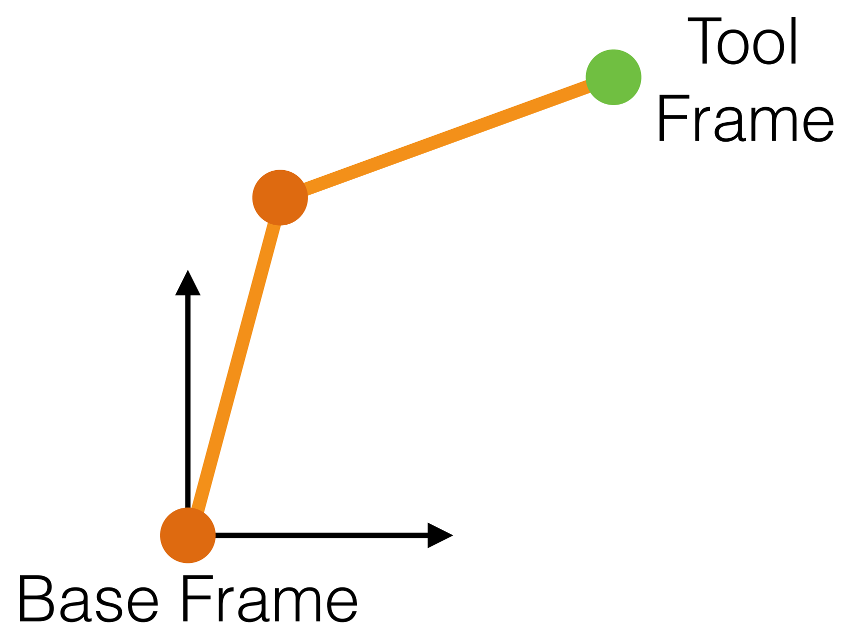
Robot Autonomy

Lecture 4: Kinematics

Oliver Kroemer

Motivation

- Interactions are often performed with endeffector/tool



- Want tool to perform a motion in Cartesian space
- Need to control the individual joints of the robot
- Map between the joint angles and Cartesian tool frame

Forward Kinematics

Joint Space
Robot Configuration

Joint Angles:

$$q_t = \begin{bmatrix} q_{t1} \\ q_{t2} \\ \vdots \\ q_{tn} \end{bmatrix}$$

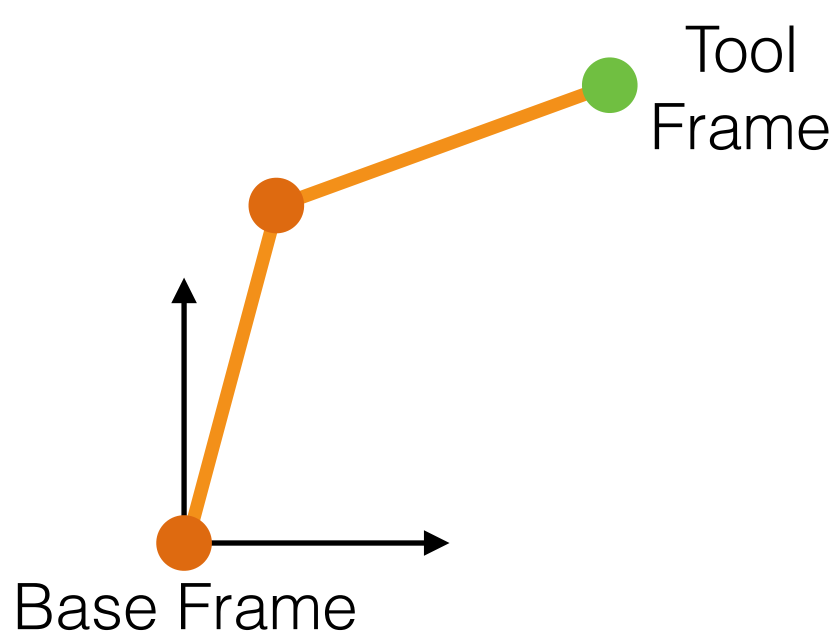
Forward Kinematics →

Cartesian Space
Tool Frame
relative to Base Frame

Orientation: R

Position: T

Other frames as well



Homogenous Transformations

- Consider a point in two coordinate frames

x^A = coordinates of a point in frame A
x^B = coordinates of a point in frame B

- Translation and rotation:

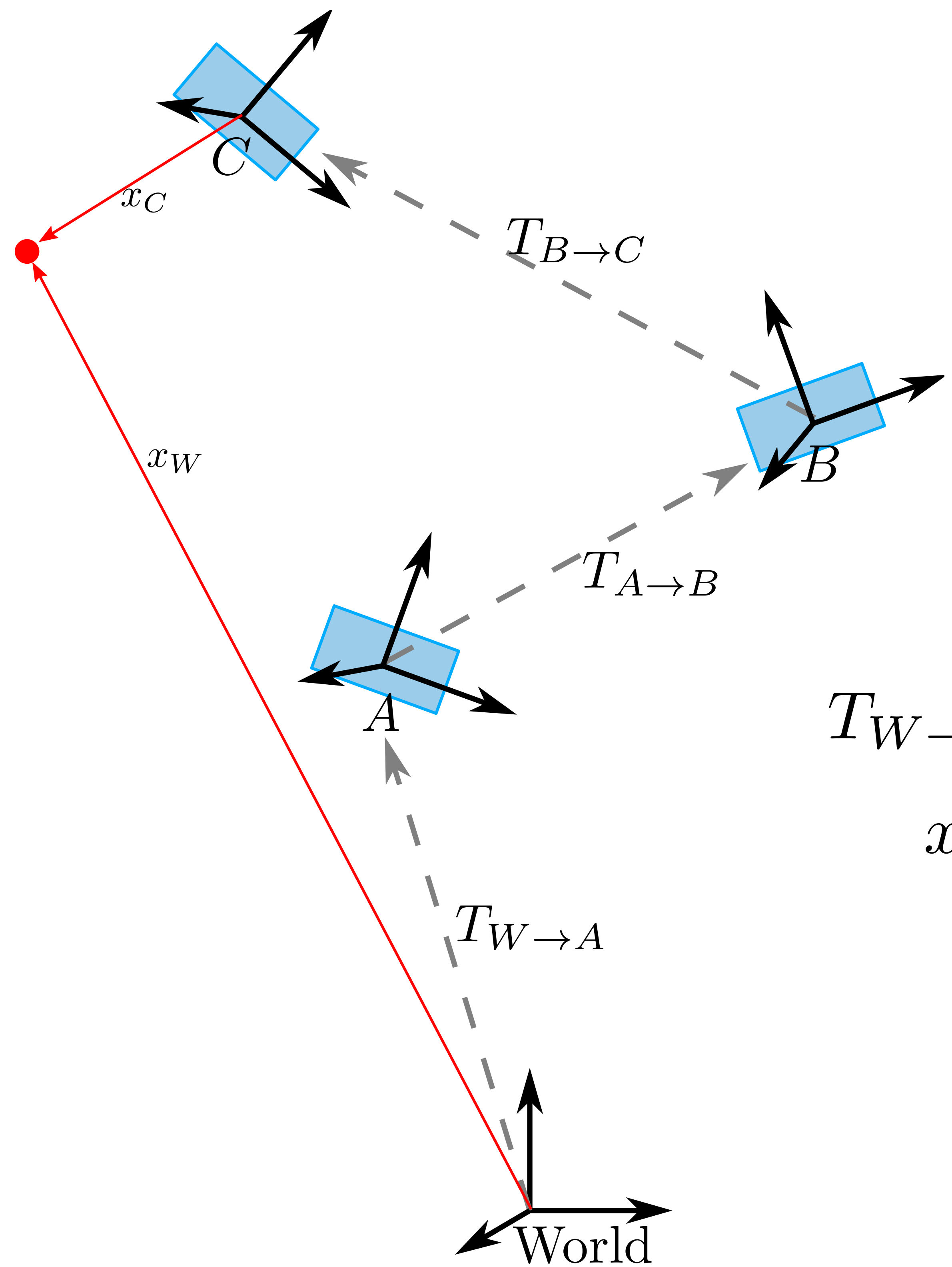
$$x^A = t + Rx^B$$

- Homogenous transform matrix $T \in \mathbb{R}^{4\times 4}$

$$T_{A\to B} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

$$x^A = T_{A\to B}\, x^B = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x^B \\ 1 \end{pmatrix} = \begin{pmatrix} Rx^B + t \\ 1 \end{pmatrix}$$

Combining Transforms

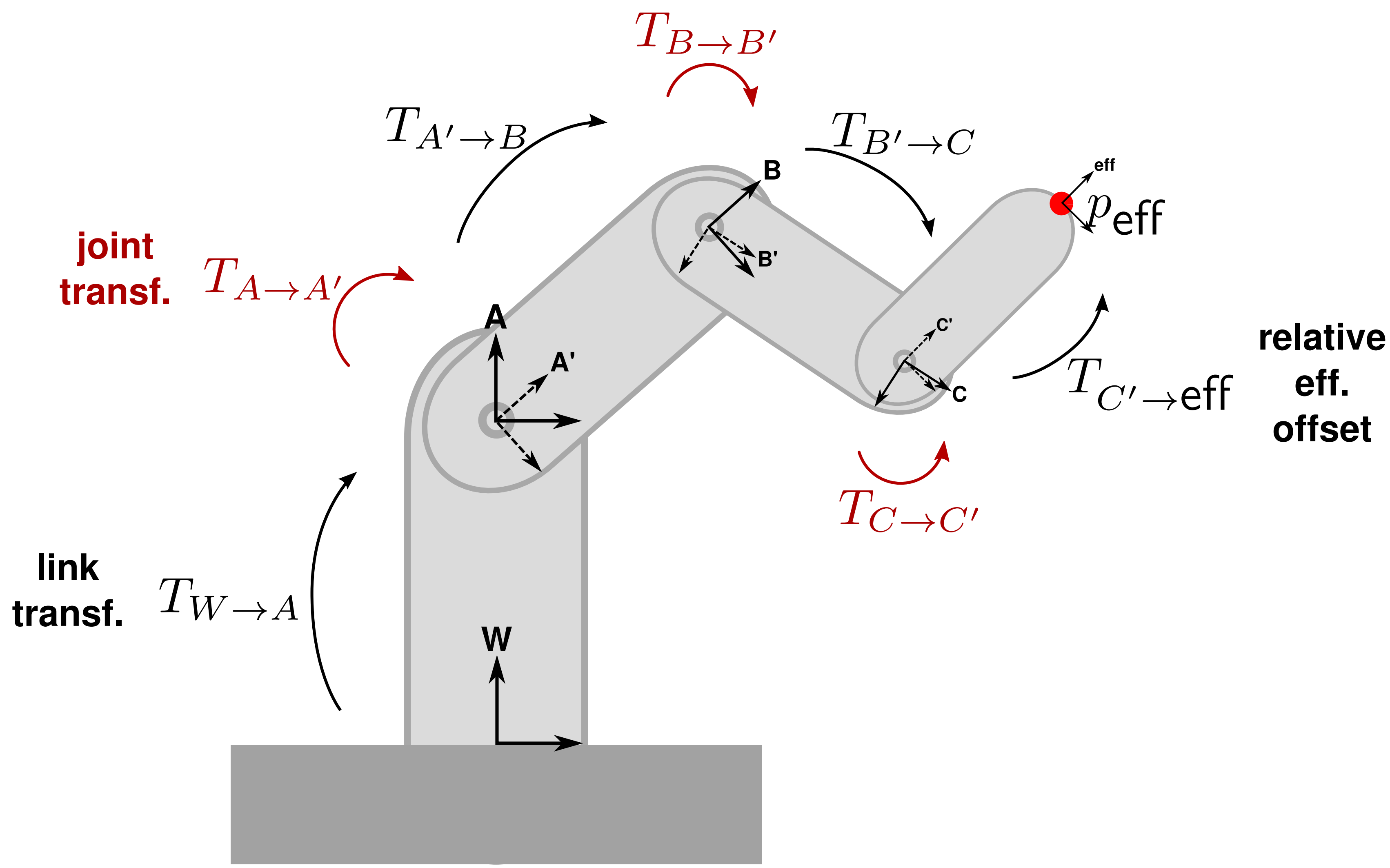


$$T_{W\to C} = T_{W\to A}\, T_{A\to B}\, T_{B\to C}$$

$$x^W = T_{W\to A}\, T_{A\to B}\, T_{B\to C}\, x^C$$

Forward Kinematics for Arm

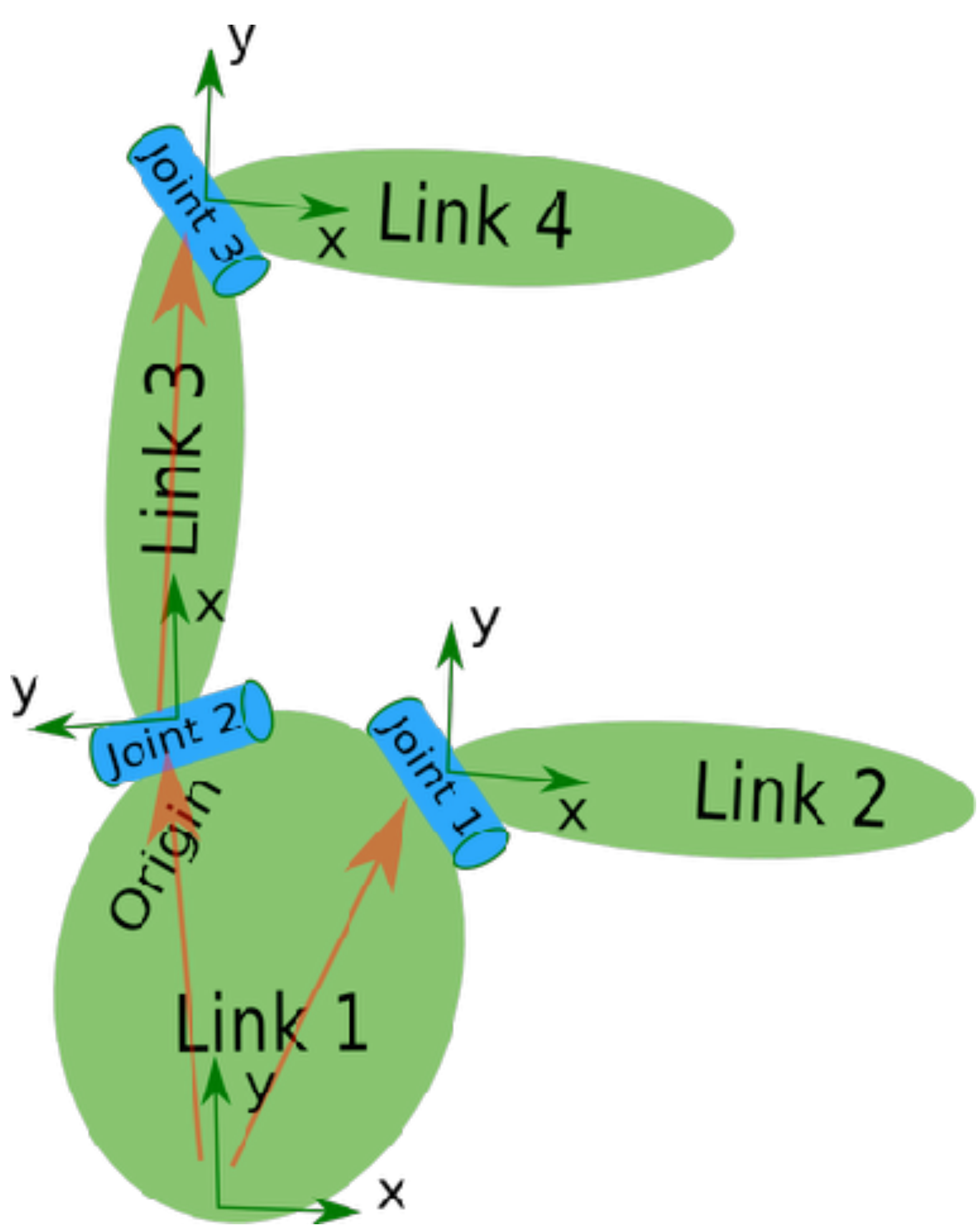
- Chain joint and link transforms from base to endeffector



$$T_{W\to\text{eff}}(q) = T_{W\to A}\, T_{A\to A'}(q)\, T_{A'\to B}\, T_{B\to B'}(q)\, T_{B'\to C}\, T_{C\to C'}(q)\, T_{C'\to\text{eff}}$$

URDF Files

- Unified Robot Description Format




```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
    <origin xyz="5 3 0" rpy="0 0 0" />
    <axis xyz="-0.9 0.15 0" />
  </joint>

  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
    <origin xyz="-2 5 0" rpy="0 0 1.57" />
    <axis xyz="-0.707 0.707 0" />
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
    <origin xyz="5 0 0" rpy="0 0 -1.57" />
    <axis xyz="0.707 -0.707 0" />
  </joint>
</robot>
```

URDF Files

- Specify link transforms

$$T = \text{Trans}(xyz)\text{Rot}_x(r)\text{Rot}_y(p)\text{Rot}_z(y)$$

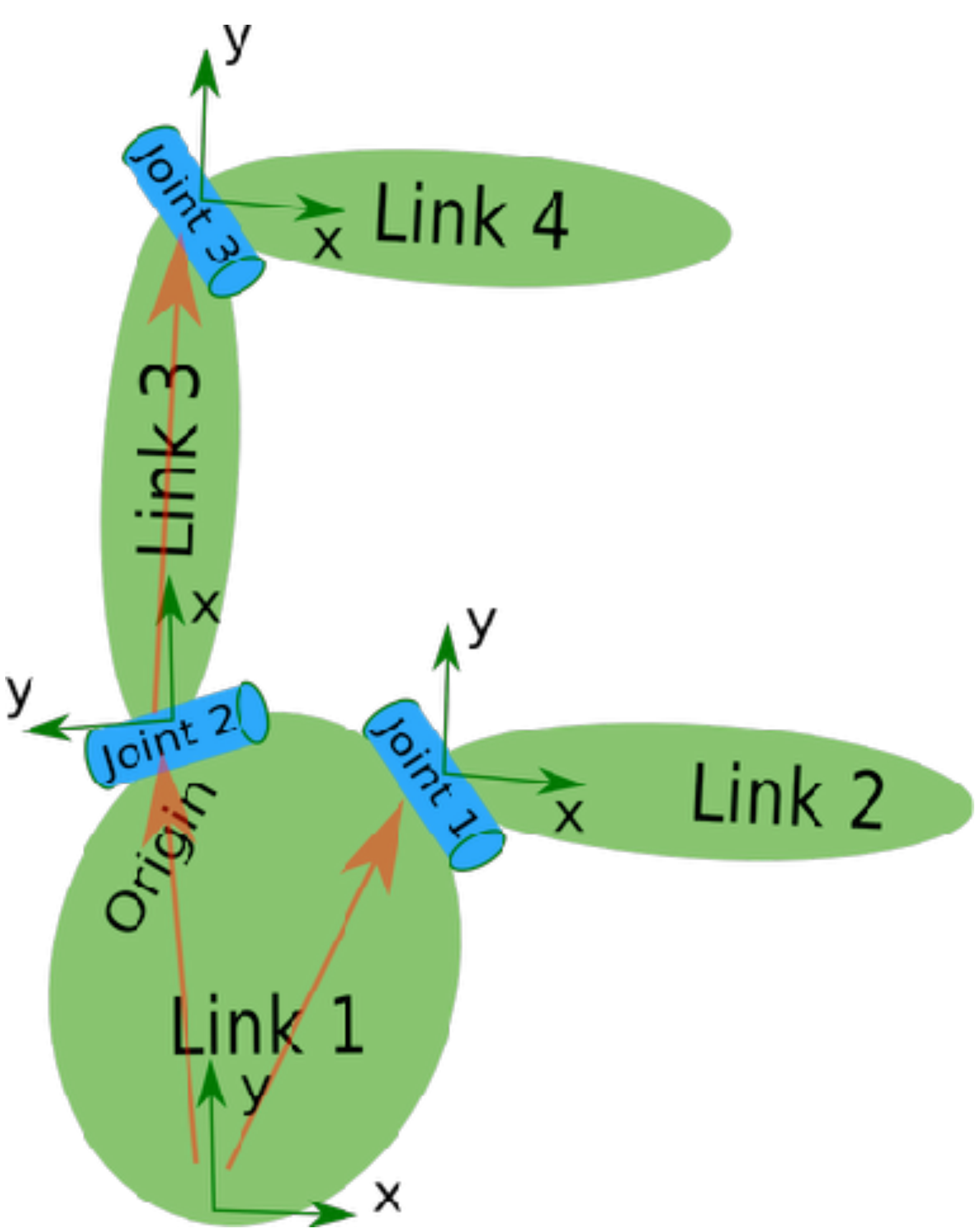
$$\begin{bmatrix} 0.0000 & -1.0000 & 0 & -2.0000 \\ 1.0000 & 0.0000 & 0 & 5.0000 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$\begin{bmatrix} 0.0000 & 1.0000 & 0 & 5.0000 \\ -1.0000 & 0.0000 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
    <origin xyz="5 3 0" rpy="0 0 0" />
    <axis xyz="-0.9 0.15 0" />
  </joint>

  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
    <origin xyz="-2 5 0" rpy="0 0 1.57" />
    <axis xyz="-0.707 0.707 0" />
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
    <origin xyz="5 0 0" rpy="0 0 -1.57" />
    <axis xyz="0.707 -0.707 0" />
  </joint>
</robot>
```

URDF Files

- Axis define axis direction w.r.t. local frame



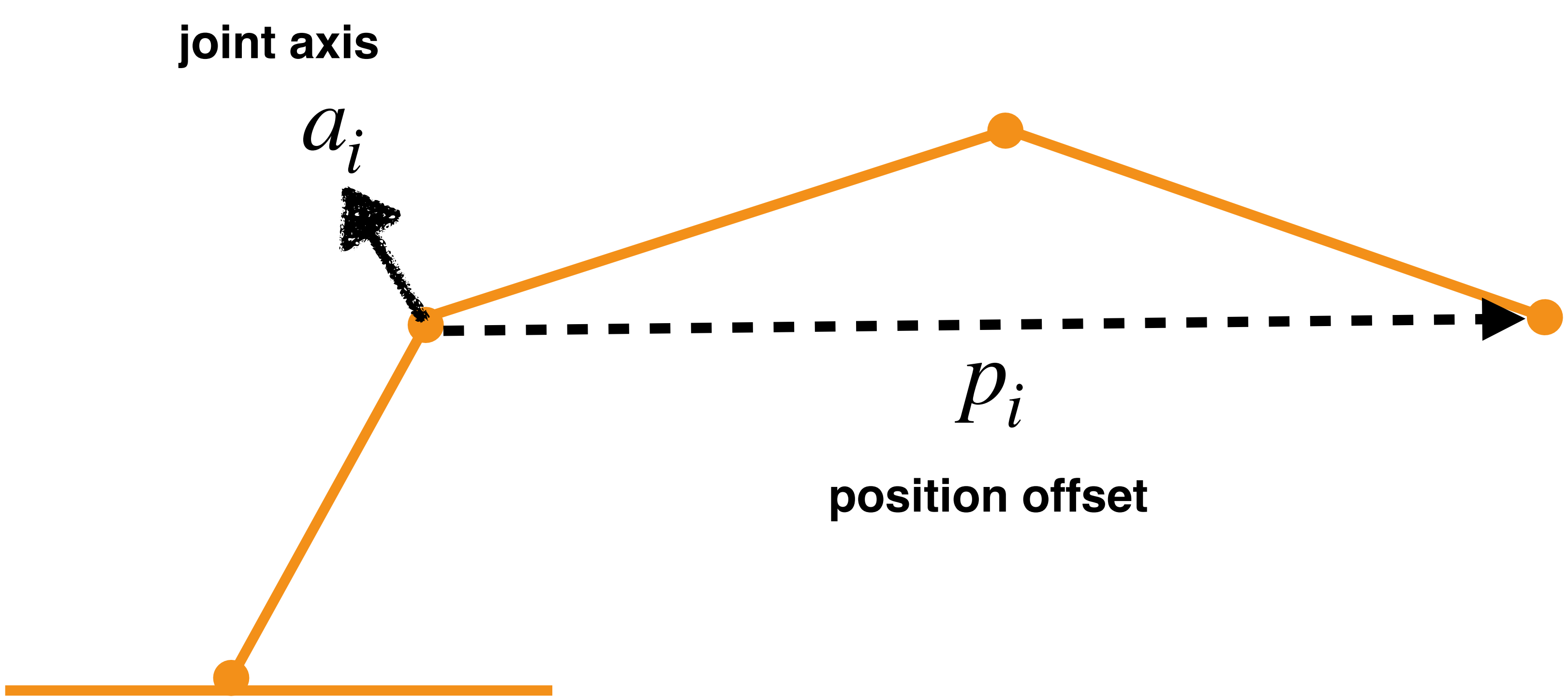

```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
    <origin xyz="5 3 0" rpy="0 0 0" />
    <axis xyz="-0.9 0.15 0" />
  </joint>

  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
    <origin xyz="-2 5 0" rpy="0 0 1.57" />
    <axis xyz="-0.707 0.707 0" />
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
    <origin xyz="5 0 0" rpy="0 0 -1.57" />
    <axis xyz="0.707 -0.707 0" />
  </joint>
</robot>
```

Jacobian

- What about velocities?

$$\dot{x} = J(q)\dot{q} \qquad \frac{\partial x}{\partial q} = \frac{\partial f(q)}{\partial q} = J(q)$$

- Compute Jacobian's i-th column for the current pose



Prismatic Joint

$$\begin{bmatrix} a_i \\ 0 \end{bmatrix}$$

Rotational Joint

$$\begin{bmatrix} a_i \times p_i \\ a_i \end{bmatrix}$$

Inverse Kinematics

Joint Space
Robot Configuration

Joint Angles:

$$q_t = \begin{bmatrix} q_{t1} \\ q_{t2} \\ \vdots \\ q_{tn} \end{bmatrix}$$

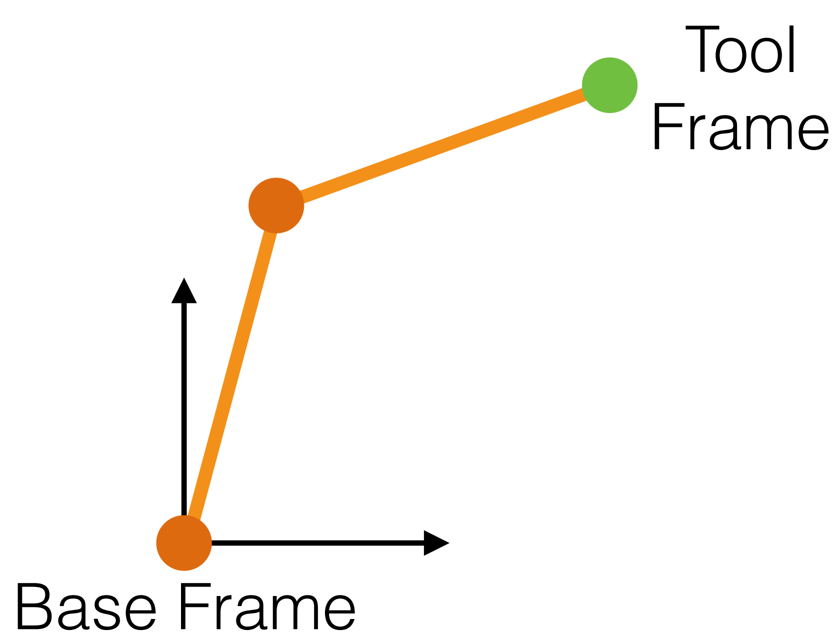
Forward Kinematics →

← Inverse Kinematics

Cartesian Space
Tool Frame
relative to Base Frame

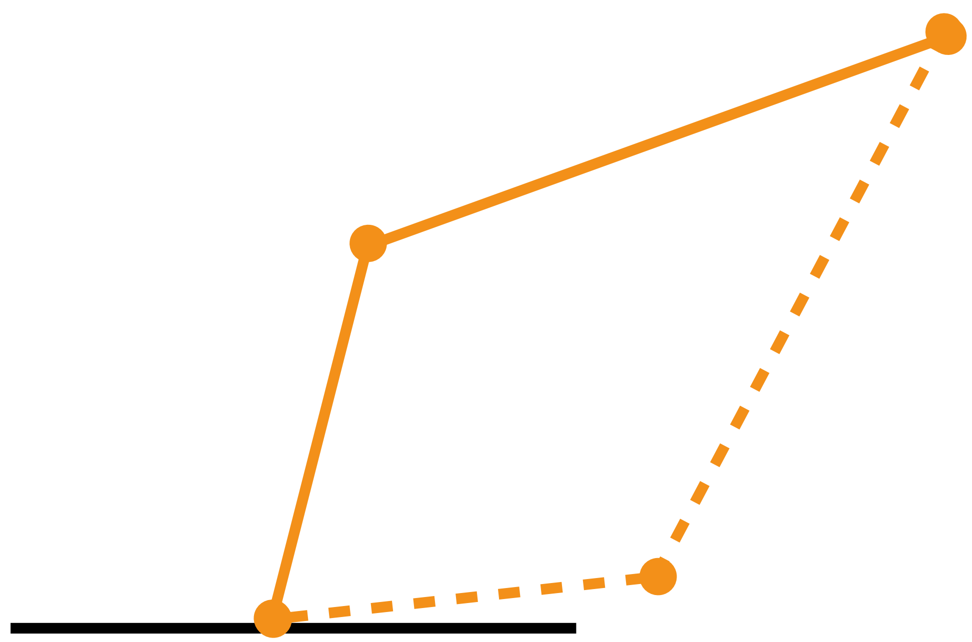
Orientation: R

Position: T



Inverse Kinematics

- Kinematics may be invertible in some cases $q = f^{-1}(x)$

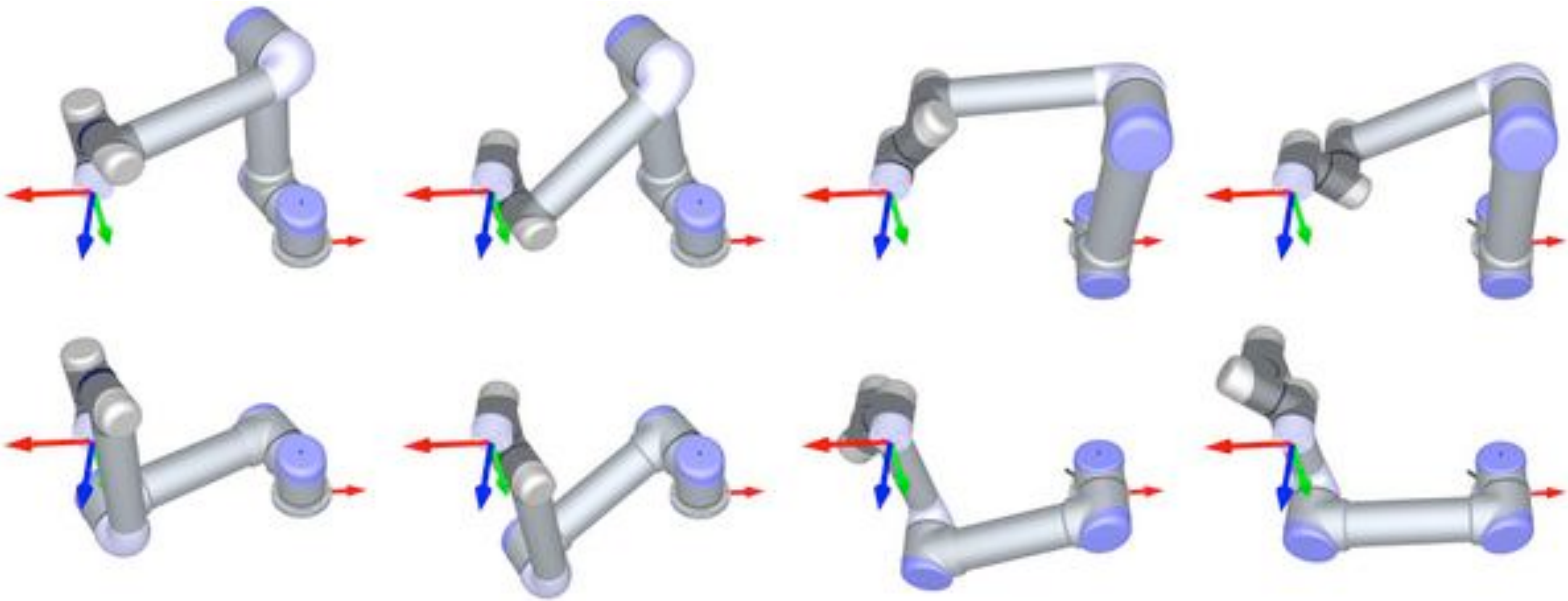


- Often have multiple solutions, or may have no solutions
- Sets of analytical solutions can be computed,
  e.g., can compute the two solutions to above problem using the law of cosines

$$\cos(A) = \frac{b^2 + c^2 - a^2}{2bc}$$

Inverse Kinematics

- The 6DoF UR5 has eight solutions

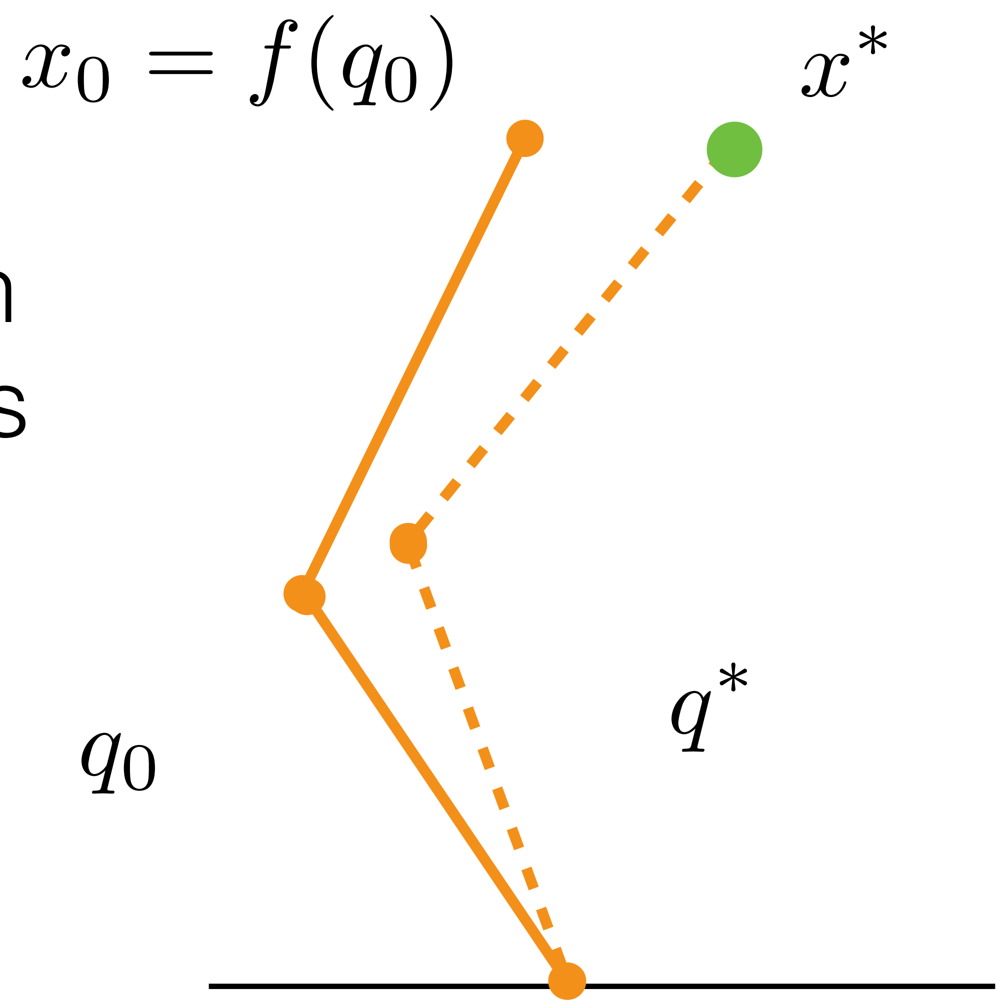


- Analytical solution to inverse kinematics is often difficult to obtain for arbitrary kinematics
- Approach inverse kinematics as a local optimization

Iterative Inverse Kinematics

- Want to find small change in joint configuration Δq to move end effector by a small amount Δx

Define Jacobian at current angles
$J = J(q_0)$



$$\Delta x = x^* - x_0$$

$$\Delta q = q - q_0$$

Jacobian Transpose Approach

Jacobian Transpose Approach

- Define a quadratic cost on end effector error:

$$E(q) = \frac{1}{2}\Delta x^T \Delta x = \frac{1}{2}(x^* - f(q))^T(x^* - f(q))$$

- Take a step along error gradient

$$\Delta q = -\alpha \left[\frac{\partial E(q_0)}{\partial q_0}\right]^T$$

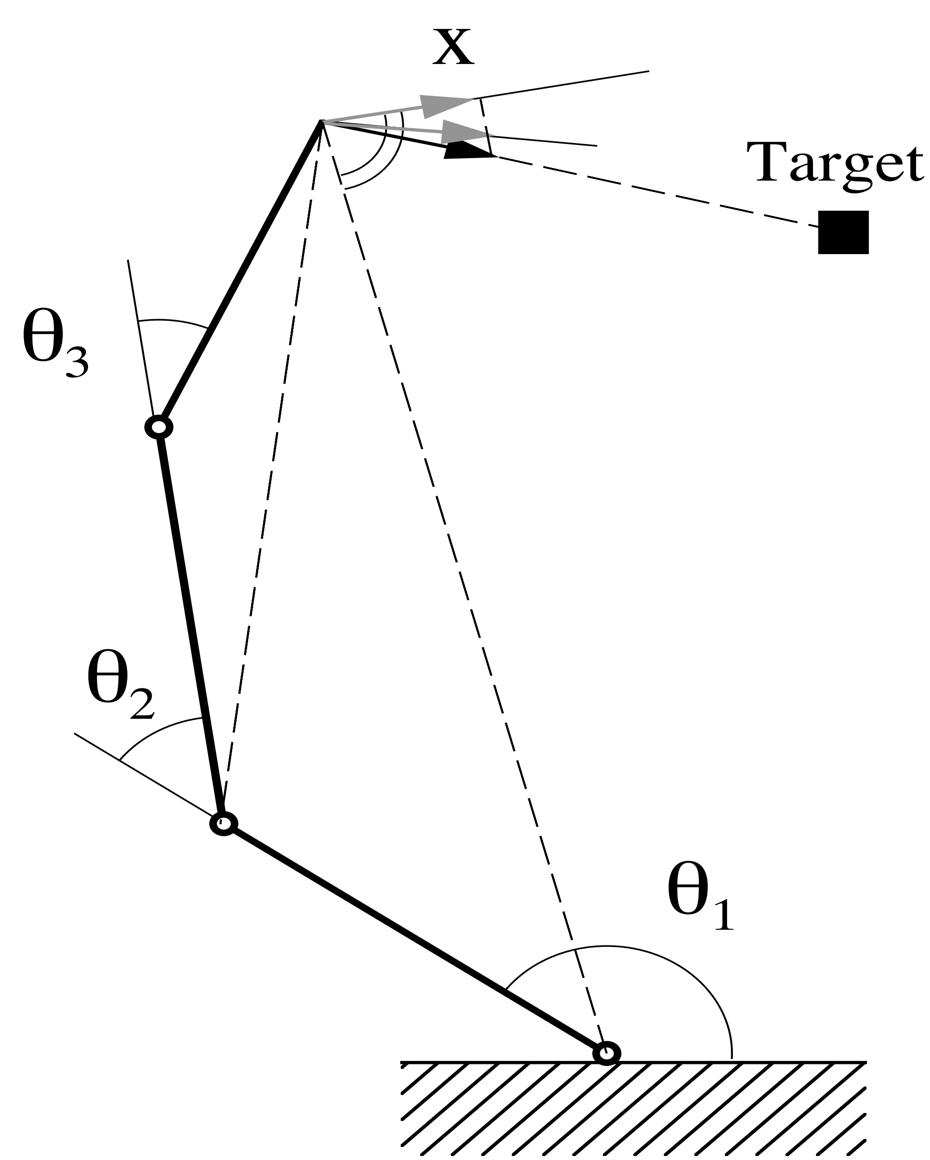
$$\Delta q = \alpha[(x^* - f(q_0))^T \frac{\partial f(q_0)}{\partial q_0}]^T$$

$$\Delta q = \alpha[\Delta x^T J]^T$$

$$\Delta q = \alpha J^T \Delta x$$

Jacobian Transpose

Projects vector Δx onto those joints that can reduce it the most



Pseudo-Inverse Approach

Jacobian Pseudo-Inverse Approach

- Define a quadratic cost on joint motion and a constraint

$$\min \Delta q^T \Delta q \text{ s.t. } \Delta x = J \Delta q$$

- Applying Lagrangian optimization gives

$$E(q) = \frac{1}{2} \Delta q^T \Delta q + \lambda^T (\Delta x - J \Delta q)$$

$$\frac{\partial E(q)}{\partial \Delta q} = 0 \Rightarrow \Delta q^T - \lambda^T J = 0 \Rightarrow \Delta q = J^T \lambda$$

$$\Delta q = J^T \lambda \Rightarrow J \Delta q = J J^T \lambda \Rightarrow \lambda = (J J^T)^{-1} J \Delta q$$

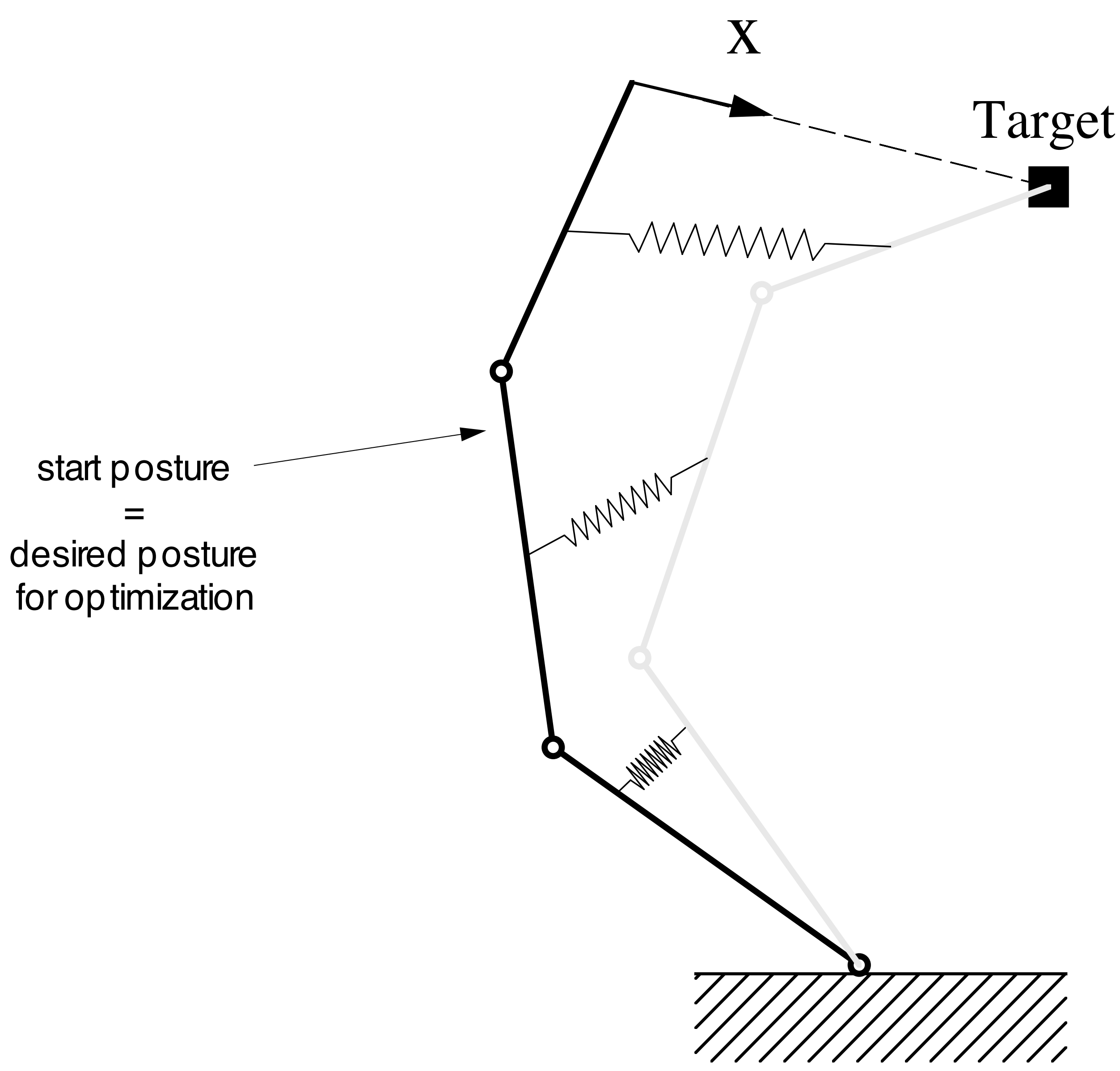
$$\lambda = (J J^T)^{-1} \Delta x$$

$$\Delta q = J^T \lambda = J^T (J J^T)^{-1} \Delta x$$

$$\Delta q = J^{\#} \Delta x \text{ where } J^{\#} = J^T (J J^T)^{-1}$$

Physical Interpretation

Quadratic cost can be interpreted as springs to start posture



Picture:Stefan Schaal's slides

Damped Least Squares Approach

Damped Least Squares Approach

- Define a quadratic cost as

$$E(q) = \frac{1}{2}\|f(q) - x^*\|_C^2 + \frac{1}{2}\|q - q_0\|_W^2$$

$$f(q) \approx f(q_0) + J(q - q_0) = x_0 + J(q - q_0)$$

- Larger C value:
  larger cost for error in that Cartesian component
- Larger W value:
  larger cost for change in that joint angle

Damped Least Squares Approach

- Define a quadratic cost as

$$E(q) = \frac{1}{2}\left[x_0 - x^* + J(q - q_0)\right]^T C \left[x_0 - x^* + J(q - q_0)\right] + \frac{1}{2}\left[q - q_0\right]^T W \left[q - q_0\right]$$

- Optimizing for the joint angle shift gives

$$0 = \left[x_0 - x^* + J(q - q_0)\right]^T CJ + \left[q - q_0\right]^T W$$

$$0 = (x_0 - x^*)^T CJ + (q - q_0)^T (J^T CJ + W)$$

$$J^T C(x^* - x_0) = (J^T CJ + W)(q - q_0)$$

$$(J^T CJ + W)^{-1} J^T C \Delta x = \Delta q$$

$$W^{-1} J^T (JW^{-1} J^T + C^{-1})^{-1} \Delta x = \Delta q$$

$$\Delta q = J^{\#} \Delta x \text{ where } J^{\#} = W^{-1} J^T (JW^{-1} J^T + C^{-1})^{-1}$$

Damped Least Squares Approach

$$\Delta q = J^{\#}\Delta x \text{ where } J^{\#} = W^{-1}J^T(JW^{-1}J^T + C^{-1})^{-1}$$

Special case: $C = \infty \text{ and } W = I$

$$J^{\#} = I^{-1}J^T(JI^{-1}J^T + \infty^{-1})^{-1} = J^T(JJ^T)^{-1}$$

Iterative Inverse Kinematics

- Equations are typically used to iteratively compute steps

$$q_{t+1} = q_t + J^{\#}(x^*_{t+1} - f(q_t))$$

  - Compute Jacobian for current configuration q_t
  - Assumes that desired pose x^*_{t+1} moves slowly over time
  - Local linearisation not valid for larger steps

- Can perform multiple steps in background to estimate q^*

Questions?