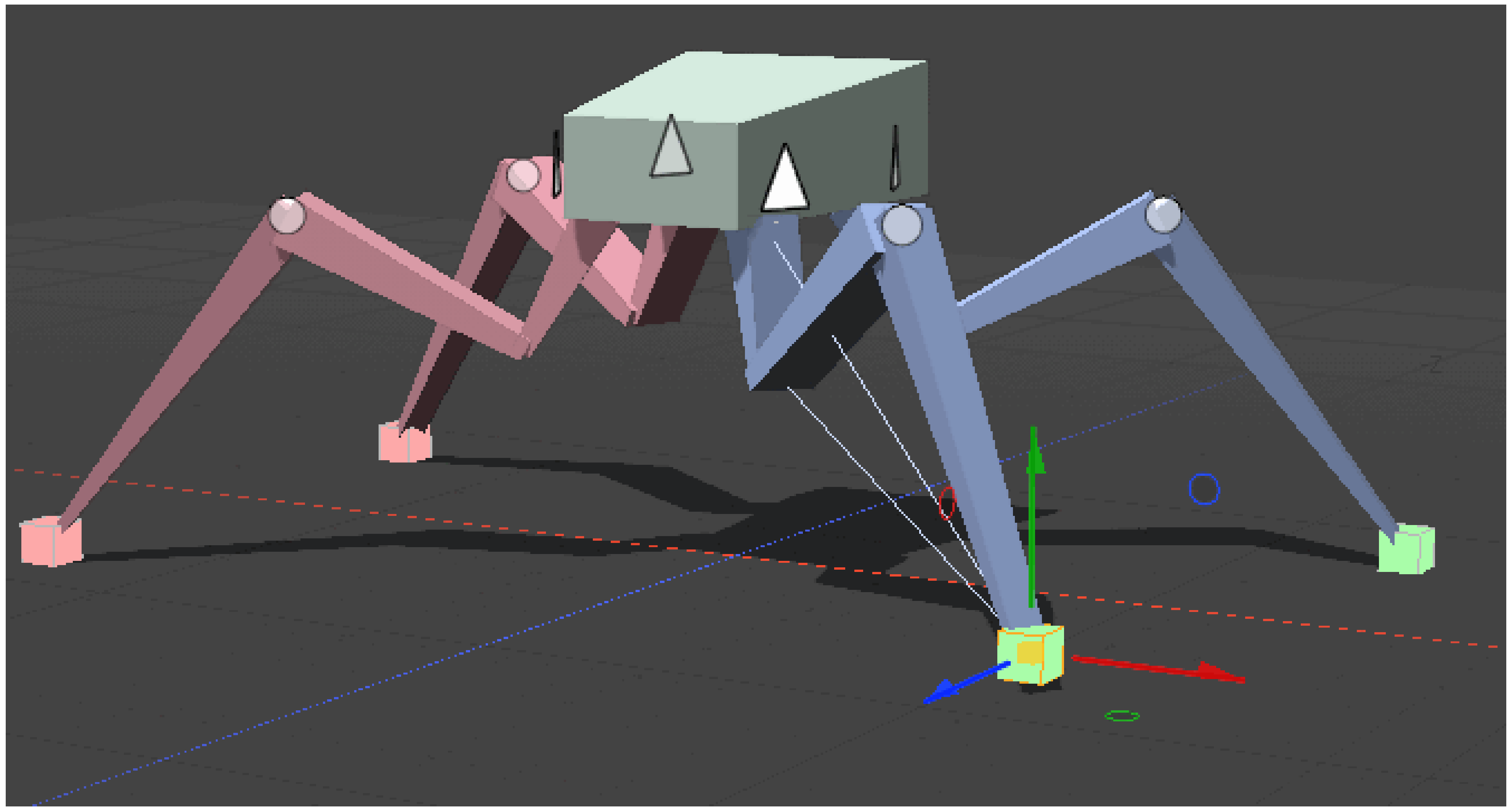
Introduction to Inverse Kinematics

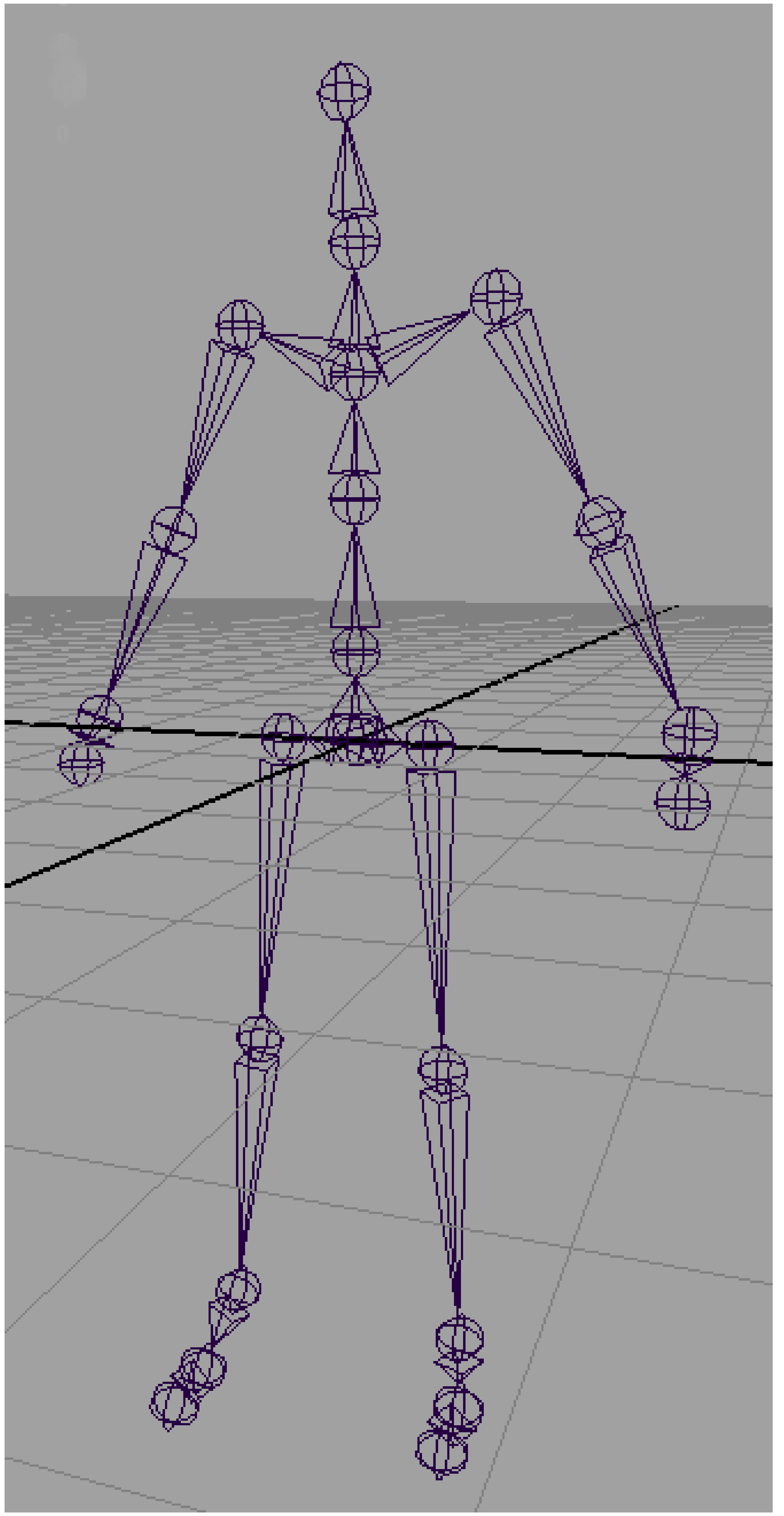


Learning Objectives

- **Learn how to formulate inverse kinematics as a numerical optimization problem**
- **Understand two of the most common techniques for solving IK, the J^t (Jacobian transpose) and the J^+ (Jacobian pseudo-inverse) methods**
- **Learn how to implement and debug gradient-based optimization methods**

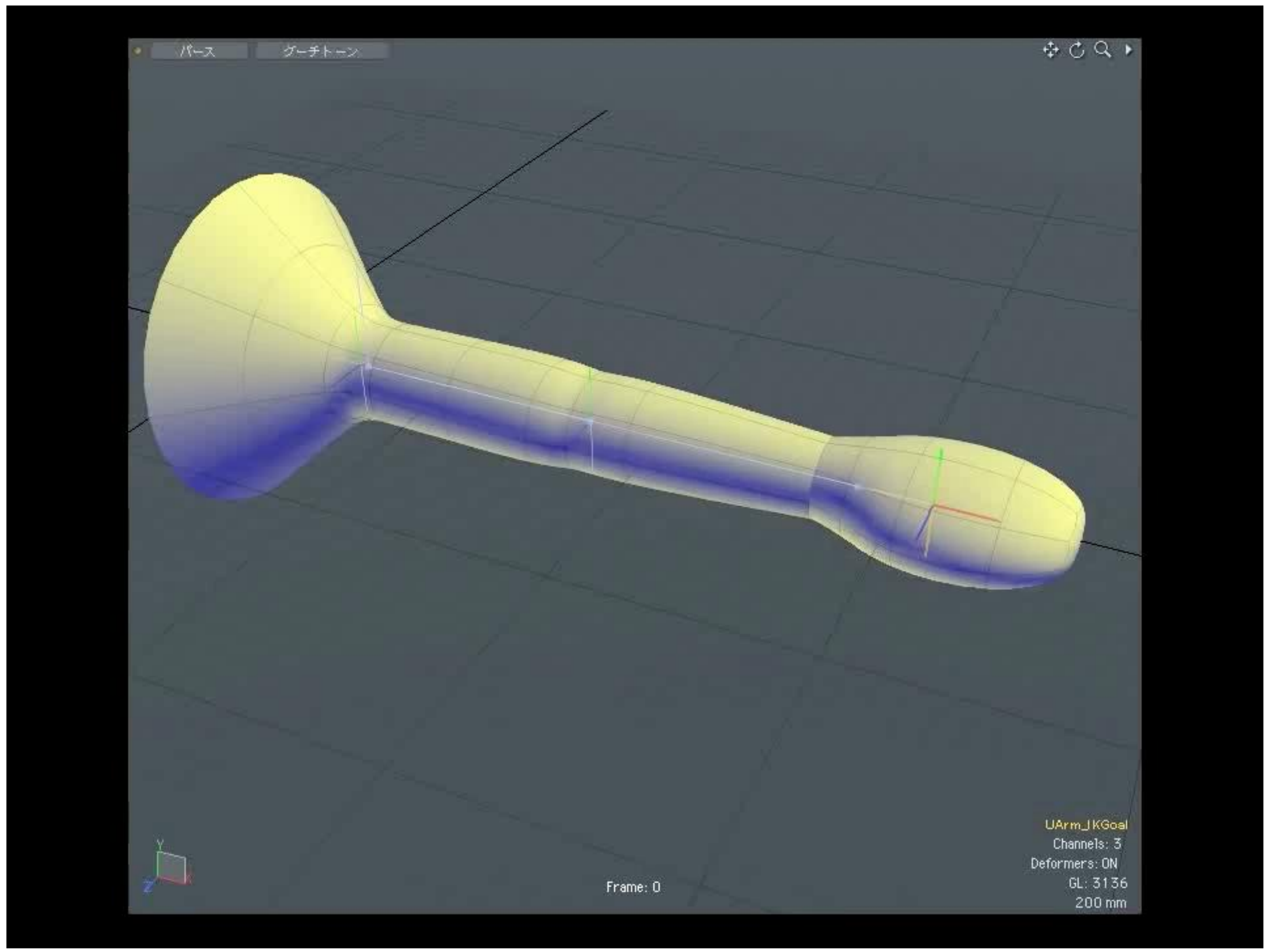
Forward Kinematics

- **Given joint angles as input, compute skeleton pose**
  - **Bones/links/body parts defined by their geometry (e.g. shape) and a coordinate frame**
  - **Joints define relative rotation between child and parent coordinate frames**
    - **They must store type, position on parent/child frames, rotation axis, etc.**



Inverse Kinematics (IK)

- **Given goal(s) for "end effector" compute joint angles**



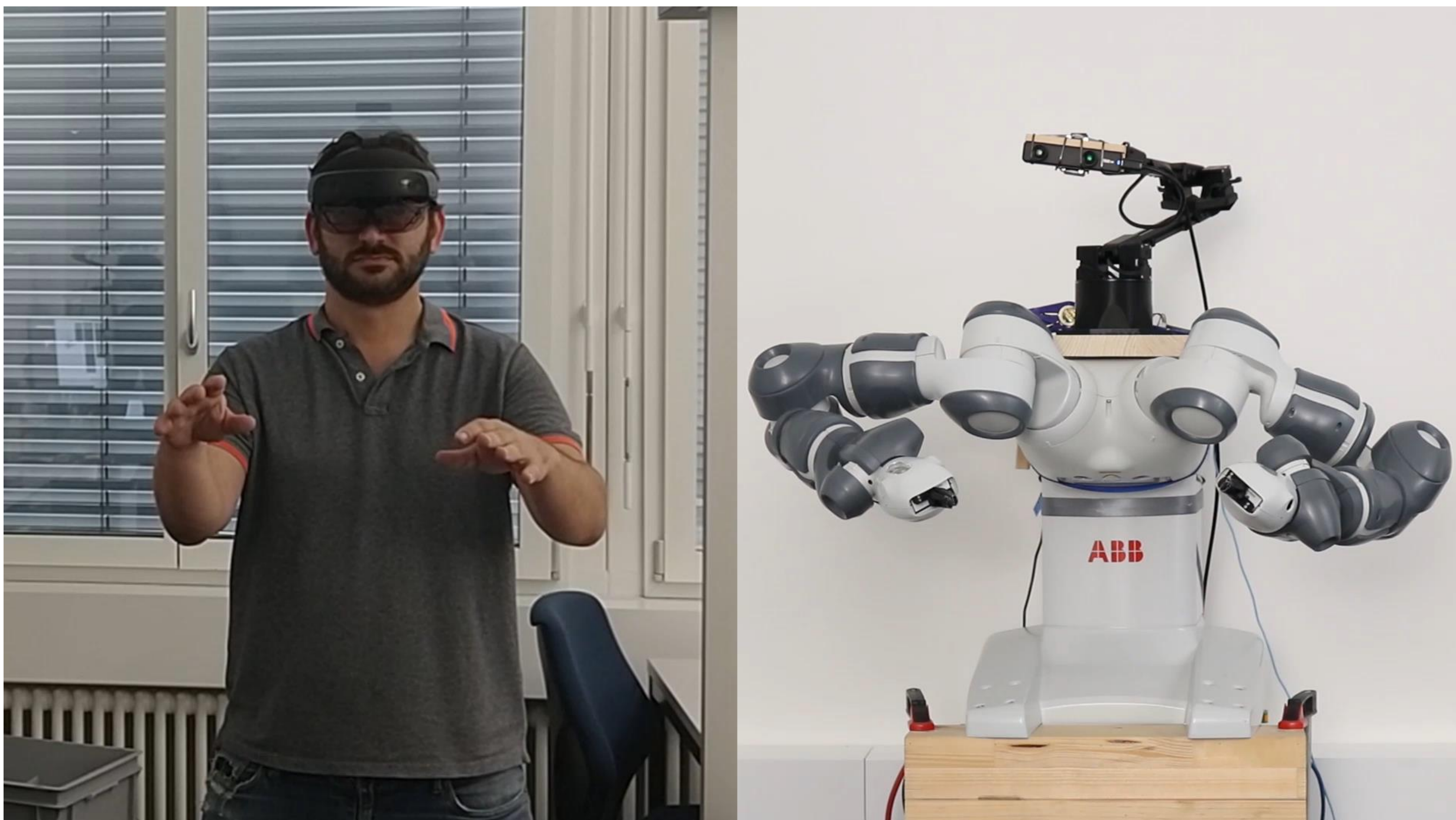
- **Many, many algorithms: analytic formulations for specific cases, cyclic coordinate descent, J^T/J^+ methods, etc**

Typical applications of inverse kinematics

- Intuitive way of creating motions via high-level goals
- Full body motion capture
- Motion retargeting and teleoperation

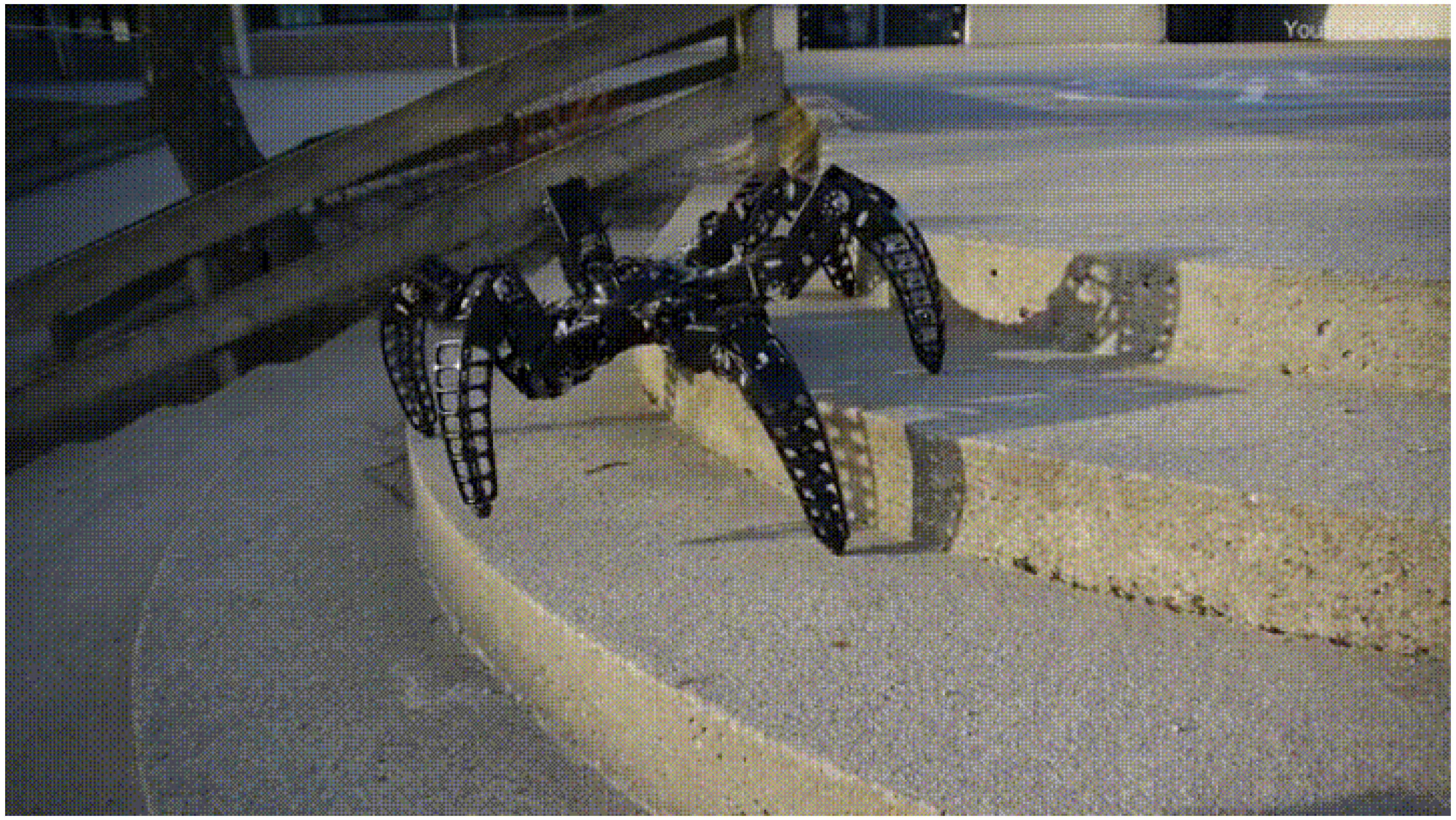
Typical applications of inverse kinematics

- Intuitive way of creating motions via high-level goals
- Full body motion capture
- Motion retargeting and teleoperation



Typical applications of inverse kinematics

- Intuitive way of creating motions via high-level goals
- Full body motion capture
- Motion retargeting and teleoperation
- Starting point for more sophisticated control methods

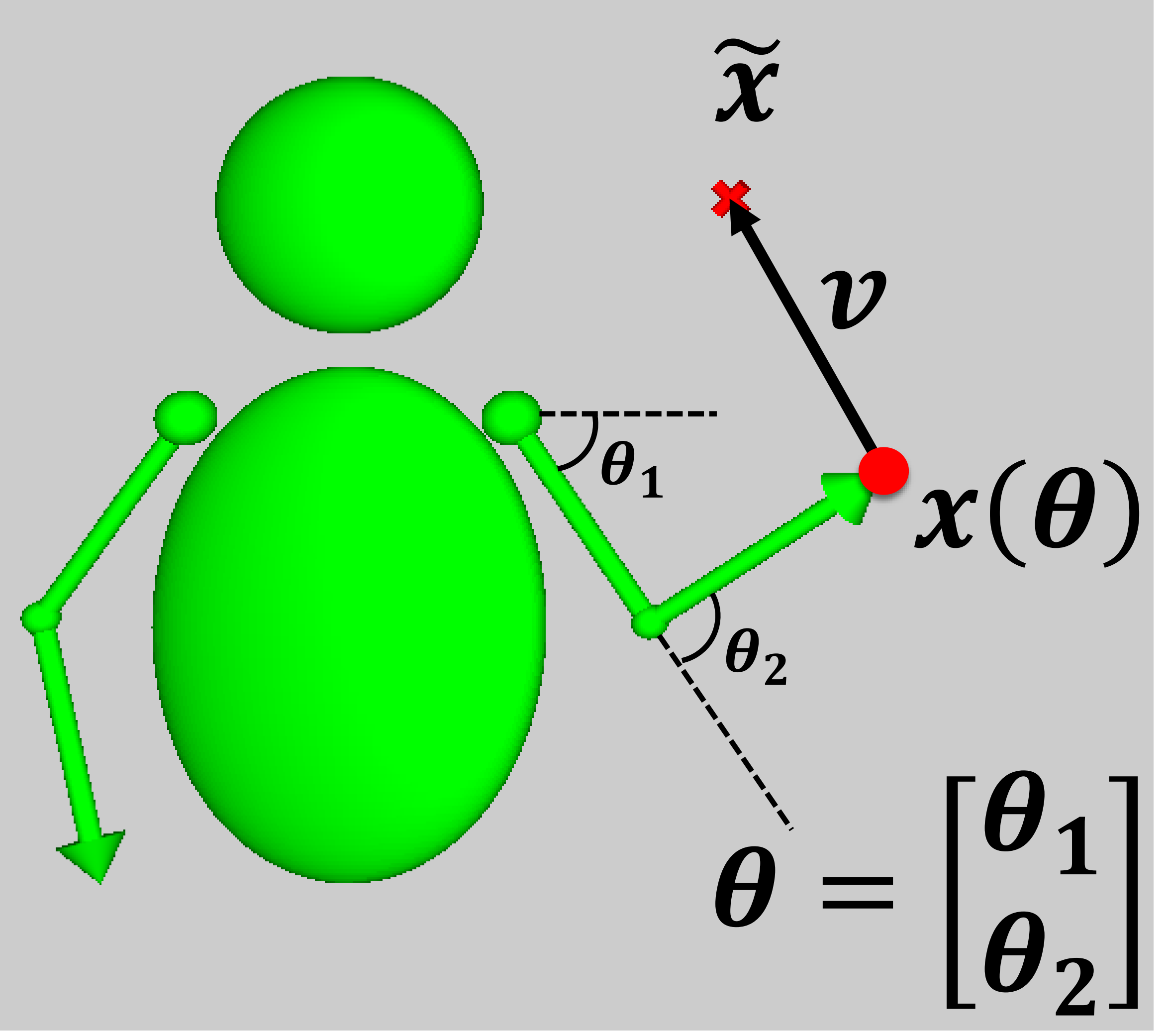


Inverse Kinematics

A numerical optimization approach

Basic idea

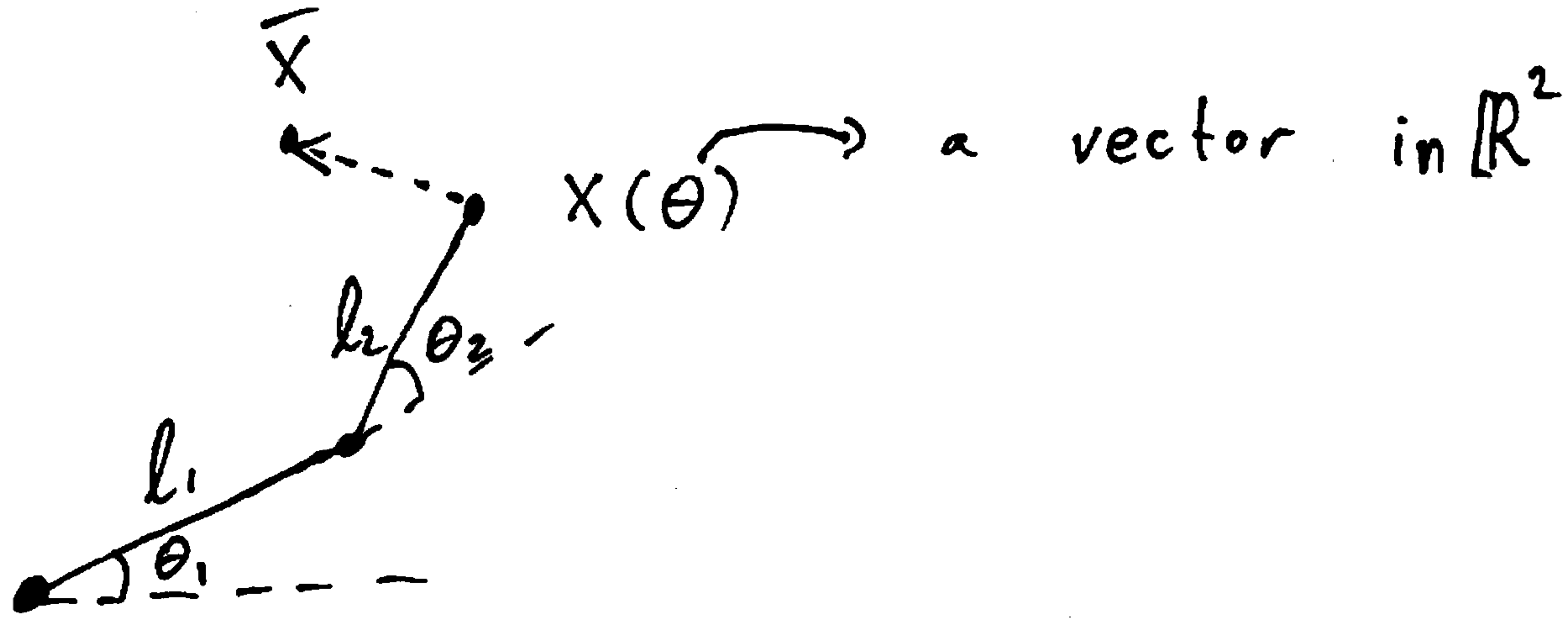
- Write down distance between final point and "target" and set up an objective/energy/error/loss function



$$E(\theta) = \frac{1}{2}(x(\theta) - \widetilde{x})^T(x(\theta) - \widetilde{x})$$

- Many possible extensions
  - Joint limits, pose regularizers, objectives on end effector orientation, etc.
- compute derivatives with respect to joint angles
- apply gradient-based methods to find minimum of objective

Forward kinematics: computing $x(\theta)$



$$x(\theta) = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) \end{bmatrix}$$

Inverse kinematics: derivatives

$$x(\theta) = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2 \end{bmatrix}$$

The energy function to minimize:

$$E(\theta) = \frac{1}{2} (x(\theta) - \bar{x})^T \underbrace{(x(\theta) - \bar{x})}_{V}$$

$$\nabla E = \frac{dE}{d\theta} = \underbrace{\frac{dx}{d\theta}^T}_{m \times n} \cdot \underbrace{\frac{dE}{dx}}_{n \times 1} = \frac{dx}{d\theta}^T \cdot V = J^T V$$

$$\frac{dx}{d\theta} = \begin{bmatrix} \frac{dx_x}{d\theta_1} & \frac{dx_x}{d\theta_2} \\ \frac{dx_y}{d\theta_1} & \frac{dx_y}{d\theta_2} \end{bmatrix} = \begin{bmatrix} \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} \end{bmatrix} = J$$

Inverse kinematics: derivatives

$$x(\theta) = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2 \end{bmatrix}$$

The energy function to minimize:

$$E(\theta) = \frac{1}{2}\,(x(\theta)-\bar{x})^T \underbrace{(x(\theta)-\bar{x})}_{V}$$

$$\nabla E = \frac{dE}{d\theta} = \underbrace{\frac{dx^T}{d\theta}}_{m \times n} \cdot \underbrace{\frac{dE}{dx}}_{n \times 1} = \frac{dx^T}{d\theta} \cdot V = J^T V$$

$$\nabla^2_\theta E = J^T J + \frac{dJ}{d\theta} \cdot V$$

These are the basic ingredients you need to get started with IK