Advanced Computer Graphics
CSE 163 [Spring 2017], Lecture 19
Ravi Ramamoorthi
http://www.cs.ucsd.edu/~ravir

To Do
- Assignment 3 due Jun 14
  - Should already be well on way
  - Contact us for difficulties etc
- Please fill out CAPE evaluations (Now!)

Course Outline
- 3D Graphics Pipeline
  - Rendering (Creating, shading images from geometry, lighting, materials)
  - Modeling (Creating 3D Geometry)

Course Outline
- 3D Graphics Pipeline
  - Rendering (Creating, shading images from geometry, lighting, materials)
  - Modeling (Creating 3D Geometry)

Course Outline
- 3D Graphics Pipeline
  - Rendering (Creating, shading images from geometry, lighting, materials)
  - Modeling (Creating 3D Geometry)

The Story So Far
- Scene ➔ Image

Animation
- scene(t) ➔ image(t)

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)

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

2 Segment Arm in 2D
- Analytically solve for parameters (not general)

\[
\theta_2 = \cos^{-1} \left( \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right)
\]

\[
\theta_1 = \tan^{-1} \left( \frac{-p_2 l_2 \sin(\theta_2) + p_x (l_1 + l_2 \cos(\theta_2))}{p_x l_2 \sin(\theta_2) + p_z (l_1 + l_2 \cos(\theta_2))} \right)
\]

\[
p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)
\]

\[
p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)
\]
**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**

\[
\begin{align*}
    p_x &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\
    p_y &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)
\end{align*}
\]

\[
\begin{align*}
    \frac{\partial p_x}{\partial \theta_1} &= -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) \\
    \frac{\partial p_x}{\partial \theta_2} &= -l_2 \sin(\theta_1 + \theta_2) \\
    \frac{\partial p_y}{\partial \theta_1} &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\
    \frac{\partial p_y}{\partial \theta_2} &= l_2 \cos(\theta_1 + \theta_2)
\end{align*}
\]

**Jacobians and Configuration Space**

**Solving for Joint Angles**

Solving for \( c_1 \) and \( c_2 \)

\[
\begin{align*}
    c &= \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \\
    dp &= \begin{bmatrix} dp_1 \\ dp_2 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
    \frac{\partial p}{\partial \theta_1} &= \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} \\ \frac{\partial p_y}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \\
    c &= J^{-1} \cdot dp
\end{align*}
\]
**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**

**Ball Joints**

- $p = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
- $p = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$

---

**More on Ball Joints**

**Ball Joints (moving axis)**

\[
\frac{dp}{dt} = \frac{d}{dt} [\begin{bmatrix} x \\ y \\ z \end{bmatrix}] = \frac{d}{dt} [\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}]
\]

**Ball Joints (fixed axis)**

\[
\frac{dp}{dt} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}
\]

---

**Multiple Links**

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

**Multiple Links (example)**

\[
d = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}
\]

\[
\frac{dp}{dt} = J \cdot dd
\]

---

**Composing Transformations**

**Transformation from body to workd**

\[
X_{b_{i-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} X_{b_{i-1}} \times X_{b_{i-2}} 
\]

**Rotation from body to world**

\[
R_{b_{i-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_{b_{i-1}} \times R_{b_{i-2}} 
\]

**Inverse Kinematics: Final Form**

\[
J = \begin{bmatrix} R_{0_{i-2}} \cdot J_3(\theta_3, p_3) \\ R_{0_{i-2}} \cdot J_2(\theta_2, \cdot) X_{2_{i-3}} \cdot p_3 \\ R_{0_{i-1}} \cdot J_1(\theta_1, X_{1_{i-2}} \cdot p_3) \\
\end{bmatrix}
\]

\[
d = \begin{bmatrix} d_3 \\ d_2 \\ d_2 \end{bmatrix}
\]

\[
\frac{dp}{dt} = J \cdot dd
\]
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