Robot Kinematics

METR 4202: Advanced Control & Robotics

Drs Surya Singh, Paul Pounds, and Hanna Kurniawati

Lecture # 3 August 6, 2012

metr4202@itee.uq.edu.au

http://itee.uq.edu.au/~metr4202/

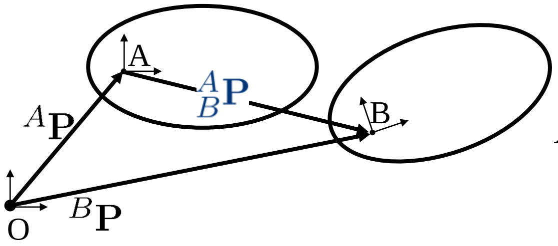
Schedule

Week	Date	Lecture (M: 12-1:30, 43-102)
1	23-Jul	Introduction
2	30-Jul	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	**6-Aug**	**Robot Kinematics and Dynamics**
4	13-Aug	Robot Dynamics & Control
5	20-Aug	Obstacle Avoidance & Motion Planning
6	27-Aug	Sensors, Measurement and Perception
7	3-Sep	Localization and Navigation
8	10-Sep	State-space modelling & Controller Design
9	17-Sep	Vision-based control
	24-Sep	*Study break*
10	1-Oct	Uncertainty/POMDPs
11	8-Oct	Robot Machine Learning
12	15-Oct	Guest Lecture
13	22-Oct	Wrap-up & Course Review

Recap from Last Week [1]

- A **position** vectors specifies the location of a **point** in 3D (Cartesian) space

$$\mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$



$$^{A}\mathbf{P} + {}^{A}\mathbf{P}^{B} - {}^{B}\mathbf{P} = 0$$

$$^{A}\mathbf{P}^{B} = {}^{A}\mathbf{P}_{B} = {}^{A}_{B}\mathbf{P} = \begin{bmatrix} ^{B}p_x \\ ^{B}p_y \\ ^{B}p_z \end{bmatrix} - \begin{bmatrix} ^{A}p_x \\ ^{A}p_y \\ ^{A}p_z \end{bmatrix}$$

- BUT we **also** concerned with its orientation in 3D space.
  This is specified as a matrix based on each **frame's unit vectors**

Recap from last week [2]…

- The components of a rotation matrix are the unit vectors projected **onto** the unit directions of the reference frame

$$\begin{array}{c|ccc} {}^{A}_{B}R & (b_x)\,\hat{i}_B & (b_y)\,\hat{j}_B & (b_z)\,\hat{k}_B \\ \hline (a_x)\,\hat{i}_A & \hat{i}_B \cdot \hat{i}_A & \hat{j}_B \cdot \hat{i}_A & \hat{k}_B \cdot \hat{i}_A \\ (a_y)\,\hat{j}_A & \hat{i}_B \cdot \hat{j}_A & \hat{j}_B \cdot \hat{j}_A & \hat{k}_B \cdot \hat{j}_A \\ (a_z)\,\hat{k}_A & \hat{i}_B \cdot \hat{k}_A & \hat{j}_B \cdot \hat{k}_A & \hat{k}_B \cdot \hat{k}_A \end{array}$$

Recap from Last Week [3]

Rotation is orthonormal ∴

→ the ***rows*** are **{A} written in {B}**

$$ {}^{B}_{A}\mathbf{R} = {}^{A}_{B}\mathbf{R}^{T} = {}^{A}_{B}\mathbf{R}^{-1} $$

→ The of a rotation matrix inverse = the transpose

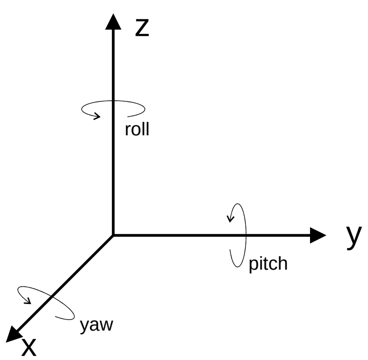
$$ \mathbf{R}\cdot\mathbf{R}^{T} = 1 $$

→ ∵ of normality → the determinant = 1

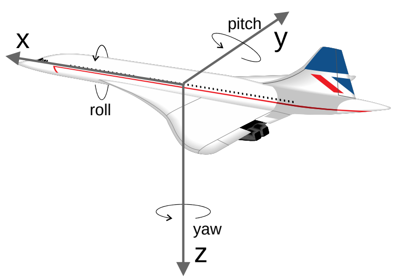
$$ det\left(\mathbf{R}\right) = 1 $$

Recall from Last Week [4]…

- In many Kinematics References:



- In many Engineering Applications:



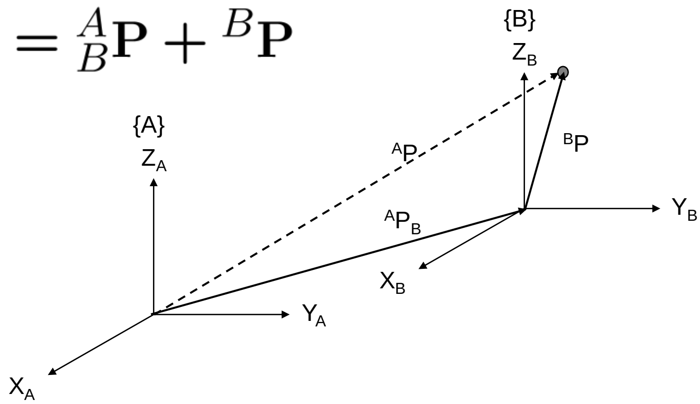
→ Be careful:
This name is given to other conventions too!

Coordinate Transformations [1]

- Translation Again:

  If {B} is translated with respect to {A} **without rotation**, then it is a vector sum

$$ {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{P} + {}^{B}\mathbf{P} $$



Coordinate Transformations [2]

- Rotation Again:

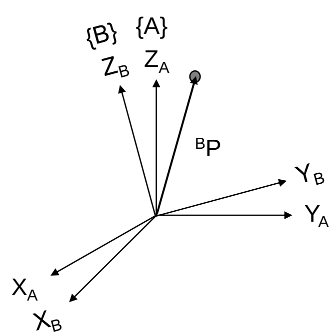
  {B} is rotated with respect to {A} then
  use rotation matrix to determine new components

$$ {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P} $$

- NOTE:
  - The Rotation matrix's ***subscript*** matches the position vector's **superscript**

$$ {}^{A}\mathbf{P} = {}^{A}_{[\![B]\!]}\mathbf{R}\,{}^{[\![B]\!]}\mathbf{P} $$

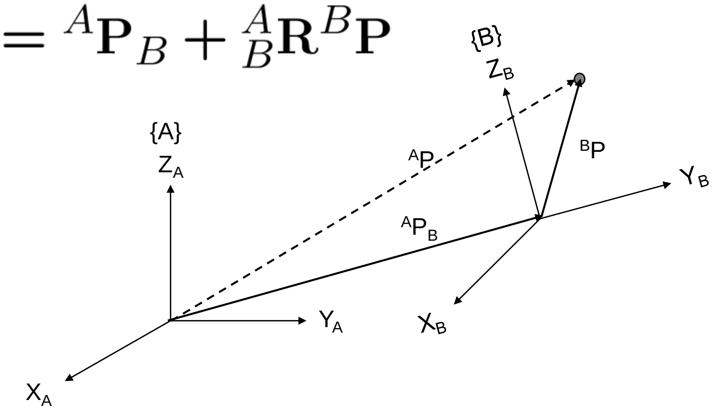
  - This gives Point Positions of {B} ORIENTED in {A}



Coordinate Transformations [3]

- Composite transformation:
  {B} is moved with respect to {A}:

$$^{A}\mathbf{P} = {}^{A}\mathbf{P}_{B} + {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}$$



Homogenous Coordinates

$$\widehat{p} = \begin{bmatrix} \rho p_x & \rho p_y & \rho p_z & \rho \end{bmatrix}^T$$

- ρ is a scaling value

Homogenous Transformation

$$\begin{bmatrix} {}^{A}R_{B} & {}^{A}p \\ \gamma & \rho \end{bmatrix}$$

- γ is a projective transformation
- ρ is a scaling value

General Coordinate Transformations [1]

- A compact representation of the translation and rotation is known as the **Homogeneous Transformation**

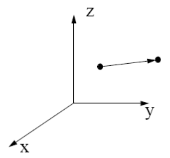
$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}\mathbf{P}_{B} \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix}$$

- This allows us to cast the rotation and translation of the general transform in a single matrix form

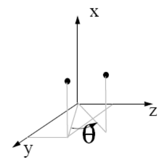
$$\begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = {}^{A}_{B}\mathbf{T} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix}$$

General Coordinate Transformations [2]

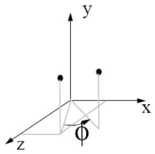
Fundamental orthonormal transformations can be represented in this form too:



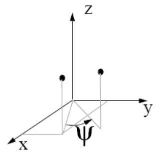
$$Trans(u, v, w) = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Rotx(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Roty(\phi) = \begin{bmatrix} c\phi & 0 & s\phi & 0 \\ 0 & 1 & 0 & 0 \\ -s\phi & 0 & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

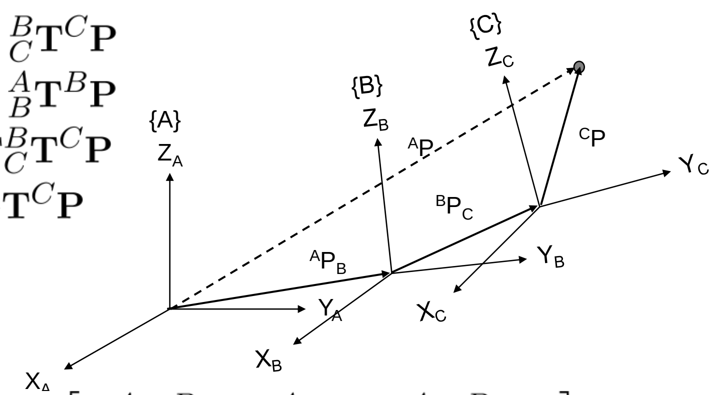


$$Rotz(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 & 0 \\ s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Coordinate Transformations [3]

- Multiple transformations compounded as a chain

$$^{B}\mathbf{P} = {}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P}$$
$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{T}\,{}^{B}\mathbf{P}$$
$$= {}^{A}_{B}\mathbf{T}\,{}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P}$$
$$= {}^{A}_{C}\mathbf{T}\,{}^{C}\mathbf{P}$$



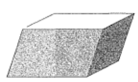
$$^{A}_{C}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R}\,{}^{B}_{C}\mathbf{R} & {}^{A}\mathbf{P}_B + {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}_C \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Projective Transformations …

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, **I**, **J** (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

p.44, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Projective Transformations & Other Transformations of 3D Space

Group	Matrix	Distortion	Invariant properties
Projective 15 dof	$\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{v}^\mathsf{T} & v \end{bmatrix}$		Intersection and tangency of surfaces in contact. Sign of Gaussian curvature.
Affine 12 dof	$\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$		Parallelism of planes, volume ratios, centroids. The plane at infinity, π_∞, (see section 3.5).
Similarity 7 dof	$\begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$		The absolute conic, Ω_∞, (see section 3.6).
Euclidean 6 dof	$\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$		Volume.

p.78, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Generalizing

Special Orthogonal & Special Euclidean Lie Algebras

- SO(n): R**o**tations

$$SO(n) = \{R \in \mathbb{R}^{n\times n} : RR^T = I, \det R = +1\}.$$

$$\exp(\widehat{\omega}\theta) = e^{\widehat{\omega}\theta} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \dots$$

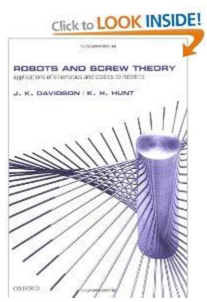
- SE(n): Transformations of EUCLIDEAN space

$$SE(n) := \mathbb{R}^n \times SO(n).$$

$$SE(3) = \{(p, R) : p \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times SO(3).$$

Screw Displacements

- Comes from the notion that all motion can be viewed as a rotation (Rodrigues formula)
- Define a vector along the axis of motion (screw vector)
  - Rotation (screw angle)
  - Translation (pitch)
  - Summations → via the screw triangle!



Denavit Hartenberg [DH] Notation

- J. Denavit and R. S. Hartenberg first proposed the use of homogeneous transforms for articulated mechanisms

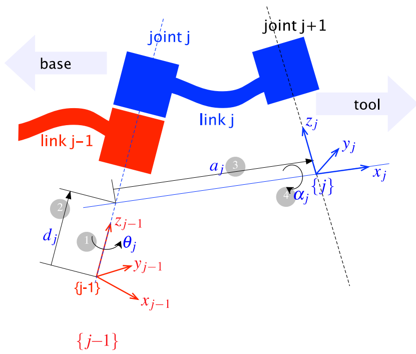
  (But B. Roth, introduced it to robotics)

- A kinematics "short-cut" that reduced the number of parameters by adding a structure to frame selection

- For two frames positioned in space, the first can be moved into coincidence with the second by a sequence of 4 operations:
  - rotate around the x_{i-1} axis by an angle α_i
  - translate along the x_{i-1} axis by a distance a_i
  - translate along the new z axis by a distance d_i
  - rotate around the new z axis by an angle θ_i

Denavit-Hartenberg Convention

- link length a_i the offset distance between the z_{i-1} and z_i axes along the x_i axis;
- link twist α_i the angle from the z_{i-1} axis to the z_i axis about the x_i axis;



Art c/o P. Corke

- link offset d_i the distance from the origin of frame i-1 to the x_i axis along the z_{i-1} axis;
- joint angle θ_i the angle between the x_{i-1} and x_i axes about the z_{i-1} axis.

DH: Where to place frame?

1. Align an axis along principal motion
   1. Rotary (**R**): align rotation axis along the z axis
   2. Prismatic (**P**): align slider travel along x axis

2. Orient so as to position x axis towards next frame

3. $\theta_{\text{(rot } z)} \rightarrow d_{\text{(trans } z)} \rightarrow a_{\text{(trans } x)} \rightarrow \alpha_{\text{(rot } x)}$

Denavit-Hartenberg → Rotation Matrix

- Each transformation is a product of 4 “basic” transformations (instead of 6)

$$
\begin{aligned}
{}^{i-1}A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdots \\
&\quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

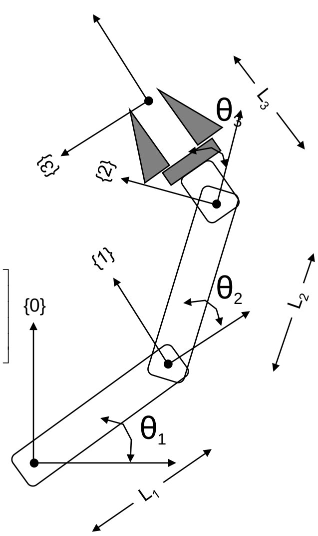
DH Example [1]: RRR Link Manipulator

1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	θ_3

$$
{}^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1 c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
{}^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
{}^2A_3 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & L_3 c_{\theta_3} \\ s_{\theta_3} & c_{\theta_3} & 0 & L_3 s_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_3 = {}^0A_1 \, {}^1A_2 \, {}^2A_3
= \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1 c_{\theta_1} + L_2 c_{\theta_{12}} + L_3 c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1 s_{\theta_1} + L_2 s_{\theta_{12}} + L_3 s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$



DH Example [2]: RRP Link Manipulator

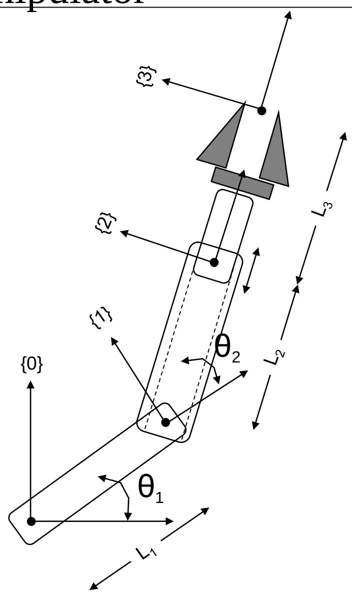
1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	0

$$ {}^{0}A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1 c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}A_3 = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_3 = {}^{0}A_1 \, {}^{1}A_2 \, {}^{2}A_3 $$

$$ = \begin{bmatrix} c_{\theta_{12}} & -s_{\theta_{12}} & 0 & L_1 c_{\theta_1} + (L_2 + L_3) c_{\theta_{12}} \\ s_{\theta_{12}} & c_{\theta_{12}} & 0 & L_1 s_{\theta_1} + (L_2 + L_3) s_{\theta_{12}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



DH Example [3]: Puma 560

- "Simple" 6R robot exercise for the reader …

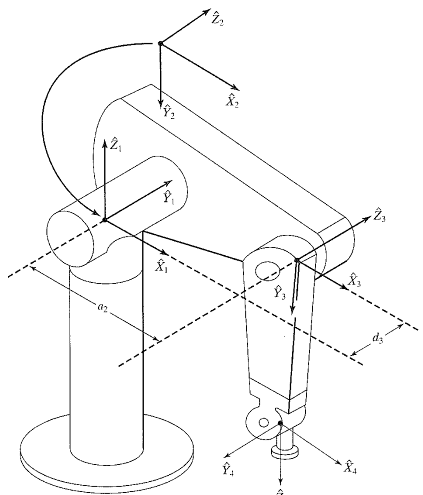
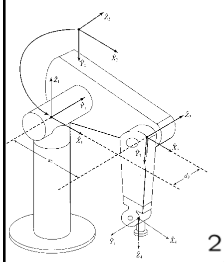


Image: J. Craig, *Introduction to Robotics* *3rd Ed.*, 2005

Link	a_i	α_i	d_i	θ_i
1	0	0	0	θ_1
2	0	$-\pi/2$	0	θ_2
3	L_2	0	D_3	θ_3
4	L_3	$-\pi/2$	D_4	θ_4
5	0	$\pi/2$	0	θ_5
6	0	$-\pi/2$	0	θ_6

DH Example [3]: Puma 560 [2]



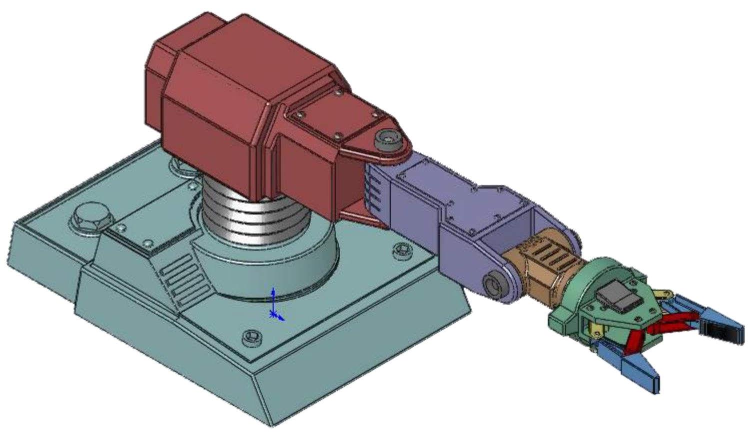
$$^0A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^1A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^2A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^3A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^4A_5 = \begin{bmatrix} c_4 & -s_5 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^5A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & L_3 \\ 0 & 0 & -1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

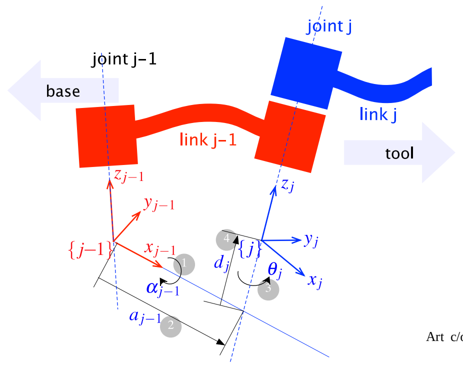
$$^0T_6 = {}^0A_1{}^1A_2{}^2A_3{}^3A_4{}^4A_5{}^5A_6$$

Demonstration: Matlab & Solidworks



Modified DH

- Made “popular” by Craig’s *Intro. to Robotics* book
- Link coordinates attached to the near by joint

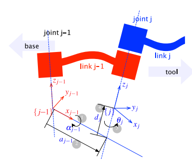


Art c/o P. Corke

- a (trans *x-1*) → α (rot *x-1*) → θ (rot *z*) → d (trans *z*)

Modified DH [2]

- Gives a similar result
  (but it’s not commutative)



$$\Rightarrow {}^{i-1}A_i = R_x(\alpha_{i-1})\,T_x(a_{i-1})\,R_z(\theta_i)\,T_x(d_i)$$

- Refactoring Standard → to Modified

$$\underbrace{\{R_z(\theta_1)\,T_z(d_1)\,T_x(a_1)\,R_x(\alpha_1)\}}_{\text{DH}_1}\cdot\underbrace{\{R_z(\theta_2)\,T_z(d_2)\,T_x(a_2)\,R_x(\alpha_2)\}}_{\text{DH}_2}\cdot\underbrace{\{R_z(\theta_3)\,T_z(d_3)\}}_{\text{End Effector}}$$

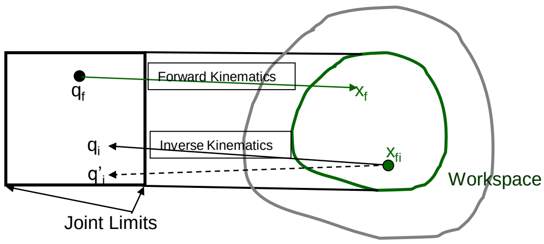
$$=\underbrace{\{R_z(\theta_1)\,T_z(d_1)\}}_{\text{Base}}\cdot\underbrace{\{T_x(a_1)\,R_x(\alpha_1)\,R_z(\theta_2)\,T_z(d_2)\}}_{\text{MDH}_1}\cdot\underbrace{\{T_x(a_2)\,R_x(\alpha_2)\,R_z(\theta_3)\,T_z(d_3)\}}_{\text{MDH}_2}$$

Forward Kinematics [1]

- Forward kinematics is the process of chaining homogeneous transforms together. For example to:
  - Find the articulations of a mechanism, or
  - the fixed transformation between two frames which is known in terms of linear and rotary parameters.
- Calculates the final position from the **machine** (**<u>joint variables</u>**)
- Unique for an open kinematic chain (**serial arm**)
- "Complicated" (multiple solutions, etc.) for a closed kinematic chain (**parallel arm**)

Forward Kinematics [2]

- Can think of this as “spaces”:
  - Operation space (x,y,z,α,β,γ): The robot’s position & orientation
    $$\vec{\mathbf{x}} = \begin{bmatrix} \vec{\mathbf{p}} \\ \vec{\mathbf{\Theta}} \end{bmatrix}$$
  - Joint space ($\theta_1 \ldots \theta_n$): A state-space vector of joint variables
    $$\vec{\mathrm{q}} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

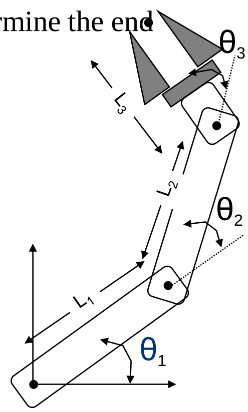


Forward Kinematics [3]

- Consider a planar RRR manipular
- Given the joint angles and link lengths, we can determine the end effector pose:

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1+\theta_2) + \ldots$$
$$L_3 \cos(\theta_1+\theta_2+\theta_3)$$

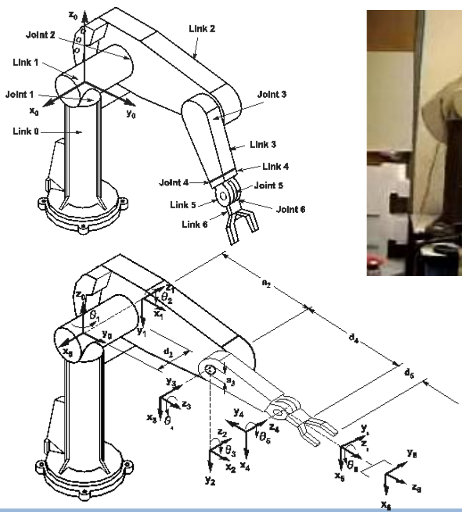
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1+\theta_2) + \ldots$$
$$L_3 \sin(\theta_1+\theta_2+\theta_3)$$



- This isn’t too difficult to determine for a simple, planar manipulator. BUT …

Forward kinematics [4]: The PUMA 560!

- What about a more complicated mechanism?



$$\begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$) - s_1(s_4c_5c_6 + c_4s_6)$

$) + c_1(s_4c_5c_6 + c_4s_6)$

$s_6) - s_1(-s_4c_5s_6 + c_4c_6)$

$s_6) + c_1(-s_4c_5s_6 + c_4c_6)$

$a_x = c_1(c_{23}c_4s$

$a_y = s_1(c_{23}c_4s$

$a_z = -s_{23}c_4s_5$

$p_x = c_1(d_6(c_{23}$

$p_y = s_1(d_6(c_{23}$

$p_z = d_6(c_{23}c_5$

Inverse Kinematics

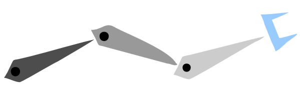
- Forward: angles → position

  x = *f* (**θ**)

- Inverse: position → angles

  θ = *f*$^{-1}$(**x**)

- Analytic Approach



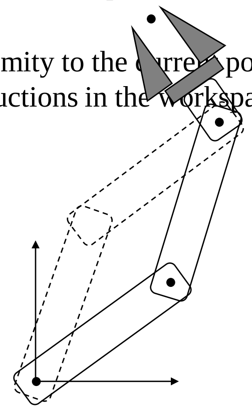
- Numerical Approaches:
  - Jacobian: $J = \frac{\delta x}{\delta q} \rightarrow \delta q \approx J^{-1}\delta x$
  - J^T Approximation: $\tau = J^T \cdot \mathbf{F} \rightarrow \Delta q \approx J^T \Delta x$
    - Slotine & Sheridan method
  - Cyclical Coordinate Descent

Inverse Kinematics

- Inverse Kinematics is the problem of finding the joint parameters given only the values of the homogeneous transforms which model the mechanism
(i.e., the pose of the end effector)
- Solves the problem of where to drive the joints in order to get the hand of an arm or the foot of a leg in the right place
- In general, this involves the solution of a set of simultaneous, non-linear equations
- Hard for serial mechanisms, easy for parallel

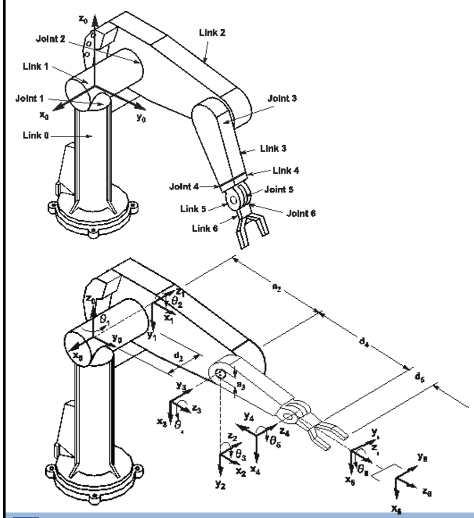
Multiple Solutions

- There will often be multiple solutions for a particular inverse kinematic analysis
- Consider the three link manipulator shown. Given a particular end effector pose, two solutions are possible
- The choice of solution is a function of proximity to the current pose, limits on the joint angles and possible obstructions in the workspace



Inverse kinematics

- What about a more complicated mechanism?



$$ {}^0T_6 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_5\,{}^5T_6 = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

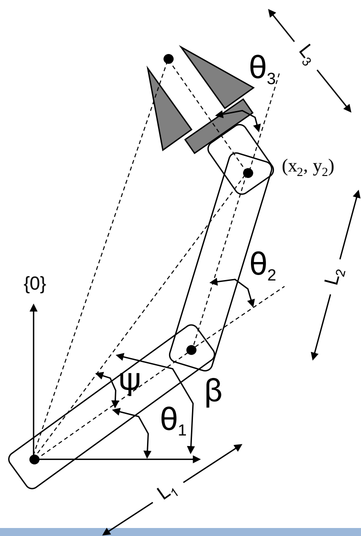
$$
\begin{aligned}
n_x &= c_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) - s_1(s_4c_5c_6 + c_4s_6) \\
n_y &= s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) + c_1(s_4c_5c_6 + c_4s_6) \\
n_z &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\
s_x &= c_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) - s_1(-s_4c_5s_6 + c_4c_6) \\
s_y &= s_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) + c_1(-s_4c_5s_6 + c_4c_6) \\
s_z &= s_{23}(c_4c_5s_6 + s_4c_6) - c_{23}s_5s_6 \\
a_x &= c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \\
a_y &= s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \\
a_z &= -s_{23}c_4s_5 + c_{23}c_5) \\
p_x &= c_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) - s_1(d_6s_4s_5 + d_2) \\
p_y &= s_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) + c_1(d_6s_4s_5 + d_2) \\
p_z &= d_6(c_{23}c_5 - s_{23}c_4s_5) + c_{23}d_4 - a_3s_{23} - a_2s_2
\end{aligned}
$$

Solution Methods

- Unlike with systems of linear equations, there are no general algorithms that may be employed to solve a set of nonlinear equation
- **Closed-form** and **numerical** methods exist
- We will concentrate on analytical, closed-form methods
- These can be characterized by two methods of obtaining a solution: **algebraic** and **geometric**

Inverse Kinematics: Geometrical Approach

- We can also consider the geometric relationships defined by the arm



Inverse Kinematics: Algebraic Approach

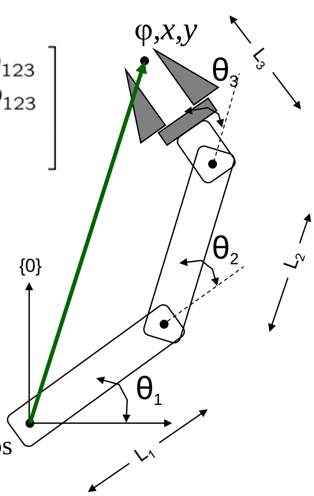
- We have a series of equations which define this system
- Recall, from Forward Kinematics:

$$^0T_3 = \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1c_{\theta_1} + L_2c_{\theta_{12}} + L_3c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1s_{\theta_1} + L_2s_{\theta_{12}} + L_3s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The end-effector pose is given by

$$^0T_3 = \begin{bmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Equating terms gives us a set of algebraic relationships

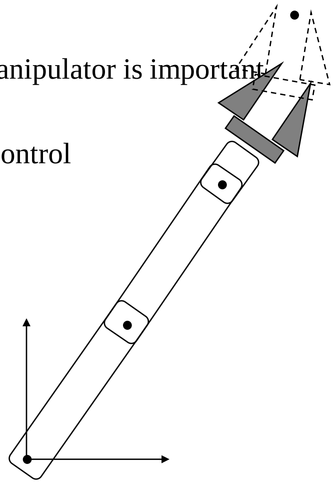


No Solution - Singularity

- Singular positions:

- An understanding of the workspace of the manipulator is important
- There will be poses that are not achievable
- There will be poses where there is a loss of control

- Singularities also occur when the manipulator loses a DOF
  - This typically happens when joints are aligned
  - **det[Jacobian]=0**

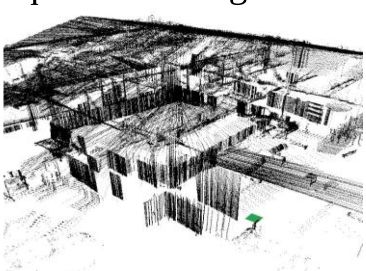


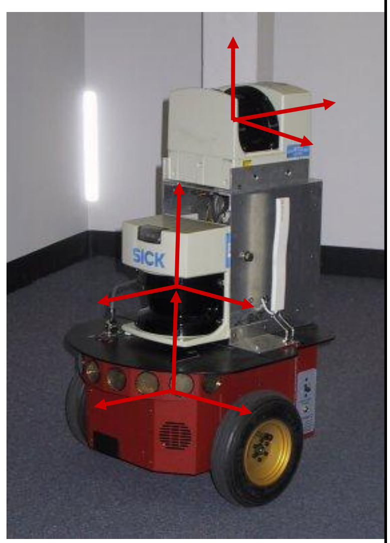
Mobile Platforms

- The preceding kinematic relationships are also important in mobile applications
- When we have sensors mounted on a platform, we need the ability to translate from the sensor frame into some world frame in which the vehicle is operating
- Should we just treat this as a P(*) mechanism?

Mobile Platforms [2]

- We typically assign a frame to the base of the vehicle
- Additional frames are assigned to the sensors
- We will develop these techniques in coming lectures





Summary

- Many ways to view a rotation
  - Rotation matrix
  - Euler angles
  - Quaternions
  - Direction Cosines
  - Screw Vectors

- Homogenous transformations
  - Based on homogeneous coordinates

Cool Robotics Share