Announcements

Final Project: multiple due dates
- Project proposal due Wed Nov 17, 11 pm
- Progress report 1 due Mon Nov 22, 11 pm
- Progress report 2 due Wed Dec 1, 11 pm
- Final report due Wed Dec 8, 11 pm
Today

Forward kinematics
Inverse kinematics
  • Pin joints
  • Ball joints
  • Prismatic joints

Forward Kinematics

Articulated skeleton
  • Topology (what’s connected to what)
  • Geometric relations from joints
  • Independent of display geometry
  • Tree structure
    • Loop joints break “tree-ness”

Forward Kinematics

Root body
  • Position set by “global” transformation
  • Root joint
    • Position
    • Rotation
  • Other bodies relative to root
    • Inboard toward the root
    • Outboard away from root
Forward Kinematics

A joint
- Joint's inboard body
- Joint's outboard body

A body
- Body's inboard joint
- Body's outboard joint
  - May have several outboard joints
- Body's parent
- Body's child
  - May have several children
Forward Kinematics

Interior joints

- Typically not 6 DOF joints
- Pin - rotate about one axis
- Ball - arbitrary rotation
- Prism - translation along one axis

Forward Kinematics

Pin Joints

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

Forward Kinematics

Ball Joints

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

Prismatic Joints

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

Composite transformations up the hierarchy
Forward Kinematics

Composite transformations up the hierarchy
Inverse Kinematics

Given
- Root transformation
- Initial configuration
- Desired end point location

Find
- Interior parameter settings

Inverse Kinematics

A simple two segment arm in 2D

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

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

Direct IK: solve for the parameters

\[ \theta_2 = \cos^{-1}\left(\frac{p_y^2 + p_z^2 - l_1^2 - l_2^2}{2l_1l_2}\right) \]

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

Inverse Kinematics

Why is the problem hard?

- Multiple solutions separated in configuration space

Inverse Kinematics

Why is the problem hard?

- Multiple solutions connected in configuration space
Inverse Kinematics

Why is the problem hard?

- Solutions may not always exist

Inverse Kinematics

Numerical Solution

- Start in some initial configuration
- Define an error metric (e.g. goal pos - current pos)
- Compute Jacobian of error w.r.t. inputs
- Apply Newton’s method (or other procedure)
- Iterate...

Inverse Kinematics

Recall simple two segment arm:

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

We can write the derivatives:

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

Direction in Config. Space

\[
\begin{align*}
\theta_1(\theta_*) &= c_1 \theta_* \\
\theta_2(\theta_*) &= c_2 \theta_* \\
\frac{d\theta_1}{d\theta_*} &= c_1 \\
\frac{d\theta_2}{d\theta_*} &= c_2
\end{align*}
\]

\[
\frac{dp_x(\theta_1(\theta_*), \theta_2(\theta_*))}{d\theta_*} = \frac{dp_x}{d\theta_1} \frac{d\theta_1}{d\theta_*} + \frac{dp_x}{d\theta_2} \frac{d\theta_2}{d\theta_*}
\]

The Jacobian (of \( p \) w.r.t. \( \theta \))

\[
J_{ij} = \frac{\partial p_i}{\partial \theta_j}
\]

Example for two segment arm

\[
J = \begin{bmatrix}
\frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \\
\frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2}
\end{bmatrix}
\]}
Inverse Kinematics

The Jacobian (of \( p \) w.r.t. \( \theta \))

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

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

Inverse Kinematics

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

\[
c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad \text{d}p = \begin{bmatrix} \text{d}p_x \\ \text{d}p_y \end{bmatrix}
\]

\[
\text{d}p = J \cdot c \\
c = J^{-1} \cdot \text{d}p
\]

Inverse Kinematics

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

\[
\text{d}p = J \cdot c \\
c = J^{-1} \cdot \text{d}p
\]
Inverse Kinematics

Jacobian is not always invertible
- Use pseudo inverse (SVD)

Computing a linear approximation
- End effector only locally moves linearly
- So iterate (choosing proper step size) and update Jacobian after each step
- Choosing step size requires line search at each step
  - Choose some step size (say 5 degrees) and compute how to update joint parameters
  - Calculate distance of end effector from goal
  - If distance decreased take step
  - If distance did not decrease set parameters to be half the current change and try again

Inverse Kinematics

More complex systems
- More complex joints (prism and ball)
- More links
- Hard constraints (joint limits)
- Multiple criteria and multiple chains

Inverse Kinematics

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

**Prism Joints**

\[ p_z = l_1 \]
\[ p_x = d \]

**Ball Joints**

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

Inverse Kinematics

**Ball Joints (moving axis)**

\[ \mathbf{d} \mathbf{p} = [\mathbf{dr}] \cdot [\mathbf{r}] \cdot \mathbf{x} = [\mathbf{dr}] \cdot \mathbf{p} = -[\mathbf{p}] \cdot \mathbf{dr} \]

That is the Jacobian for this joint.

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

\[ [\mathbf{r}] \cdot \mathbf{x} = \mathbf{r} \times \mathbf{x} \]
Inverse Kinematics

**Ball Joints (fixed axis)**

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

That is the Jacobian for this joint.

---

**Inverse Kinematics**

Many links / joints

* Need a generic method for building Jacobian

---

**Inverse Kinematics**

Can’t just concatenate individual matrices

\[ \mathbf{J} = \begin{bmatrix} J_3 & J_{2b} & J_{2a} & J_{1b} \end{bmatrix} \]

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

\[ \mathbf{dp} \neq \tilde{\mathbf{J}} \cdot \mathbf{dd} \]
Inverse Kinematics

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

\[ J = \begin{bmatrix} R_{0 \leftarrow 2b} \cdot J_{3}(\theta_{3}, p_{3}) \\ R_{0 \leftarrow 2a} \cdot J_{2b}(\theta_{2b}, X_{2b \leftarrow 3} \cdot p_{3}) \\ R_{0 \leftarrow 1} \cdot J_{2a}(\theta_{2a}, X_{2a \leftarrow 3} \cdot p_{3}) \\ J_{1}(\theta_{1}, X_{1 \leftarrow 3} \cdot p_{3}) \end{bmatrix}^T \]

\[ d = \begin{bmatrix} d_{3} \\ d_{2b} \\ d_{2a} \\ d_{1b} \end{bmatrix} \]

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

\[ dp = J \cdot dd \]
Suggested Reading

Numerical Methods for Inverse Kinematics by Niels Joubert (see wiki)

Advanced Animation and Rendering Techniques by Watt and Watt
  • Chapters 15 and 16