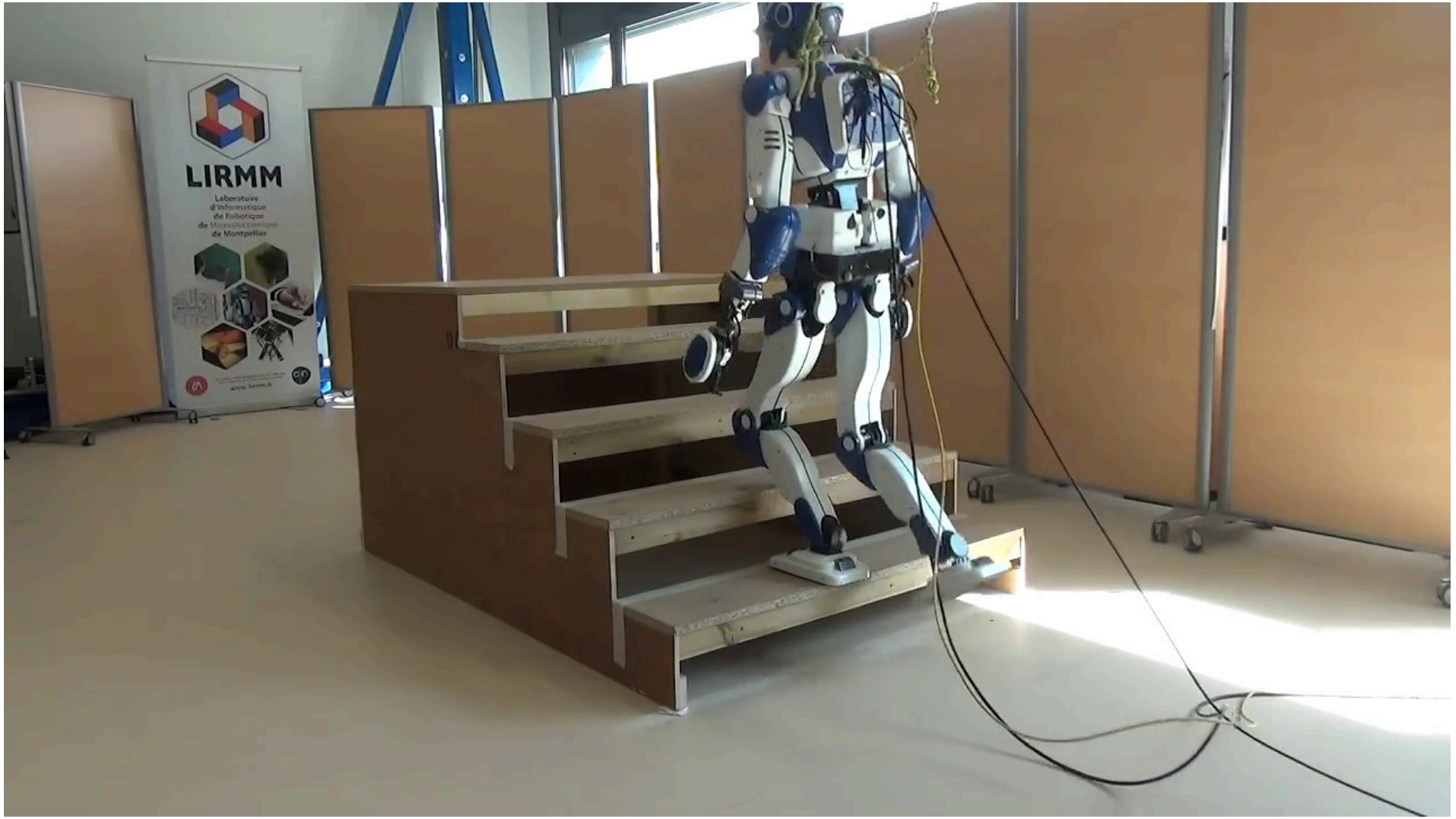
Inverse kinematics

Stéphane Caron

Robotics - Master MVA

24 October 2024

By the end of today's lecture



Quadratic programming 🔨

Quadratic programming

A quadratic program is a convex optimization problem:

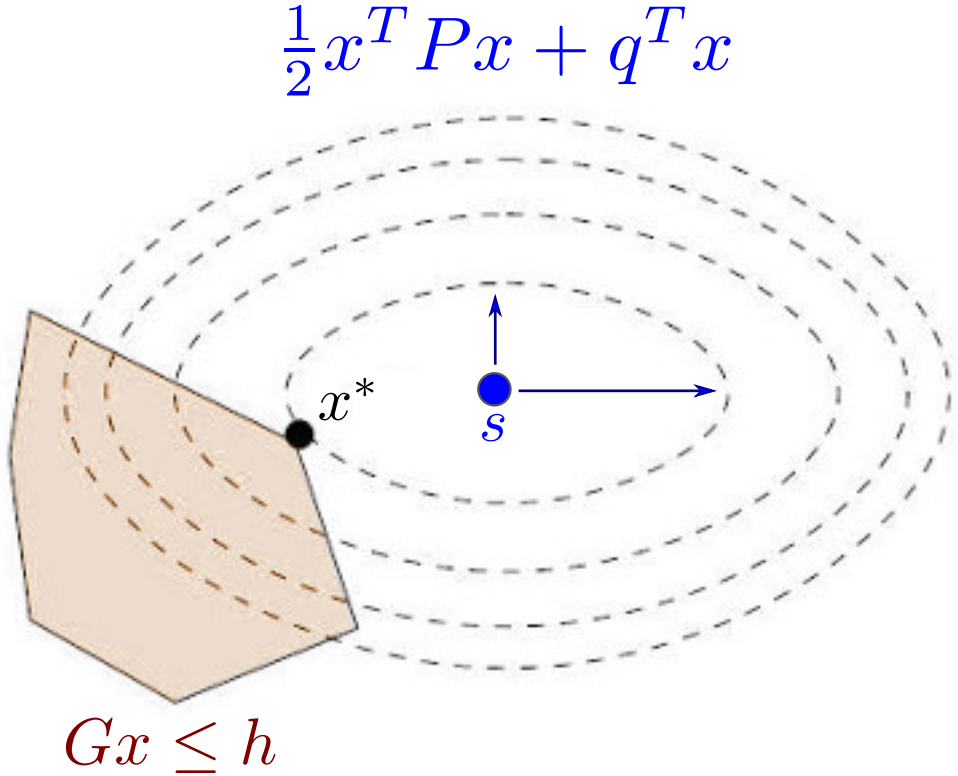
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T P x + q^T x \\ \text{subject to} \quad & Gx \leq h \end{aligned}$$

Efficient open-source solvers available, for example:

```
from qpsolvers import solve_qp

M = np.array([[1., 2., 0.], [-8., 3., 2.], [0., 1., 1.]])
P = M.T @ M # this is a positive definite matrix
q = np.array([3., 2., 3.]) @ M
G = np.array([[1., 2., 1.], [2., 0., 1.], [-1., 2., -1.]])
h = np.array([3., 2., -2.])

x = solve_qp(P, q, G, h, solver="proxqp")
```



End-effector velocity

Forward kinematics

- **Pose** $SE(3)$: position and orientation of a frame B with respect to another W:

$$M_{WB} = \begin{bmatrix} R_{WB} & {}_W p_{WB} \\ 0_{1\times 3} & 1 \end{bmatrix}$$

- **Velocity** $\mathfrak{se}(3)$: angular and linear velocities ${}_B\xi_{WB} = ({}_B\omega_{WB}, {}_B v_{WB})$:

$$M_{WB}(t) = M_{WB}(0) \oplus {}_B\xi_{WB} := M_{WB}(0)\exp(t[{}_B\xi_{WB}])$$
$${}_B\xi_{BC} = M_{WC} \ominus M_{WB} := \log(M_{WC}^{-1} M_{WB})$$

- Contrary to 3D Euclidean space: things **don't commute**.
  - Go forward, take a quarter turn, go forward $\neq$ take a quarter turn, go forward, go forward

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \neq \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Configuration space

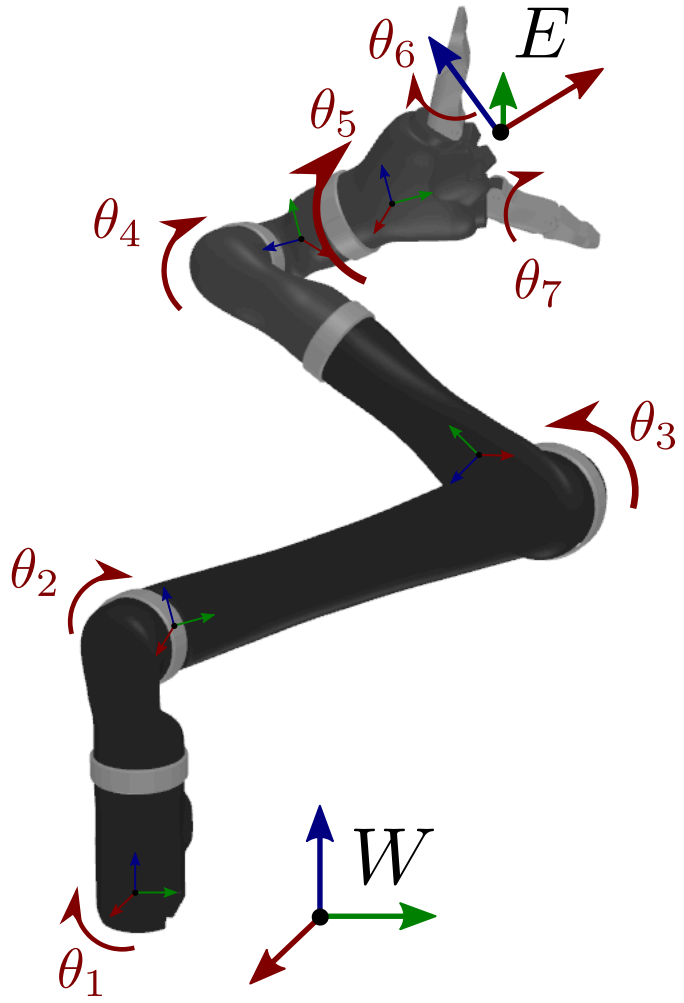
Configuration $q \in \mathcal{C}$: set of transformations to apply to robot bodies. In practice:

$$q = (M_{WB}, q_1, \ldots, q_n)$$

- $M_{WB} \in SE(3)$ for the floating base,
- q_i for joint angles.

Kinematic chain to an end-effector frame E:

$$M_{WE}(q) = M_{WB} \cdot M_{BL_1}(q_1) \cdots M_{L_{i-1}L_i}(q_i)$$



Joint velocity

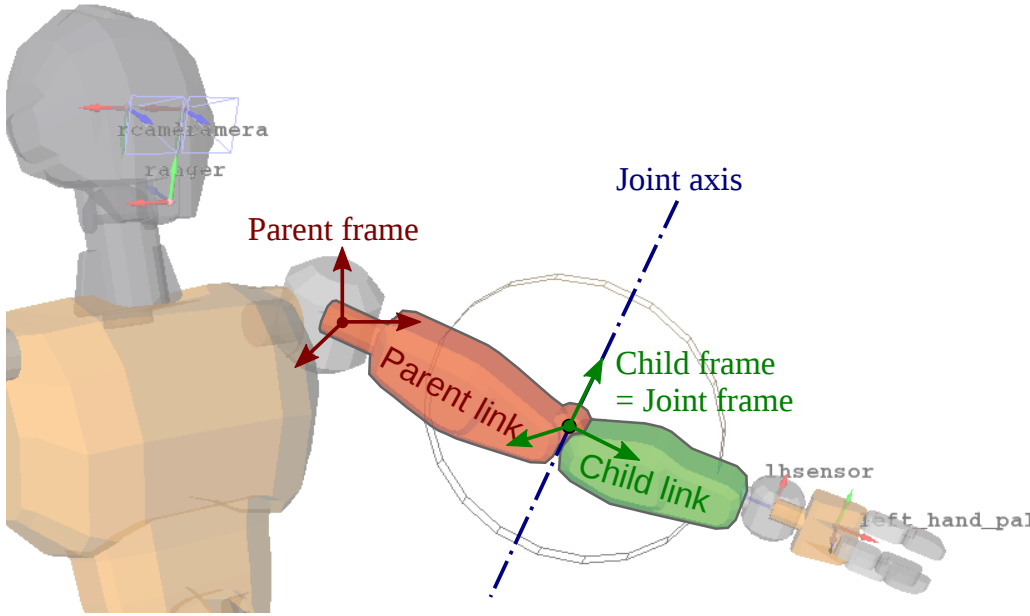
Configuration displacements result from joint velocities:

- ${}_B\xi_{WB} \in \mathfrak{se}(3)$ for the floating base,
- $\dot{q}_i$ for joint angles.

For a joint i between P_i (parent) and C_i (child):

- Pose: $M_i(q_i) := M_{P_iC_i}(q_i) \in SE(3)$
- Velocity: $\xi_i(q_i) = S_i(q_i)\dot{q}_i \in \mathfrak{se}(3)$

The velocity can always be written in this form.



Revolute joint velocity

Example where the joint rotates around the local x-axis:

$$M_i(q_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q_i & -\sin q_i & 0 \\ 0 & \sin q_i & \cos q_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \dot{M}_i(q_i) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\sin q_i & -\cos q_i & 0 \\ 0 & \cos q_i & -\sin q_i & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \dot{q}_i$$

Recall that $\dot{M} = M[\xi\times]$. Here:

$$\dot{M}_i(q_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q_i & -\sin q_i & 0 \\ 0 & \sin q_i & \cos q_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \dot{q}_i = M_i(q) S_i(q_i) \dot{q}_i$$

And we can identify ξ_i: $\omega_i = (\dot{q}_i, 0, 0)$ from the top-left 3×3 block, and $v_i = 0_3$ from the last column.

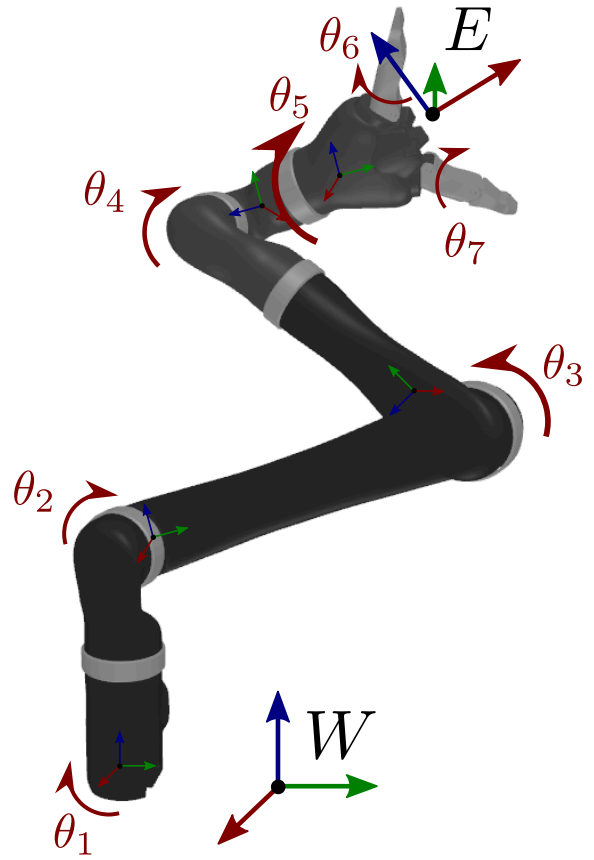
Jacobian matrix

Kinematic chain to an end-effector frame E:

$$M_{WE}(q) = M_1(q_1) \cdots M_n(q_n)$$

Velocity of the end-effector frame ${}_E\xi_{WE}$:

$$\begin{aligned}
[{}_E\xi_{WE}] &= M_{WE}^{-1}(q)\dot{M}_{WE}(q) \\
&= \sum_{i=1}^{n} M_n^{-1}(q_n) \cdots M_{i+1}^{-1}(q_{i+1}) S_i(q_i)\dot{q}_i M_{i+1}(q_{i+1}) \cdots M_n(q_n) \\
&= \begin{bmatrix} J_1(q) & \cdots J_n(q) \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J(q)v
\end{aligned}$$



Thus ${}_E\xi_{WE} := {}_E J_{WE}(q)\dot{q}$, where the matrix J is the **Jacobian** of the end-effector frame. (To be precise: the local Jacobian from the end-effector frame to the inertial frame.)

Inverse kinematics

Inverse geometry

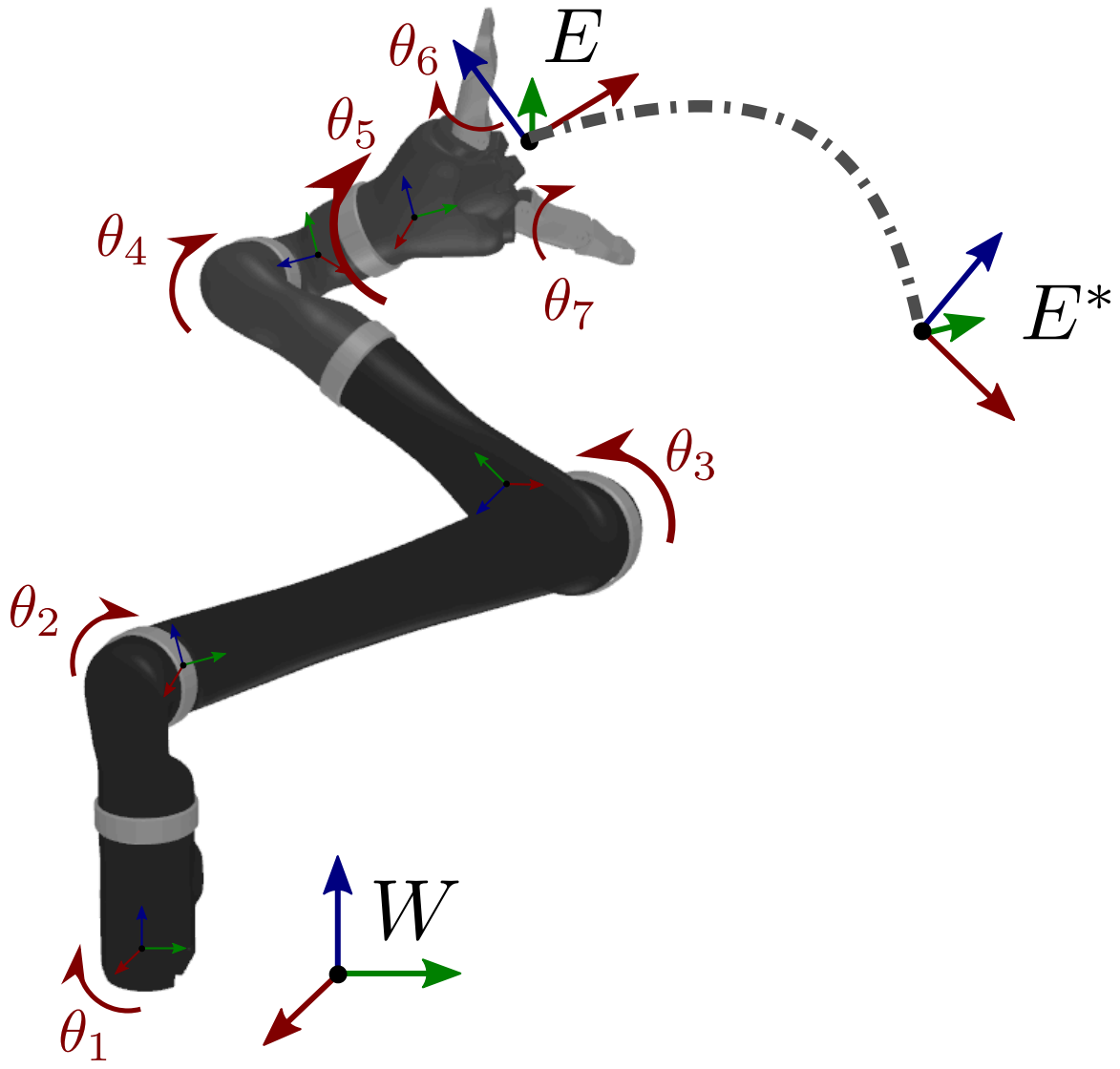
Goal: find $q \in \mathcal{C}$ s.t. $M_{WE}(q) = M_{WE^*}$.

We can cast this as a nonlinear optimization:

$$\underset{q \in \mathcal{C}}{\text{minimize}} \;\; \text{dist}(M_{WE}(q), M_{WE^*})$$

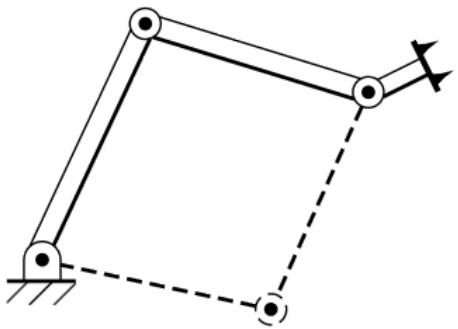
Using the logarithmic metric on $SE(3)$:

$$\underset{q \in \mathcal{C}}{\text{minimize}} \;\; \| \log(M_{WE}(q)^{-1} M_{WE^*}) \|$$

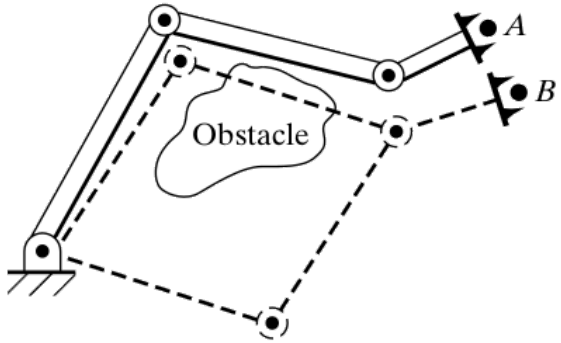


Redundancy

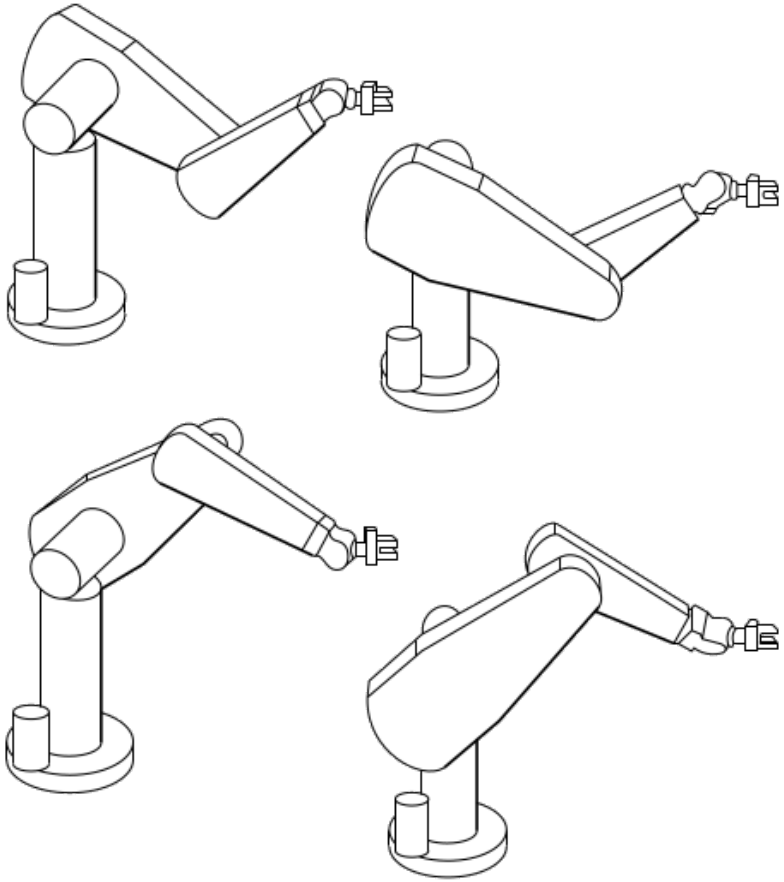
The inverse geometry problem may have several solutions:



one may be better than the other...



Redundancy



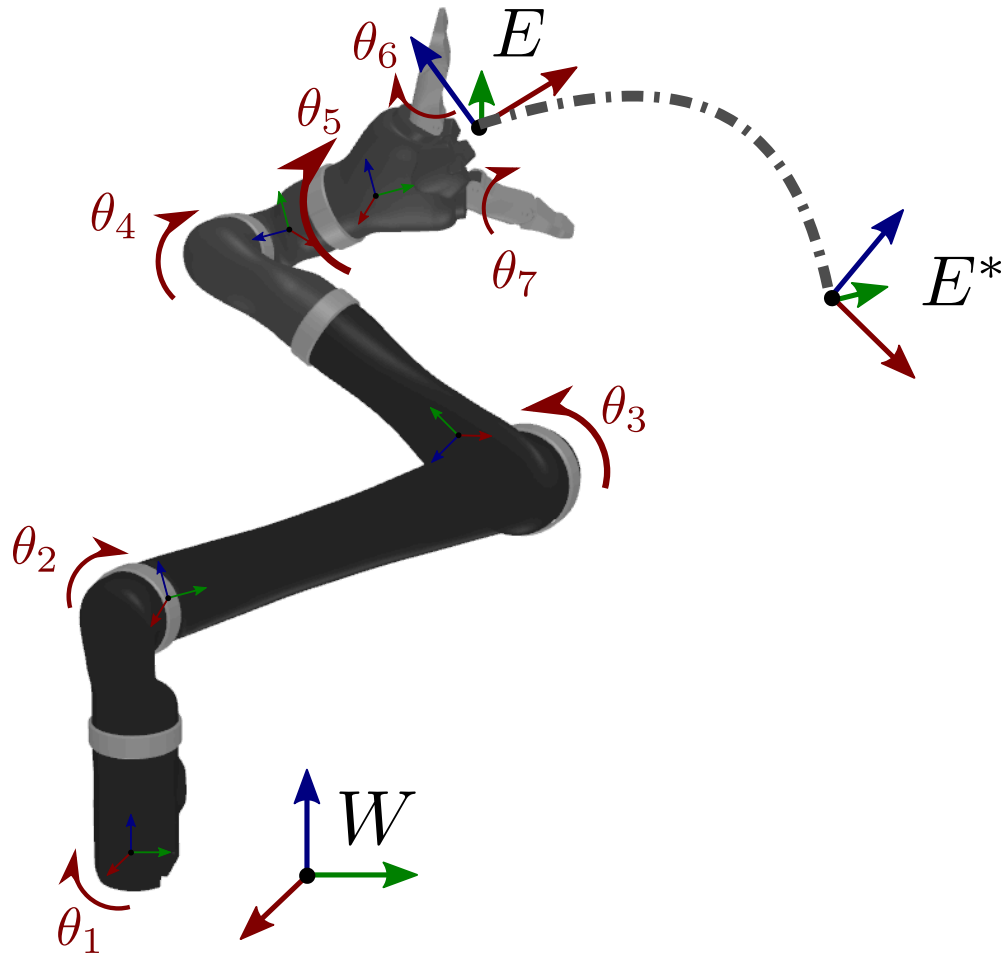
Inverse kinematics

- **Tracking error:** we take an error $e(q) \in \mathfrak{se}(3)$:

$$e(q) = M^* \ominus M(q) = \log(M(q)^{-1} M^*)$$

- Our goal is to bring the tracking error to zero.
- Derivative of the tracking error: $\dot{e}(q, v) \approx J(q)v$
- We now **choose** $\dot{e}(q) = -\lambda e(q)$, $\lambda > 0$.
- This results in a quadratic program:

$$\begin{aligned} &\underset{v \in \mathcal{T}_q \mathcal{C}}{\text{minimize}} \quad \|J(q)v + \lambda e(q)\|_2^2 \\ &\text{subject to} \quad v_{\min} \leq v \leq v_{\max} \end{aligned}$$



Multi-task inverse kinematics

There are two main paradigms to handle multiple tasks:

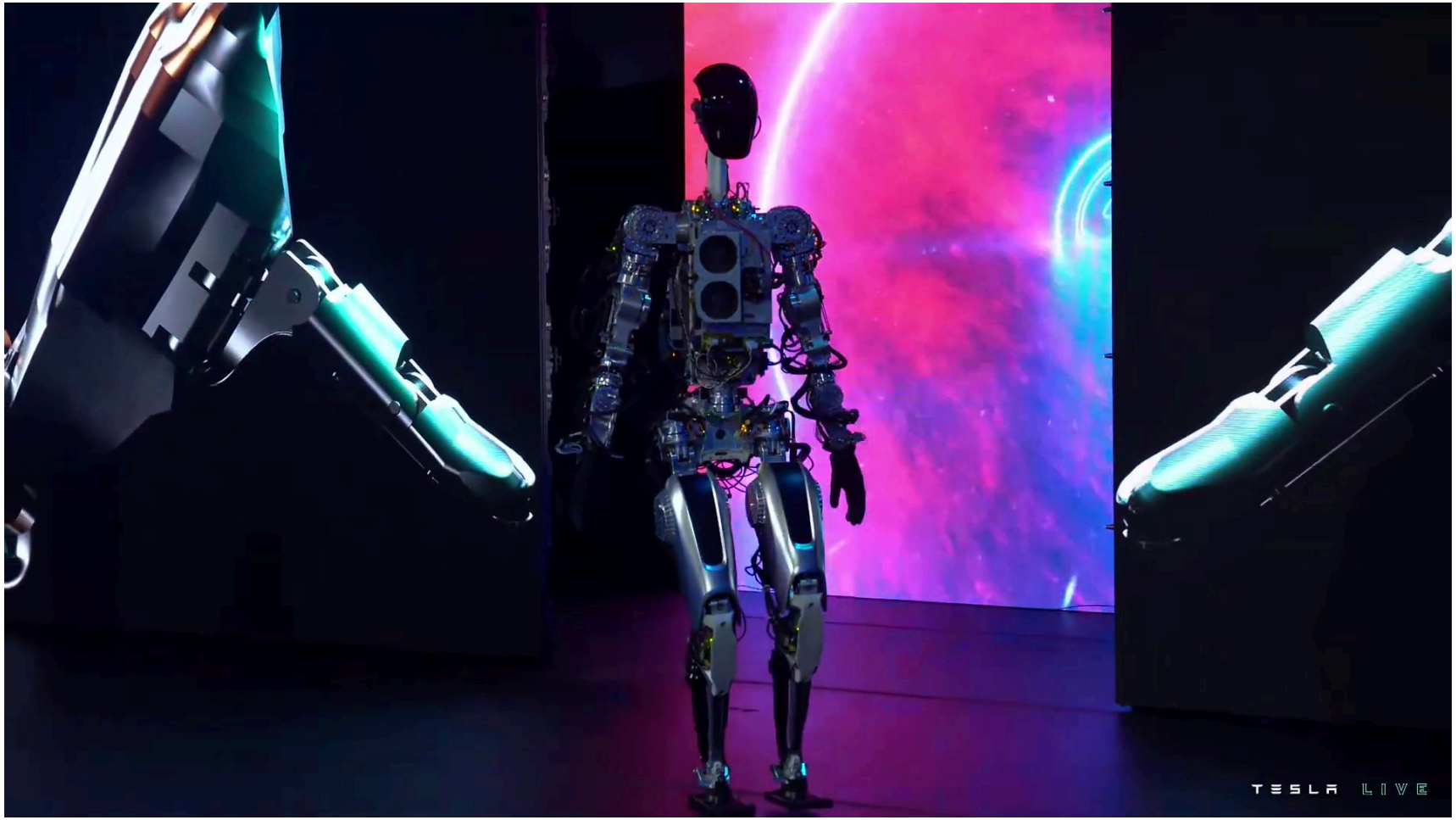
- **Weighted approach:** (*e.g.* followed in Pink)

$$\min_{v \in \mathcal{T}_q\mathcal{C}} \sum_{i \in \text{tasks}} w_i \| J_i(q)v + \lambda_i e_i(q) \|_2^2$$

$$\text{s.t. } v_{\min} \leq v \leq v_{\max}$$

- **Hierarchical approach:** (assignment for next week)
  - $\min_{v_1 \in \mathcal{T}_q\mathcal{C}} \| J_1(q)v + \lambda_1 e_1(q) \|_2^2$
  - $\min_{v_2 \in \mathcal{T}_q\mathcal{C}} \| J_2(q)v_2 + \lambda_2 e_2(q) \|_2^2$
    - s.t. $J_1(q)v_2 = J_1(q)v_1$
  - etc.

Tesla humanoid prototype (2022)



Slides continue in...

Quadratic programming also appears in Model predictive control for legged locomotion.