Inverse Kinematics

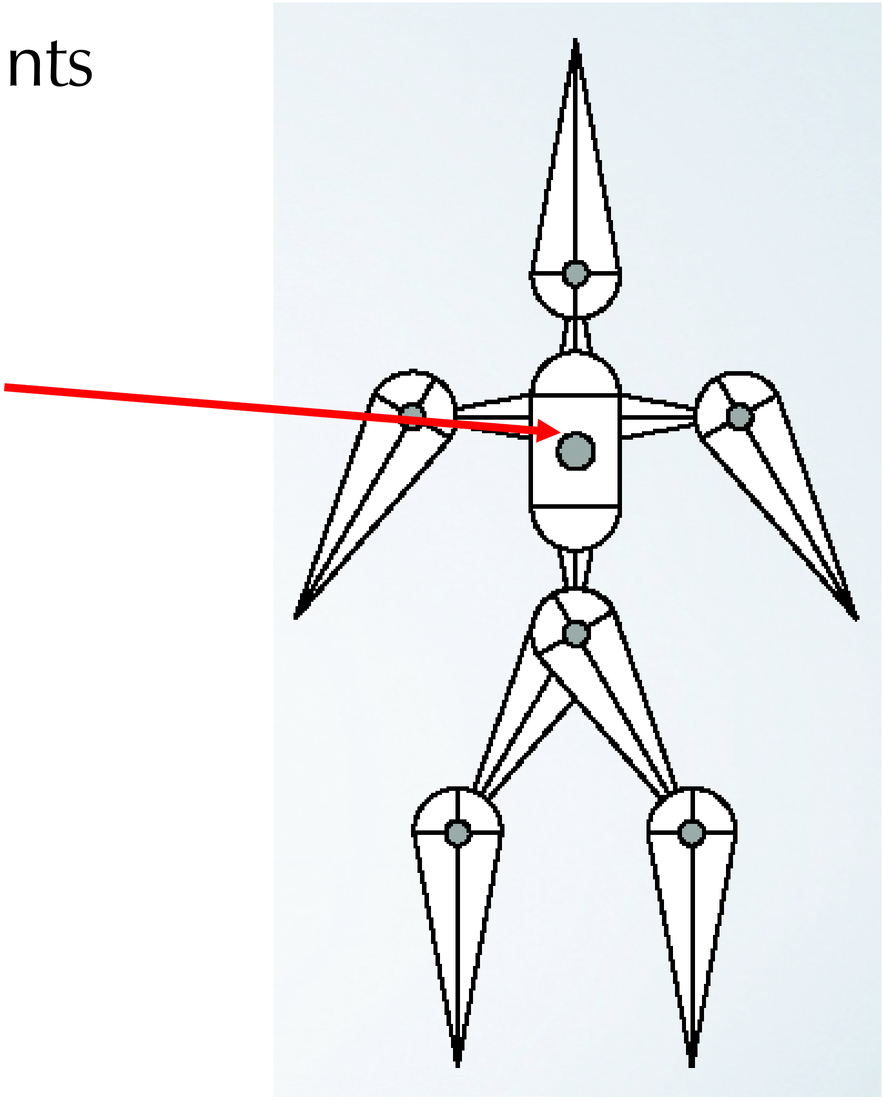
COS 526: Advanced Computer Graphics

PRINCETON UNIVERSITY

Slide credits: Rahul Narain, James O'Brien, Ravi Ramamoorthi.

Kinematic Tree / Skeleton

- Collection of bodies and joints
  – Tree-structured: loop joints would break "tree-ness"
- Root joint
  – Position, rotation set by global transformation
- Root body
  – Other bodies relative to root
  – "Inboard" vs "outboard": towards vs. away from root



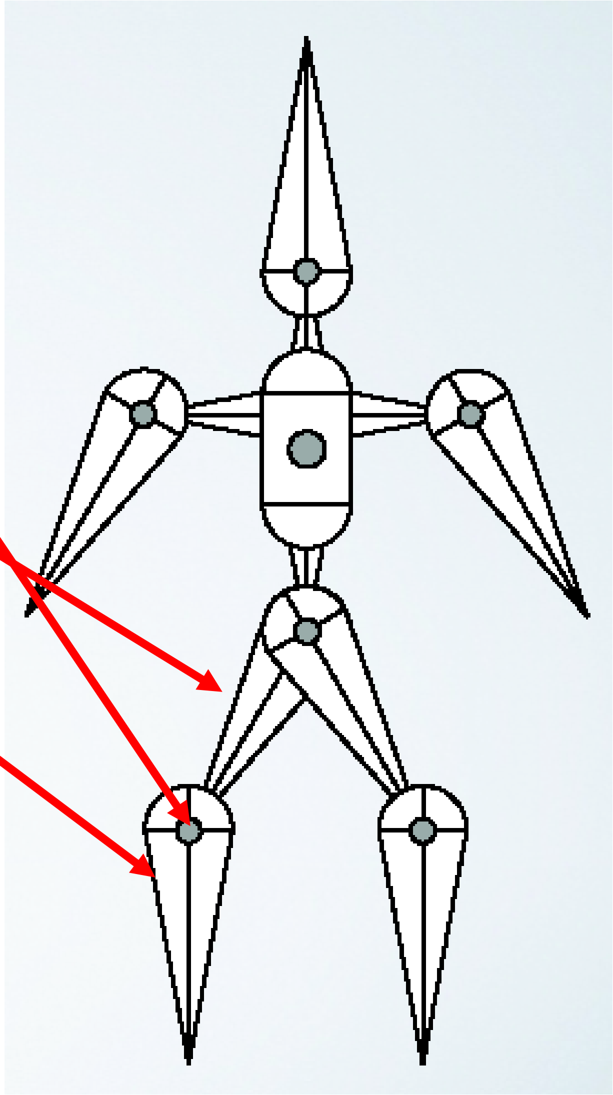
Inboard and Outboard

Joints

- Inboard body
- Outboard body

Body

- Inboard joint
- Outboard joint (may be several)



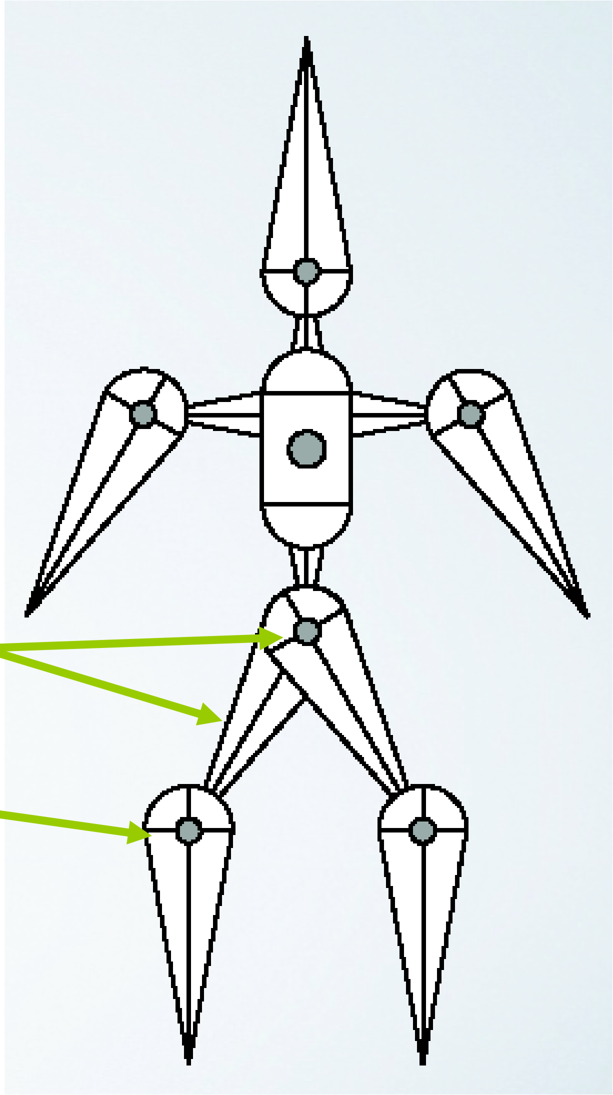
Inboard and Outboard

Joints

- Inboard body
- Outboard body

Body

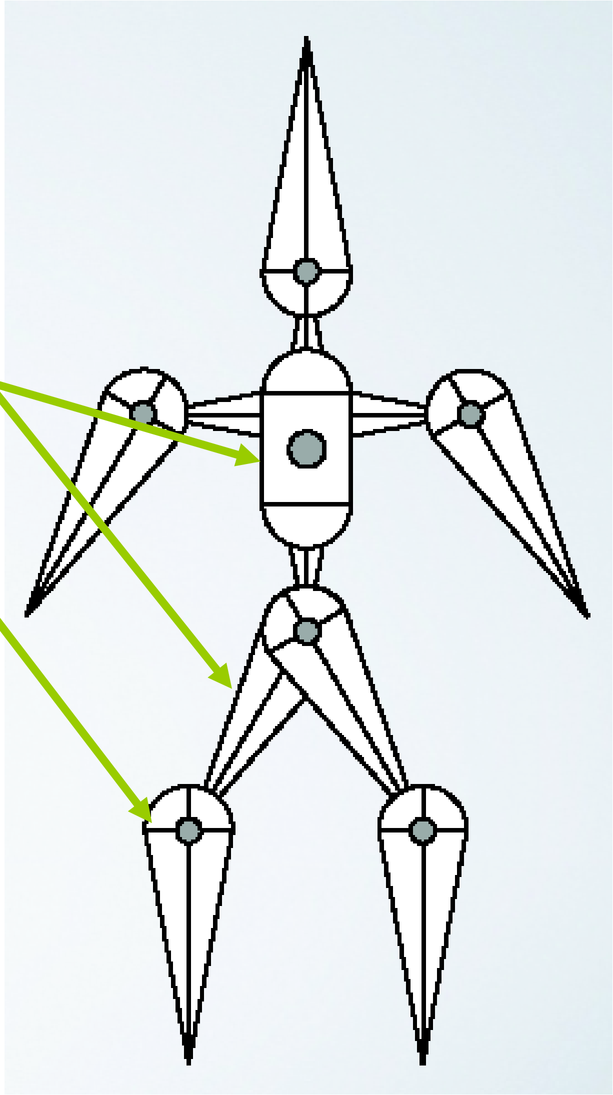
- Inboard joint
- Outboard joint (may be several)



Bodies

Bodies arranged in a tree

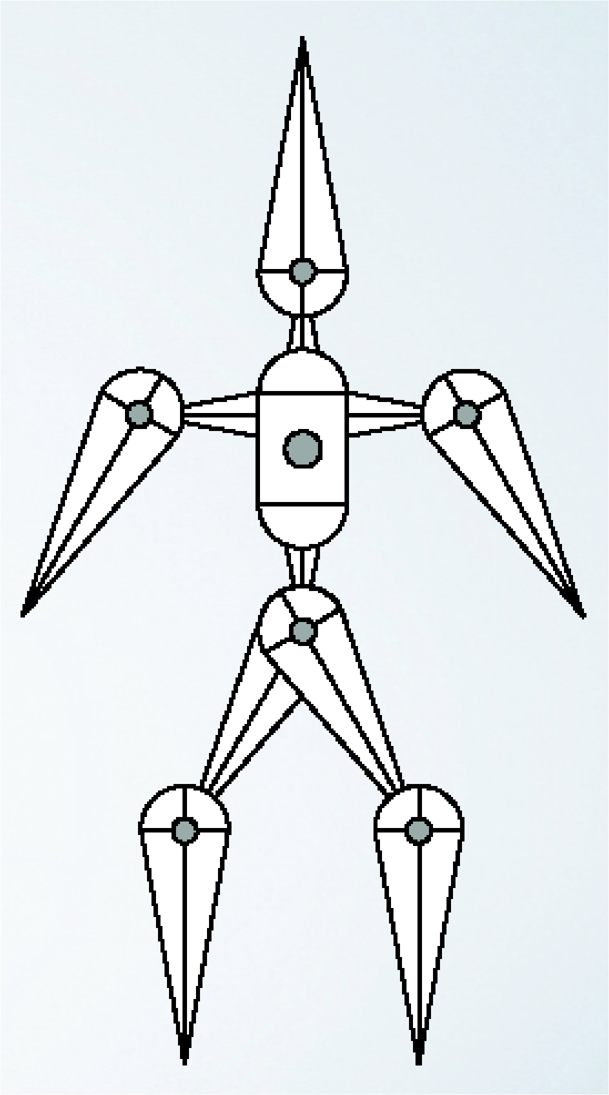
- For now, assume no loops
- Body's parent (except root)
- Body's child (may have many)



Joints

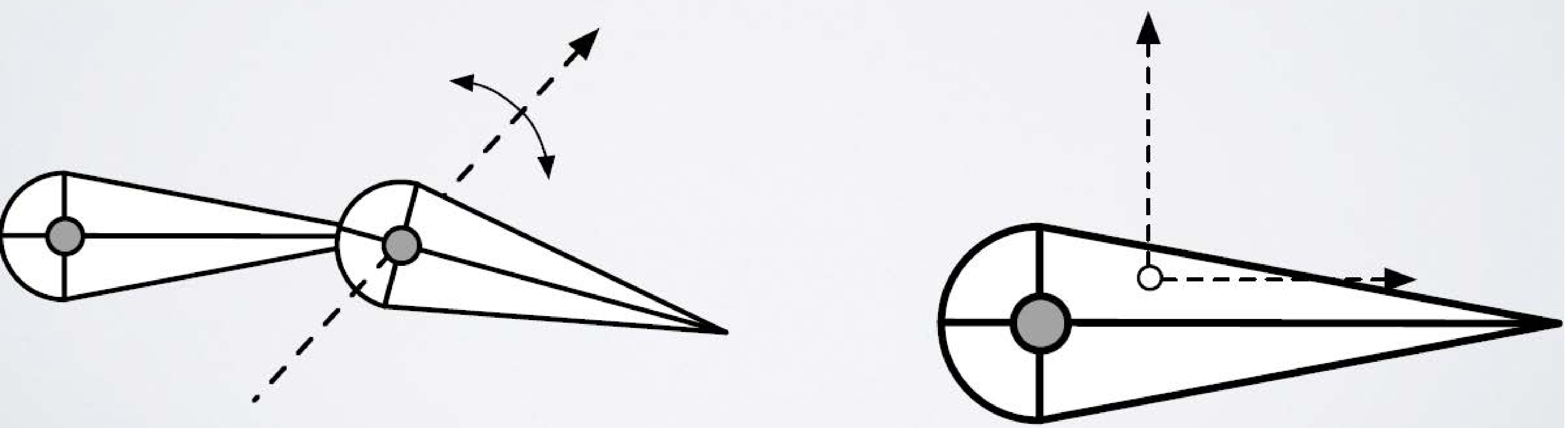
Interior Joints (typically not 6 DOF)

- Pin – rotate about one axis
- Ball – arbitrary rotation
- Prism – translate along one axis



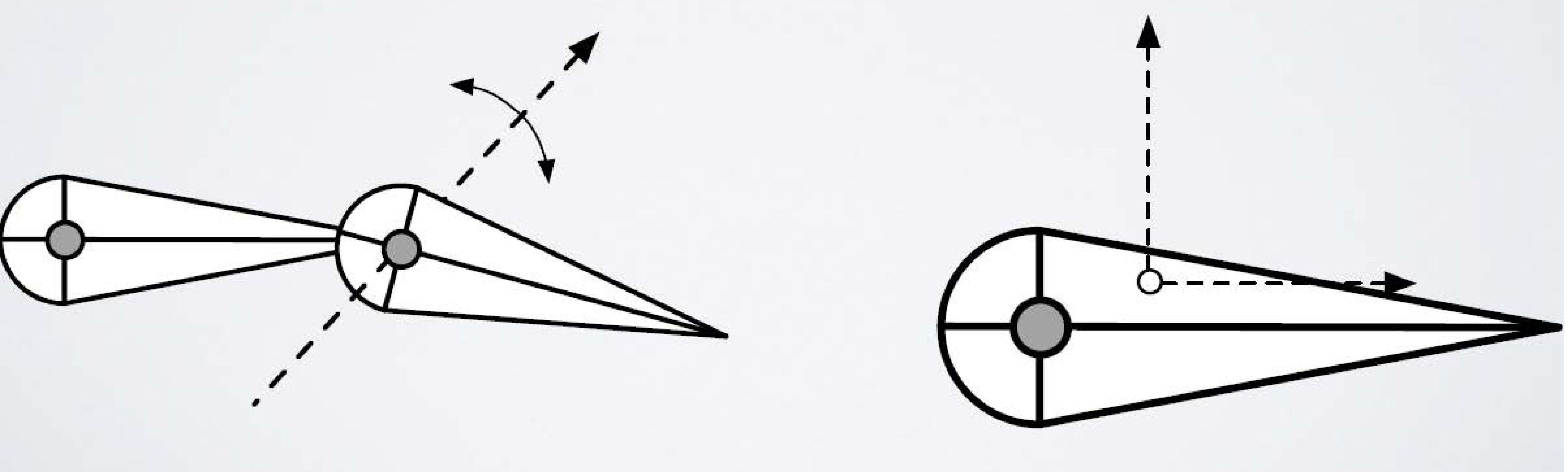
Pin Joints

- Relative to coordinate system at inboard joint…
- Apply rotation about fixed axis
- Translate origin to outboard joint



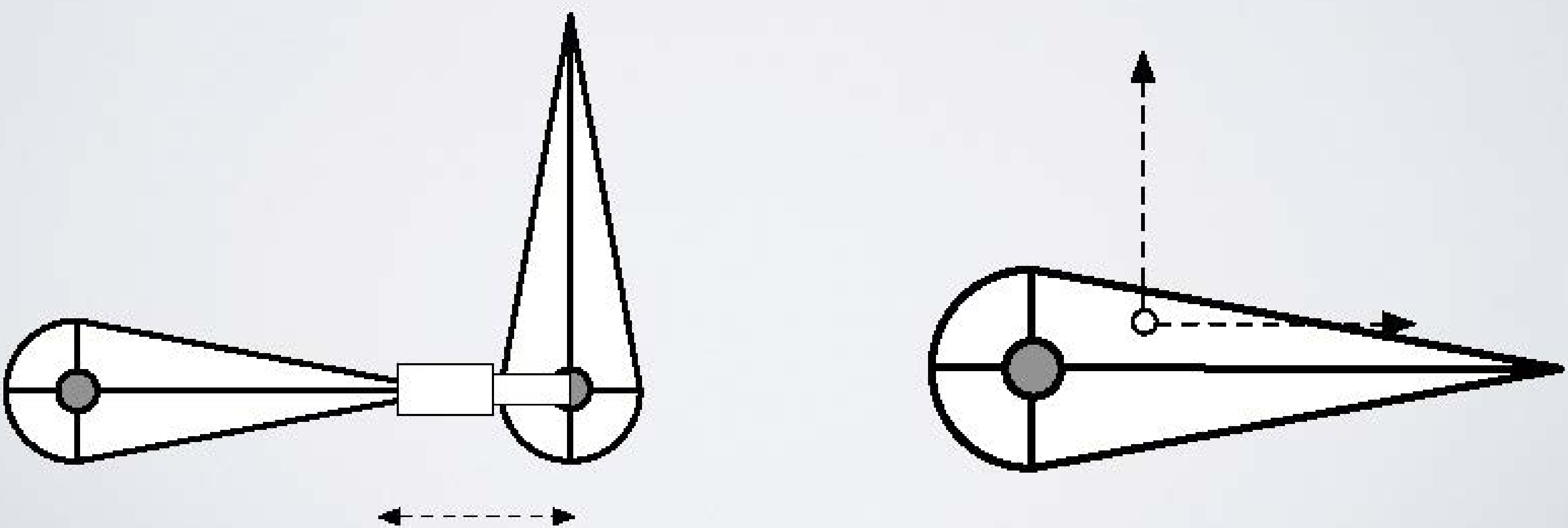
Ball Joints

- Relative to coordinate system at inboard joint…
- Apply rotation about arbitrary axis
- Translate origin to outboard joint



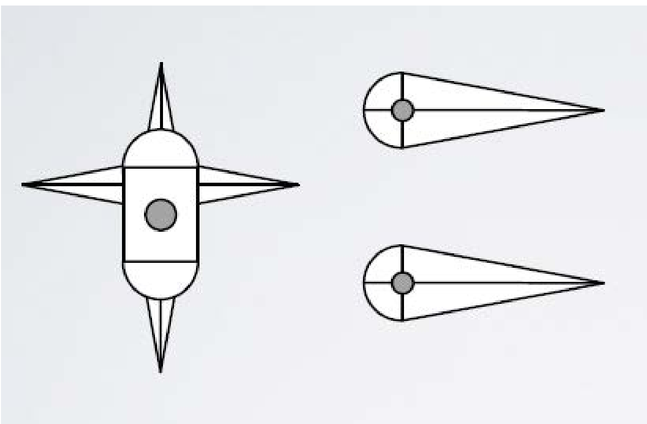
Prism Joints

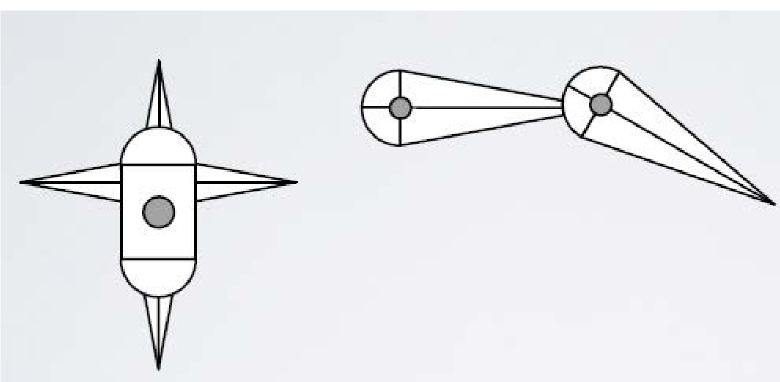
- Relative to coordinate system at inboard joint…
- Translate along fixed axis
- Translate origin to outboard joint

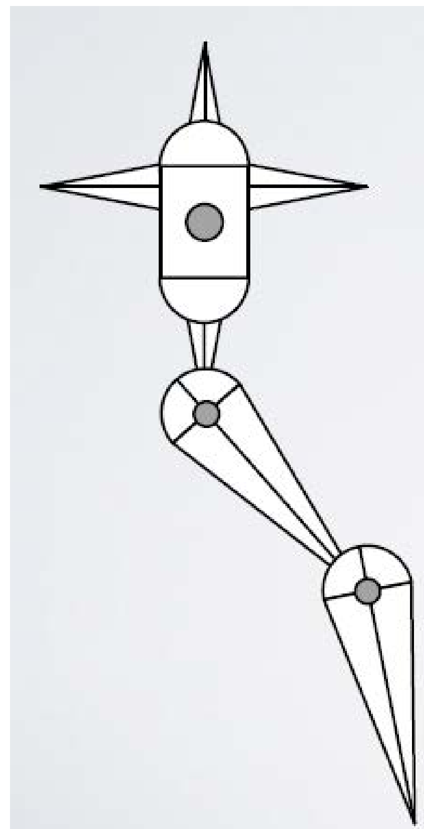


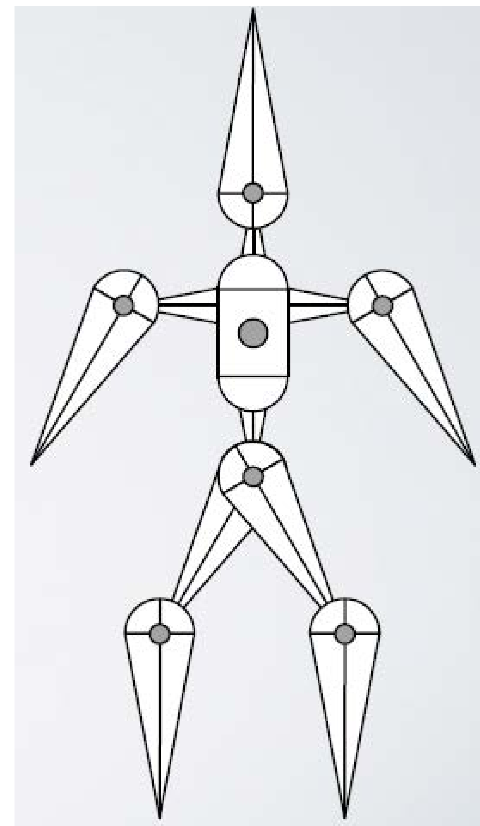
Forward Kinematics

- Composite transformations down the hierarchy



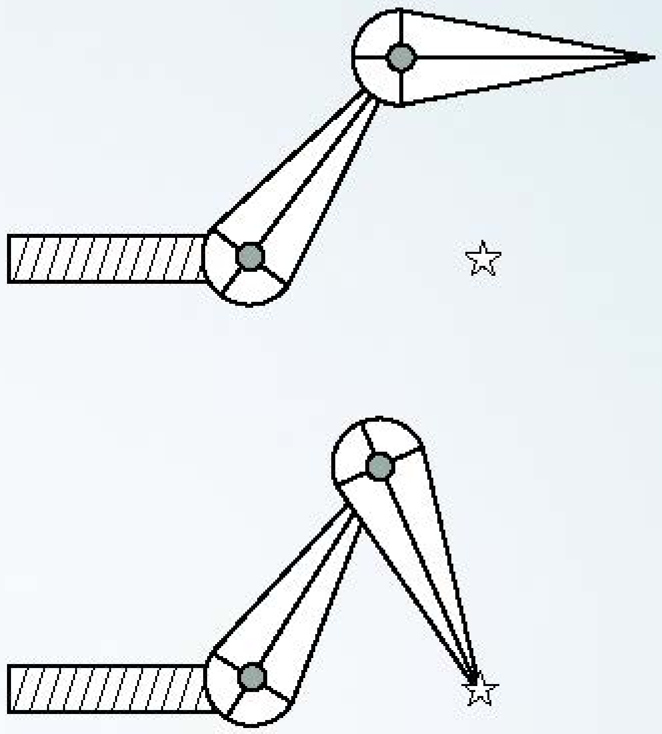




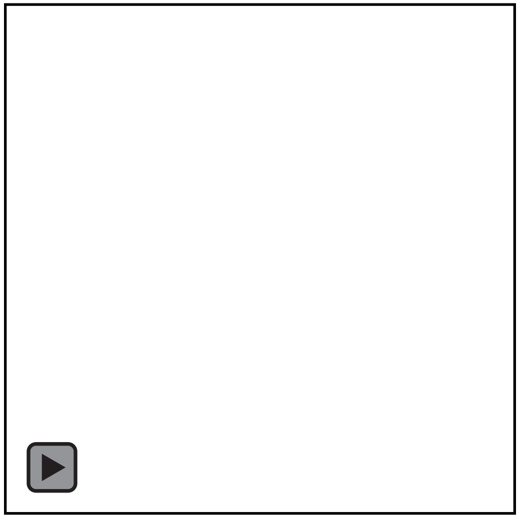


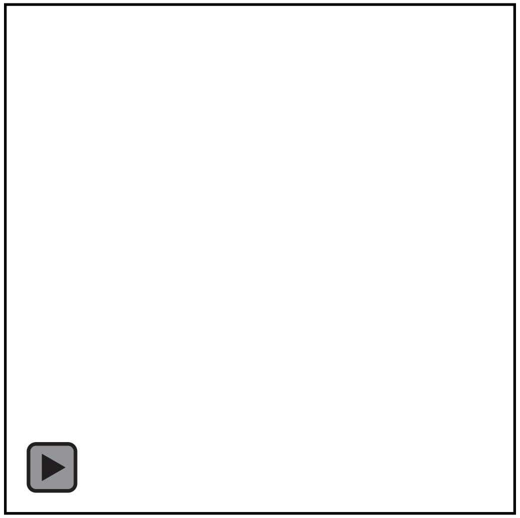
Inverse Kinematics

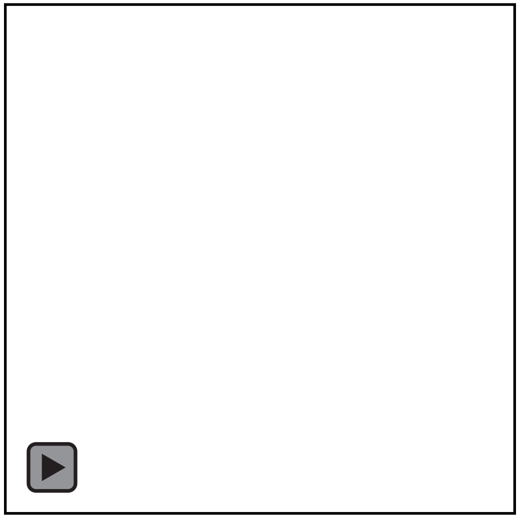
- Given
  - Root transformation
  - Initial configuration
  - Desired end point location
- Find
  - Interior parameter settings

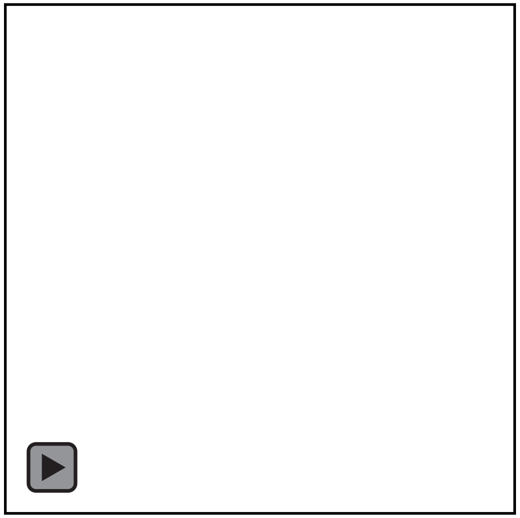


Inverse Kinematics

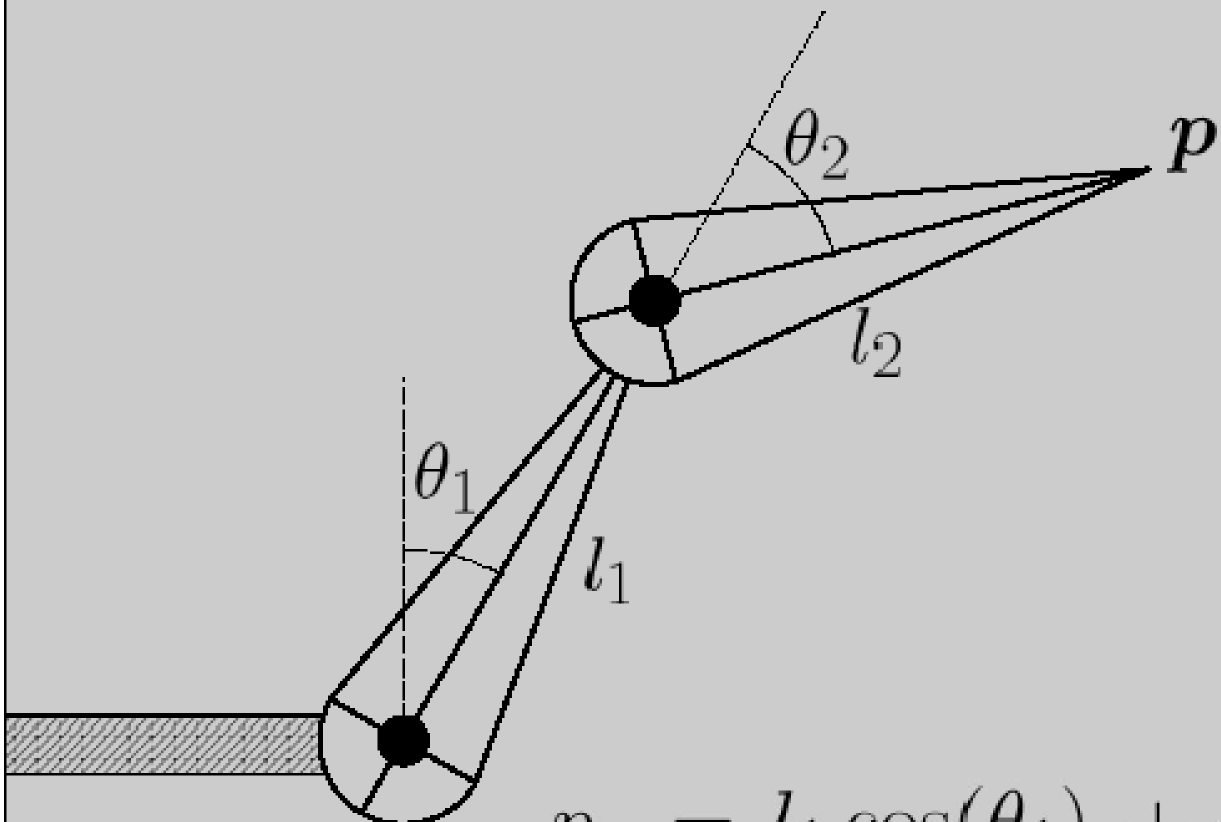








2-Segment Arm in 2D



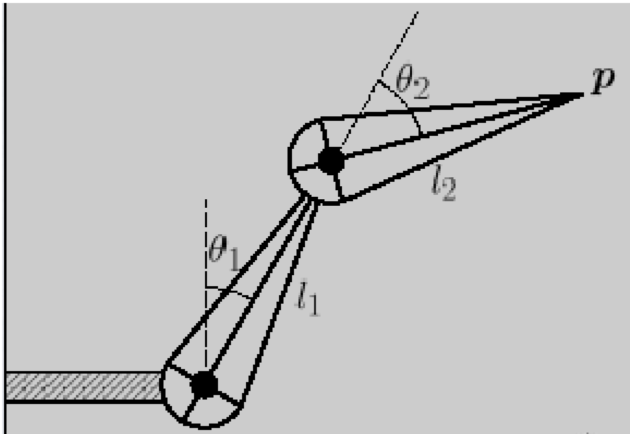
$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Warning: Z–up Coordinate System

Direct IK

- Analytically solve for parameters (not general)

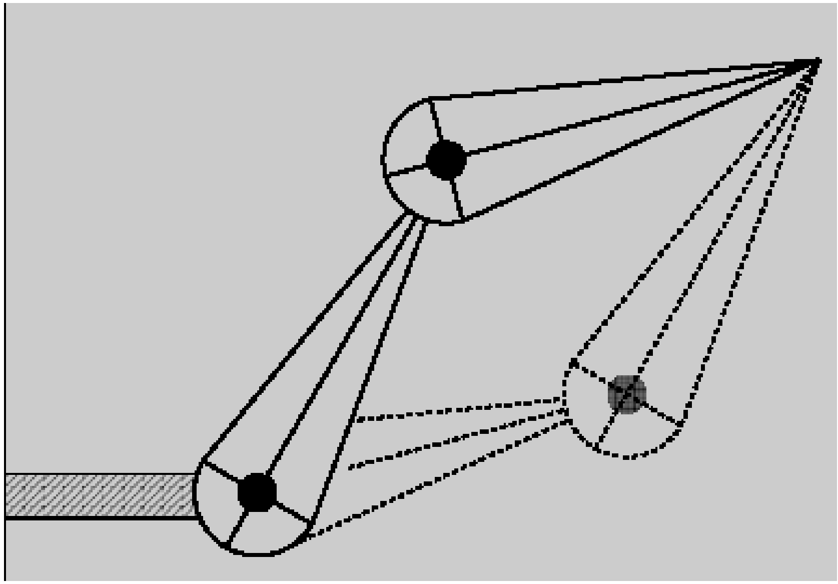


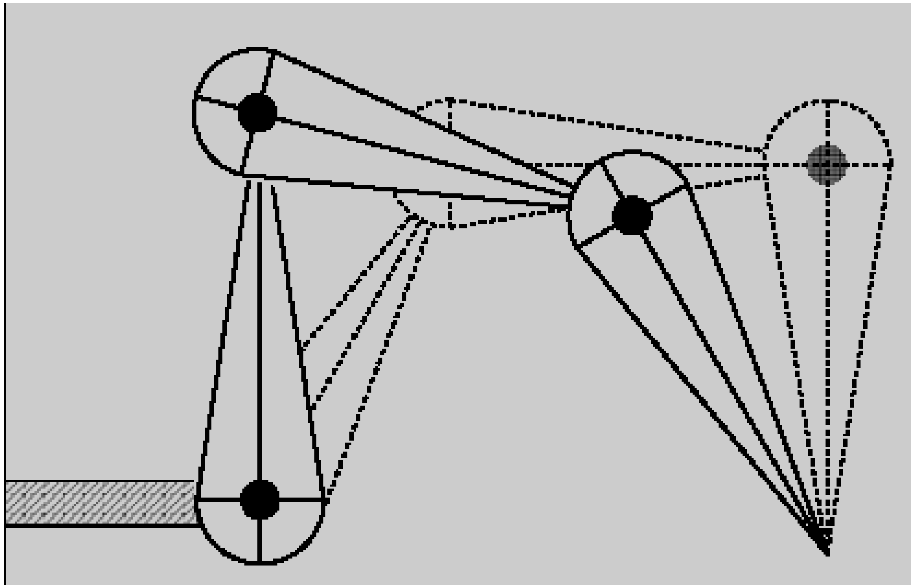
$$\theta_2 = \cos^{-1}\left(\frac{p_z^2 + p_x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\theta_1 = \tan^{-1} \frac{-p_z l_2 \sin(\theta_2) + p_x(l_1 + l_2\cos(\theta_2))}{p_x l_2 \sin(\theta_2) + p_z(l_1 + l_2\cos(\theta_2))}$$

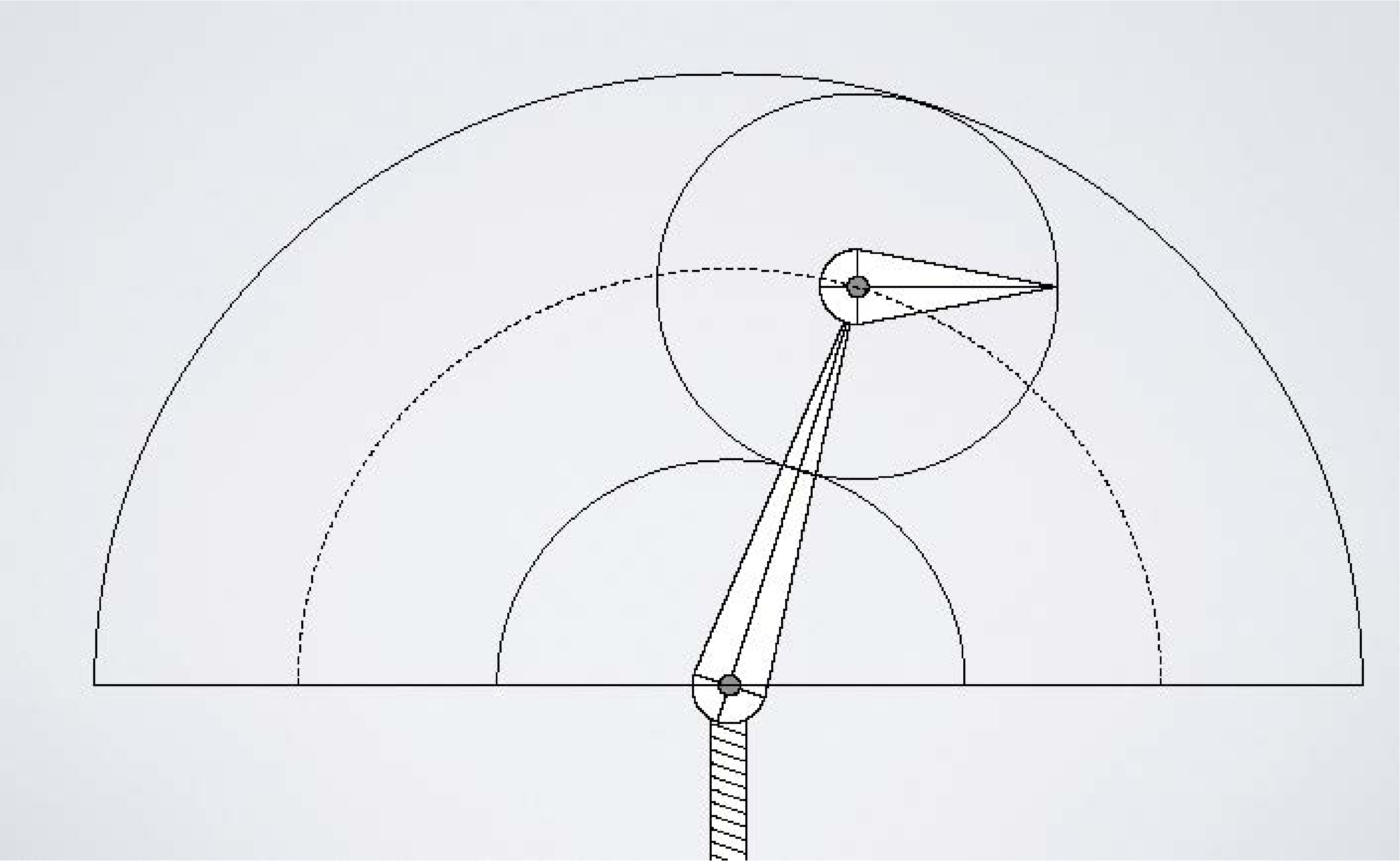
Difficult Issues

- Multiple configurations distinct in config space
- Or connected in config space





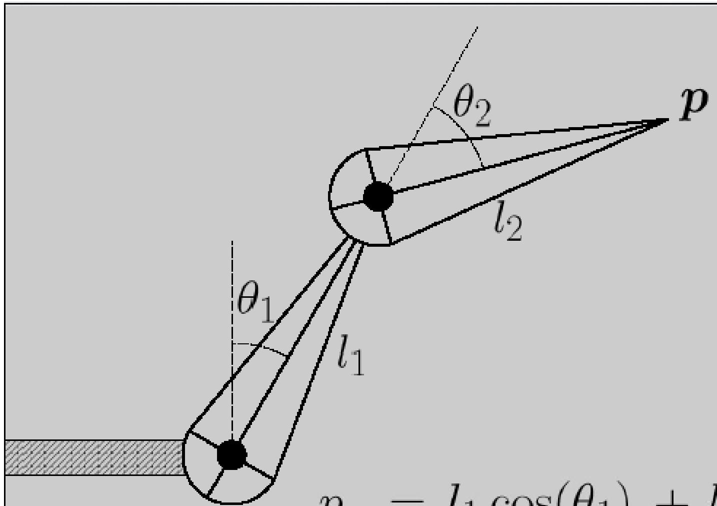
Infeasible Regions



Numerical Solution

- Start in some initial config. (previous frame)
- Define error metric (goal pos – current pos)
- Compute Jacobian with respect to inputs
- Iterate with gradient descent, Newton's method, etc.
- General principle of goal optimization

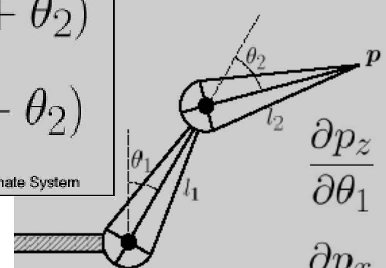
Back to 2 Segment Arm



$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Warning: Z–up Coordinate System



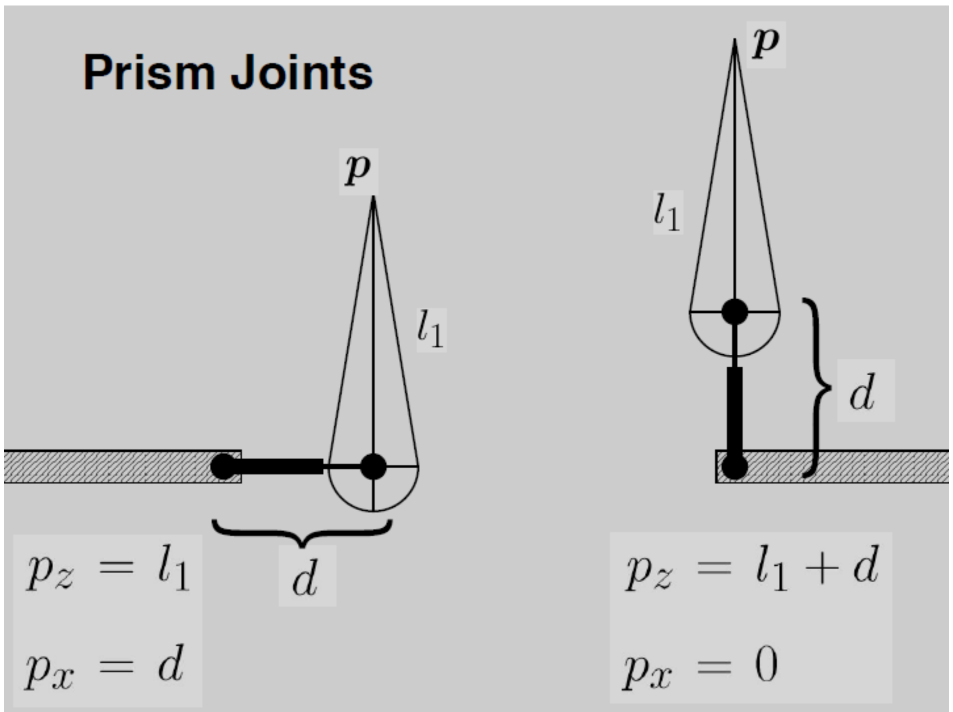
$$\frac{\partial p_z}{\partial \theta_1} = -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2)$$

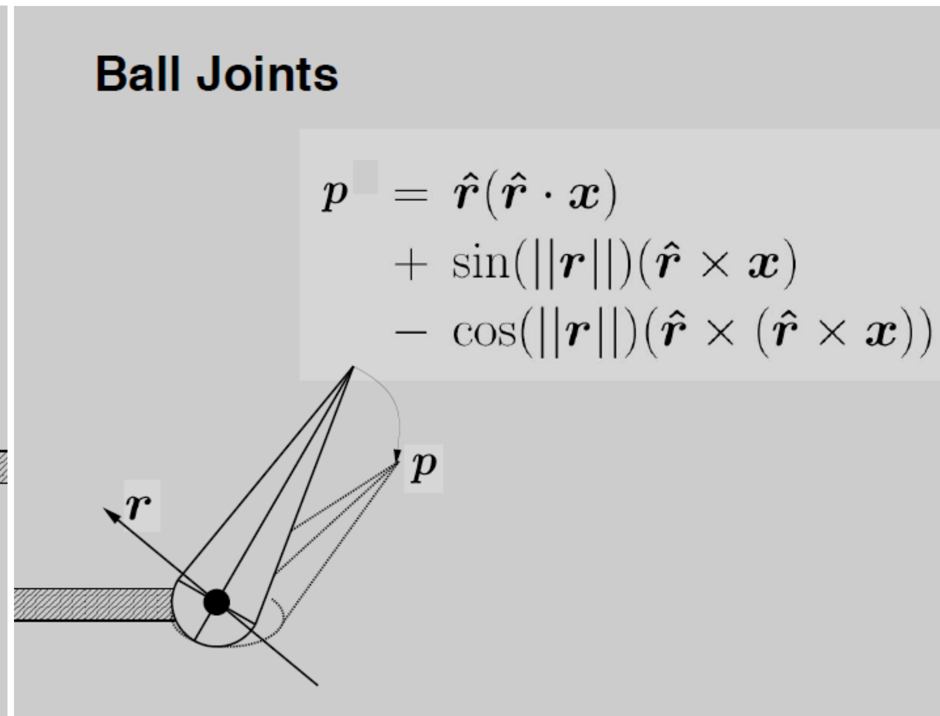
$$\frac{\partial p_x}{\partial \theta_1} = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$\frac{\partial p_z}{\partial \theta_2} = - l_2 \sin(\theta_1 + \theta_2)$$

$$\frac{\partial p_x}{\partial \theta_2} = + l_2 \cos(\theta_1 + \theta_2)$$

Prism and Ball Joints in 3D…





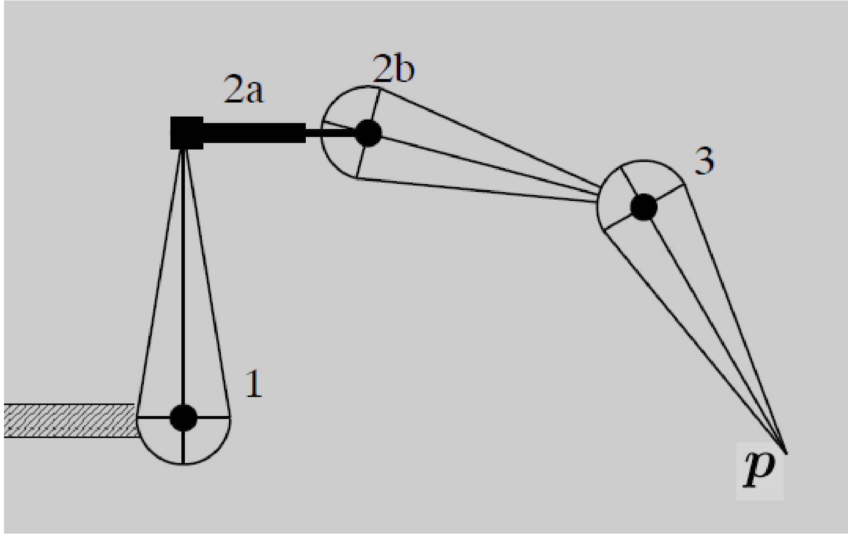
Issues

- Jacobian not always invertible
  - Use an SVD and pseudo-inverse
- Iterative approach, not direct
  - The Jacobian is a linearization, changes
- Practical implementation
  - Analytic forms for prism, ball joints
  - Composing transformations
  - Or quick and dirty: finite differencing
  - Cyclic coordinate descent (each DOF one at a time)

Multiple Links

- IK requires Jacobian
  - Need generic method for building one
- Won't work to just concatenate matrices

$$\tilde{J} = [J_3 \; J_{2b} \; J_{2a} \; J_{1b}]$$



$$\boldsymbol{d} = \begin{bmatrix} d_3 \\ d_{2b} \\ d_{2a} \\ d_{1b} \end{bmatrix}$$

$$\mathrm{d}\boldsymbol{p} \neq \tilde{J} \cdot \mathrm{d}\boldsymbol{d}$$

Composing Transformations

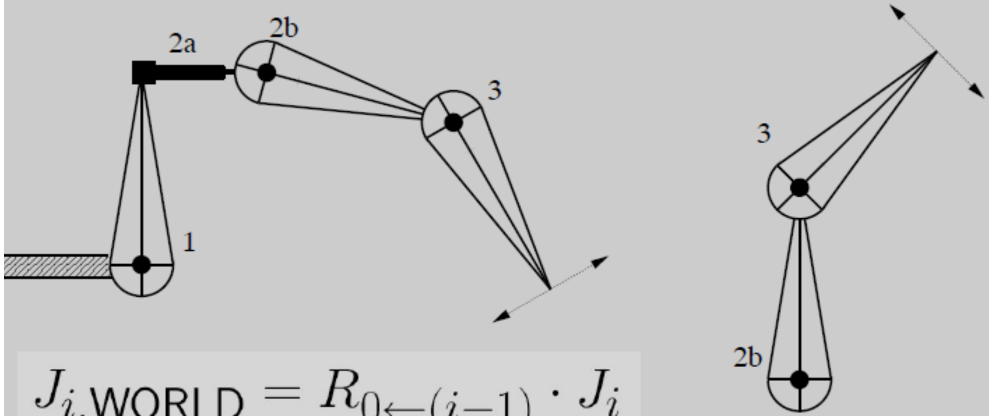
Transformation from body to world

$$X_{0\leftarrow i} = \prod_{j=1}^{i} X_{(j-1)\leftarrow j} = X_{0\leftarrow 1} \cdot X_{1\leftarrow 2} \cdots$$

Rotation from body to world

$$R_{0\leftarrow i} = \prod_{j=1}^{i} R_{(j-1)\leftarrow j} = R_{0\leftarrow 1} \cdot R_{1\leftarrow 2} \cdots$$

Need to transform Jacobians to common coordinate system (WORLD)



$$J_{i,\mathrm{WORLD}} = R_{0\leftarrow(i-1)} \cdot J_i$$

Inverse Kinematics: Final Form

$$J = \begin{bmatrix} R_{0\leftarrow 2\mathrm{b}} \cdot & J_3(\theta_3, \boldsymbol{p_3}) \\ R_{0\leftarrow 2\mathrm{a}} \cdot & J_{2\mathrm{b}}(\theta_{2\mathrm{b}}, X_{2\mathrm{b}\leftarrow 3} \cdot \boldsymbol{p_3}) \\ R_{0\leftarrow 1} \cdot & J_{2\mathrm{a}}(\theta_{2\mathrm{a}}, X_{2\mathrm{a}\leftarrow 3} \cdot \boldsymbol{p_3}) \\ & J_1(\theta_1, X_{1\leftarrow 3} \cdot \boldsymbol{p_3}) \end{bmatrix}^{\mathsf{T}}$$

$$\boldsymbol{d} = \begin{bmatrix} d_3 \\ d_{2\mathrm{b}} \\ d_{2\mathrm{a}} \\ d_{1\mathrm{b}} \end{bmatrix}$$

Note: Each row in the above should be transposed....

$$\mathrm{d}\boldsymbol{p} = J \cdot \mathrm{d}\boldsymbol{d}$$

Issues

- Jacobian not always invertible
  - Use an SVD and pseudo-inverse
- Iterative approach, not direct
  - The Jacobian is a linearization, changes
- Practical implementation
  - Analytic forms for prism, ball joints
  - Composing transformations
  - Or quick and dirty: finite differencing
  - Cyclic coordinate descent (each DOF one at a time)

A Cheap Alternative

- Estimate Jacobian (or parts of it) w. finite differences
- Cyclic coordinate descent
  - Solve for each DOF one at a time
  - Iterate till good enough / run out of time

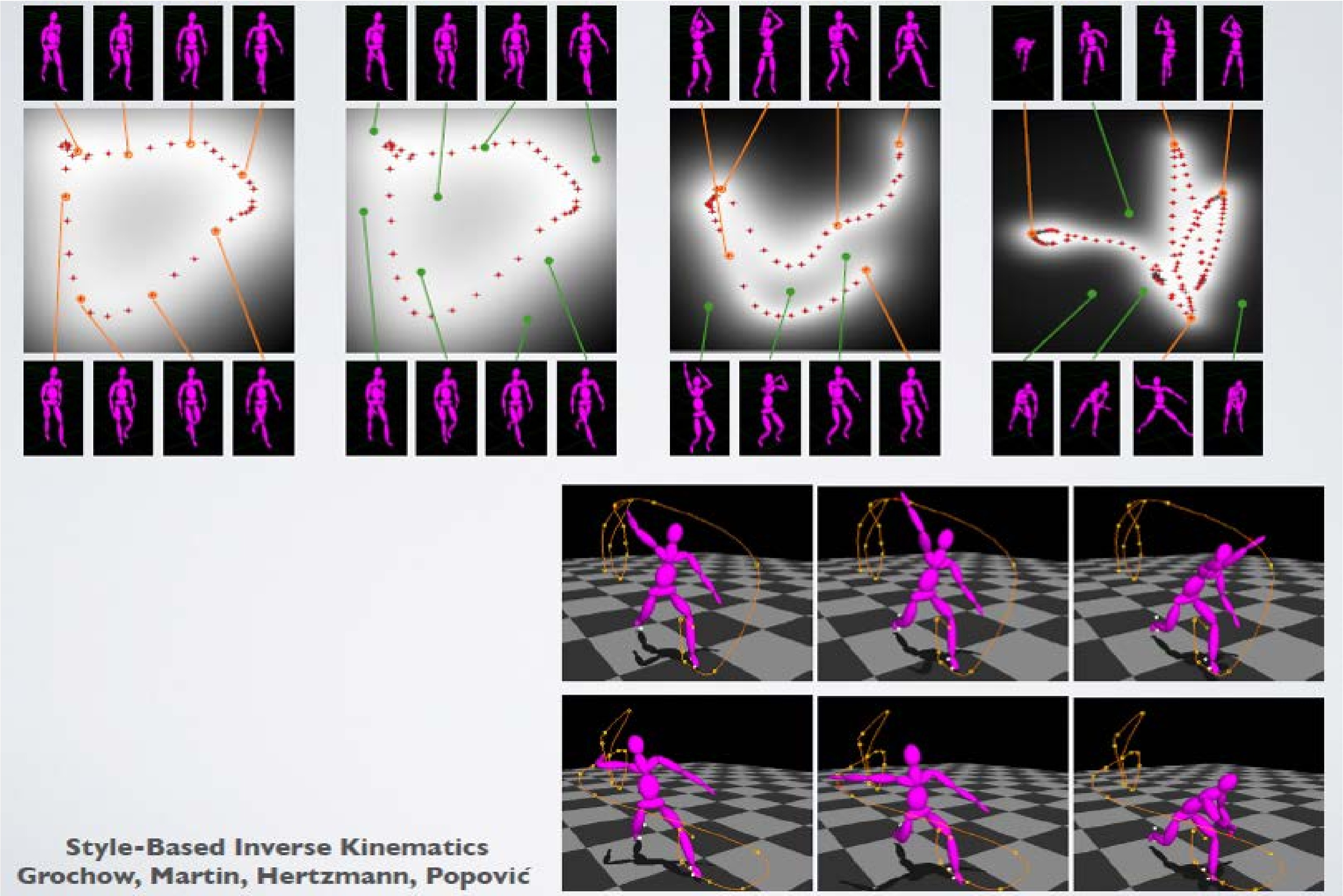
More complex systems

- More complex joints (prism and ball)
- More links
- Other criteria (center of mass or height)
- Hard constraints (e.g., foot plants)
- Unilateral constraints (e.g., joint limits)
- Multiple criteria and multiple chains
- Loops
- Smoothness over time
  - DOF determined by control points of curve (chain rule)

Practical Issues

- How to pick from multiple solutions?
- Robustness when no solutions
- Contradictory solutions
- Smooth interpolation
  - Interpolation aware of constraints

Prior on "good" configurations



Style-Based Inverse Kinematics
Grochow, Martin, Hertzmann, Popović