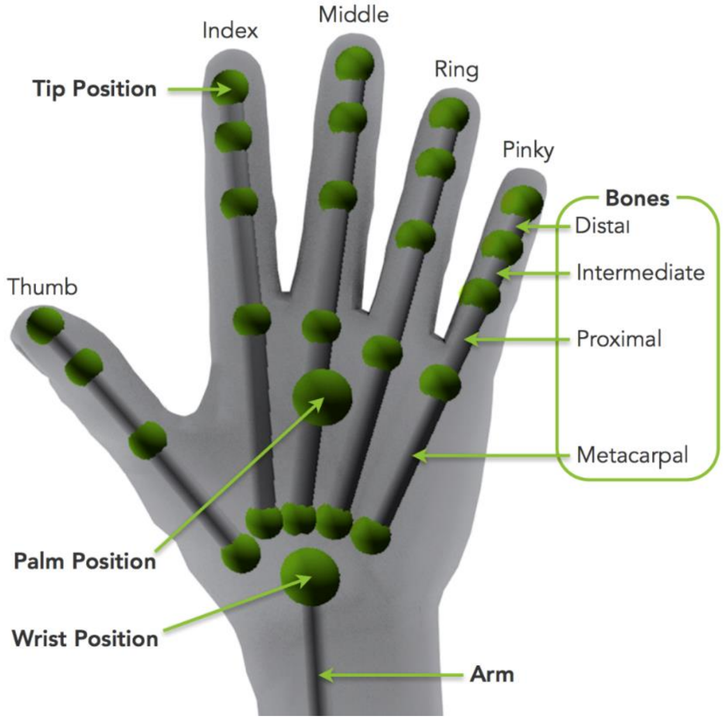
BVH skeleton (mocap file format: Biovision hierarchical data)

Motivation – Skeletal Animation

hand tracking



© wikipedia

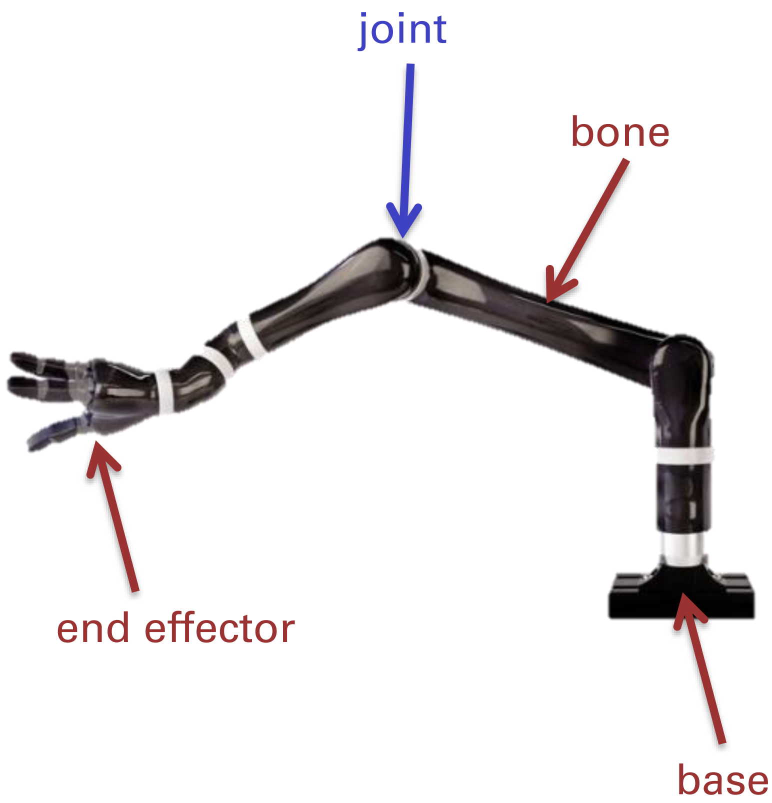


leap motion hand skeleton

Other applications: facial animations

Kinematic Chain – Definition

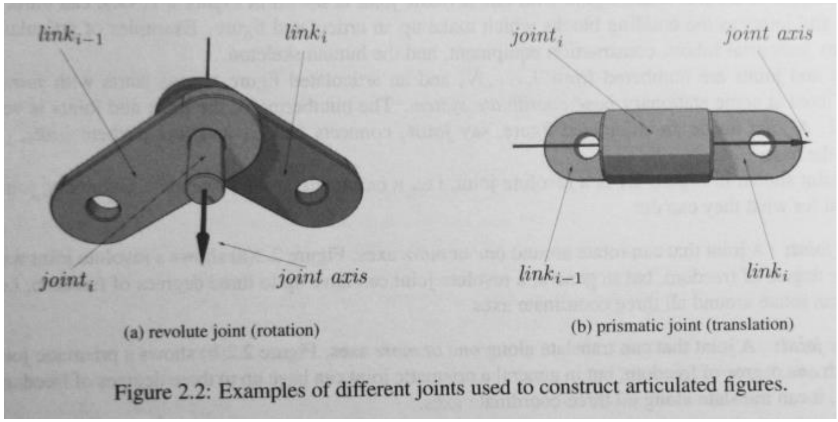
- bone/limb/link corresponds to a stiff part and a bone coordinate system
- the arm is fixed at the first bone, which is called base
- the last bone is also called end effector and used for example for grabbing
- joints connect two bones and often have an own coordinate system aligned with their rotation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

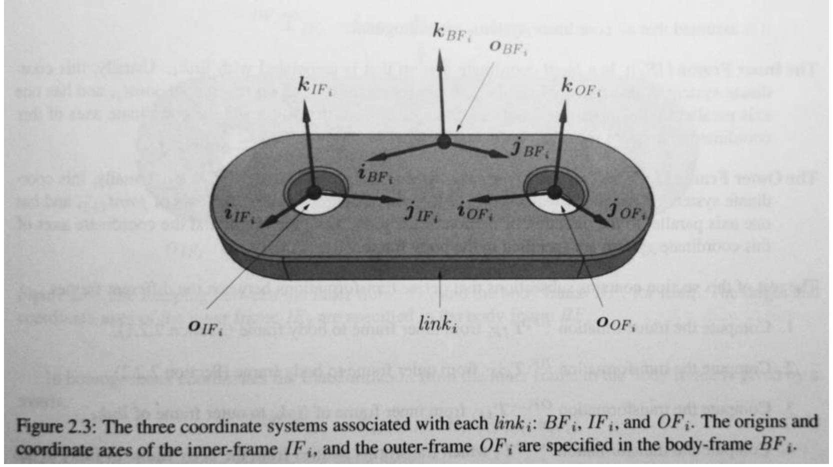


Robot arms with Bones and Joints

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





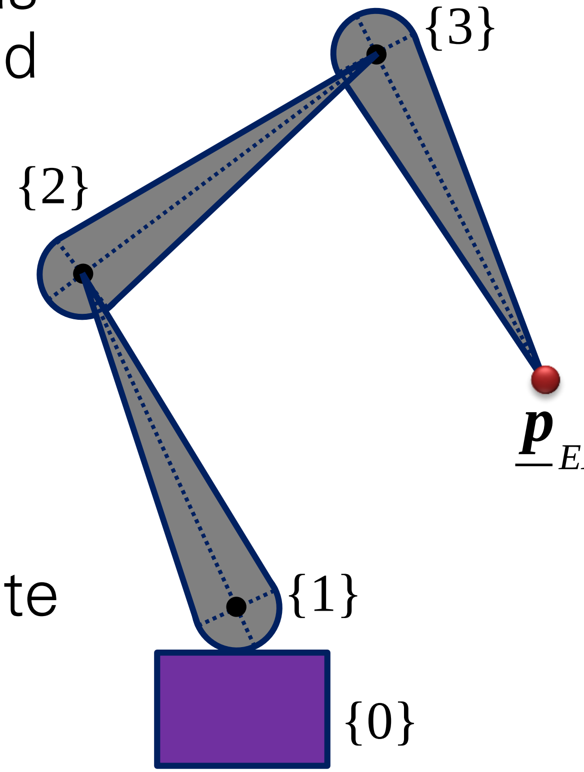
Kinematic Chain – Coordinate Systems

Computer Graphics and Visualization

- 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

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

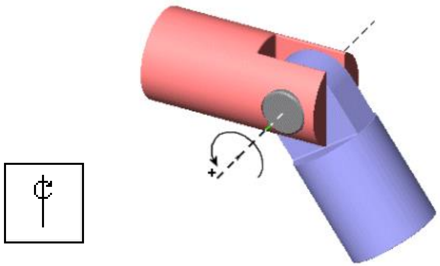
- *model transform view:* place bones from base to end effector
- *system transform view:* convert coordinate system from end effector to base
- This can be further refined into



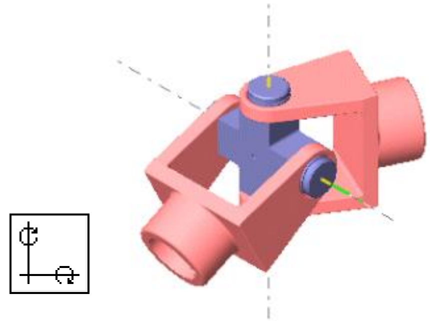
$$ \boldsymbol{T}_{\text{chain}} = \overbrace{{}^{\text{world}}\boldsymbol{T}_{OF_0} \cdot {}^{OF_0}\boldsymbol{T}_{IF_1}}^{{}^{0}\boldsymbol{T}_1} \cdot \overbrace{{}^{IF_1}\boldsymbol{T}_{BF_1} \cdot {}^{BF_1}\boldsymbol{T}_{OF_1} \cdot {}^{OF_1}\boldsymbol{T}_{IF_2}}^{{}^{1}\boldsymbol{T}_2} \cdot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

dependent on joint parameters

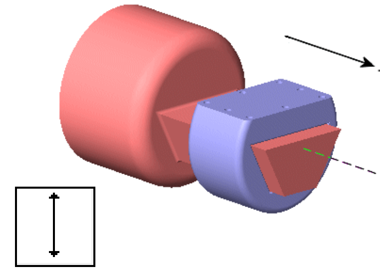
Basic Joint Types



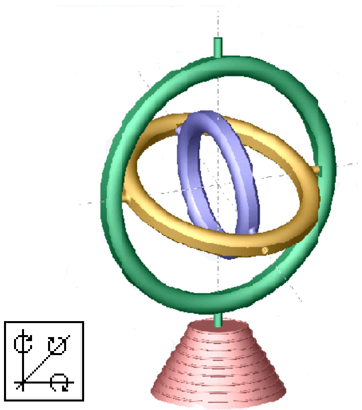
1R Revolute



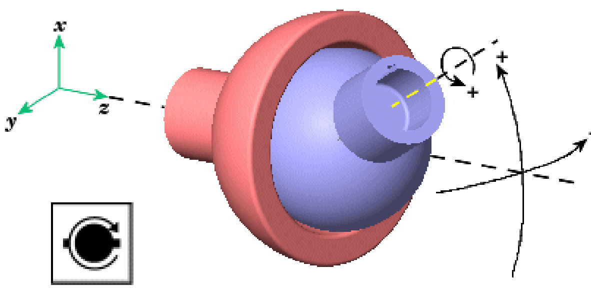
2R Universal



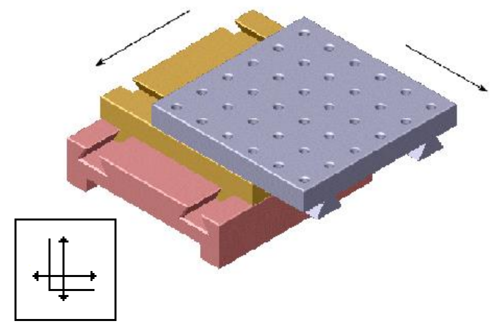
1T prismatic



3R Gimbal



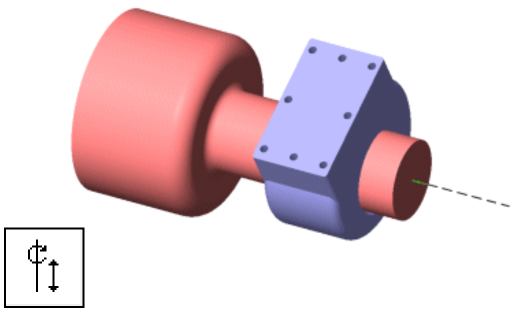
3R Spherical



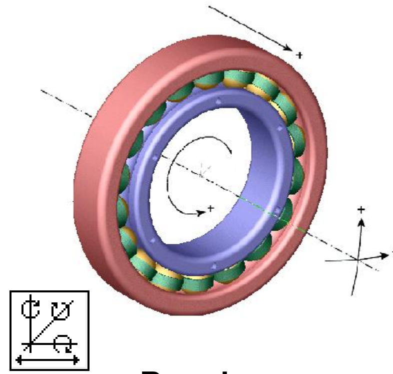
2T in-planar

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

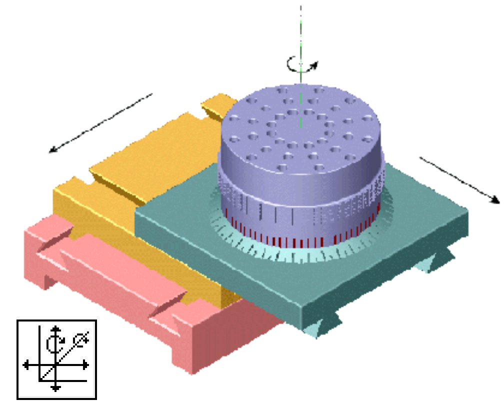
Special Joint Types



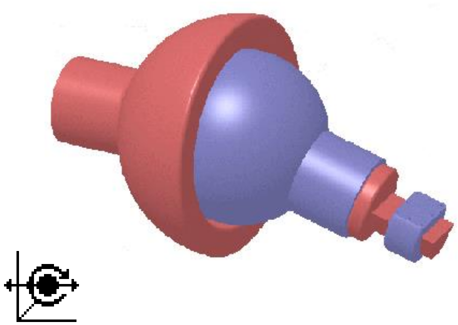
1R1T Cylindrical



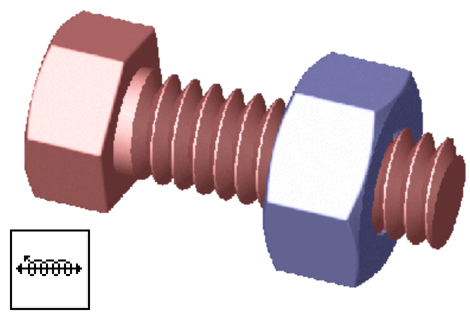
3R1T Bearing



1R2T planar

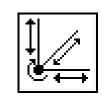


3R1T Telescoping



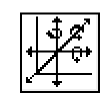
Screw

Six-DoF



3R3T

Bushing



3R3T

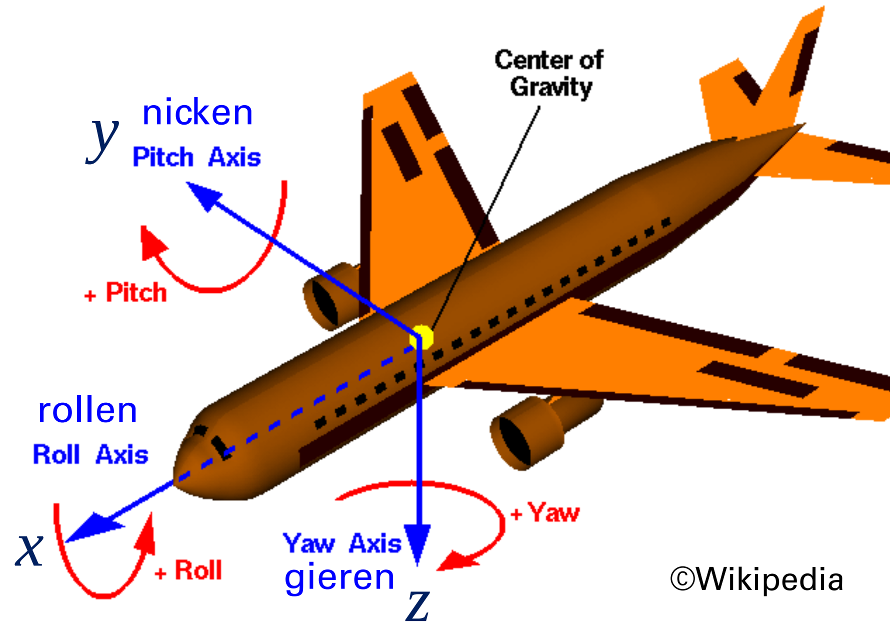
http://www.mathworks.de/de/help/physmod/sm/assembled-joints.html

Rotation from Euler Angles

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

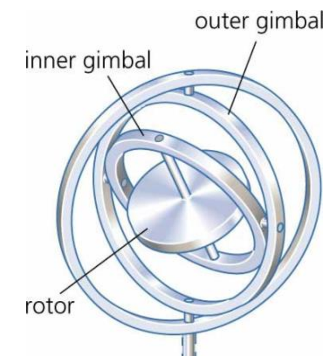
$$\boldsymbol{R}_{\text{roll-pitch-yaw}} = \boldsymbol{R}_Z(\phi_{\text{yaw}})\boldsymbol{R}_Y(\phi_{\text{pitch}})\boldsymbol{R}_X(\phi_{\text{roll}})$$



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

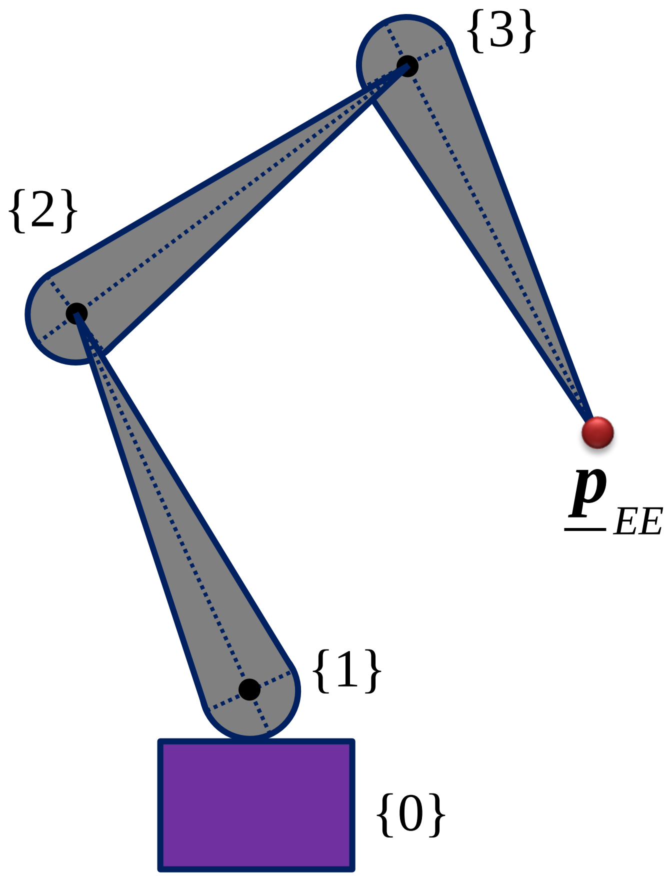
Forward Kinematics

- 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:

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

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

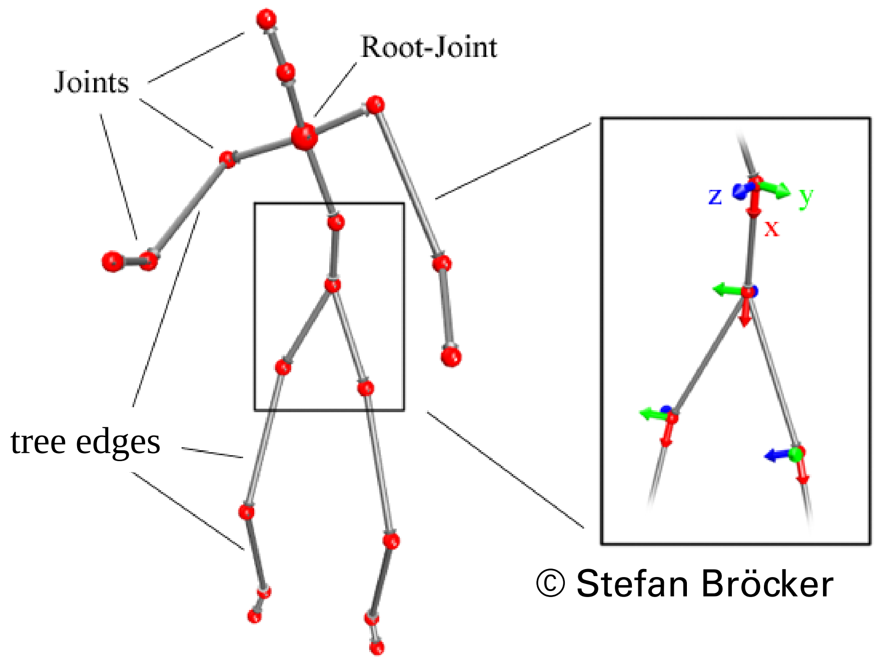
Orientation for example given as Euler angles and computed from 3x3-rotation matrix



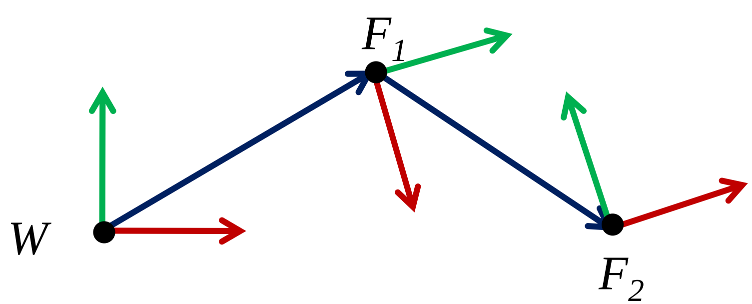
$${}^0\boldsymbol{T}_N = {}^0\boldsymbol{T}_1 \cdot {}^1\boldsymbol{T}_2 \cdot \ldots \cdot {}^{N-1}\boldsymbol{T}_N$$

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 coordinate frame F_i is defined
- the pose of the skeleton is defined with one rigid body transformation ${}^{p(i)}T_i$ per joint mapping F_i to the frame of the parent joint
- the rigid body transformation ${}^{p(i)}T_i$ between frames can be represent-ted as
  - translation and rotation, or
  - rotation and translation (used in the following)



© Stefan Bröcker



representation of transformations

Computer Graphics and Visualization

- 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

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

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

- Using quaternions one has 7 parameters plus one normalization constraint

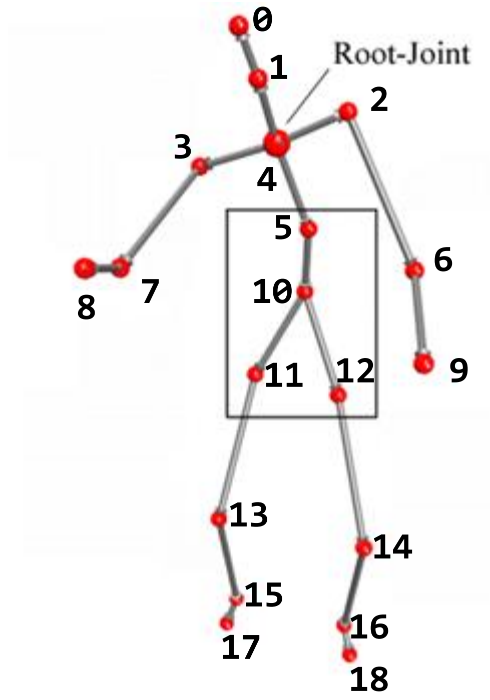
$$s^2 + x^2 + y^2 + z^2 = 1$$

$$^{i-1}\boldsymbol{T}_i(q_i = (s,x,y,z), \vec{t}_i) = \begin{pmatrix} 1-2y^2-2z^2 & 2xy-2sz & 2xz+2ys & t_x \\ 2xy+2sz & 1-2x^2-2z^2 & 2yz-2sx & t_y \\ 2xz-2sy & 2yz+2sx & 1-2x^2-2y^2 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Computing World to Bone Transforms

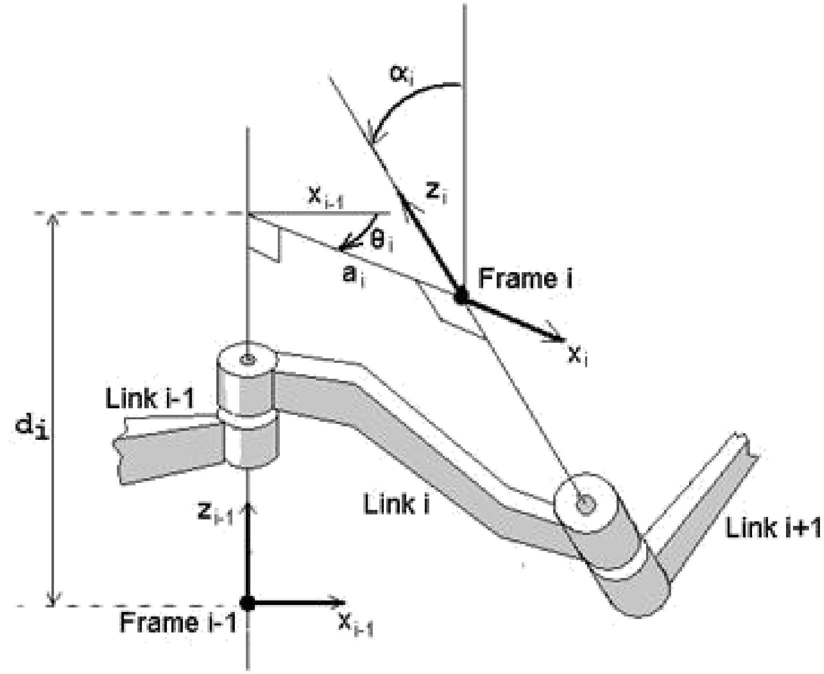
joint index	4	3	2	1	5	7	6	0	10	8	9	11	12	13	14	15	16	17	18
parent position	-1	0	0	0	0	1	2	3	4	5	6	8	8	11	12	13	14	15	16

- the skeleton tree can be linearized for example in a depth first traversal
- for rendering we need for each joint the transformation ${}^0\boldsymbol{T}_i$ from world to joint frame
- these transformations can be stored linearly in breadth or depth first order
- for a given set of parameters q_{ik} the transformations can be efficiently computed in this order as it ensures that parents are always computed first: initialize ${}^0\boldsymbol{T}_1; \forall i = 1\ldots n : {}^0\boldsymbol{T}_i = {}^0\boldsymbol{T}_{\mathrm{p}(i)} \cdot {}^{\mathrm{p}(i)}\boldsymbol{T}_i\left(q_{ik}\right)$



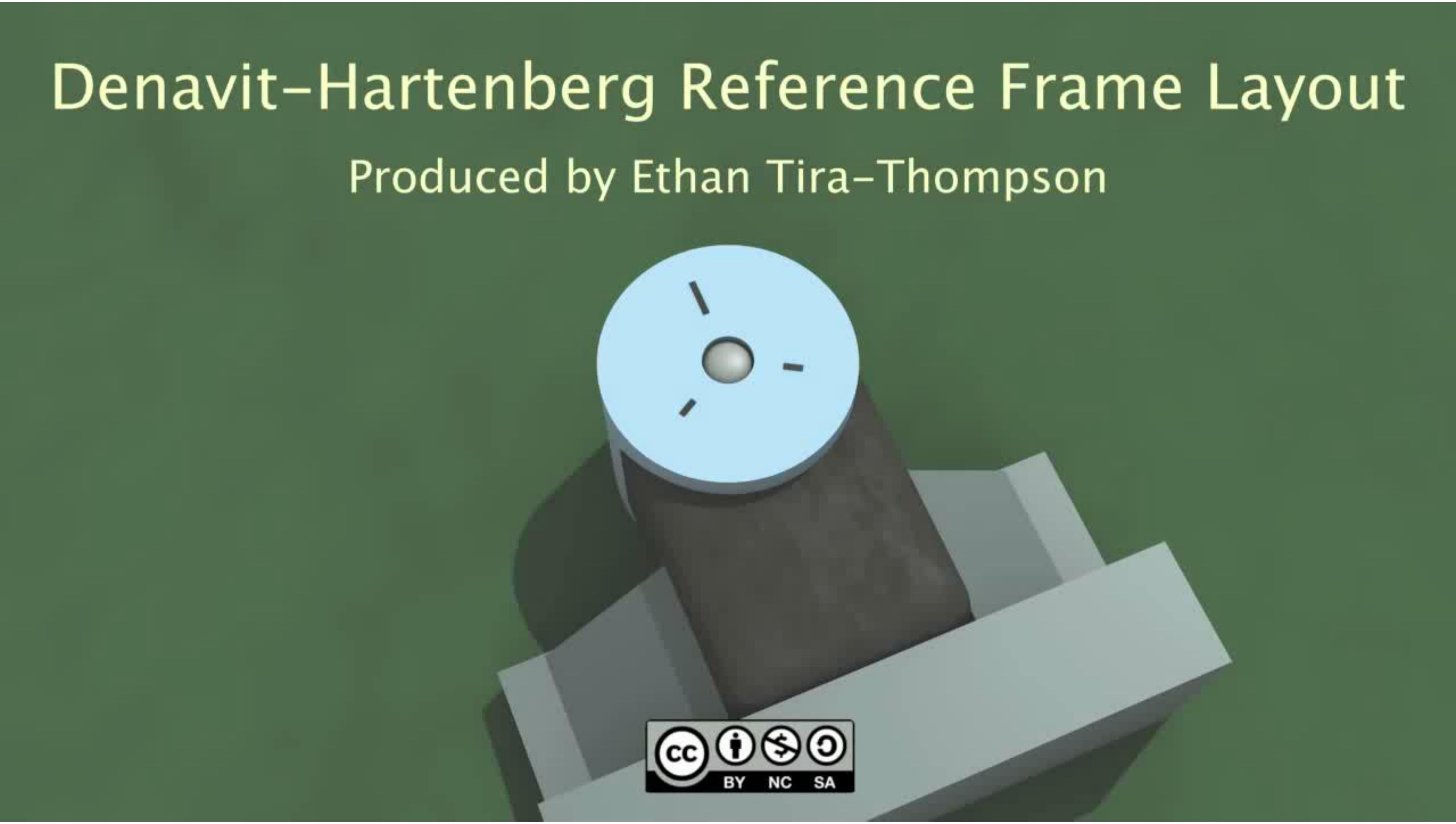
Denavit-Hartenberg notation

- **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 algebraic 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)



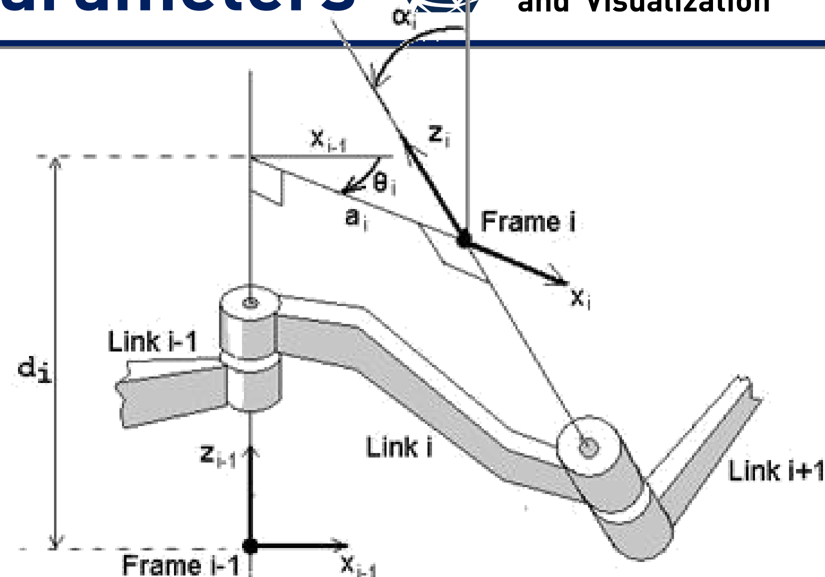
  - $\theta_i \ldots$ joint angle / rotation angle around $\hat{\boldsymbol{z}}_{i-1}$ that rotates $\hat{\boldsymbol{x}}_{i-1}$ axis onto $\hat{\boldsymbol{x}}_i$ axis (parameter of revolute joint)
  - $a_i \ldots$ link length / perp. distance between joint axes
  - $\alpha_i \ldots$ link twist / rotation angle between joint axes (around $\hat{\boldsymbol{x}}_i$)
- ➔ ${}^{i-1}\boldsymbol{T}_i = \mathrm{Rot}_{\boldsymbol{z}}(\theta_i) \cdot \mathrm{Trans}_{\boldsymbol{z}}(d_i) \cdot \mathrm{Trans}_{\boldsymbol{x}}(a_i) \cdot \mathrm{Rot}_{\boldsymbol{x}}(\alpha_i)$
- end effector axis 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

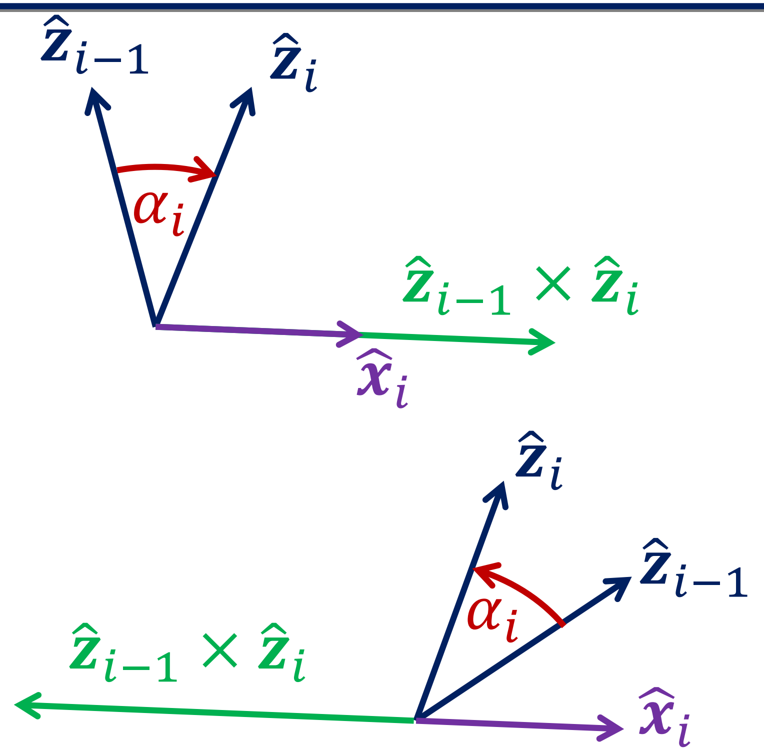
DH Frames and Translation Parameters



- 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 sinus 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.



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

- Similarly one gets

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

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