CS 294-13
Advanced Computer Graphics

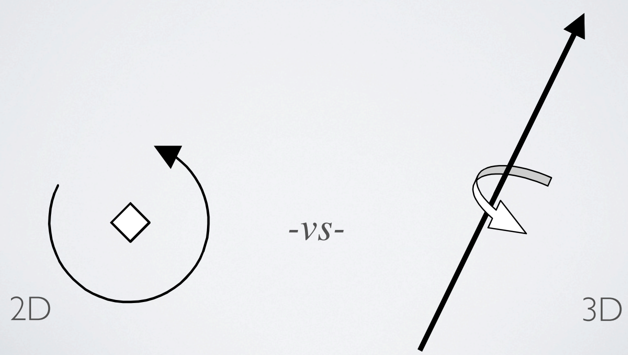
Rotations and Inverse Kinematics

James F. O'Brien

Associate Professor
U.C. Berkeley

Rotations

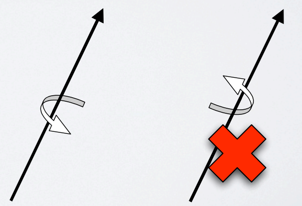
- 3D Rotations fundamentally more complex than in 2D
  - 2D: amount of rotation
  - 3D: amount and axis of rotation



Rotations

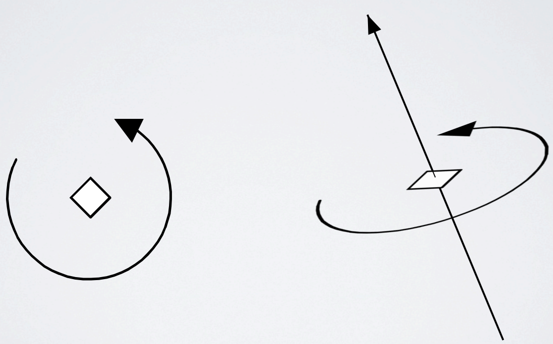
- Rotations still orthonormal
- $\mathrm{Det}(\mathbf{R}) = 1 \neq -1$
- Preserve lengths and distance to origin
- 3D rotations DO NOT COMMUTE!
- Right-hand rule
- Unique matrices

DO NOT COMMUTE!



Axis-aligned 3D Rotations

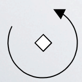
- 2D rotations implicitly rotate about a third out of plane axis

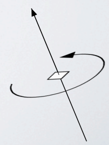


Axis-aligned 3D Rotations

- 2D rotations implicitly rotate about a third out of plane axis

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



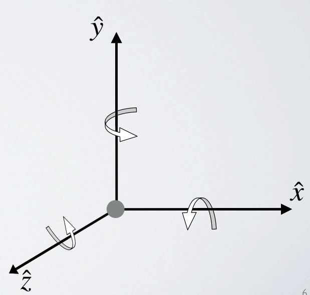


Axis-aligned 3D Rotations

$$\mathbf{R}_{\hat{x}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_{\hat{y}} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_{\hat{z}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



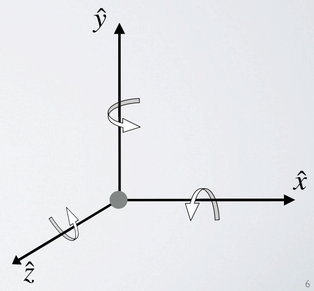
Axis-aligned 3D Rotations

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

"Z is in your face"



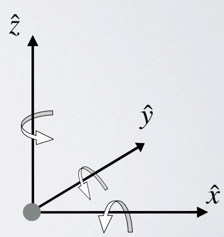
Axis-aligned 3D Rotations

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Also right handed "Zup"



Axis-aligned 3D Rotations

- Also known as "direction-cosine" matrices

$$\mathbf{R}_{\hat{x}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad \mathbf{R}_{\hat{y}} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_{\hat{z}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Arbitrary Rotations

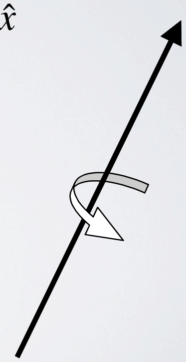
- Can be built from axis-aligned matrices:

$$\mathbf{R} = \mathbf{R}_{\hat{z}} \cdot \mathbf{R}_{\hat{y}} \cdot \mathbf{R}_{\hat{x}}$$

- Result due to Euler... hence called Euler Angles
- Easy to store in vector

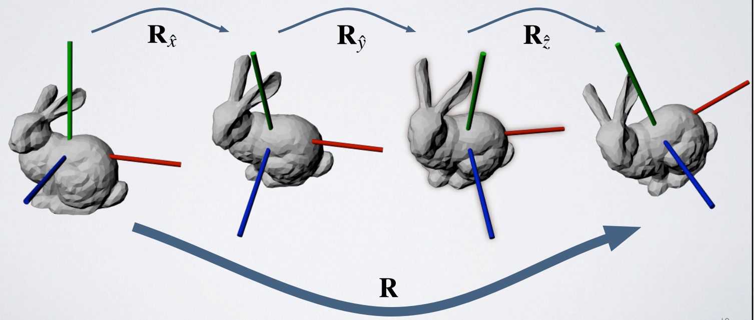
- But NOT a vector.

$$\mathbf{R} = \mathrm{rot}(x, y, z)$$



Arbitrary Rotations

$$\mathbf{R} = \mathbf{R}_{\hat{z}} \cdot \mathbf{R}_{\hat{y}} \cdot \mathbf{R}_{\hat{x}}$$



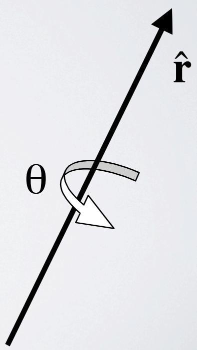
Arbitrary Rotations

- Allows tumbling
- Euler angles are non-unique
- Gimbal-lock
- Moving -vs- fixed axes
  - Reverse of each other

Exponential Maps

- Direct representation of arbitrary rotation
- AKA: axis-angle, angular displacement vector
- Rotate θ degrees about some axis
- Encode θ by length of vector

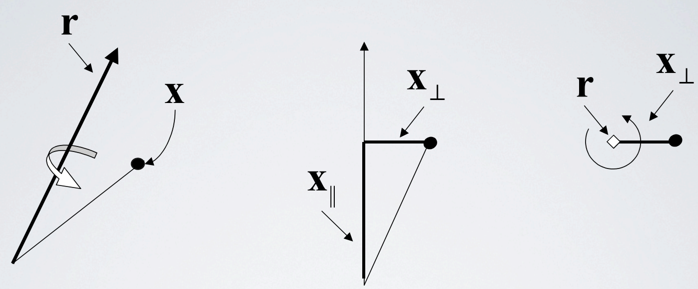
$$\theta = |\mathbf{r}|$$



Exponential Maps

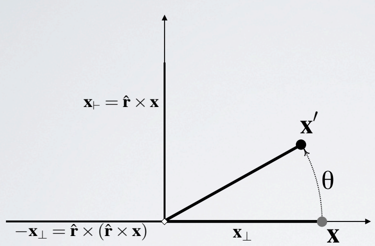
- Given vector $\mathbf{r}$, how to get matrix $\mathbf{R}$
- Method from text:
  1. rotate about x axis to put $\mathbf{r}$ into the x-y plane
  2. rotate about z axis align $\mathbf{r}$ with the x axis
  3. rotate θ degrees about x axis
  4. undo #2 and then #1
  5. composite together

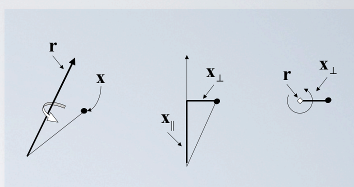
Exponential Maps



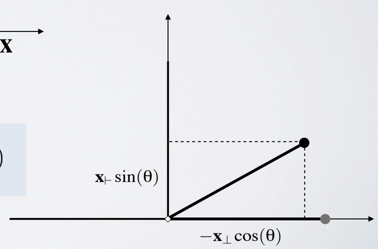
- Vector expressing a point has two parts
  - $\mathbf{x}_\parallel$ does not change
  - $\mathbf{x}_\perp$ rotates like a 2D point

Exponential Maps





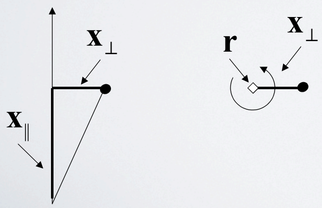
$$\mathbf{x}' = \mathbf{x}_{||} + \mathbf{x}_{\vdash}\sin(\theta) + \mathbf{x}_\perp\cos(\theta)$$



Exponential Maps

- Rodriguez Formula

$$\begin{aligned}\mathbf{x}' = {} & \hat{\mathbf{r}}(\hat{\mathbf{r}}\cdot\mathbf{x}) \\ & +\sin(\theta)(\hat{\mathbf{r}}\times\mathbf{x}) \\ & -\cos(\theta)(\hat{\mathbf{r}}\times(\hat{\mathbf{r}}\times\mathbf{x}))\end{aligned}$$



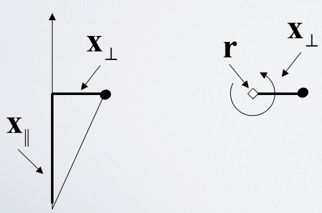
Actually a minor variation ...

Exponential Maps

- Rodriguez Formula

$$\begin{aligned}\mathbf{x}' = {} & \hat{\mathbf{r}}(\hat{\mathbf{r}}\cdot\mathbf{x}) \\ & +\sin(\theta)(\hat{\mathbf{r}}\times\mathbf{x}) \\ & -\cos(\theta)(\hat{\mathbf{r}}\times(\hat{\mathbf{r}}\times\mathbf{x}))\end{aligned}$$

Linear in $\mathbf{x}$



Actually a minor variation ...

Exponential Maps

- Building the matrix

$$\mathbf{x}' = \left((\hat{\mathbf{r}}\hat{\mathbf{r}}^{\mathrm{t}}) + \sin(\theta)(\hat{\mathbf{r}}\times) - \cos(\theta)(\hat{\mathbf{r}}\times)(\hat{\mathbf{r}}\times) \right)\mathbf{x}$$

$$(\hat{\mathbf{r}}\times) = \begin{bmatrix} 0 & -\hat{r}_z & \hat{r}_y \\ \hat{r}_z & 0 & -\hat{r}_x \\ -\hat{r}_y & \hat{r}_x & 0 \end{bmatrix}$$

Antisymmetric matrix

$(\mathbf{a}\times)\mathbf{b} = \mathbf{a}\times\mathbf{b}$

Easy to verify by expansion

Exponential Maps

- Allows tumbling
- No gimbal-lock!
- Orientations are space within π-radius ball
- Nearly unique representation
- Singularities on shells at 2π
- Nice for interpolation

Exponential Maps

- Why exponential?
- Recall series expansion of e^x

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Exponential Maps

- Why exponential?
- Recall series expansion of e^x
- Euler: what happens if you put in $i\theta$ for x

$$e^{i\theta} = 1 + \frac{i\theta}{1!} + \frac{-\theta^2}{2!} + \frac{-i\theta^3}{3!} + \frac{\theta^4}{4!} + \cdots$$

$$= \left(1 + \frac{-\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots\right) + i\left(\frac{\theta}{1!} + \frac{-\theta^3}{3!} + \cdots\right)$$

$$= \cos(\theta) + i\sin(\theta)$$

Exponential Maps

- Why exponential?

$$e^{(\hat{\mathbf{r}}\times)\theta} = \mathbf{I} + \frac{(\hat{\mathbf{r}}\times)\theta}{1!} + \frac{(\hat{\mathbf{r}}\times)^2\theta^2}{2!} + \frac{(\hat{\mathbf{r}}\times)^3\theta^3}{3!} + \frac{(\hat{\mathbf{r}}\times)^4\theta^4}{4!} + \cdots$$

But notice that: $(\hat{\mathbf{r}}\times)^3 = -(\hat{\mathbf{r}}\times)$

$$e^{(\hat{\mathbf{r}}\times)\theta} = \mathbf{I} + \frac{(\hat{\mathbf{r}}\times)\theta}{1!} + \frac{(\hat{\mathbf{r}}\times)^2\theta^2}{2!} + \frac{-(\hat{\mathbf{r}}\times)\theta^3}{3!} + \frac{-(\hat{\mathbf{r}}\times)^2\theta^4}{4!} + \cdots$$

Exponential Maps

$$e^{(\hat{\mathbf{r}}\times)\theta} = \mathbf{I} + \frac{(\hat{\mathbf{r}}\times)\theta}{1!} + \frac{(\hat{\mathbf{r}}\times)^2\theta^2}{2!} + \frac{-(\hat{\mathbf{r}}\times)\theta^3}{3!} + \frac{-(\hat{\mathbf{r}}\times)^2\theta^4}{4!} + \cdots$$

$$e^{(\hat{\mathbf{r}}\times)\theta} = (\hat{\mathbf{r}}\times)\left(\frac{\theta}{1!} - \frac{\theta^3}{3!} + \cdots\right) + \mathbf{I} + (\hat{\mathbf{r}}\times)^2\left(+\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \cdots\right)$$

$$e^{(\hat{\mathbf{r}}\times)\theta} = (\hat{\mathbf{r}}\times)\sin(\theta) + \mathbf{I} + (\hat{\mathbf{r}}\times)^2(1-\cos(\theta))$$

Quaternions

- More popular than exponential maps
- Natural extension of $e^{i\theta} = \cos(\theta) + i\sin(\theta)$
- Due to Hamilton (1843)
  - Interesting history
  - Involves "hermaphroditic monsters"

Quaternions

- Uber-Complex Numbers

$$\mathrm{q} = (z_1, z_2, z_3, s) = (\mathbf{z}, s)$$

$$\mathrm{q} = iz_1 + jz_2 + kz_3 + s$$

$$i^2 = j^2 = k^2 = -1 \qquad \begin{array}{ll} ij = k & ji = -k \\ jk = i & kj = -i \\ ki = j & ik = -j \end{array}$$

Quaternions

- Multiplication natural consequence of defn.

$$\mathsf{q} \cdot \mathsf{p} = (\mathbf{z}_q s_p + \mathbf{z}_p s_q + \mathbf{z}_p \times \mathbf{z}_q \ , \ s_p s_q - \mathbf{z}_p \cdot \mathbf{z}_q)$$

- Conjugate

$$\mathsf{q}^* = (-\mathbf{z}, s)$$

- Magnitude

$$||\mathsf{q}||^2 = \mathbf{z} \cdot \mathbf{z} + s^2 = \mathsf{q} \cdot \mathsf{q}^*$$

Quaternions

- Vectors as quaternions

$$\mathsf{v} = (\mathbf{v}, 0)$$

- Rotations as quaternions

$$\mathsf{r} = (\hat{\mathbf{r}} \sin\frac{\theta}{2}, \cos\frac{\theta}{2})$$

- Rotating a vector

$$\mathsf{x}' = \mathsf{r} \cdot \mathsf{x} \cdot \mathsf{r}^*$$

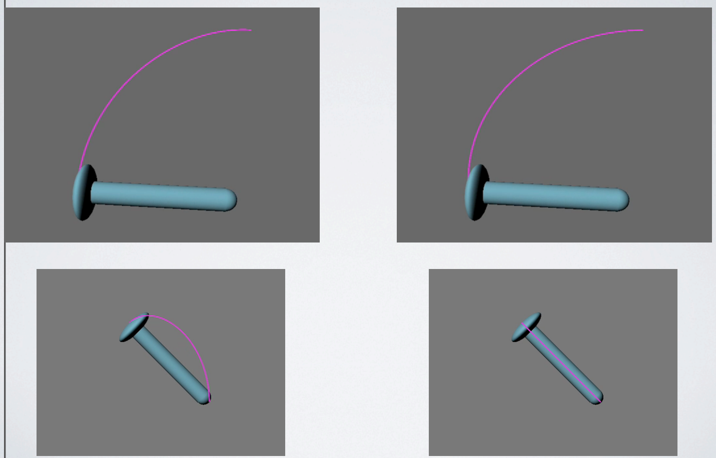
- Composing rotations

$$\mathsf{r} = \mathsf{r}_1 \cdot \mathsf{r}_2$$ ← Compare to Exp. Map

Quaternions

- No tumbling
- No gimbal-lock
- Orientations are "double unique"
- Surface of a 3-sphere in 4D $||\mathrm{r}|| = 1$
- Nice for interpolation

Interpolation



Rotation Matrices

- Eigen system
  - One real eigenvalue
  - Real axis is axis of rotation
  - Imaginary values are 2D rotation as complex number
- Logarithmic formula

$$(\hat{\mathbf{r}}\times) = \ln(\mathbf{R}) = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^{\mathsf{T}})$$

$$\theta = \cos^{-1}\left(\frac{\mathrm{Tr}(\mathbf{R}) - 1}{2}\right)$$

Similar formulae as for exponential...

Rotation Matrices

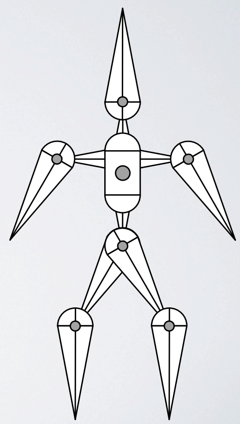
- Consider:

$$\mathbf{RI} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Columns are coordinate axes after transformation (true for general matrices)
- Rows are original axes in original system (not true for general matrices)

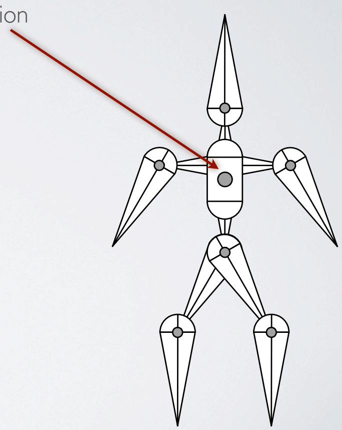
Forward Kinematics

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



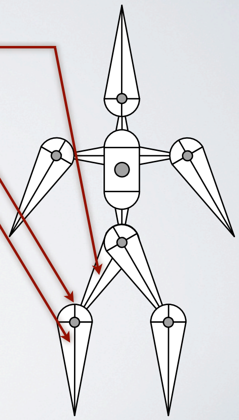
Forward Kinematics

- 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

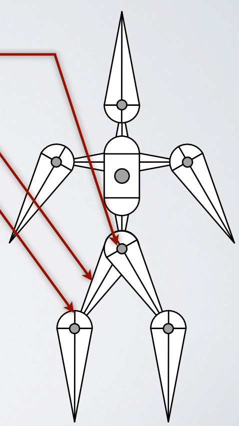
- A joint
  - Joint's inboard body
  - Joint's outboard body



33

Forward Kinematics

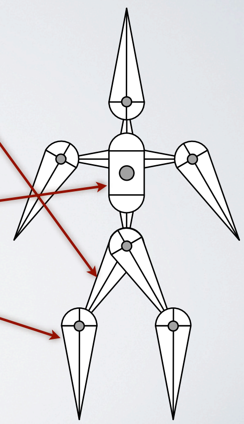
- A body
  - Body's inboard joint
  - Body's outboard joint
    - May have several outboard joints



34

Forward Kinematics

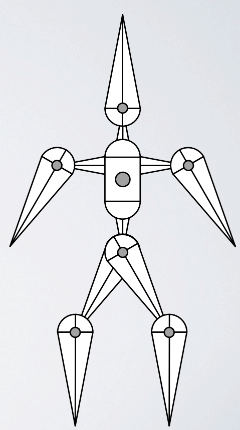
- A body
  - Body's inboard joint
  - Body's outboard joint
    - May have several outboard joints
  - Body's parent
  - Body's child
    - May have several children



35

Forward Kinematics

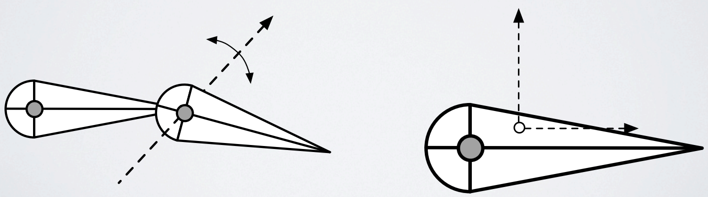
- Interior joints
  - Typically not 6 DOF joints
  - Pin - rotate about one axis
  - Ball - arbitrary rotation
  - Prism - translation along one axis



36

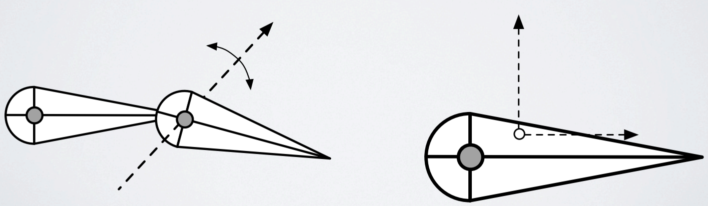
Forward Kinematics

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



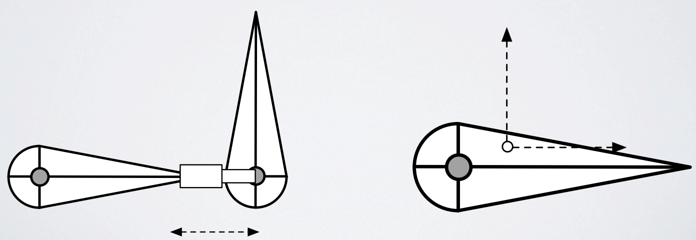
Forward Kinematics

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



Forward Kinematics

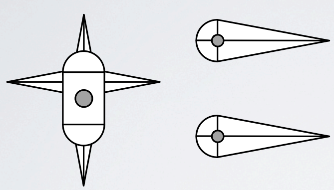
- Prismatic Joints
  - Translate inboard joint to local origin
  - Translate along axis
  - Translate origin to location of joint on outboard body



39

Forward Kinematics

- Composite transformations up the hierarchy

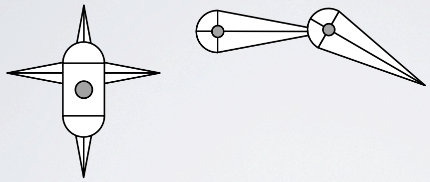


40

Thursday, November 12, 2009

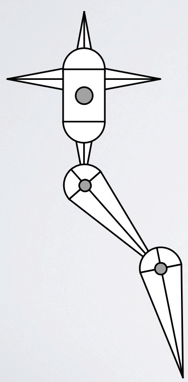
Forward Kinematics

- Composite transformations up the hierarchy



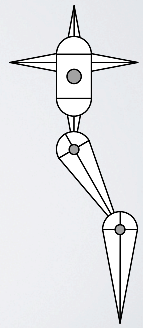
Forward Kinematics

- Composite transformations up the hierarchy



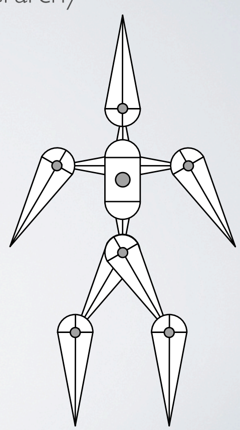
Forward Kinematics

- Composite transformations up the hierarchy



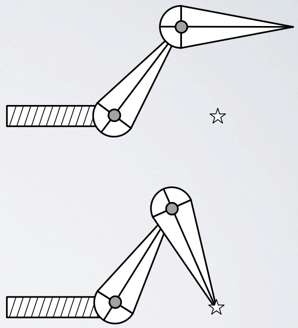
Forward Kinematics

- Composite transformations up the hierarchy



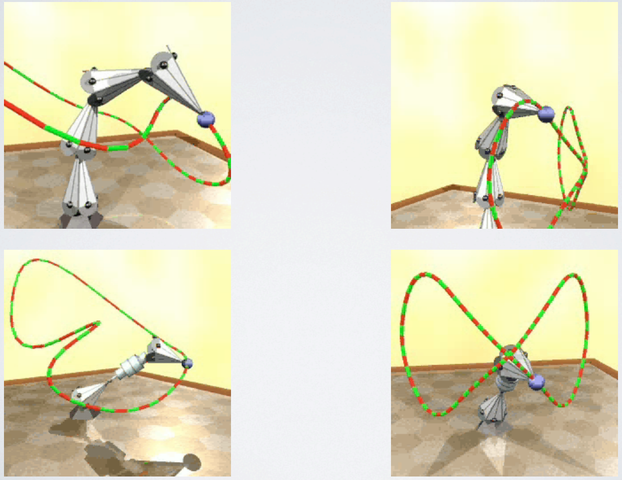
Inverse Kinematics

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



45

Inverse Kinematics



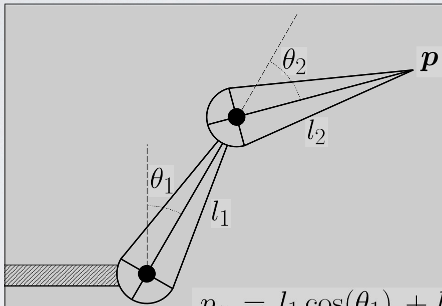
Egon Pasztor

46

Thursday, November 12, 2009

Inverse Kinematics

- A simple two segment arm in 2D



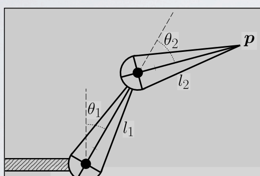
$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

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

Warning: Z-up Coordinate System

Inverse Kinematics

- Direct IK: solve for the parameters

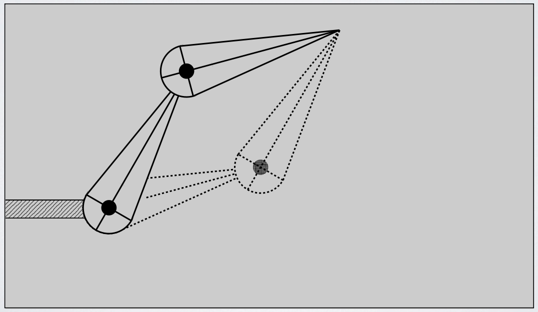


$$\theta_2 = \cos^{-1}\left(\frac{p_z^2 + p_x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

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

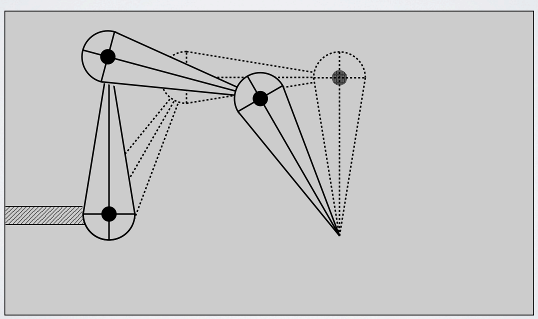
Inverse Kinematics

- Why is the problem hard?
  - Multiple solutions separated in configuration space



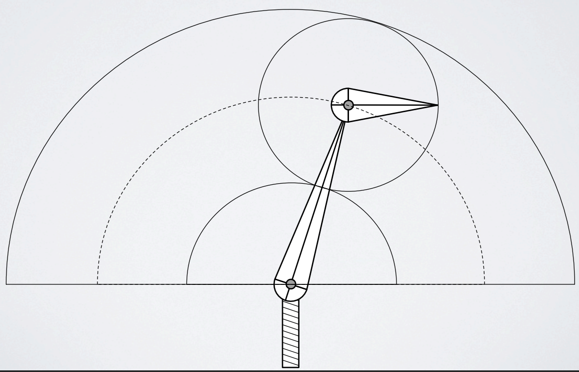
Inverse Kinematics

- Why is the problem hard?
  - Multiple solutions connected in configuration space



Inverse Kinematics

- Why is the problem hard?
  - Solutions may not always exist

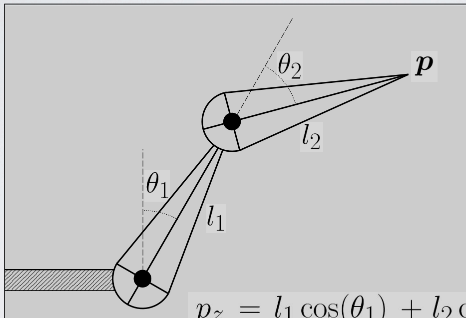


Inverse Kinematics

- 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:



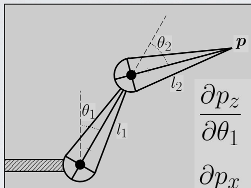
$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

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

Warning: Z-up Coordinate System

Inverse Kinematics

- We can write of the derivatives



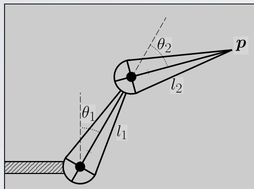
$$\frac{\partial p_z}{\partial \theta_1} = -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2)$$

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

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

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

Inverse Kinematics



Direction in Config. Space

$$\theta_1 = c_1\theta_*$$

$$\theta_2 = c_2\theta_*$$

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

Inverse Kinematics

The Jacobian (of p w.r.t. θ)

$$J_{ij} = \frac{\partial p_i}{\partial \theta_j}$$

Example for two segment arm

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

Inverse Kinematics

The Jacobian (of p w.r.t. θ)

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

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

Inverse Kinematics

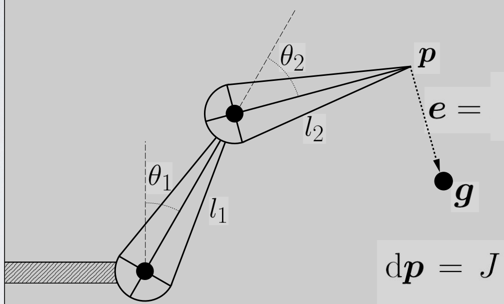
Solving for c_1 and c_2

$$\boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \qquad \mathrm{d}\boldsymbol{p} = \begin{bmatrix} \mathrm{d}p_z \\ \mathrm{d}p_x \end{bmatrix}$$

$$\begin{aligned} \mathrm{d}\boldsymbol{p} &= J \cdot \boldsymbol{c} \\ \boldsymbol{c} &= J^{-1} \cdot \mathrm{d}\boldsymbol{p} \end{aligned}$$

Inverse Kinematics

Solving for c_1 and c_2



$$\boldsymbol{e} = \mathrm{d}\boldsymbol{p}$$

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

$$\boldsymbol{c} = J^{-1} \cdot \mathrm{d}\boldsymbol{p}$$

Is the Jacobian invertible?

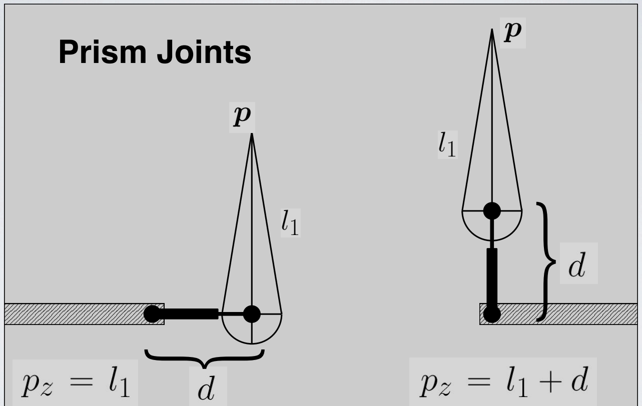
Inverse Kinematics

- Problems
  - Jacobian may (will!) not always be invertible
    - Use pseudo inverse (SVD)
    - Robust iterative method
  - Jacobian is not constant
    - Nonlinear ly) well behaved

$$J = \begin{bmatrix} \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} \\ \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \end{bmatrix} = J(\theta)$$

Inverse Kinematics

Prism Joints



$p_z = l_1$

$p_x = d$

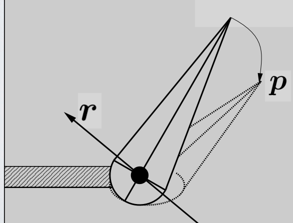
$p_z = l_1 + d$

$p_x = 0$

Inverse Kinematics

Ball Joints

$$\begin{aligned}\boldsymbol{p} &= \hat{\boldsymbol{r}}(\hat{\boldsymbol{r}} \cdot \boldsymbol{x}) \\ &+ \sin(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times \boldsymbol{x}) \\ &- \cos(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times (\hat{\boldsymbol{r}} \times \boldsymbol{x}))\end{aligned}$$



Inverse Kinematics

Ball Joints (moving axis)

$$\mathrm{d}\boldsymbol{p} = [\mathrm{d}\boldsymbol{r}]\cdot e^{[\boldsymbol{r}]}\cdot\boldsymbol{x} = [\mathrm{d}\boldsymbol{r}]\cdot\boldsymbol{p} = -\underbrace{[\boldsymbol{p}]}_{\text{That is the Jacobian for this joint}}\cdot\mathrm{d}\boldsymbol{r}$$

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

$$[\boldsymbol{r}]\cdot\boldsymbol{x} = \boldsymbol{r}\times\boldsymbol{x}$$

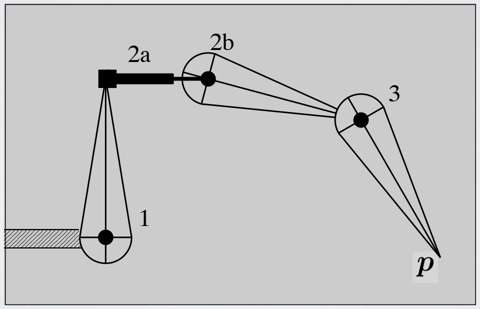
Inverse Kinematics

Ball Joints (fixed axis)

$$\mathrm{d}\boldsymbol{p} = (\mathrm{d}\theta)[\hat{\boldsymbol{r}}]\cdot\boldsymbol{x} = -\underbrace{[\boldsymbol{x}]\cdot\hat{\boldsymbol{r}}}_{\text{That is the Jacobian for this joint}}\mathrm{d}\theta$$

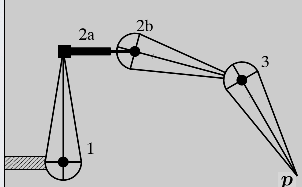
Inverse Kinematics

- Many links / joints
  - Need a generic method for building Jacobian



Inverse Kinematics

- Can't just concatenate individual matrices



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

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

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

Inverse Kinematics

Transformation from body to world

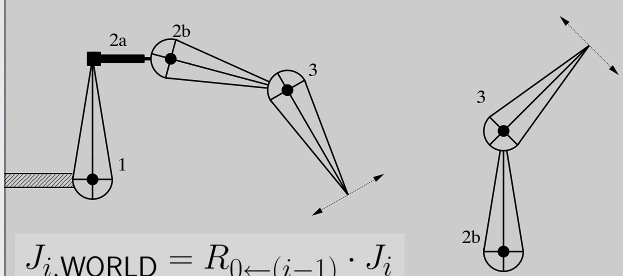
$$X_{0\leftarrow i} = \prod_{j=1}^{i} X_{(j-1)\leftarrow j} = X_{0\leftarrow 1} \cdot X_{1\leftarrow 2} \cdots$$

Rotation from body to world

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

Inverse Kinematics

Need to transform Jacobians to common coordinate system (WORLD)



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

Inverse Kinematics

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

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

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

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

A Cheap Alternative

- Estimate Jacobian (or parts of it) using finite differences
- Cyclic Coordinate Descent
  - Solve for each DOF one at a time
  - Iterate till good enough / run out of time

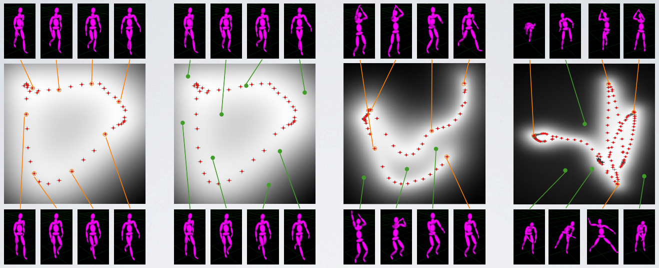
Inverse Kinematics

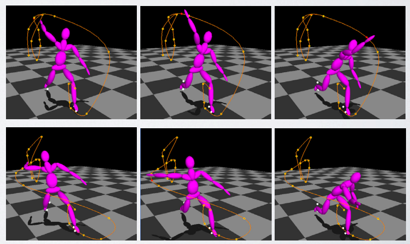
- More complex systems
  - More complex joints (prism and ball)
  - More links
  - Other criteria (COM or height)
  - Hard constraints (*eg* foot plants)
  - Unilateral constraints (*eg* joint limits)
  - Multiple criteria and multiple chains
- Smoothness over time
  - DOF are determined by control points of a curve (chain rule)

Inverse Kinematics

- Some 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