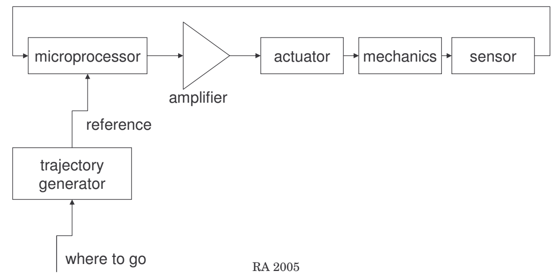
Robotica Antropomorfa

Lezione 10

Back to the global view



Kinematics

- Kinematics:
  - Given the joint angles, compute the hand position

$$\mathbf{x} = \Lambda(\mathbf{q})$$

- Inverse kinematics:
  - Given the hand position, compute the joint angles to attain that position

$$\mathbf{q} = \Lambda^{-1}(\mathbf{x})$$

- As usual, inverse problems might be troublesome!

Kinematics

- Inverting:
  - Geometrically: closed form solution exists in certain cases
  - By minimization:

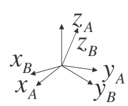
$$J = \frac{1}{2}\left\|\mathbf{x} - \Lambda(\mathbf{q})\right\|^2 \Rightarrow \mathbf{q} = \arg\min_{\mathbf{q}} J$$

- Kinematic redundancy: more joints than constraints
  - E.g. a rigid body (hand) in space is described by 6 numbers (position + orientation). A robot (or human) arm might have 7 or more joints (degrees of freedom)

Representing kinematics

- Representing rotations and translations between coordinate frames of reference

$${}^{A}v = [?]\,{}^{B}v$$



$${}^{A}v = [\,{}^{A}x_B \mid {}^{A}y_B \mid {}^{A}z_B\,]\,{}^{B}v = {}^{A}R_B\,{}^{B}v \qquad B \rightarrow A$$

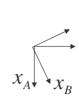
$${}^{A}x_B = {}^{A}R_B\,{}^{B}x_B = {}^{A}R_B[1,0,0]^T$$

Rotation matrix

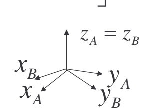
$${}^{A}R_B({}^{A}R_B)^T = I \Leftrightarrow ({}^{A}R_B)^T = ({}^{A}R_B)^{-1} = {}^{B}R_A$$

Orthogonal matrix

Example: rotation along the Z axis $\begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix}$



$${}^{A}x_B = \begin{bmatrix} \cos\vartheta \\ \sin\vartheta \\ 0 \end{bmatrix}$$



More simple rotations

Example: rotation along the Y axis $\begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\vartheta & 0 & \cos\vartheta \end{bmatrix}$

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\vartheta & -\sin\vartheta \\ 0 & \sin\vartheta & \cos\vartheta \end{bmatrix}$ Example: rotation along the X axis

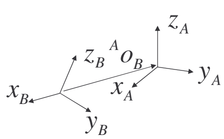
Representing 3D rotations

- Sequences of elementary rotations
  - Euler angles: z, y, z or z, x, z
  - Roll, pitch, yaw angles: z, y, x
  - Vector (axis of rotation) and angle

Roto-translation

- Rotation combined with translation

$${}^{A}v = {}^{A}R_{B}\,{}^{B}v + {}^{A}o_{B}$$



Homogeneous representation

- To make things uniform

$${}^{A}v = {}^{A}R_{B}\,{}^{B}v + {}^{A}o_{B}$$

$$\begin{bmatrix} {}^{A}v \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}R_{B} & {}^{A}o_{B} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{B}v \\ 1 \end{bmatrix}$$

$${}^{A}v = {}^{A}T_{B}\,{}^{B}v \qquad \dim(v) = 4$$

Clearly

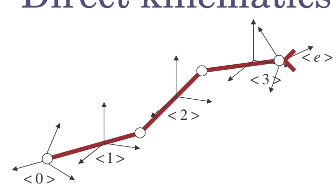
$${}^{A}v = {}^{A}T_{B}\,{}^{B}T_{C}\,{}^{C}v \qquad C \to A$$

$$\begin{bmatrix} {}^{A}R_{B} & {}^{A}o_{B} \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} {}^{A}R_{B}^{T} & -{}^{A}R_{B}^{T}\,{}^{A}o_{B} \\ 0 & 1 \end{bmatrix}$$

$${}^{A}T_{B}^{-1} = {}^{B}T_{A}$$

- Composition of transforms
- Inverse of a rototranslation

Direct kinematics



$${}^{0}T_{1}(q_{1}) \cdots {}^{n-1}T_{n}(q_{n})$$

$$(x, y, z) = {}^{0}T_{e}(q_{1}, q_{2}, q_{3}, q_{4}) \cdot (0,0,0)^{T}$$

$$\mathbf{x} = \Lambda(\mathbf{q})$$

$$\textbf{orientation} = \tilde{\Lambda}(\mathbf{q})$$

Conventions

- For placing the reference frames on each link
  - Denavit-Hartenberg
- Many times DH parameters are given for a manipulator (and various useful equations are also given wrt DH convention)

Inverse kinematics

- Direct approach
- Geometric
- Minimization
- Neural network, learning

Inverse kinematics

- Direct approach
  - Try solving:

$$x = NL_x(q_1, q_2, q_3, q_4)$$
$$y = NL_y(q_1, q_2, q_3, q_4)$$
$$z = NL_z(q_1, q_2, q_3, q_4)$$

for q_1, q_2, q_3, q_4

Geometric approach

- For certain manipulator the solution exists in close form
  - Decomposable structures (e.g. translation and rotations can be handled separately)
  - Rotations follow certain rules
- Many industrial manipulators were designed with inverse kinematics in mind

Minimization

- Find the solution to:

$$J = \frac{1}{2}\left\|\mathbf{x} - \Lambda(\mathbf{q})\right\|^2 \Rightarrow \mathbf{q} = \arg\min_{\mathbf{q}} J$$

- Neural network/learning:

$$(\mathbf{q}, \mathbf{x}) \rightarrow \Lambda^{-1}$$

- Approximate the inverse out of a family of functions (NN approach) starting from examples

What about velocity?

- Jacobian matrix

$$\mathbf{x} = \Lambda(\mathbf{q}) \Rightarrow \frac{d\mathbf{x}}{dt} = \begin{bmatrix} \frac{dx_1}{dq_1} & \cdots & \frac{dx_1}{dq_m} \\ \vdots & \ddots & \vdots \\ \frac{dx_n}{dq_1} & \cdots & \frac{dx_n}{dq_m} \end{bmatrix} \cdot \frac{d\mathbf{q}}{dt}$$

$$\frac{d\mathbf{x}}{dt} = J(\mathbf{q}) \cdot \frac{d\mathbf{q}}{dt}$$

Note on representing velocities

- If **x** is:

$$\mathbf{x} = (x, y, z, \vartheta, \varphi, \psi)$$

- Position + Euler angles

$$\mathbf{v} = (v_x, v_y, v_z, \dot{\vartheta}, \dot{\varphi}, \dot{\psi})$$

- Euler angles derivatives do not have any clear physical meaning

$$\mathbf{v} = (v_x, v_y, v_z, \boldsymbol{\omega})$$

- Angular velocity (rate of rotation along the axis

Anyway…

- Just make sure the representation and the equations are consistent

$$\mathbf{v} = (v_x, v_y, v_z, \dot{\vartheta}, \dot{\varphi}, \dot{\psi}) \Rightarrow J_r$$

$$\mathbf{v} = (v_x, v_y, v_z, \boldsymbol{\omega}) \Rightarrow J_v$$

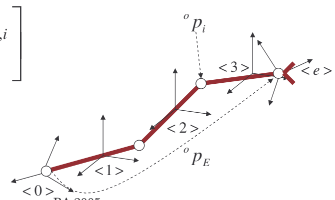
Jacobian

- Formula
  - Given the DH representation of transformations
  - Considering only rotational joints

$$J_v = [J_1 \mid J_2 \cdots J_n] \quad \text{for } n \text{ joints}$$

$$J_i = \begin{bmatrix} {}^o z_i \times {}^o p_{E,i} \\ {}^o z_i \end{bmatrix}$$

$${}^o p_{E,i} = {}^o p_E - {}^o p_i$$



Having written

$${}^0T_i = \begin{bmatrix} {}^0x_i & {}^0y_i & {}^0z_i & {}^0p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0T_i = {}^0T_1 \, {}^1T_2 \cdots {}^{i-1}T_i$$

When J is invertible

- Can compute the joint velocities to obtain a certain hand velocity

$$\dot{\mathbf{q}} = J^{-1}\dot{\mathbf{x}}$$

- If n>6, redundancy:

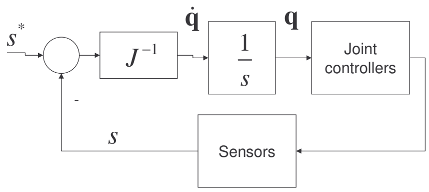
$$\dot{\mathbf{q}} = J^{+}\dot{\mathbf{x}} + (I - J^{+}J)\mathbf{k}$$

- **k** is a constant vector

Troubles

- Even if n≤6 there are many situations where *J* cannot be inverted (singularities)
  - Movement singularities (chain of rotations)
  - *J* not invertible because certain elements go to zero

Resolved rate controller



Static

- Relationship between forces and torques

$$d\mathbf{x} = Jd\mathbf{q}$$
$$d\mathbf{q}^T\boldsymbol{\tau} = d\mathbf{x}^T\mathbf{F}$$
$$d\mathbf{q}^T\boldsymbol{\tau} = d\mathbf{q}^T J^T\mathbf{F}$$
$$\Downarrow$$
$$\boldsymbol{\tau} = J^T\mathbf{F}$$

- Imagining the integrals where appropriate

Another idea

$$\boldsymbol{\tau} = J^T\mathbf{F}$$

- Use this equation to design a force controller:
  - Given **F** compute the torques to drive the joints

Dynamics

- Two methods to derive the equation of motion (differential equations)
  - Newton-Euler
  - Lagrange formalism

Newton-Euler

- Start from: $\begin{cases} \mathbf{F} = \dfrac{d}{dt}(m\mathbf{v}) \\ \boldsymbol{\tau} = \dfrac{d}{dt}(I\boldsymbol{\omega}) \end{cases}$

$$\begin{cases} \mathbf{F} = \dfrac{d}{dt}(m\mathbf{v}) \\ \boldsymbol{\tau} = \dfrac{d}{dt}(I\boldsymbol{\omega}) = \boldsymbol{\omega}\times(I\boldsymbol{\omega}) + I\dot{\boldsymbol{\omega}} \\ \text{kinematics} \end{cases}$$

⇨ Write down every equation (6): find the angular velocity and *I* with respect to a base frame

Lagrange formulation

- Lagrange equations:

$$\begin{cases} L = K - P \\ \displaystyle\sum_\mu F_\mu \frac{\partial x_\mu}{\partial q_i} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \end{cases}$$

$$x_\mu = x_\mu(q_1 \cdots q_N, t)$$

External forces (no potential)

$$K = \frac{1}{2}m\mathbf{v}^T\mathbf{v} + \frac{1}{2}\boldsymbol{\omega}^T I\boldsymbol{\omega}$$

For a manipulator

- Take the joint angles as variable, write the position x of the links, write down K, P and the external forces

$$\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + h(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$$

External forces (control)
Inertia (generalized)
Coriolis, centrifugal effects
Gravity

Complexity

- Newton-Euler: $o(n)$
- Lagrange: $o(n^4)$

Estimation

- Kinematics → just measure the params
- Dynamics → estimate from data

Dynamics

- Direct dynamics:

$$\tau(t) \rightarrow q(t)$$

- Simulation (integrate the equations – Runge-Kutta, Euler, etc.)
- Inverse dynamics:

$$q(t) \rightarrow \tau(t)$$

Dynamics and control

- Case 1: parameters are such that feedback gain at each joint is >> gravity, Coriolis, centrifugal, disturbances, etc.
- Case 2: feedback in not enough for high-speed, precision, etc. → compensation is required

Case 1

- Approx behavior:

$$A\ddot{\mathbf{q}} + B\dot{\mathbf{q}} + k[\mathbf{q} - \mathbf{q}^*] = 0$$

- Can design k or a PID controller to make this system behave as desired

Case 2

- Let's imagine we know all the parameters with a certain precision:

$$\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + h(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$$
$$\boldsymbol{\tau}_{control} = M(\mathbf{q})\mathbf{u} + h(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$$
$$M(\mathbf{q})\ddot{\mathbf{q}} + h(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = M(\mathbf{q})\mathbf{u} + h(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$$
$$M(\mathbf{q})\ddot{\mathbf{q}} = M(\mathbf{q})\mathbf{u}$$

$$\mathbf{u} = \ddot{\mathbf{q}}^* + k_d(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}) + k_p(\mathbf{q}^* - \mathbf{q})$$

Case 2 (continued)

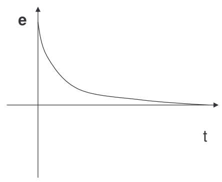
$$\ddot{\mathbf{q}} = \mathbf{u}$$

$$\mathbf{u} = \ddot{\mathbf{q}}^* + k_d(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}) + k_p(\mathbf{q}^* - \mathbf{q})$$

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^* + k_d(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}) + k_p(\mathbf{q}^* - \mathbf{q})$$

$$\mathbf{e} = \mathbf{q}^* - \mathbf{q}$$

$$0 = \ddot{\mathbf{e}} + k_d\dot{\mathbf{e}} + k_p\mathbf{e}$$



- Appropriate design of the gains can get arbitrary exponential behavior of the error