Introduction to Robotics

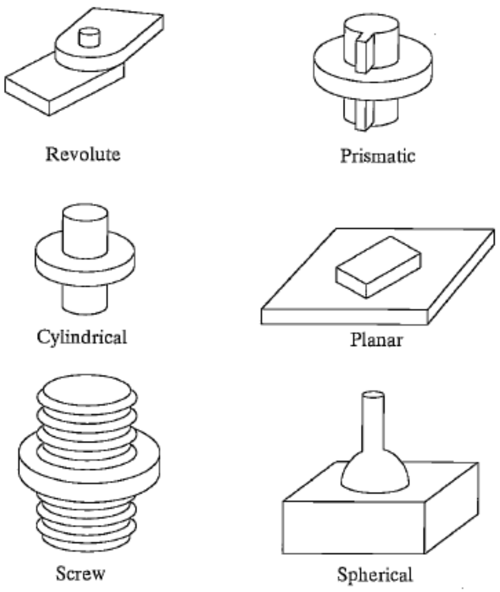
Kinematic Tree, Denavit-Hartenberg Notation, Jacobian, Inverse Kinematics

WS 2015 / 16

Prof. Dr. Daniel Göhring
Intelligent Systems and Robotics
Department of Computer Science
Freie Universität Berlin

Manipulator Kinematics

- **Idea:** We want to be able to calculate the end-effector‘s (manipulator‘s) position with respect to a base frame as a function of joint variables
- Link Description
- Denavit Hartenberg Notation
- How to attach different frames



All six lower-pair joints *(Craig)*

Frame Attachment

- At the center of mass?

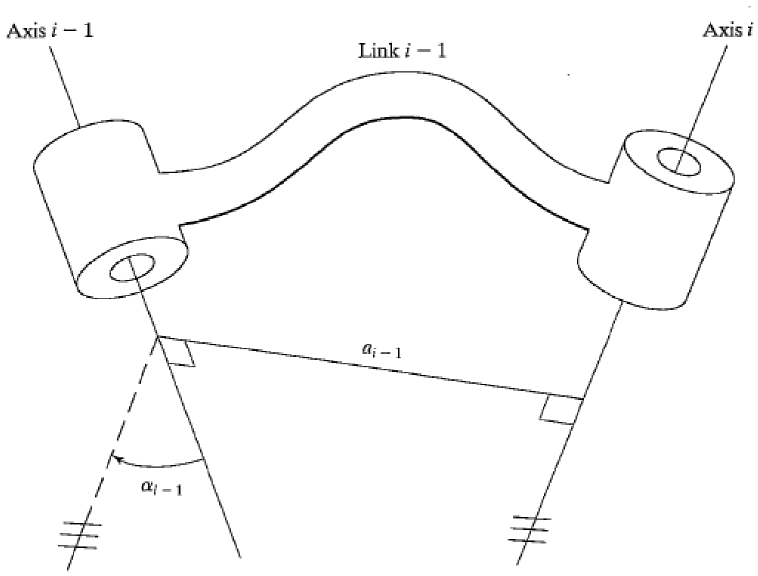
- How can we take advantage of joint constraints?

- Minimal description required.

- Link description
- Link connection
- Link variables and constants

Link Description

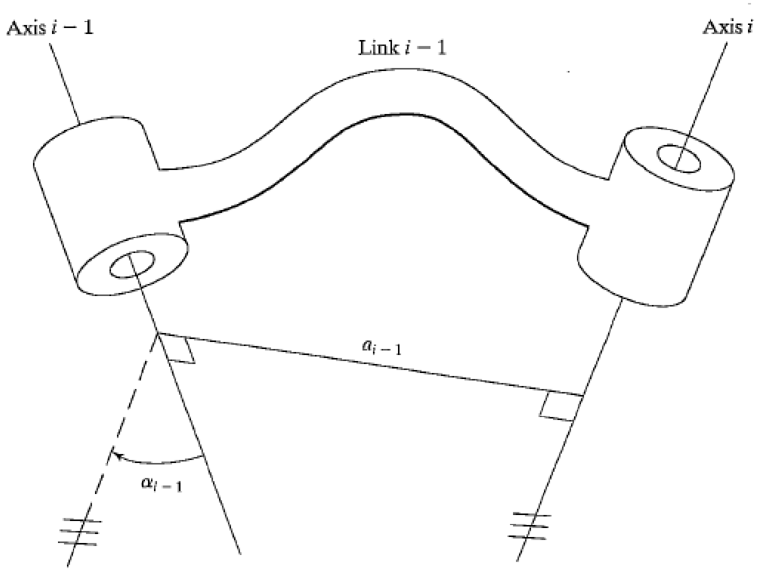
- Links are numbered, starting with 0 at the base
- First moving link is link 1
- A link is considered as a rigid body that defines the relationship between two neighboring joint axes of a manipulator
- Constants for link: link length a_{i-1} and twist α_{i-1}



(Craig)

Link Description

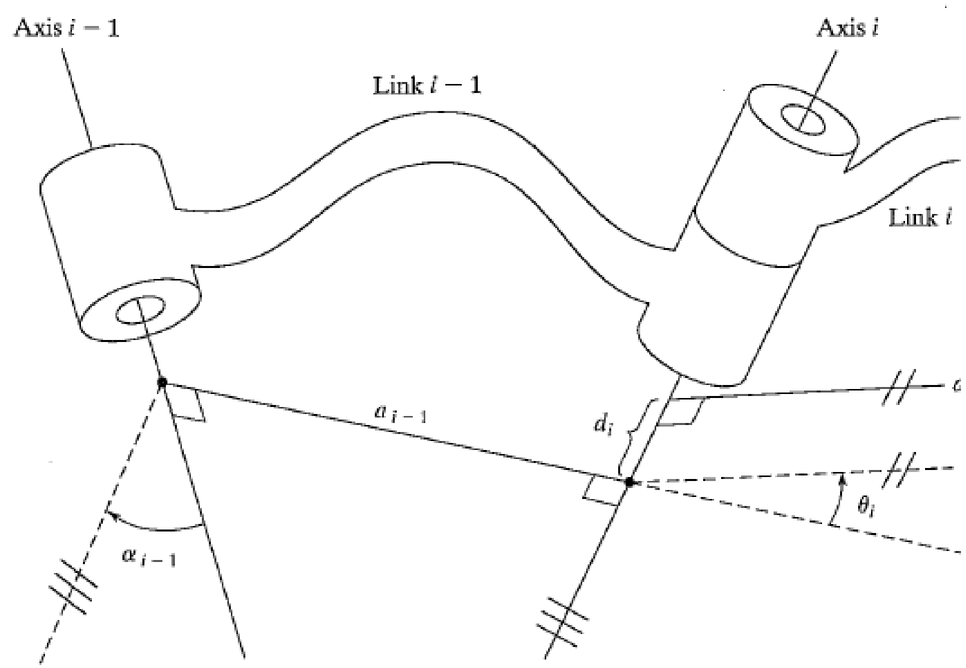
- Find common perpendicular a_{i-1}, defined to be between axis i-1 and i – unique except for parallel axes
- a_{i-1} : **Link Length**
- α_{i-1} : **Link Twist**, measured in the right hand sense about a_{i-1}
- a_{i-1} and α_{i-1} are constant
- For intersecting axes, sign of α_{i-1} is free, usually points to end-effector



Link Connections

- New link will create a new common normal a_i with axis i
- d_i : **Link offset**, variable for prismatic joints, fix for revolute joints
- θ_i : **Joint angle**, variable for revolute joints, fix for prismatic joints
- 4 variables per joint:

  a_{i-1} , α_{i-1} , d_i, θ_i

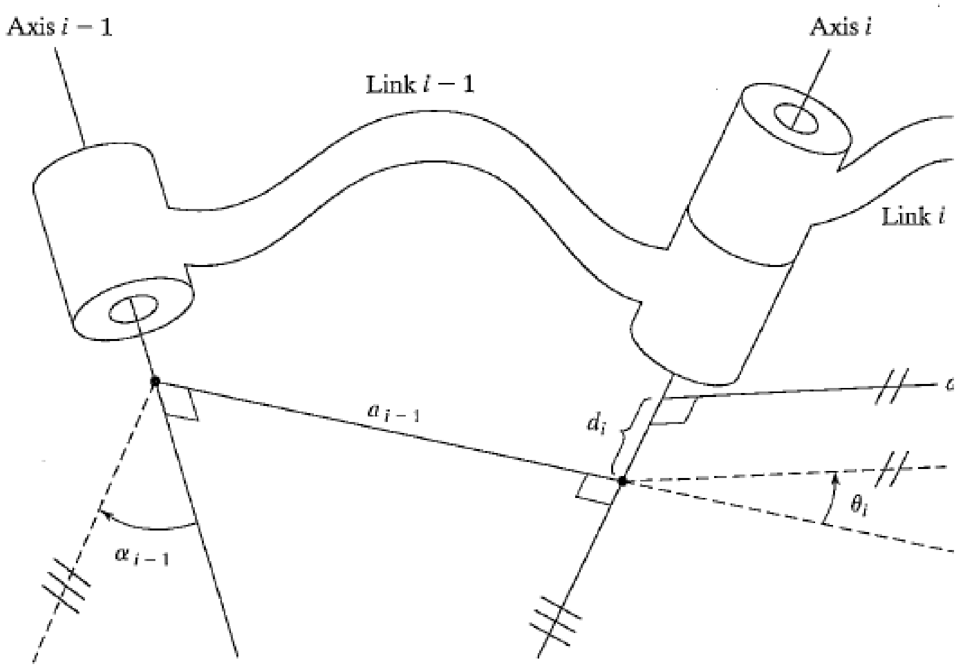


Parallel Joint Axes

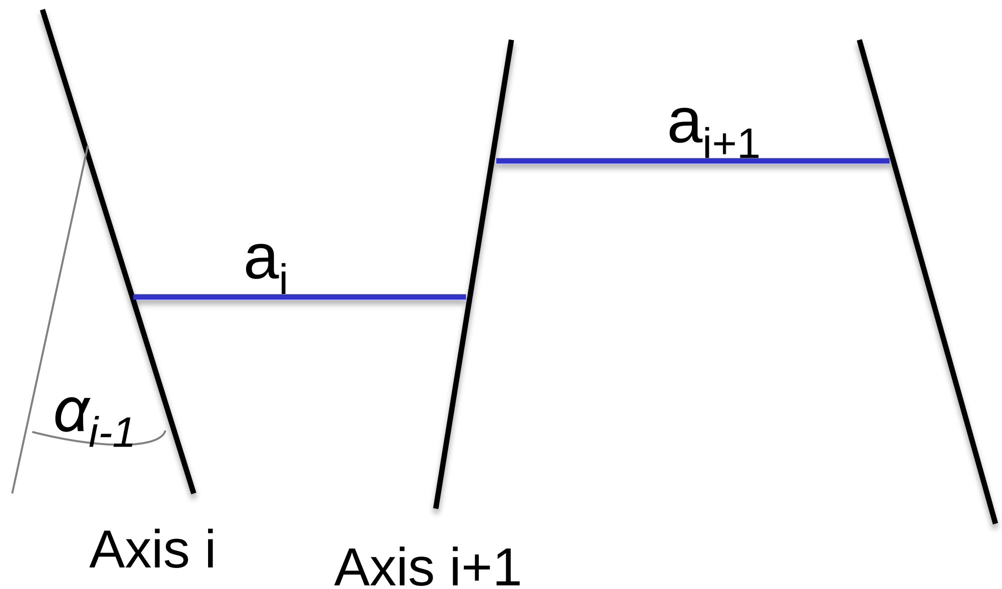
- What if two axis are parallel – infinite perpendiculars are possible
  - Go to next joint and find perpendicular, chose first perpendicular that it intersects with second one

Intermediate Links

- θ_i and d_i depend on links *i-1* and *i*
- $\theta_2 \ldots \theta_{i-1}$ and $d_2 \ldots d_{i-1}$ defined by link configuration



First and Last Links



- a_i , α_i depend on joint axis i and i+1
- axes *1* to *n* determine $a_1 \dots a_{n-1}$ and $\alpha_1 \dots \alpha_{n-1}$
- what about a_0 and α_0 ?
- Convention: try to set as many variables as possible to *0*

$$\boxed{a_0 = a_n = 0 \quad \text{and } \alpha_n = \alpha_n = 0}$$

First and Last Links, contd.

- axis *0* is set to be identical to axis *1*
- If joint *1* is revolute,
  - choose zero position for θ_1 arbitrarily,
  - $d_1 = 0$
- If joint *1* is prismatic
  - choose zero position for d_1 arbitrarily,
  - $\theta_1 = 0$
- Exactly the same applies for joint *n*

End-effector Frame

- End-effector frame can be arbitrary placed with respect to last rigid body

- Last axis *n* to be set identical to *n-1*

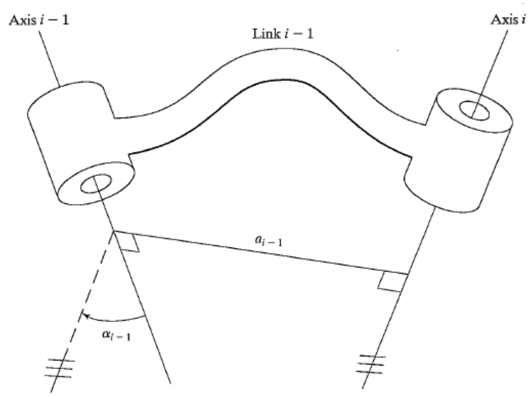
Summary

- Any robot can be described by 4 parameters for each joint
- Two parameters describe the link, two describe the connection to the neighboring link
- For each joint, 3 parameters are fix, 1 is variable
  - revolute joints: θ_i is joint variable
  - prismatic joints: d_i is joint variable

Denavit-Hartenberg Notation

Denavit-Hartenberg Parameters

- 4 D-H parameters (α_i , a_i, d_i , θ_i) for each joint
- Revolute joint: $d_1 = 0$ / $d_n = 0$
- Prismatic: $\theta_1 = 0$ / $\theta_n = 0$



- α_i and a_i describe a link

- θ and d describe how one link is connected to the next one
- d describes the translation between to links, θ describes the angle

Frame Attachment

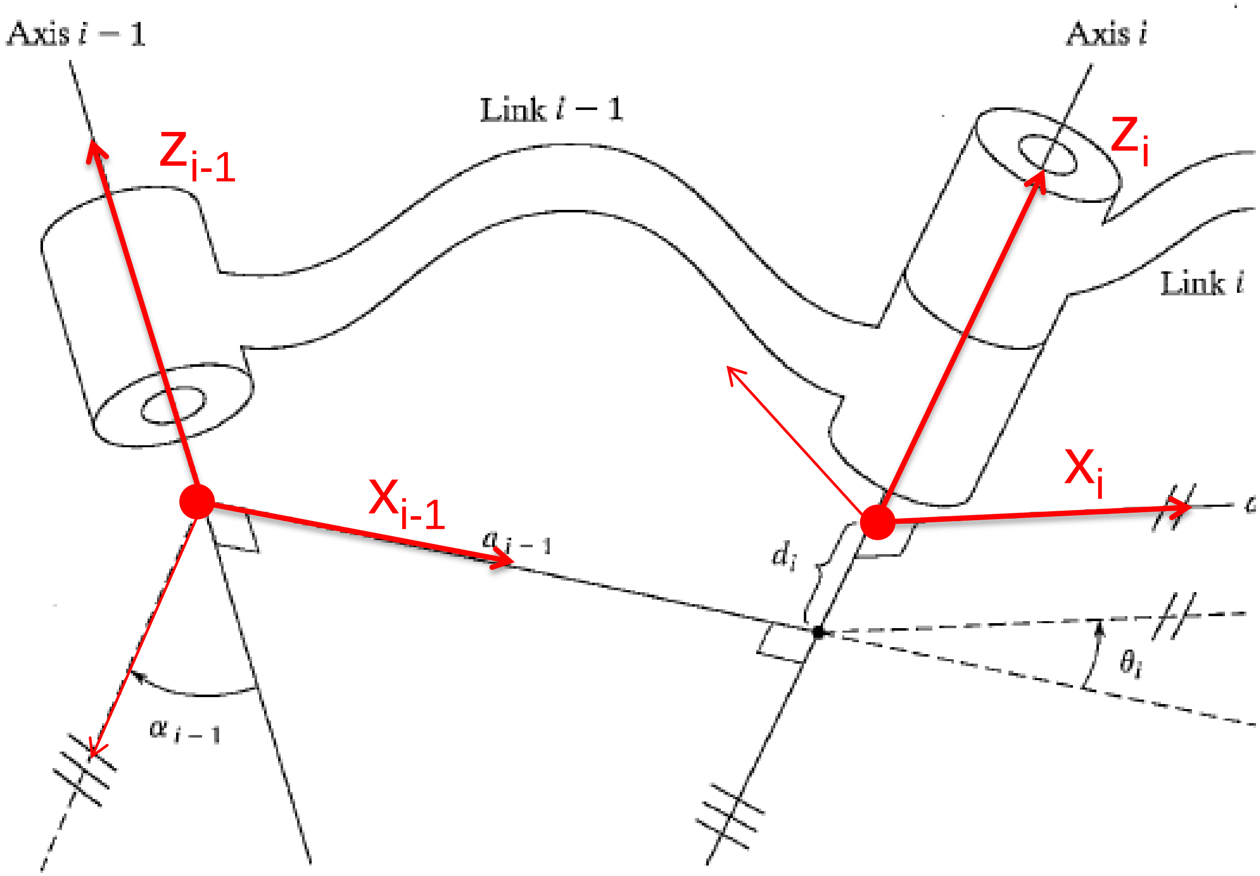
- How to attach a frame from D-H parameters?
- Necessary to transform coordinates in base frame into coordinates in the end-effector frame and back
- Origin?
- Axes?

Frame Attachment

- Define the *z*-axis of the new frame along axis *i* (e.g. joint angle with axis *i*)
- Point of intersection of axis *i* with common normal from axis *i* to the next axis *i+1* defines the origin
- *x*-axis points along the common normal to the next joint

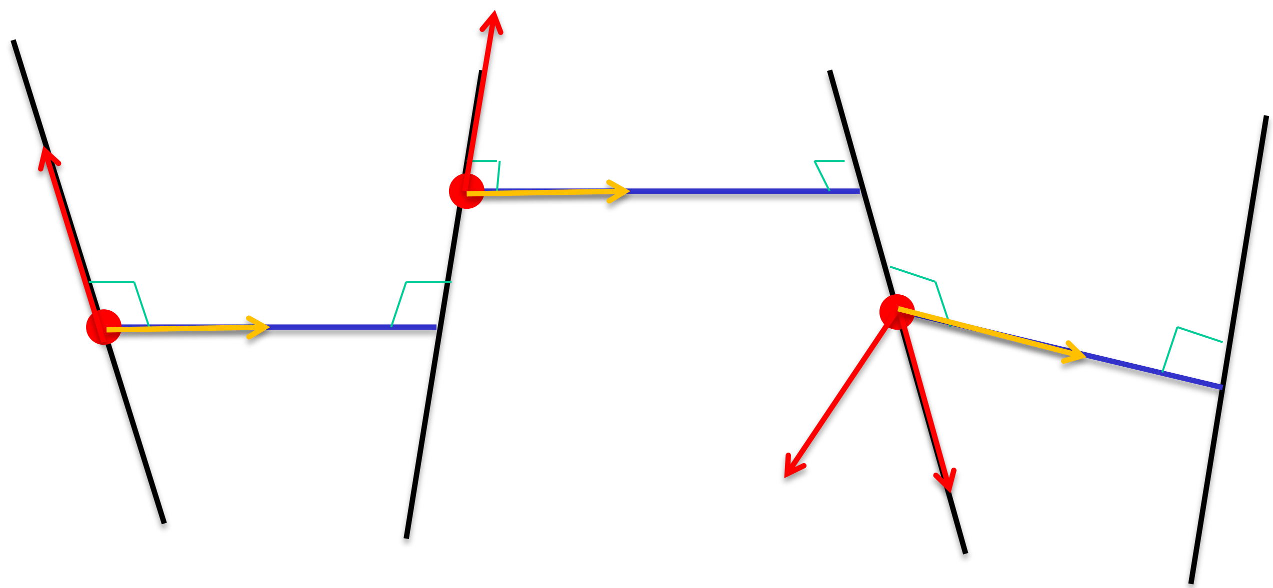
Freie Universität Berlin

Frame Attachment



y-vectors according to right hand frames

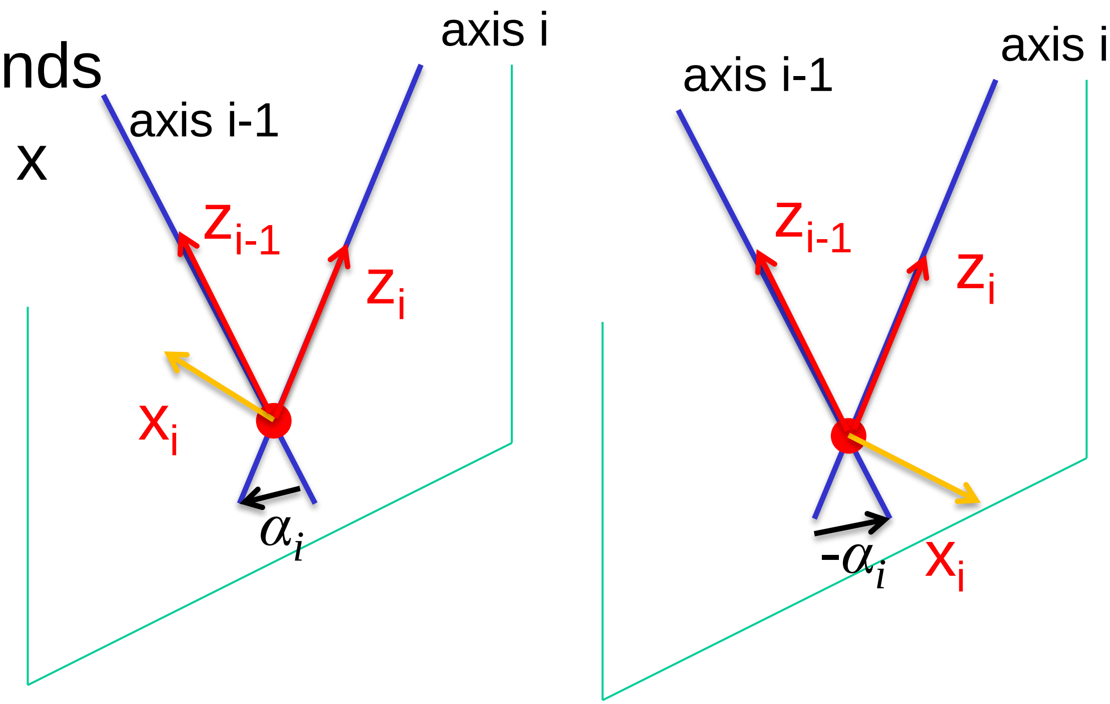
Summary: Frame Attachment



1. Normals
2. Origins
3. z-axis
4. x-axis

Intersecting Joint Axes

- Sometimes axes intersect
- Where to put the x-direction
- Sign of *α* depends on selection of x
- Free choice



- Captured by *α* and repr. in homogen. transformation

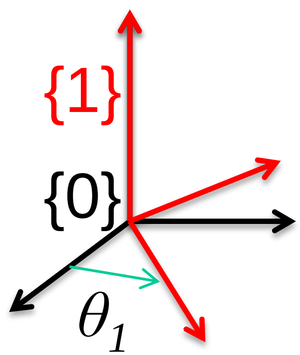
Homogenious Transformation Matrix

- How to generate a homogenious transformation matrix from D-H parameters

First Link

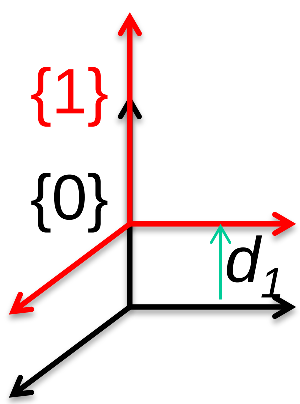
(Khatib)

- Revolute



$a_0 = 0$
$\alpha_0 = 0$
$d_1 = 0$
if $\theta_1 = 0 \rightarrow \{0\} \equiv \{1\}$

- Prismatic



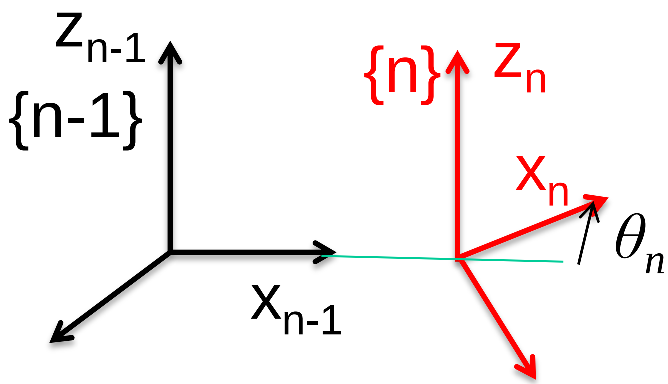
$a_0 = 0$
$\alpha_0 = 0$
$\theta_1 = 0$
if $d_1 = 0 \rightarrow \{0\} \equiv \{1\}$

Freie Universität Berlin

(Khatib)

Last Link

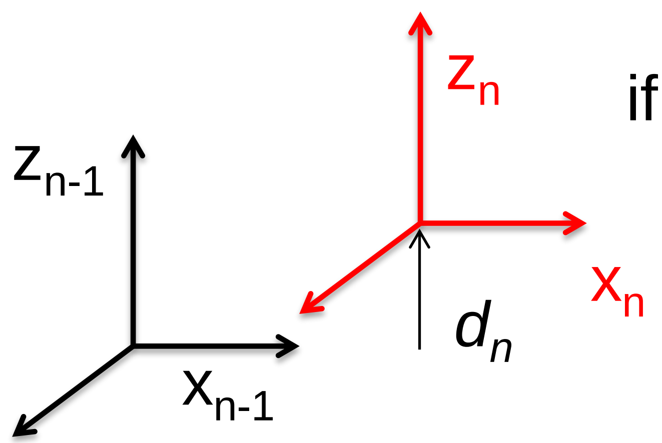
- Revolute



$$d_n = 0$$

if $\theta_n = 0 \rightarrow x_n = x_{n-1}$

- Prismatic



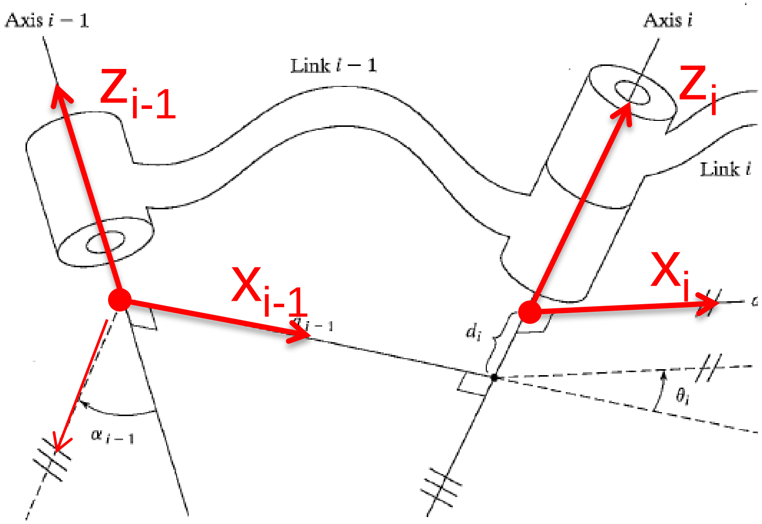
$$\theta_n = 0$$

if $d_n = 0 \rightarrow x_n = x_{n-1}$

Freie Universität Berlin

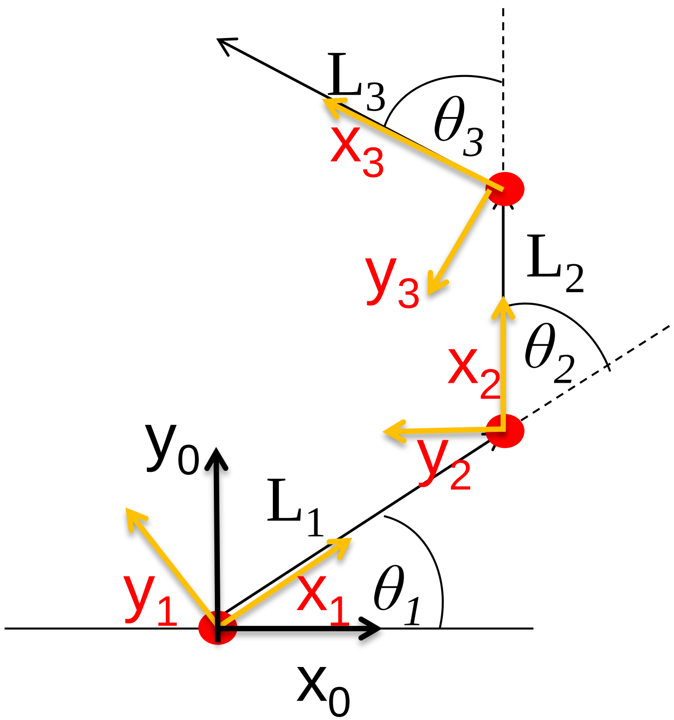
Link to Link, Homogen. Transformation

- a_{i-1} : distance z_{i-1} , z_i through x_{i-1}
- α_{i-1}: angle (z_{i-1}, z_i) around x_{i-1}
- d_i: distance x_{i-1}, x_i along z_i
- θ_i : angle (x_{i-1}, x_i) around z_i



Example RRR-Arm

- Given three angle robot



Where are the axis?
Common normals?
Frame origins?

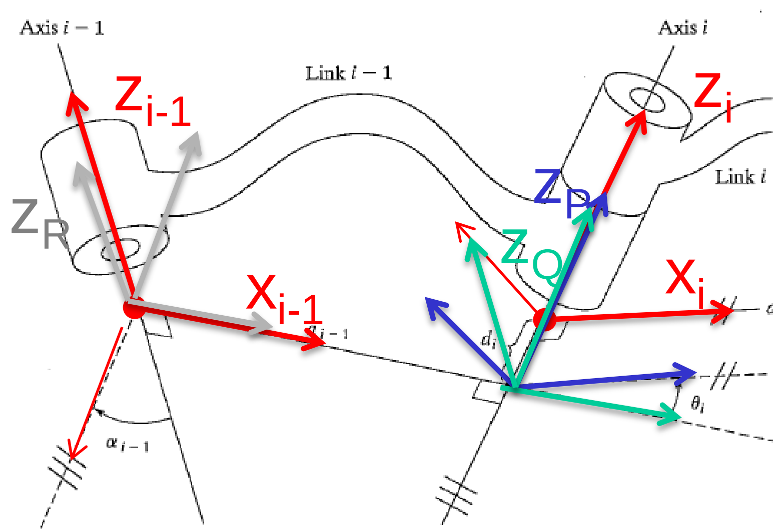
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3
4	…optional			

Configuration shown:

$\theta_{1,} = \ldots$

Transformation

- Transformation from frame to frame can be split up into 4 transformations
- Each operation requires only one operator: d_i, θ_i, a_{i-1}, α_{i-1}

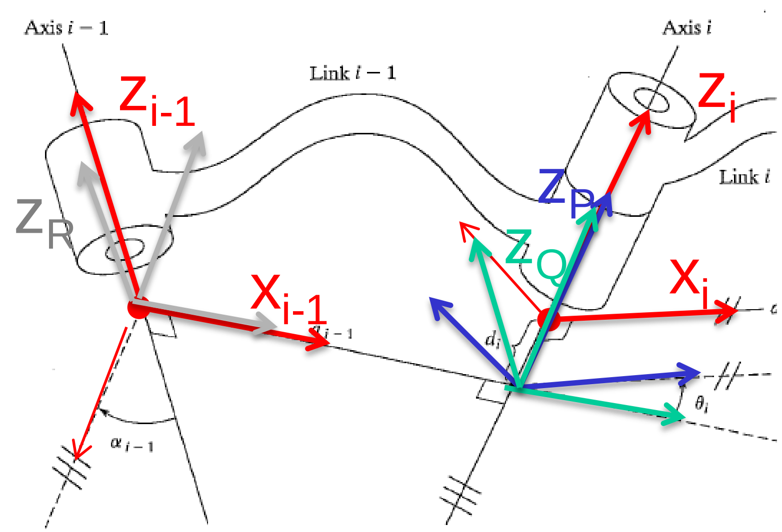


$$ {}^{i-1}_{i}T = {}^{i-1}_{R}T \cdot {}^{R}_{Q}T \cdot {}^{Q}_{P}T \cdot {}^{P}_{i}T $$

$$ {}^{i-1}_{i}T_{(\alpha_{i-1},a_{i-1},\theta_i,d_i)} = R_X(\alpha_{i-1}) \cdot D_X(a_{i-1}) \cdot R_Z(\theta_i) \cdot D_Z(d_i) $$

Transformation

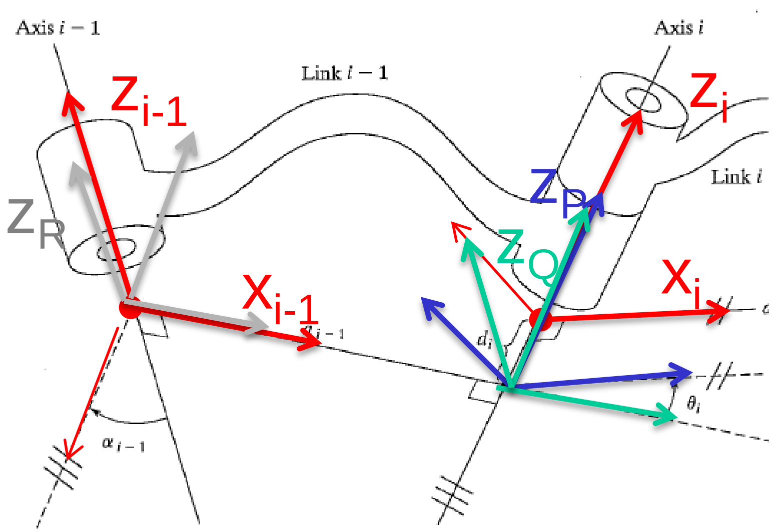
- Transformation from frame to frame can be split up into 4 transformations
- Each operation requires only one operator: d_i, θ_i, a_{i-1}, α_{i-1}



$$
{}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Forward Kinematics

- Once we have the transformations between frames we can transform coordinates for all joints of the robot



- Forward Kinematics

$$ {}^{0}_{N}T = {}^{O}_{1}T \cdot {}^{1}_{2}T \cdot {}^{Q}_{P}T \ldots {}^{N-1}_{N}T $$

Freie Universität Berlin

Kinematic Map

- For every joint angle vector we can compute the pose (position and orientation) of the end-effector, represented in the base frame by forward chaining of transformations

- Position: just transform the null vector in end-effector space to base frame
- Rotation: e.g., transform x-component $(1,0,0)^T$ from end-effector space to base frame

- Kinematic map consists of map for **position** and **orientation**

Inverse Kinematics

- Problem: How to move all joints in a coordinated fashion so that the end-effector performs a desired motion?

- Subproblems:
  1. given the joint settings, where is the end-effector
  2. given the change of a joint, how does the end-effector change
  3. given a certain end-effector position in work space, how to set the joints
  4. given a change of end-effector position, what change in joint space is required

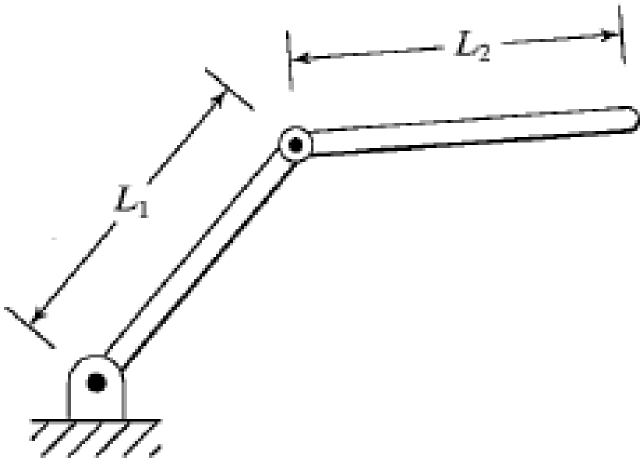
Solvability of Inverse Kinematics Problems

- Given a certain end-effector position, how to set the joints?
  - Existance of solutions?
  - Multiple solutions?
  - Method of solution?

- Existance:
  - Depends on manipulator workspace
    - Dextrous workspace: the space the robot can reach with all orientations
    - Reachable workspace: with at least one orientation

Example

- Consider a robot with two joints and two links:
  - Dextrous / Reachable workspace, for given L_1 and L_2?

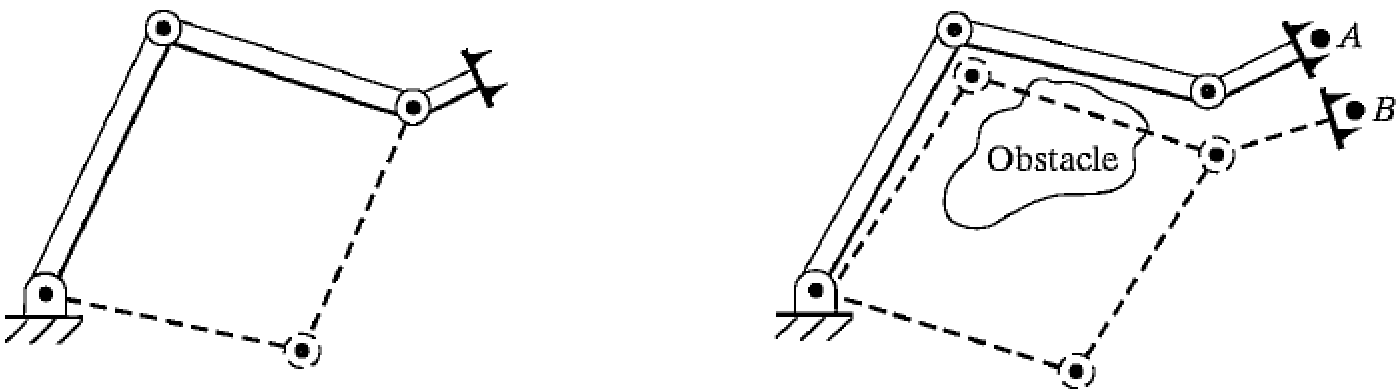


  - Not every joint can reach all 360°

Freie Universität Berlin

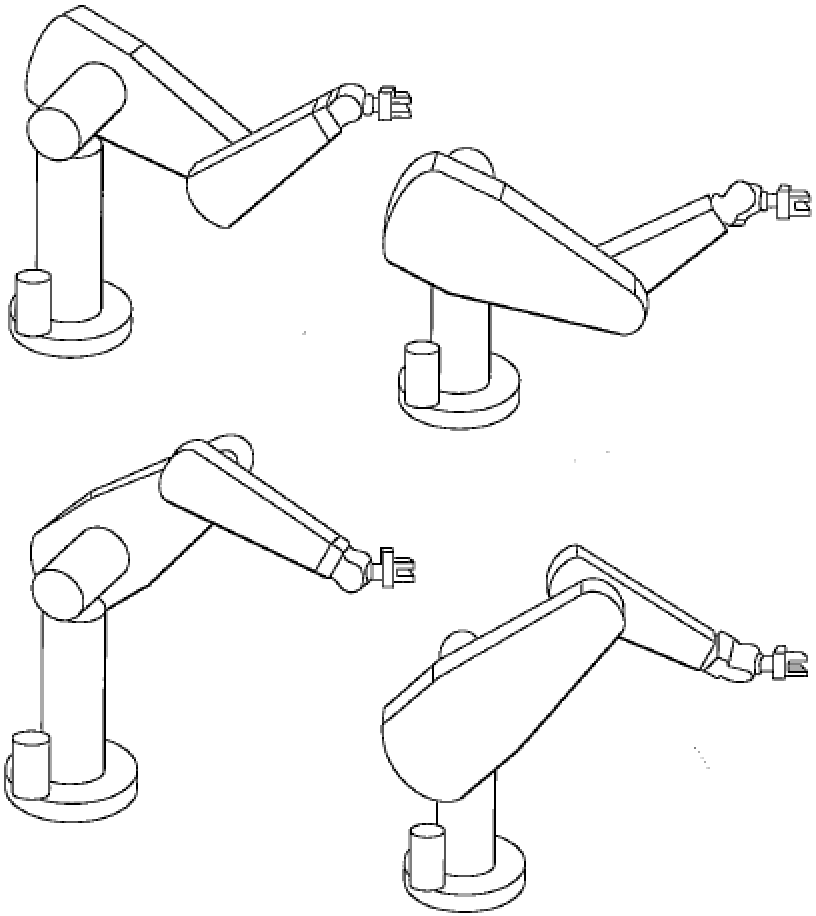
Example: Multiple Solutions

- Sometimes multiple solutions exist for a given position+orientation, sometimes one possible solution causes a collision with an obstacle or even a self-collision:



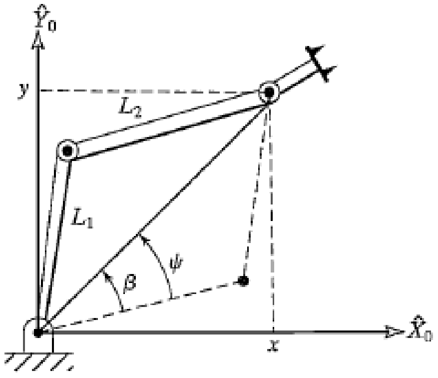
Multiple Solutions

- Puma 560:



Methods of Solution

- Closed Form
  - not available for robots with arbitrary joint configurations (transcendent equations)
  - Design consideration to enable analytic solutions, e.g., many α_i=+/- *90°*
    - Algebraic solutions (e.g. using transformation matrix)
    - Geometric solutions

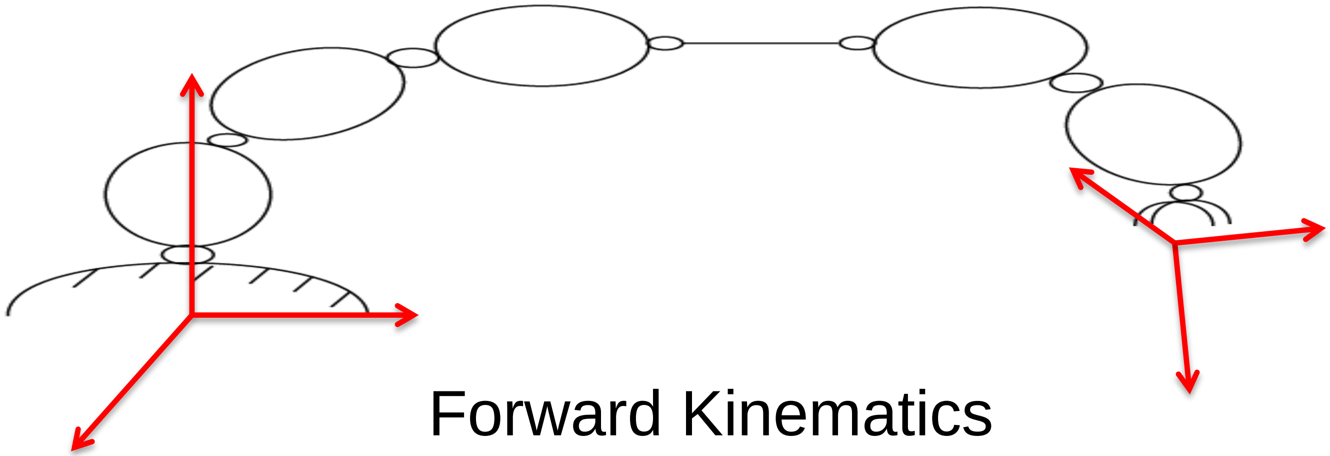


- Numerical Solutions
  - need multiple iterations
  - usually slower computation

Inverse Kinematics: Jacobian

- So far: study of static robot positioning problems
- Study of velocities and static forces leads to a matrix, called the Jacobian
- Why is it important?
  - Differential Motion
  - Linear and Angular Motion
  - Velocity Propagation
  - Static Forces

Inverse Kinematics: Jacobian



Forward Kinematics

$\theta \rightarrow x$, defined by Kinematic map

Instantaneous Kinematics

$\theta + \delta\theta \rightarrow x + \delta x$

Relationship: $\delta\theta \leftrightarrow \delta x$

$\boxed{\dot{\theta} \leftrightarrow \dot{x}}$ defined by Jacobian

Joint Coordinates

- Annotation of generalized coordinates:

- Coordinate i : θ_i *: revolute*
  d_i *: prismatic*

- Joint coordinate i: $q_i = \overline{\varepsilon_i}\theta_i + \varepsilon_i d_i$

with: $$\varepsilon_i = \begin{cases} 0: & \text{revolute} \\ 1: & \text{prismatic} \end{cases}$$

Jacobians: Direct Differentiation

- Let ϕ be a kinematic map (posit. and orientat.):

$$x = \phi(q) \qquad \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} \phi_1(q) \\ \phi_2(q) \\ \vdots \\ \phi_m(q) \end{pmatrix}$$

$$\begin{aligned} \delta x_1 &= \frac{\partial \phi_1(q)}{\partial q_1} \delta q_1 + \cdots + \frac{\partial \phi_1(q)}{\partial q_n} \delta q_n \\ &\vdots \\ \delta x_m &= \frac{\partial \phi_m(q)}{\partial q_1} \delta q_1 + \cdots + \frac{\partial \phi_m(q)}{\partial q_n} \delta q_n \end{aligned}$$

Jacobians: Direct Differentiation

- What is the Jacobian *J(q)*:

$$J(q) = \frac{\partial}{\partial q}\phi(q) = \begin{pmatrix} \frac{\partial \phi_1(q)}{\partial q_1} & \frac{\partial \phi_1(q)}{\partial q_2} & \cdots & \frac{\partial \phi_1(q)}{\partial q_n} \\ \frac{\partial \phi_2(q)}{\partial q_1} & \frac{\partial \phi_2(q)}{\partial q_2} & \cdots & \frac{\partial \phi_2(q)}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi_m(q)}{\partial q_1} & \frac{\partial \phi_m(q)}{\partial q_2} & \cdots & \frac{\partial \phi_m(q)}{\partial q_n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

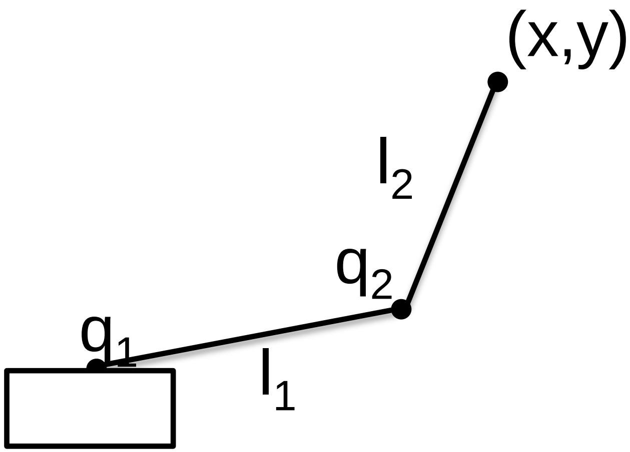
- rows: for *m* work space coordinates (e.g. position and orientation)
- columns: *n* joint space coordinates

$$\boxed{\delta x_{(m\times 1)} = J_{(m\times n)}(q)\delta q_{(n\times 1)}}$$

Jacobian

$$\begin{aligned} \delta x_{(m\times 1)} &= J_{(m\times n)}(q)\delta q_{(n\times 1)} \\ \dot{x}_{(m\times 1)} &= J_{(m\times n)}(q)\dot{q}_{(n\times 1)} \end{aligned}$$

Example



DH-parameters, transformation matrix:

$$ {}^0_2T = \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_2 & -s_2 & l_1 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ x = l_1 c_1 + l_2 c_{12} $$

$$ y = l_1 s_1 + l_2 s_{12} $$

$$ \delta x = -(l_1 s_1 + l_2 s_{12})\delta\theta_1 - l_2 s_{12} \delta\theta_2 $$

$$ \delta y = (l_1 c_1 + l_2 c_{12})\delta\theta_1 + l_2 c_{12} \delta\theta_2 $$

$$ \boxed{\delta X = \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \begin{bmatrix} -y & -l_2 s_{12} \\ x & l_2 c_{12} \end{bmatrix} \begin{pmatrix} \delta\theta_1 \\ \delta\theta_2 \end{pmatrix}} $$

How to find the joint values δθ?

Freie Universität Berlin

Representations

$$X = \begin{bmatrix} x_P \\ x_R \end{bmatrix}$$

- x_P : cartesian, spherical, cylindrical, …
- x_R : Euler angles, Direction cosines, Euler parameters, …

Jacobian for X

- Resulting Jacobian is dependent on the representation

$$\begin{aligned} \dot{x}_P &= J_{X_P}(q)\dot{q} \\ \dot{x}_R &= J_{X_R}(q)\dot{q} \end{aligned} \qquad \begin{pmatrix} \dot{x}_P \\ \dot{x}_R \end{pmatrix} = \begin{pmatrix} J_{X_P}(q) \\ J_{X_R}(q) \end{pmatrix} \dot{q}$$

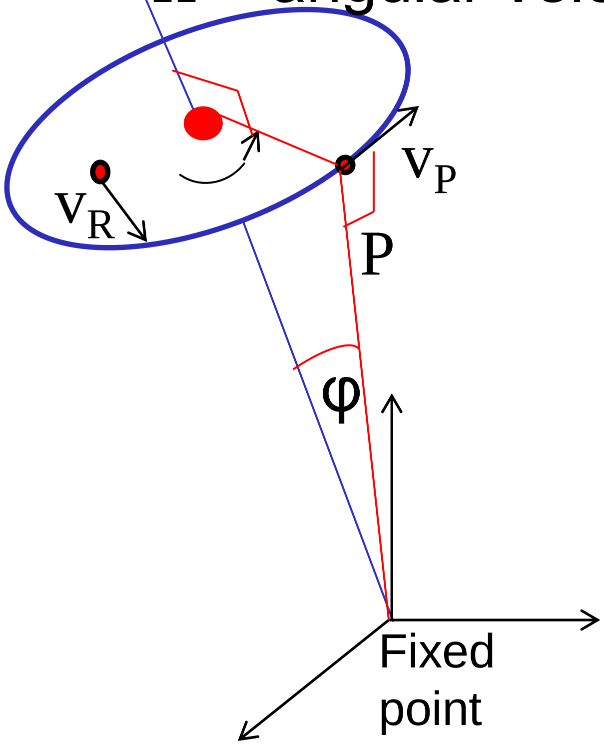
- Cartesian and Direction Cosines:

$$\dot{X}_{(12\times1)} = J_X(q)_{(12\times6)}\dot{q}_{(6\times1)}$$

Rotational Motion

(Khatib)

Ω = angular velocity (vector + magnitude)



v_P is proportional to

$\|\Omega\|$ (rotational velocity)

$\|P \sin \varphi\|$ (distance fr. center)

and

v_P is orthogonal to Ω

v_P is orthogonal to P

$$\mathbf{v_P = \Omega \times P}$$

Cross Product

$$a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \quad c = a \times b \rightarrow c = \hat{a} b$$

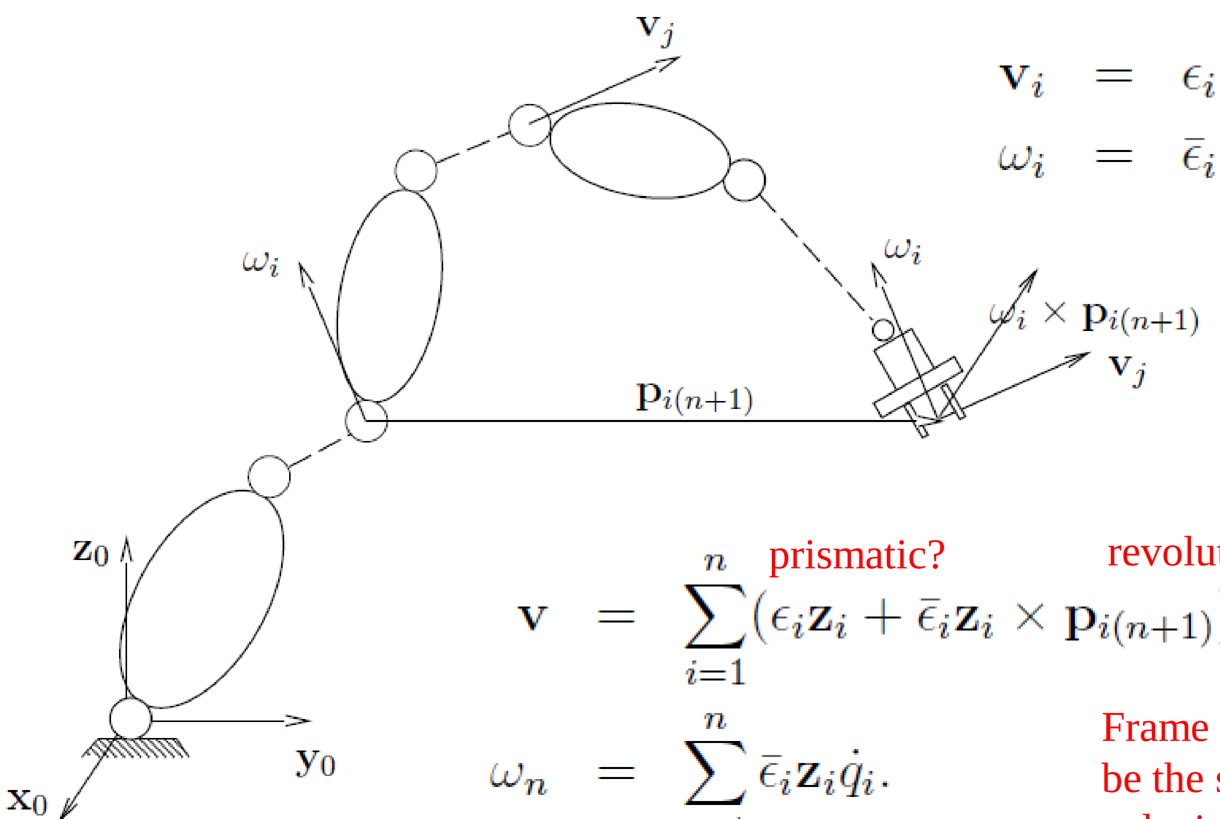
$$a\times = \hat{a} : \text{a skew-symmetric matrix}$$

$$c = \hat{a} b = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

$$\boxed{v_P = \Omega \times P \rightarrow \hat{\Omega} P}$$

Freie Universität Berlin

Linear and Rotational Velocities



$$\mathbf{v}_i = \epsilon_i \ \mathbf{z}_i \ \dot{q}_i$$
$$\omega_i = \bar{\epsilon}_i \ \mathbf{z}_i \ \dot{q}_i$$

nonzero for:
prismatic,

revolute

prismatic? revolute?

$$\mathbf{v} = \sum_{i=1}^{n} \left(\epsilon_i \mathbf{z}_i + \bar{\epsilon}_i \mathbf{z}_i \times \mathbf{p}_{i(n+1)} \right) \dot{q}_i$$

$$\omega_n = \sum_{i=1}^{n} \bar{\epsilon}_i \mathbf{z}_i \dot{q}_i .$$

Frame of reference must be the same for all velocities

Basic Jacobian J_0

- Idea: Transform all linear velocities v and angular velocities into frame {0}, S_{0i} means transforming from frame i to frame 0:

$$\mathbf{v} = \sum_{i=1}^{n} S_{0i} \left(\epsilon_i \mathbf{z}_i + \bar{\epsilon}_i \widehat{\mathbf{z}}_i \; \mathbf{p}_{i(n+1)_{(\mathcal{R}_i)}}\right) \dot{q}_i$$

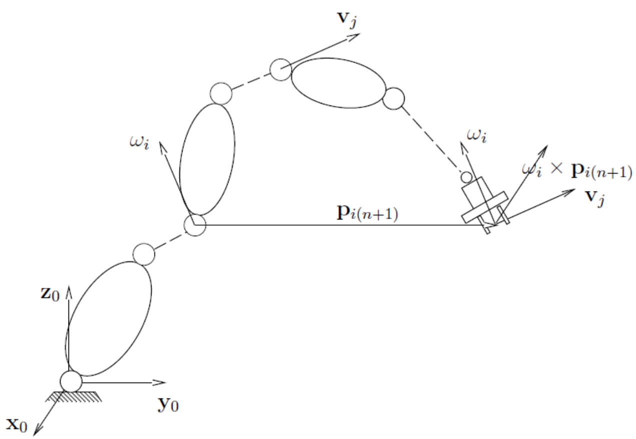
$$\omega = \sum_{i=1}^{n} S_{0i} \; \bar{\epsilon}_i \mathbf{z}_i \; \dot{q}_i;$$

$$\mathbf{z} \triangleq \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \mathbf{z}_i = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

z_i is our rotational axis Ω in frame {i}

Basic Jacobian J_0

$$\mathbf{z} \triangleq \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \mathbf{z}_i = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



$$\mathbf{v} = \sum_{i=1}^{n} S_{0i} \left(\epsilon_i \mathbf{z}_i + \bar{\epsilon}_i \widehat{\mathbf{z}}_i \ \mathbf{p}_{i(n+1)}{}_{(\mathcal{R}_i)}\right) \dot{q}_i$$

$$\omega = \sum_{i=1}^{n} S_{0i} \ \bar{\epsilon}_i \mathbf{z}_i \ \dot{q}_i;$$

$$\boxed{J_0(\mathbf{q}) = \begin{pmatrix} S_{01}\left(\epsilon_1 \mathbf{z} + \bar{\epsilon}_1 \widehat{\mathbf{z}} \mathbf{p}_{1(n+1)}{}_{(\mathcal{R}_1)}\right) & \cdots & S_{0n}\left(\epsilon_n \mathbf{z} + \bar{\epsilon}_n \widehat{\mathbf{z}} \mathbf{p}_{n(n+1)}{}_{(\mathcal{R}_n)}\right) \\ \\ \bar{\epsilon}_1 S_{01} \mathbf{z} & \cdots & \bar{\epsilon}_n S_{0n} \mathbf{z} \end{pmatrix}}$$

Basic Jacobian J_0

Summary: Joints types, linear and rotational motion

- A prismatic joint contributes to
  linear velocity, but no rotational velicity of the endeffector

- A rotational joint contributes to
  rotational velocity and linear velocity of the endeffector

Transforming Jacobians

$$J(\mathbf{q}) = E(\mathbf{x}) J_O(\mathbf{q})$$

$$E(\mathbf{x}) = \begin{pmatrix} E_p(\mathbf{x}_p) & 0 \\ 0 & E_r(\mathbf{x}_r) \end{pmatrix} \qquad \begin{bmatrix} {}^{A}v \\ {}^{A}\omega \end{bmatrix} = \left[\begin{array}{c|c} {}^{A}_{B}R & 0 \\ \hline 0 & {}^{A}_{B}R \end{array}\right] \begin{bmatrix} {}^{B}v \\ {}^{B}\omega \end{bmatrix}$$

Cartesian Coordinates: $E_P(x)$ is the identity matrix of order 3
Cylindrical coordinates:

$$E_p(\mathbf{x}) = \begin{pmatrix} \cos\theta/\rho & \sin\theta/\rho & 0 \\ -\sin\theta/\rho & \cos\theta/\rho & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Singularities

Two types of Singularities of the Mechanism:

- Workspace Boundary Singularities:
  - Manipulator is fully stretched or folded,
  - The endeffector is at or very near the boundary of the workspace
- Workspace-interior singularities:
  - Caused by lining up two or more joint axes
- Example:

$$DET[J(\Theta)] = \begin{bmatrix} l_1 s_2 & 0 \\ l_1 c_2 + l_2 & l_2 \end{bmatrix} = l_1 l_2 s_2 = 0$$

Singular for $\theta_2 = 0°$ or $180°$

Inverse Kinematics Problem

- If we want a certain change in workspace *δx*, how to change the joints *δq*
- We know: $\delta x = J(q)\delta q$
- Iff the Jacobian was invertible:

$$\delta q = J^{-1}(q)\delta x$$

but usually it is not:

$$J \in \mathbb{R}^{m \times n} \quad \text{with } m \neq n$$

- Formulate an optimality principle to choose δq for given δx
  - related to taking the pseudo-inverse $J^\sharp$ instead of J^{-1}

Freie Universität Berlin

Inverse Kinematics Optimality Principle

(Toussaint)

- Given: current joint state q_t ,
  current end-effector position $x_t = \phi(q_t)$
  and desired end-effector position x^* , such that:

1) $\phi(q_{t+1})$ is close to x^* (move end-effector)
2) q_t is close to q_{t+1} (be lazy / efficient)

- Formalized as an objective function:

$$\boxed{f(q_{t+1}) = \|q_{t+1} - q_t\|_W^2 + \|\phi(q_{t+1}) - x^*\|_C^2}$$

Inverse Kinematics

$$\boxed{f(q_{t+1}) = \|q_{t+1} - q_t\|_W^2 + \|\phi(q_{t+1}) - x^*\|_C^2}$$

- When using local linearization, $\phi(q_{t+1}) \approx \phi(q_t) + J(q_{t+1} - (q_t))$, the optimal next joint state q_{t+1}, that minimizes $f(q_{t+1})$ is:

$$\begin{aligned} q_{t+1} &= q_t + J^\sharp(x^* - x_t) \\ \delta q &= J^\sharp \delta x \\ J^\sharp &= (J^T C J + W)^{-1} J^T C \\ &= W^{-1} J^T (J W^{-1} J^T + C^{-1})^{-1} \end{aligned}$$

- Last equality: Woodbury identity

Inverse Kinematics

$$J^\sharp \;=\; W^{-1}J^T(JW^{-1}J^T + C^{-1})^{-1}$$

- for $C \to \infty$ and $W = \mathbf{I}$,

$\boxed{J^\sharp = J^T(JJ^T)^{-1}}$ is called pseudo-inverse

- *W* generalizes the metric space
- *C* regularizes this pseudo inverse (see later section on singularities)

Freie Universität Berlin

Iterating Inverse Kinematics

(Toussaint)

- Assume initial posture q_0. We want to reach a desired endeff position x^* in T steps:

Input: initial state q_0, desired x^*, methods ϕ_{pos} and J_{pos}
Output: trajectory $q_{0:T}$
Set $x_0 = \phi_{\text{pos}}(q_0)$ $\quad\triangleright$ current (old) endeff position
for $t = 1 : T$ **do**
  $x \leftarrow \phi_{\text{pos}}(q_{t\text{-}1})$ $\quad\triangleright$ current endeff position
  $J \leftarrow J_{\text{pos}}(q_{t\text{-}1})$ $\quad\triangleright$ current endeff Jacobian
  $\hat{x} \leftarrow x_0 + (t/T)(x^* - x_0)$ $\quad\triangleright$ interpolated endeff target
  $q_t = q_{t\text{-}1} + J^\sharp(\hat{x} - x)$ $\quad\triangleright$ new joint positions
  Command q_t to all robot motors and compute all ${}^0_iT(q)$
end for

Why does this not follow the interpolated trajectory $\hat{x}_{0:T}$ exactly? What if T = 1 and x is far?*

(Marc Toussaint)

Summary

- We know how to:
  - map points between different frames
  - attach frames to a robot (DH-parameters)
  - derive basic motion generation principles using forward and inverse kinematics
  - apply a basic notion of optimality for path generation

Aspects of Inverse Kinematics

- Inverse Kinematics and Motion Rate Control
- Singularities
- Null-Space
- Multiple Tasks

Remarks on Notion of Terms

- Notion of „**Kinematics**“ describes mapping $\phi : q \rightarrow x$ usually a many-to-one-function
- Notion of „Inverse Kinematics“, in a strict sense describes a mapping $g : x \rightarrow q$, so that $\phi(g(x)) = x$, which is usually not unique (and non-optimal in our setting)
- In practice with **Inverse Kinematics** is meant

$$\delta q = J^\sharp \delta x$$

- When iterating $\delta q = J^\sharp \delta x$ in a control cycle with time step τ, $\tau \approx 1 - 10ms$, $\dot{x} = \delta x / \tau,$ $\dot{q} = \delta q / \tau,$ $\dot{q} = J^\sharp \dot{x}$. Thus, the control effectively controls the end-effector velocity, therefore it is called **motion rate control**.

Null-Space

- The space of all $q \in \mathbb{R}^n$ is called **joint/configuration space**
  The space of all $x \in \mathbb{R}^m$ is called **task/operational space**
  Usually $m < n$, which is called **redundancy**

- For a desired endeffector state x^* there exists a whole manifold (assuming ϕ is smooth) of joint configurations q:

$$\textbf{nullspace}(x^*) = \{q \mid \phi(q) = x^*\}$$

- Plain $\delta q = J^\sharp \, \delta x$ resolves redundancy based on the "be lazy" criterion.
  One can also add **null space motion**: an additional drift $h \in \mathbb{R}^n$ in the nullspace of the task:

$$\delta q = J^\sharp \, \delta x + (I - J^\sharp J) \, h$$

  This corresponds to a cost term $\|q_{t+1} - q_t - h\|_W^2$ in $f(q_{t+1})$!

(Marc Toussaint)

Singularities, continued

- In general: A matrix J **singular** $\iff \mathrm{rank}(J) < m$
  - rows of J are linearly dependent
  - dimension of image is $< m$
  - $\delta x = J \delta q \quad \Rightarrow$ dimensions of δy limited
  - Intuition: arm fully stretched

- Implications:
  $\det(J J^\top) = 0$
  - $\to$ pseudo-inverse $J^\top (J J^\top)^{-1}$ is ill-defined!
  - $\to$ inverse kinematics $\delta q = J^\top (J J^\top)^{-1} \delta x$ computes "infinite" steps!

- **Singularity robust pseudo inverse** $J^\top (J J^\top + \epsilon \mathbf{I})^{-1}$
  The term $\epsilon \mathbf{I}$ is called **regularization**
- Recall our general solution (for $W = \mathbf{I}$)
  $$J^\sharp = J^\top (J J^\top + C^{-1})^{-1}$$
  is already singularity robust

(Marc Toussaint)

Multiple Tasks

- Assume we have d simultaneous tasks; for each task i we have:
  - a kinematic mapping $x_i = \phi_i(q) \in \mathbb{R}^{m_i}$
  - a current value $x_{i,t} = \phi_i(q_t)$
  - a desired value x_i^*
  - a metric C_i or precision ϱ_i (related via $C_i = \varrho_i \, \mathbf{I}$)

- Each task contributes a term to the objective function

$$\begin{aligned} f(q_{t+1}) &= \|q_{t+1} - q_t\|_W^2 \\ &\quad + \|\phi_1(q_{t+1}) - x_1^*\|_{C_1}^2 \\ &\quad + \varrho_2 \, \|\phi_2(q_{t+1}) - x_2^*\|^2 \\ &\quad + \cdots \end{aligned}$$

- A bit algebra $\rightarrow$ the optimal joint step is:

$$q_{t+1} = q_t + \Big[\sum_{i=1}^{d} J^\top C_i J + W\Big]^{-1} \Big[\sum_{i=1}^{d} J^\top C_i (x_i^* - x_{i,t})\Big]$$

(Marc Toussaint)

Multiple Tasks

- A much nicer way to write (and code) exactly the same:

$$f(q_{t+1}) = \|q_{t+1} - q_t\|^2_W + \Phi(q_{t+1})^\top \Phi(q_{t+1})$$

with the "big task vector" $\Phi(q_{t+1}) := \begin{pmatrix} M_1\ (\phi_1(q_{t+1}) - x_1^*) \\ \varrho_2\ (\phi_2(q_{t+1}) - x_2^*) \\ \vdots \end{pmatrix} \quad \in \mathbb{R}^{\sum_i m_i}$

where M_1 is the Cholesky decomposition $C_1 = M_1^\top M_1$.

- The optimal joint step now is:

$$q_{t+1} = q_t - (J^\top J + W)^{-1} J^\top\ \Phi(q_t)$$

with $J \equiv \frac{\partial \Phi(q)}{\partial q}$ the "big Jacobian".

(Marc Toussaint)

Literature

- John J. Craig: Introduction to Robotics, Mechanics and Control, Third Edition, Pearson, Prentice Hall, 2005.
- Oussama Khatib: Advanced Robotic Manipulation, Lecture Notes (CS327A), 2005.
- Oussama Khatib: Introduction to Robotics, Online Lecture, 2008
- Marc Toussaint: Lecture Notes on „Robotics“, 2010