

Advanced Computer Graphics (Spring 2013)

CS 283, Lecture 20:

Inverse Kinematics

Ravi Ramamoorthi

<http://inst.eecs.berkeley.edu/~cs283/sp13>

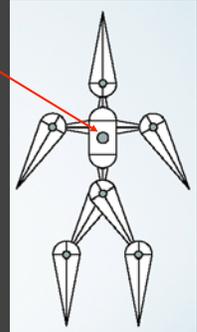


Most slides courtesy James O' Brien from CS294-13 Fall 2009

Forward Kinematics

Root body

- Position set by global transform
- Root joint: position, rotation
- Other bodies relative to root
- *Inboard* toward the root
- *Outboard* away from the root
- Tree structure: loop joints break "tree-ness"



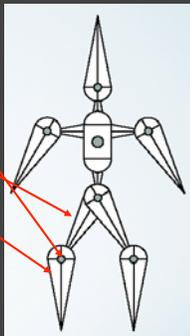
Inboard and Outboard

Joints

- Inboard body
- Outboard body

Body

- Inboard joint
- Outboard joint (may be several)



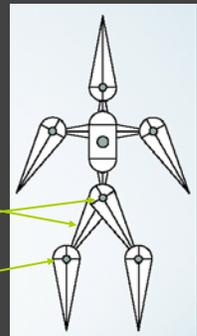
Inboard and Outboard

Joints

- Inboard body
- Outboard body

Body

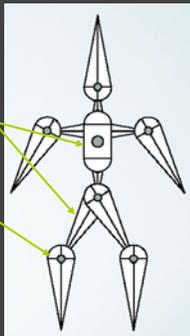
- Inboard joint
- Outboard joint (may be several)



Bodies

Bodies arranged in a tree

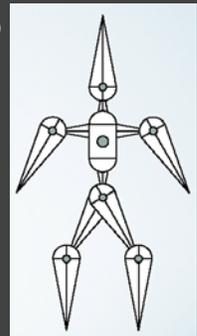
- For now, assume no loops
- Body's parent (except root)
- Body's child (may have many children)



Joints

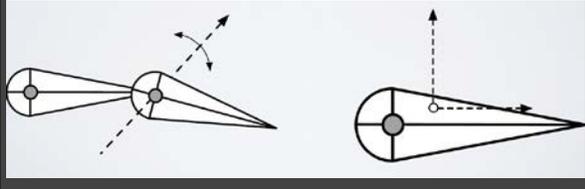
Interior Joints (typically not 6 DOF)

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



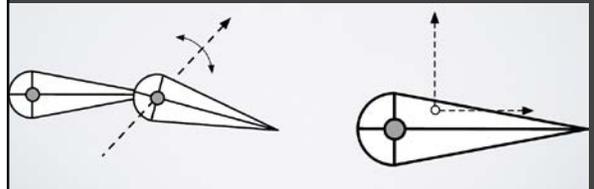
Pin Joints

- Translate inboard joint to local origin
- Apply rotation about axis
- Translate origin to location of joint on outboard body



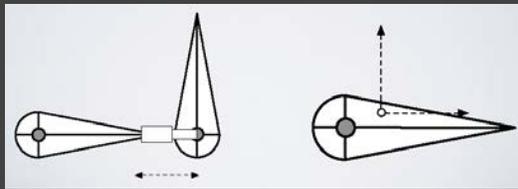
Ball Joints

- Translate inboard point to local origin
- Apply rotation about arbitrary axis
- Translate origin to location of joint on outboard body



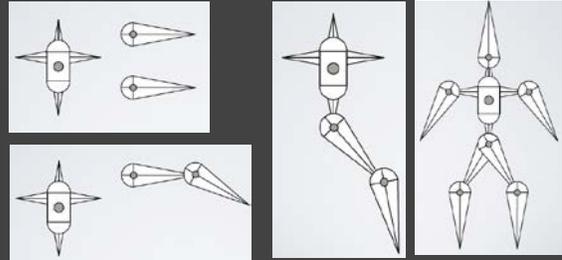
Prism Joint

- Translate inboard joint to local origin
- Translate along axis
- Translate origin to location of joint on outboard



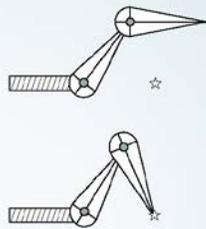
Forward Kinematics

- Composite transformations up the hierarchy

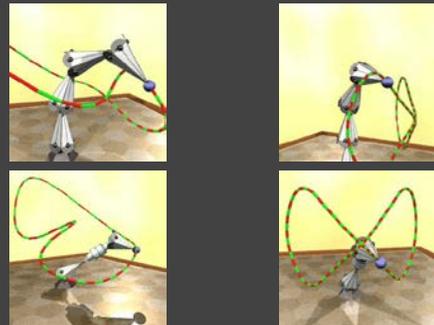


Inverse Kinematics

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

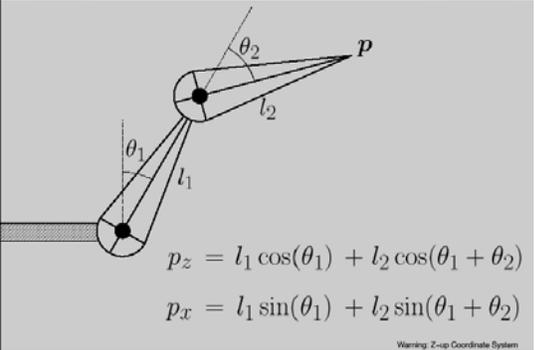


Inverse Kinematics



Egon Pasztor

2 Segment Arm in 2D



Direct IK

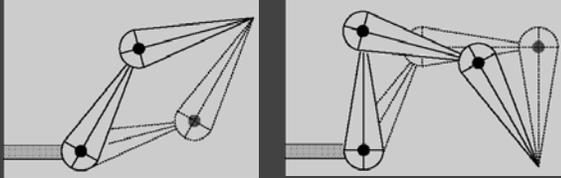
- Analytically solve for parameters (not general)

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

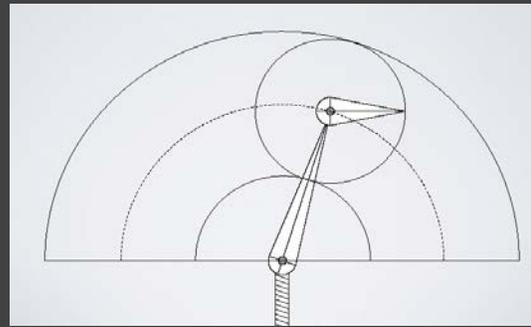
$$\theta_1 = \frac{-p_z l_2 \sin(\theta_2) + p_x (l_1 + l_2 \cos(\theta_2))}{p_x l_2 \sin(\theta_2) + p_z (l_1 + l_2 \cos(\theta_2))}$$

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
- Use Newton's or other method to iterate
- General principle of goal optimization

Back to 2 Segment Arm

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

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

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

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

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

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

Jacobians and Configuration Space



Direction in Config. Space

$$\theta_1 = c_1 \theta_*$$

$$\theta_2 = c_2 \theta_*$$

The Jacobian (of p w.r.t. θ)

$$\frac{\partial p_z}{\partial \theta_*} = c_1 \frac{\partial p_z}{\partial \theta_1} + c_2 \frac{\partial p_z}{\partial \theta_2}$$

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} \end{bmatrix}$$

$$\frac{\partial p}{\partial \theta_*} = J \cdot \begin{bmatrix} \frac{\partial \theta_1}{\partial \theta_*} \\ \frac{\partial \theta_2}{\partial \theta_*} \end{bmatrix} = J \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Solving for Joint Angles

Solving for c_1 and c_2

$$c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad dp = \begin{bmatrix} dp_x \\ dp_y \end{bmatrix}$$

$$dp = J \cdot c$$

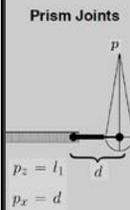
$$c = J^{-1} \cdot dp$$

Issues

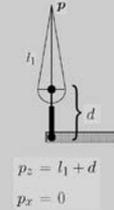
- Jacobian not always invertible
 - Use an SVD and pseudo-inverse
- Iterative approach, not direct
 - The Jacobian is a linearization, changes
- Practical implementation
 - Analytic forms for prism, ball joints
 - Composing transformations
 - Or quick and dirty: finite differencing
 - Cyclic coordinate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

Prism and Ball Joints

Prism Joints

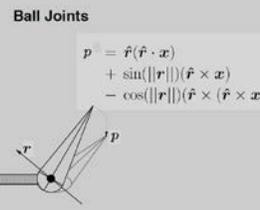


$p_z = l_1$
 $p_x = d$



$p_z = l_1 + d$
 $p_x = 0$

Ball Joints

$$p = \hat{r}(\hat{r} \cdot x) + \sin(|\hat{r}|)(\hat{r} \times x) - \cos(|\hat{r}|)(\hat{r} \times (\hat{r} \times x))$$


More on Ball Joints

Ball Joints (moving axis)

$$dp = [dr] \cdot e^{[r]} \cdot x = [dr] \cdot p - [p] \cdot dr$$

Ball Joints (fixed axis)

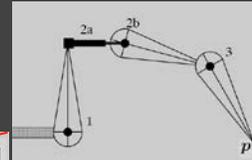
$$dp = (d\theta) [\hat{r}] \cdot x = -[x] \cdot \hat{r} d\theta$$

$$[r] = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$$

$[r] \cdot x = r \times x$

Multiple Links

- IK requires Jacobian
 - Need generic method for building one
- Can't just concatenate matrices



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

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

$$dp \neq \tilde{J} \cdot dd$$

Composing Transformations

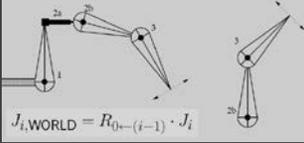
Transformation from body to world

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

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)



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

Inverse Kinematics: Final Form

$$J = \begin{bmatrix} R_{0 \leftarrow 2b} \cdot J_3(\theta_3, \mathbf{p}_3) \\ R_{0 \leftarrow 2a} \cdot J_{2b}(\theta_{2b}, X_{2b \leftarrow 3} \cdot \mathbf{p}_3) \\ R_{0 \leftarrow 1} \cdot J_{2a}(\theta_{2a}, X_{2a \leftarrow 3} \cdot \mathbf{p}_3) \\ J_1(\theta_1, X_{1 \leftarrow 3} \cdot \mathbf{p}_3) \end{bmatrix}^T$$

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

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

$$d\mathbf{p} = J \cdot d\mathbf{d}$$

A Cheap Alternative

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

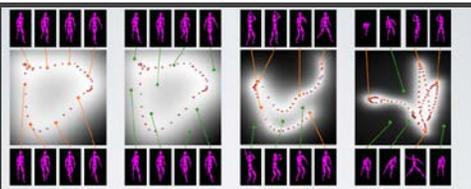
More complex systems

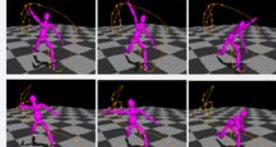
- More complex joints (prism and ball)
- More links
- Other criteria (center of mass or height)
- Hard constraints (e.g., foot plants)
- Unilateral constraints (e.g., joint limits)
- Multiple criteria and multiple chains
- Loops
- Smoothness over time
 - DOF determined by control points of curve (chain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