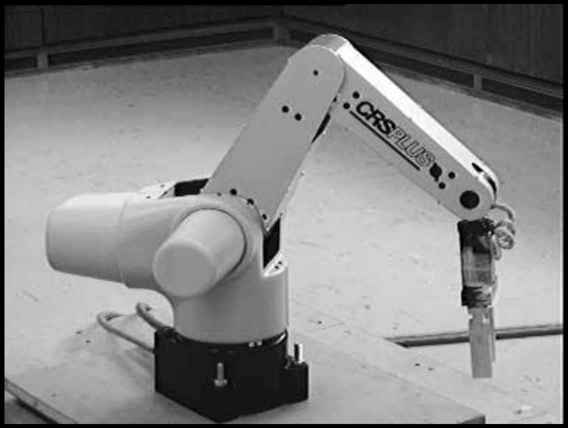
Kinematics

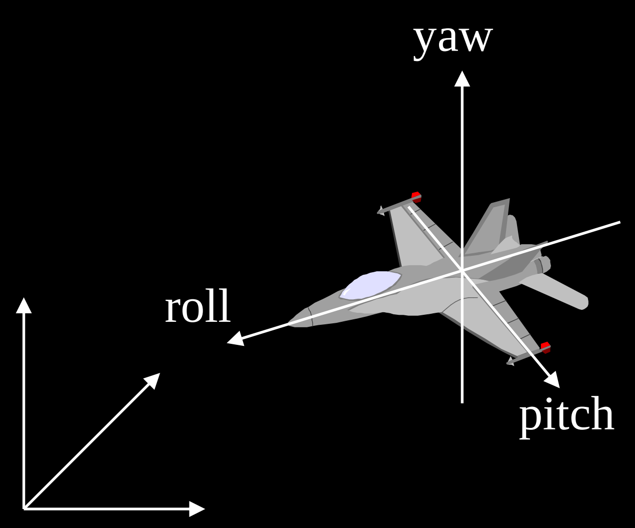
Kinematics

- The branch of mechanics concerned with the motions of objects without regard to the forces that cause the motion
- Why kinematics?
  - Hierarchical articulated model
  - Posing a character




Degrees of Freedom (DOF)

- The minimum number of coordinates required to specify completely the motion of an object



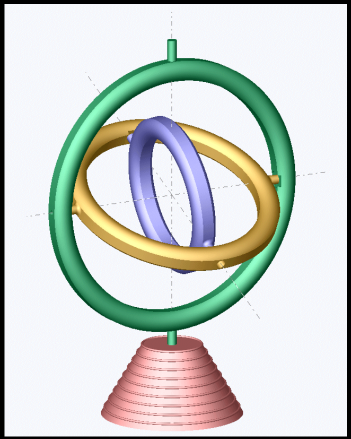
6 DOF: x, y, z, raw, pitch, yaw

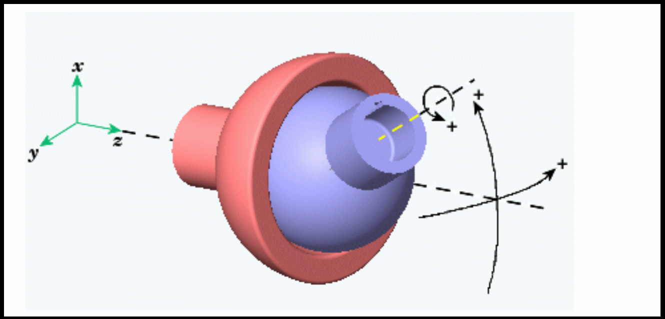
Degrees of Freedom in Human Model

- Root: 3 translational DOF + 3 rotational DOF
- Rotational joints are commonly used
- Each joint can have up to 3 DOF
  - Shoulder: 3 DOF
  - Wrist: 2 DOF
  - Knee: 1 DOF

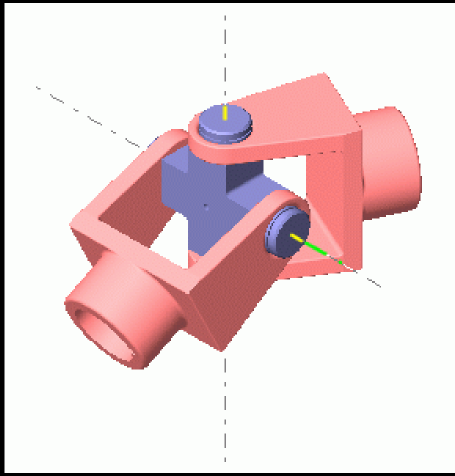
Revolute Joints

- 3 DOF joint
  - gimbal
  - ball and socket



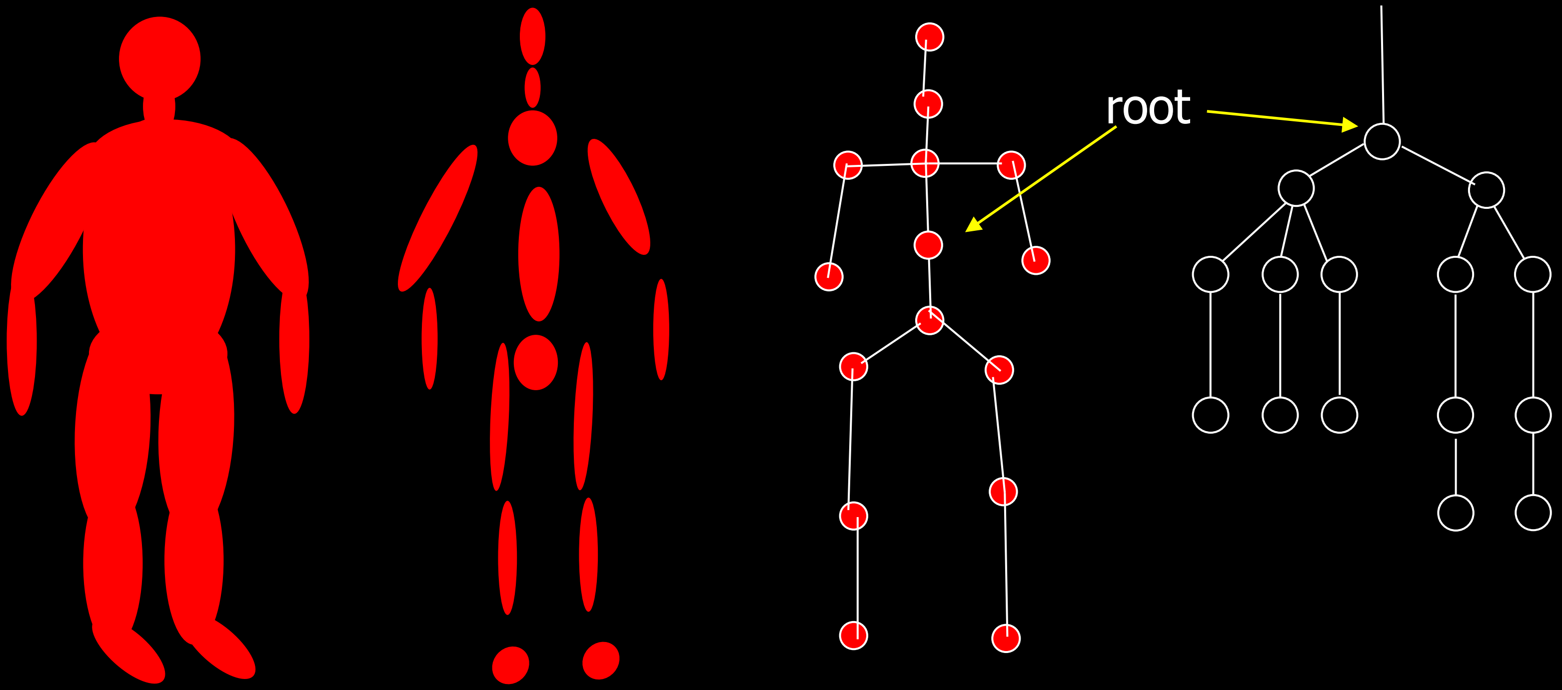


- 2 DOF joint
  - universal



Hierarchical Articulated Model

- Represent an articulated figure as a series of links connected by joints
- Enforce limb connectivity in a tree-like structure

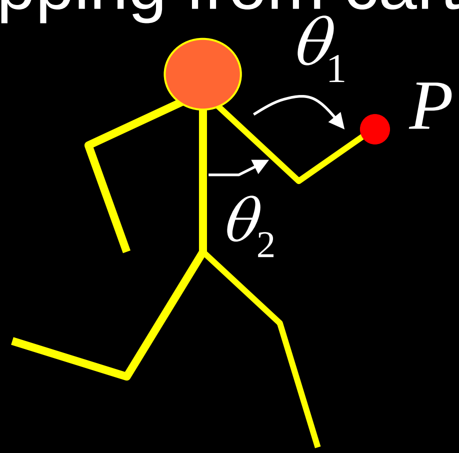


Joint Space vs. Cartesian Space

- Joint space
  - space formed by joint angles
  - position all joints—fine level control
- Cartesian space
  - 3D space
  - specify environment interactions

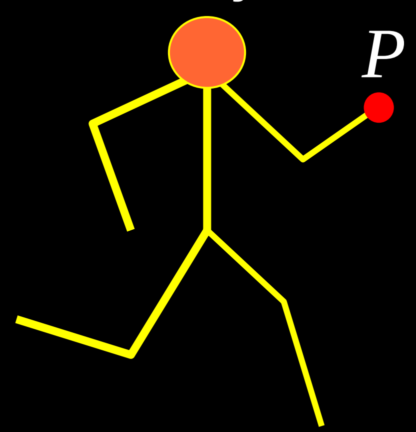
Forward and Inverse Kinematics

- Forward kinematics
  - mapping from joint space to cartesian space
- Inverse kinematics
  - mapping from cartesian space to joint space



Forward Kinematics

$$P = f(\theta_1, \theta_2)$$



Inverse Kinematics

$$\theta_1, \theta_2 = f^{-1}(P)$$

Forward and Inverse Kinematics (cont.)

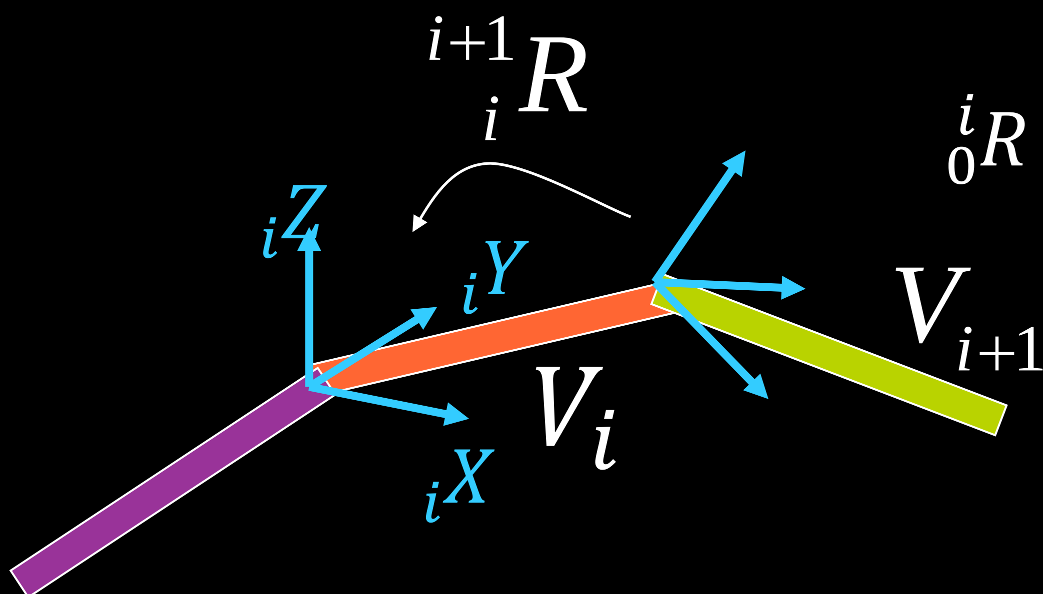
- Forward kinematics
  - rendering
- Inverse kinematics
  - good for specifying environment interaction
  - good for controlling a character—fewer parameters

Notations

- V_i : vector represented in coordinate frame i
- $_iT$: global position of the origin of coordinate frame i (global position of the *i*th joint)
- ^i_jR : rotation matrix that transforms a vector from coordinate frame *i* to coordinate frame *j*, i.e.,

$$V_j = {}^i_jR V_i$$

Forward Kinematics



$$ {}^{i+1}_{i}R = {}^{0}_{i}R \cdot {}^{i+1}_{0}R $$

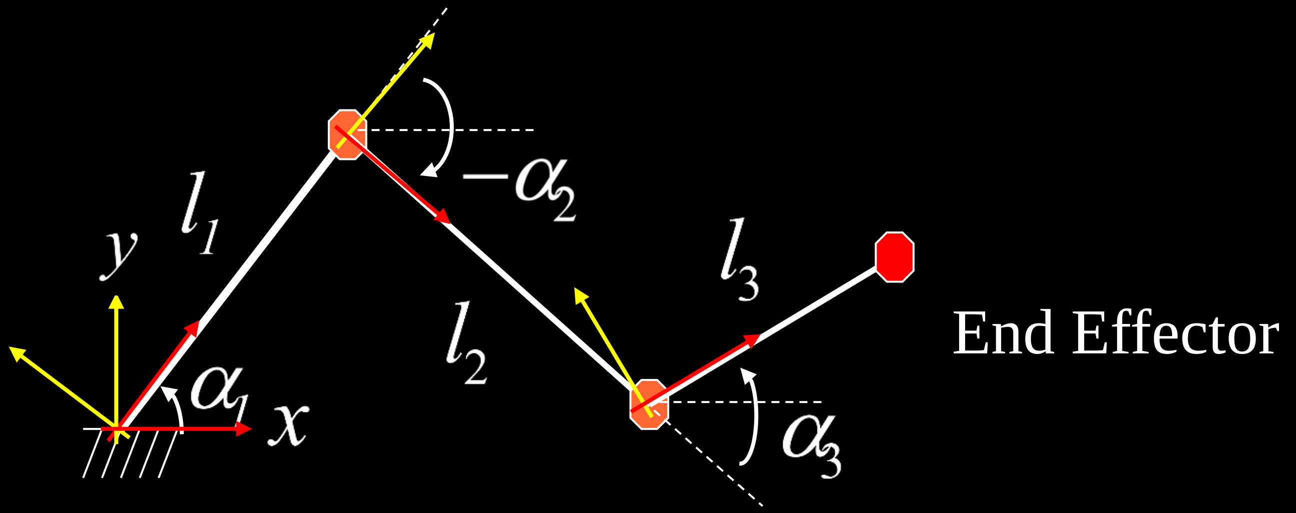
$$ {}^{i}_{0}R = \begin{bmatrix} {}_iX & {}_iY & {}_iZ \end{bmatrix} $$

$$ {}^{0}_{i}R = {}^{i}_{0}R^{-1} = {}^{i}_{0}R^{T} $$

$$ {}_{i+1}T = {}^{1}_{0}R\left\{{}^{2}_{1}R \cdots {}^{i-1}_{i-2}R\left({}^{i}_{i-1}RV_i + V_{i-1}\right) + V_{i-2}\right] \cdots \big] + V_1\right\} $$

$$ {}_{i+1}T = {}^{i}_{0}RV_i + {}_{i}T $$

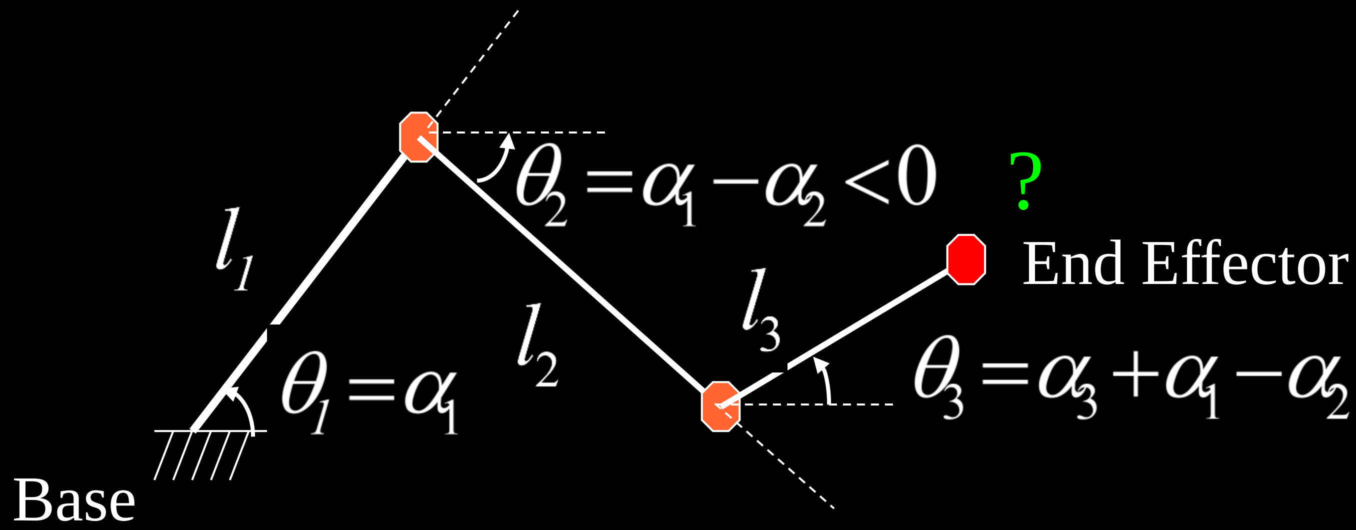
Forward Kinematics by Composing Transformations



$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = R_z(\alpha_1)\{R_z(-\alpha_2)[R_z(\alpha_3)T_x(l_3) + T_x(l_2)] + T_x(l_1)\}$$

Relative rotation from the local coordinate of parent link to the local coordinate of the child link

With simplified notations



$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_2) + l_3 \cos(\theta_3)$$

$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) + l_3 \sin(\theta_3)$$

Acclaim Format

- Skeletal animation file format including
  - .ASF: Skeleton file
  - .AMC: Motion file

- All information in ASF file is specified with respect to the global coordinate, while all information in AMC file is specified with respect to the local coordinate.

ASF File

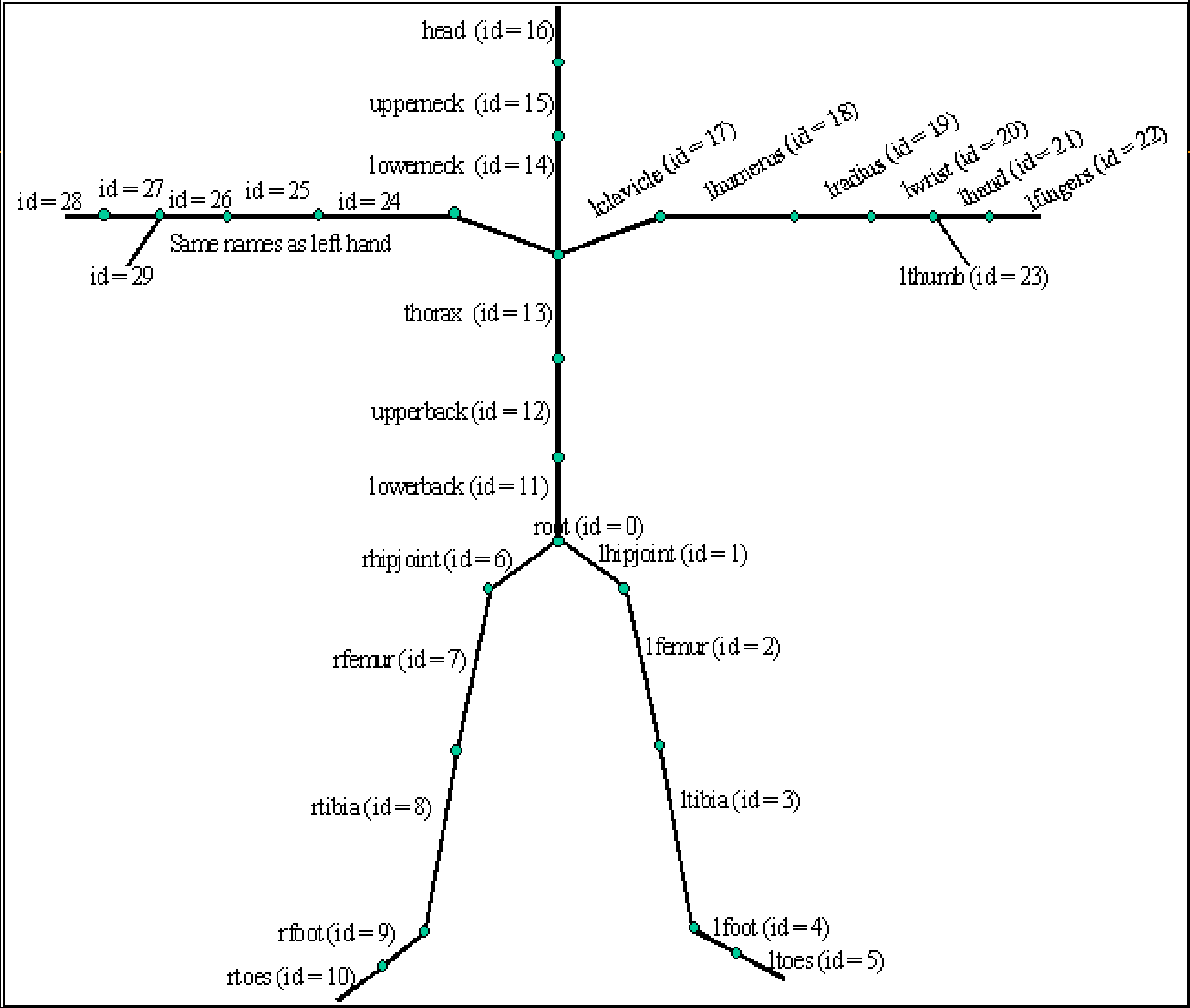
- Describes the local frame of each bone of a neutral (zero) pose in the global coordinate, e.g. Euler angle in xyz order,

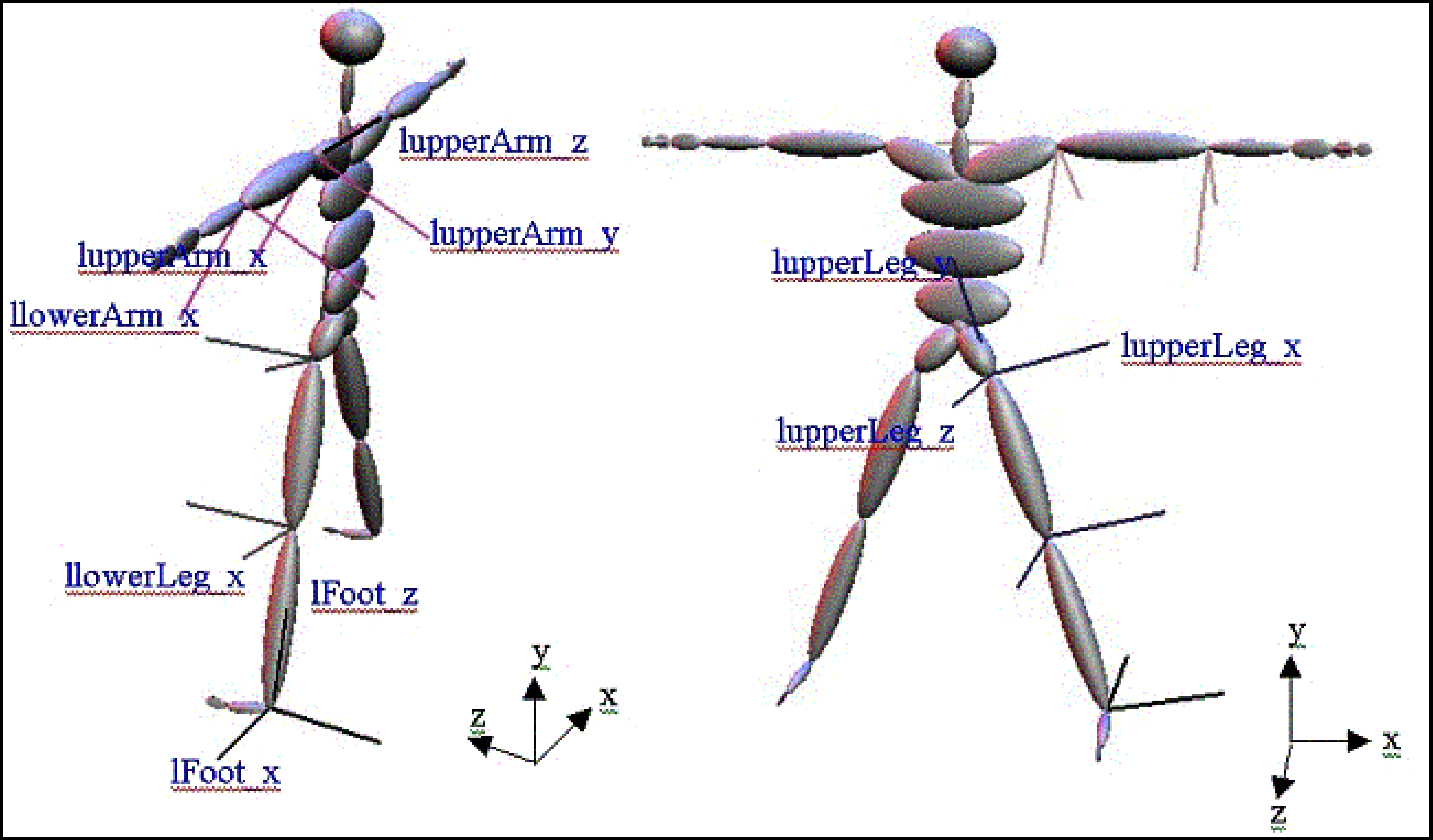
$$ {}_0^iR = {}_0^iR_z \cdot {}_0^iR_y \cdot {}_0^iR_x $$

- We need to compute the relative transformation and the unit-length bone direction vector in local coordinate

$$ {}_iR_{asf} = {}_i^{i+1}R = {}_i^0R \cdot {}_0^{i+1}R \qquad \hat{V}_i = {}_i^0R\hat{V}_0 $$

```
begin
   id 2
   name lfemur
   direction 0.342 -0.939 0
   length 7.113
   axis 0 0 20  XYZ
  dof rx ry rz
  limits (-160.0 20.0)
         (-70.0 70.0)
         (-60.0 70.0)
 end
```


head (id=16)
uppeneck (id=15)
loweneck (id=14)
lclavicle (id=17)
lhumerus (id=18)
lradius (id=19)
lwrist (id=20)
lhand (id=21)
lfingers (id=22)
lthumb (id=23)
id=28
id=27
id=26
id=25
id=24
id=29
Same names as left hand
thorax (id=13)
upperback (id=12)
lowerback (id=11)
root (id=0)
rhipjoint (id=6)
lhipjoint (id=1)
rfemur (id=7)
lfemur (id=2)
rtibia (id=8)
ltibia (id=3)
rfoot (id=9)
lfoot (id=4)
rtoes (id=10)
ltoes (id=5)


lupperArm_z
lupperArm_y
lupperArm_x
llowerArm_x
llowerLeg_x
lFoot_z
lFoot_x
y
z
x
lupperLeg_y
lupperLeg_x
lupperLeg_z
y
x
z

AMC File

$$ {}_iR_{amc} = {}_iR_z \cdot {}_iR_y \cdot {}_iR_x $$

```
#!OML:ASF F:\VICON\USERDATA\INSTALL\rory3\rory3.ASF
:FULLY-SPECIFIED
:DEGREES
1
root 3.1294 17.6906 0.576147 -69.7364 88.7134 -68.7451
lowerback 5.37529 -0.419929 3.55267
upperback -1.47894 -0.3644 -1.32457
thorax -4.58452 -0.299522 -3.33877
lowerneck -3.64552 -5.65816 -4.72229
upperneck 4.19034 -7.74441 9.40555
head 2.64463 -3.6745 3.67041
rclavicle 1.2921e-015 1.55052e-014
rhumerus -39.2113 -25.8219 -71.2854
rradius 20.028
rwrist 28.2698
rhand -0.838087 16.263
rfingers 7.12502
rthumb 24.7874 -12.1506
lclavicle 1.2921e-015 1.55052e-014
```

Forward Kinematics in ASF/AMC

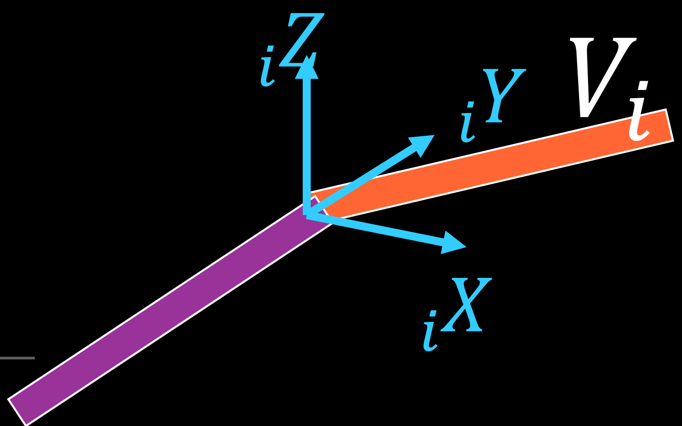
$$ {}^{i+1}_{i}R = {}_{i}R_{asf} \cdot {}_{i}R_{amc} $$

$$ {}^{i}_{0}R = {}^{1}_{0}R\,{}^{2}_{1}R \cdots {}^{i}_{i-1}R $$

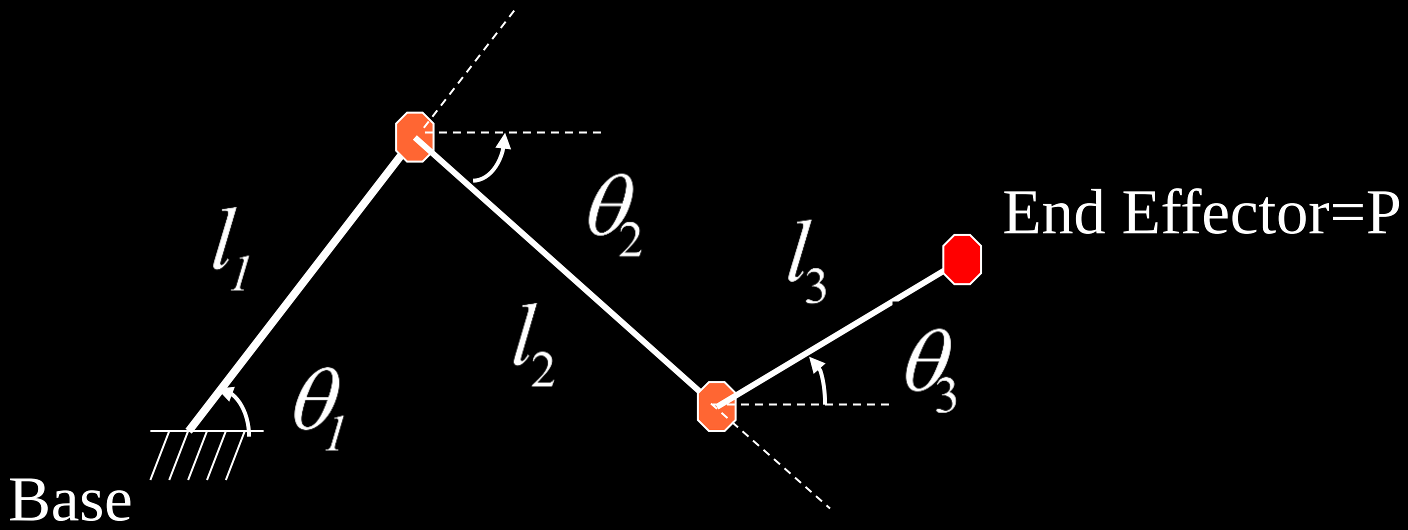
$$ V_i = \hat{V}_i \cdot l_i $$

$$ {}_{i}T = {}^{i-1}_{0}R V_{i-1} + {}_{i-1}T $$

```
begin
    id 2
    name lfemur
    direction 0.342 -0.939 0
    length 7.113
    axis 0 0 20  XYZ
   dof rx ry rz
   limits (-160.0 20.0)
         (-70.0 70.0)
         (-60.0 70.0)
end
```



Inverse Kinematics



$$\theta_1, \theta_2, \theta_3 = \mathrm{f}^{-1}(P)$$

Redundancy in IK

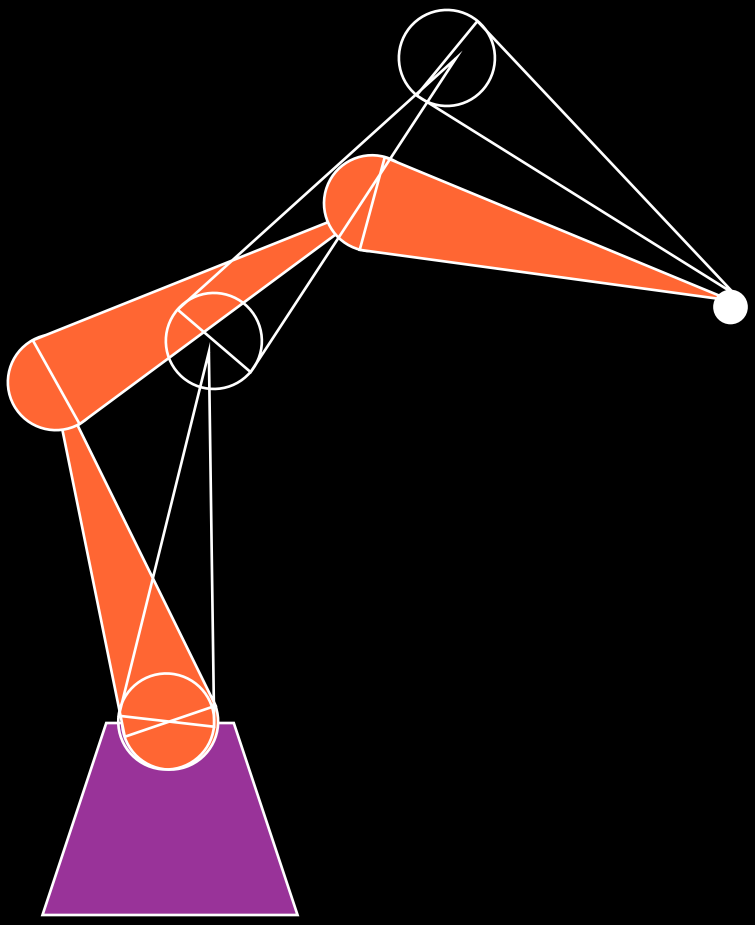
- Our example
  - 2 equations (constraints)
  - 3 unknowns

$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_2) + l_3 \cos(\theta_3)$$
$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) + l_3 \sin(\theta_3)$$

  - Multiple solutions exist!
- This is not uncommon!
  - see how you can move your elbow while keeping your finger touching your nose

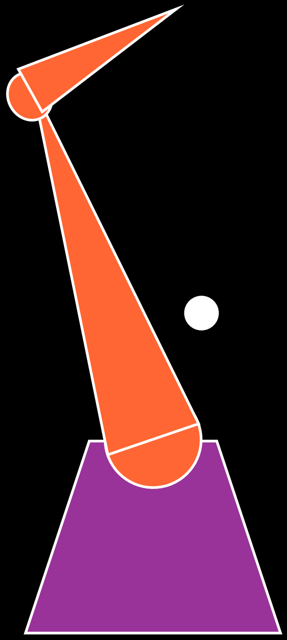
Other problems in IK

- Infinite solutions



Other problems in IK

- No solutions



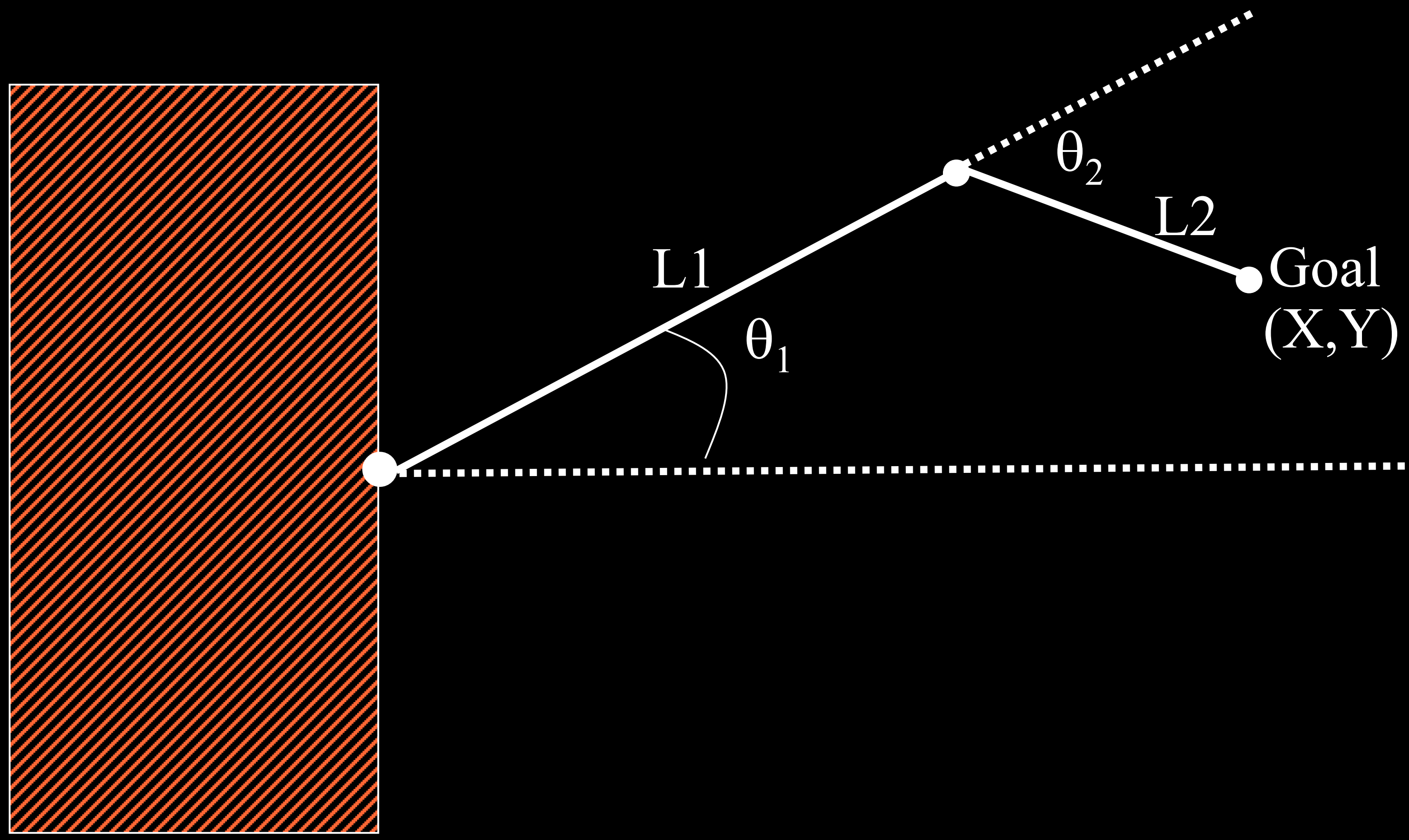
Why Is IK hard?

- Redundancy
- Natural motion
  - joint limits
  - minimum jerk
  - style?
- Singularities
  - ill-conditioned matrix
  - shown later

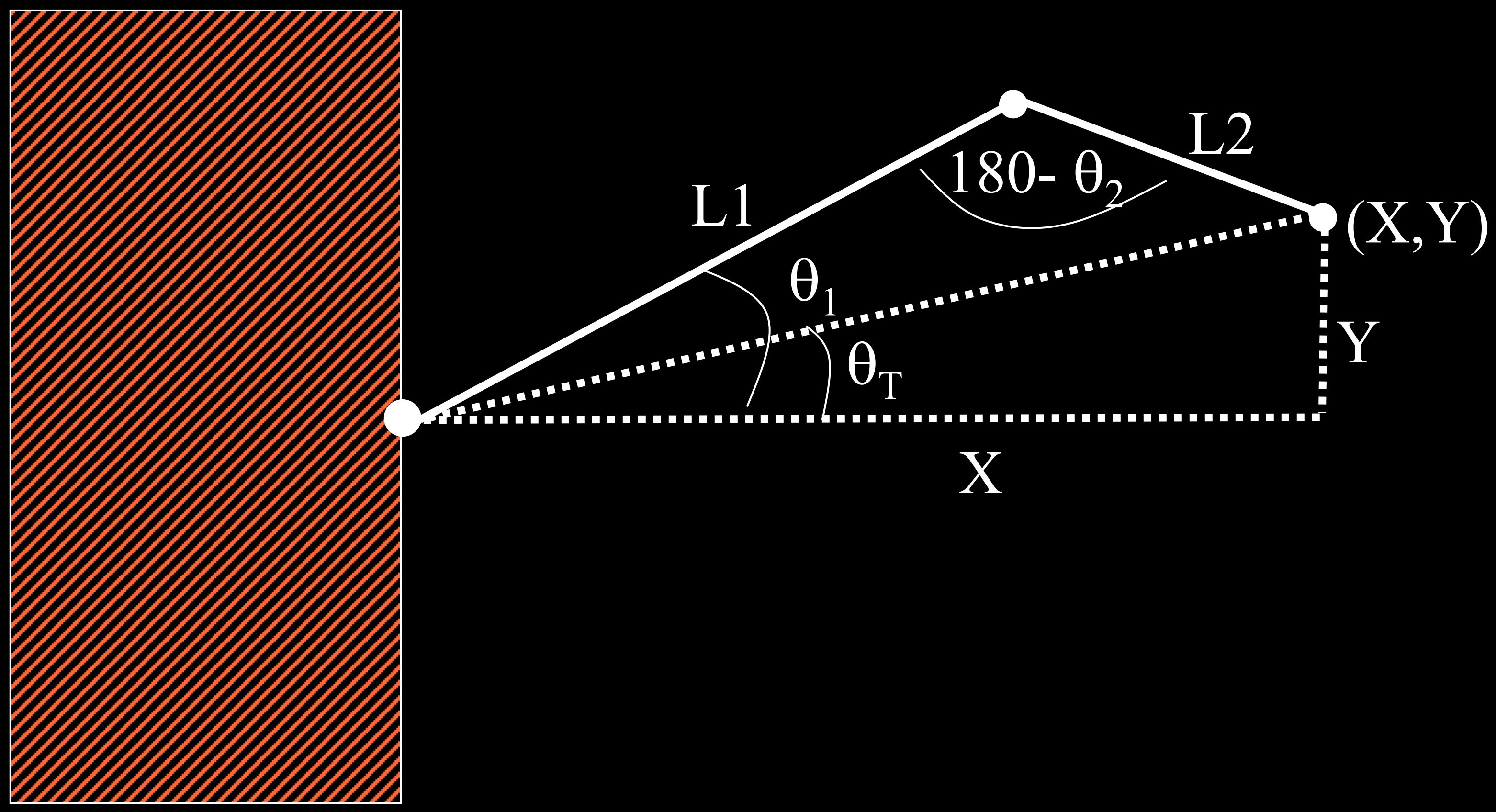
Solving Inverse Kinematics

- Analytic method
- Inverse-Jacobian method
- Optimization-based method
- Example-based method

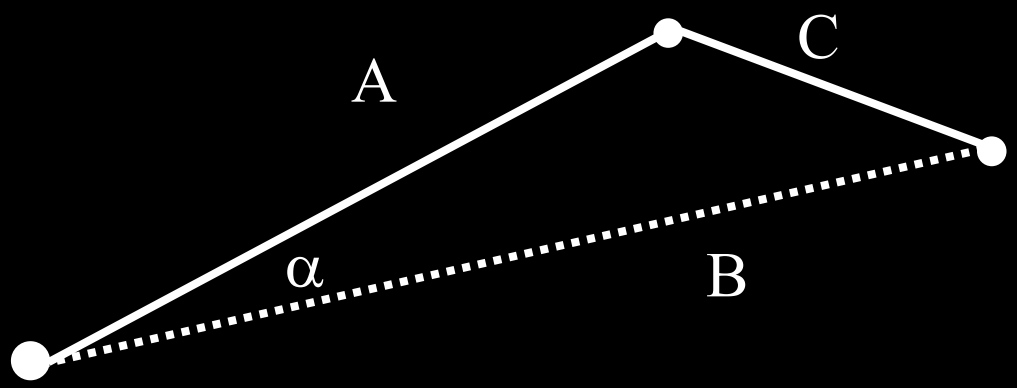
Analytic Method



Analytic Method (cont.)

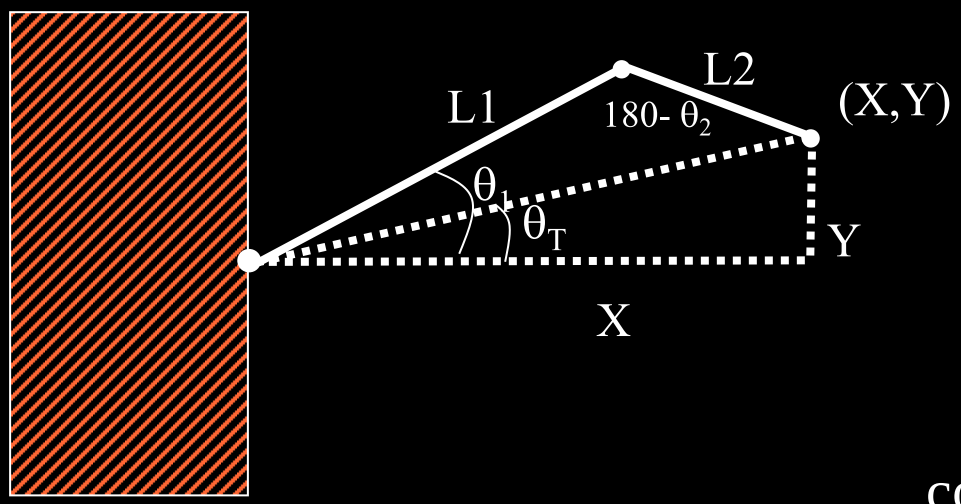


Cosine Law



$$\cos(\alpha) = \frac{A^2 + B^2 - C^2}{2AB}$$

Analytic Method



$$\cos(\theta_T) = \frac{X}{\sqrt{X^2 + Y^2}}$$

$$\theta_T = \cos^{-1}\left(\frac{X}{\sqrt{X^2 + Y^2}}\right)$$

$$\cos(\theta_1 - \theta_T) = \frac{L_1^{\,2} + X^2 + Y^2 - L_2^{\,2}}{2L_1\sqrt{X^2 + Y^2}}$$

$$\cos(180 - \theta_2) = \frac{L_1^{\,2} + L_2^{\,2} - \left(X^2 + Y^2\right)}{2L_1L_2}$$

$$\theta_1 = \cos^{-1}\left(\frac{L_1^{\,2} + X^2 + Y^2 - L_2^{\,2}}{2L_1\sqrt{X^2 + Y^2}}\right) + \theta_T$$

$$\theta_2 = 180 - \cos^{-1}\left(\frac{L_1^{\,2} + L_2^{\,2} - \left(X^2 + Y^2\right)}{2L_1L_2}\right)$$

Inverse-Jacobian method

- When linkage is complicated
- Iteratively change the joint angles to approach the goal position and orientation

Jacobian

$$f(\boldsymbol{\theta}) = \mathbf{p}$$

$$\mathbf{p} \in R^n \quad (n = 6 \text{ usually})$$

$$\boldsymbol{\theta} \in R^m (m = \text{DOFs})$$

- Jacobian is the n by m matrix relating differential changes of θ to differential changes of **p** (*d***p**)

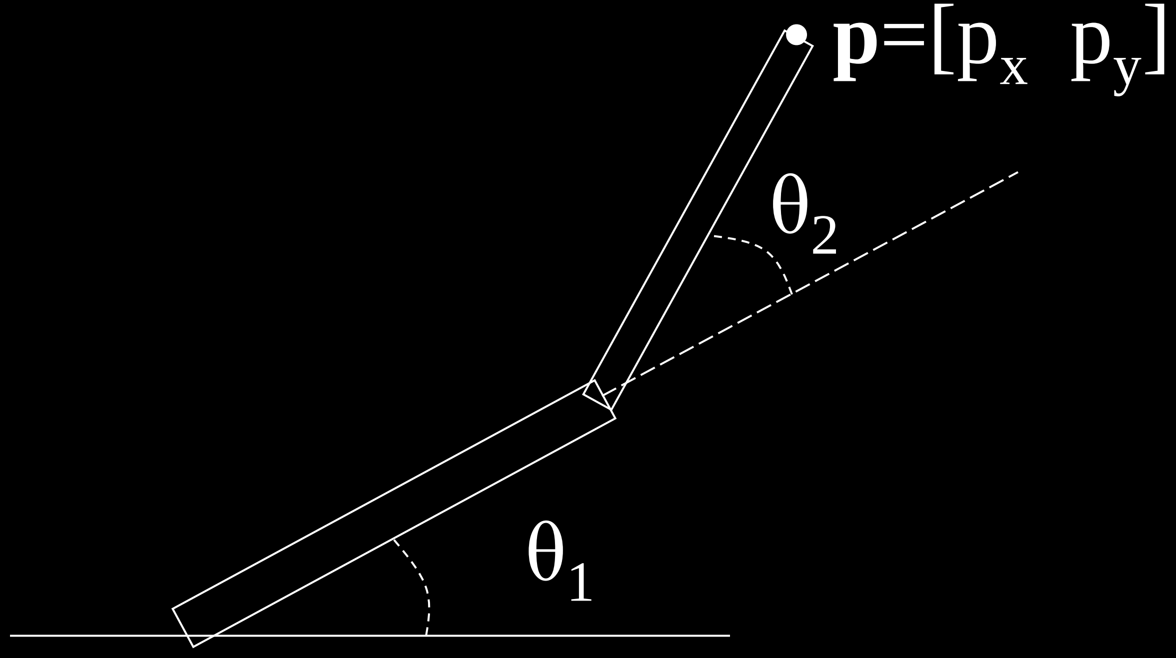
$$\frac{d\mathbf{p}}{dt} = \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{d\boldsymbol{\theta}}{dt} = J(\boldsymbol{\theta}) \frac{d\boldsymbol{\theta}}{dt} \qquad J_{ij} = \frac{\partial f_i}{\partial \theta_j}$$

- Jacobian maps velocities in joint space to velocities in cartesian space $J(\theta)\dot{\theta} = V$

Kinematic Interpretation of Jacobian

Example: Jacobian for a 2D arm

- Let’s say we have a simple 2D robot arm with two 1-DOF rotational joints:



Jacobian for a 2D arm

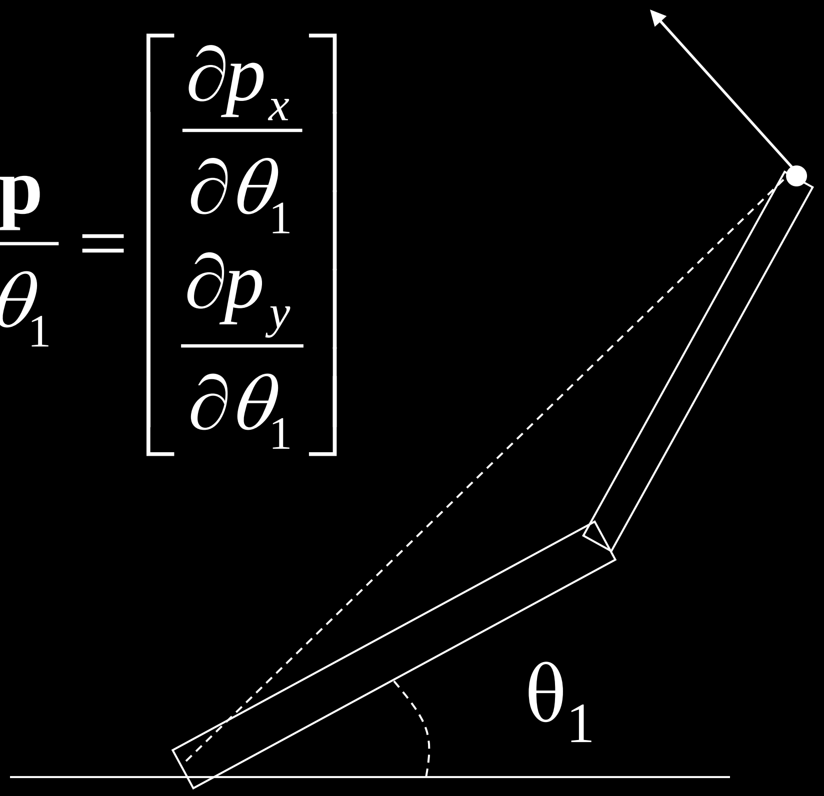
- The Jacobian matrix J(**θ**) shows how each component of **p** varies with respect to each joint angle

$$J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{p}}{\partial \theta_1} & \frac{\partial \mathbf{p}}{\partial \theta_2} \end{bmatrix}$$

Jacobian for a 2D arm

- Consider what would happen if we increased θ_1 by a small amount. What would happen to **p** ?

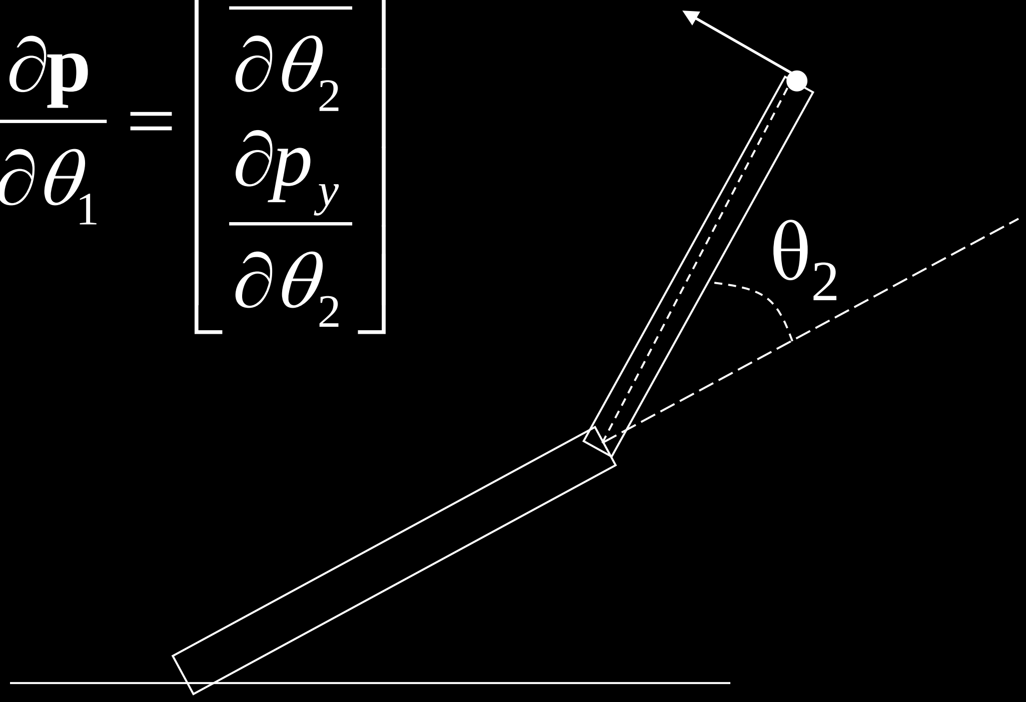
$$\frac{\partial \mathbf{p}}{\partial \theta_1} = \begin{bmatrix} \dfrac{\partial p_x}{\partial \theta_1} \\ \dfrac{\partial p_y}{\partial \theta_1} \end{bmatrix}$$



Jacobian for a 2D arm

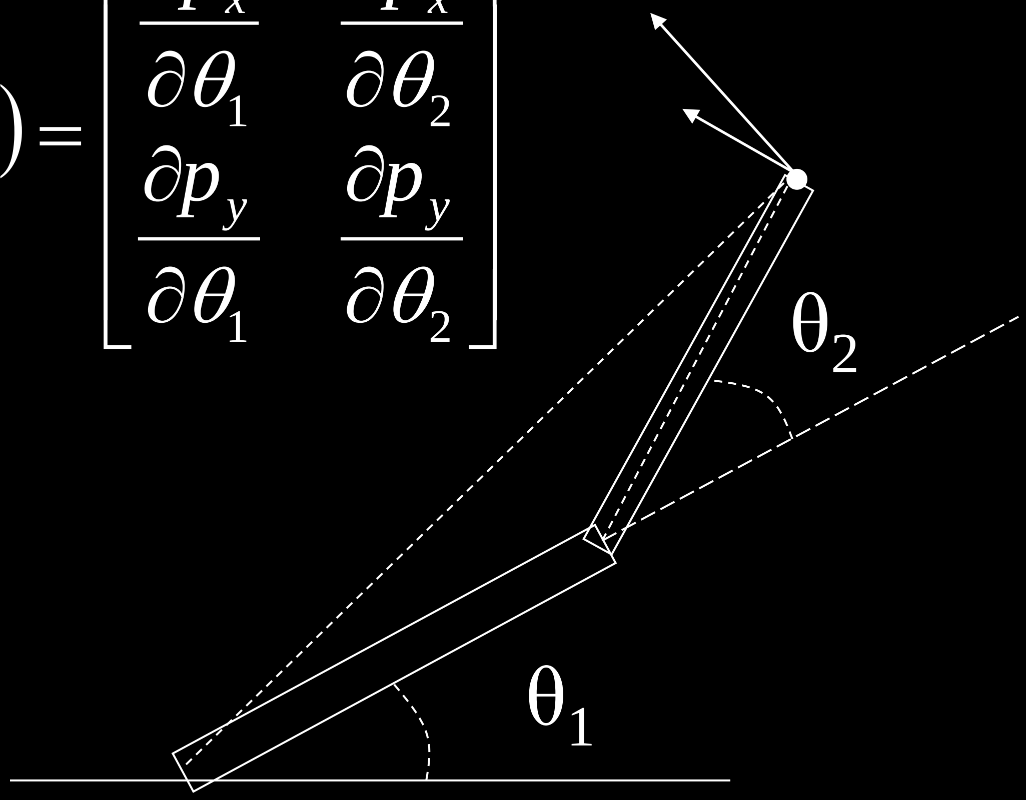
- What if we increased θ_2 by a small amount?

$$\frac{\partial \mathbf{p}}{\partial \theta_1} = \begin{bmatrix} \dfrac{\partial p_x}{\partial \theta_2} \\ \dfrac{\partial p_y}{\partial \theta_2} \end{bmatrix}$$



Jacobian for a 2D arm

$$J(\mathbf{p},\boldsymbol{\theta}) = \begin{bmatrix} \dfrac{\partial p_x}{\partial \theta_1} & \dfrac{\partial p_x}{\partial \theta_2} \\ \dfrac{\partial p_y}{\partial \theta_1} & \dfrac{\partial p_y}{\partial \theta_2} \end{bmatrix}$$



Computing Jacobian analytically

- A simple example:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{\theta}) \\ f_2(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos\theta_2 + l_3 \cos\theta_3 \\ l_1 \sin\theta_1 + l_2 \sin\theta_2 + l_3 \sin\theta_3 \end{bmatrix} \qquad \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_1(\boldsymbol{\theta})}{\partial \theta_1} & \dfrac{\partial f_1(\boldsymbol{\theta})}{\partial \theta_2} & \dfrac{\partial f_1(\boldsymbol{\theta})}{\partial \theta_3} \\ \dfrac{\partial f_2(\boldsymbol{\theta})}{\partial \theta_1} & \dfrac{\partial f_2(\boldsymbol{\theta})}{\partial \theta_2} & \dfrac{\partial f_2(\boldsymbol{\theta})}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} -l_1 \sin\theta_1 & -l_2 \sin\theta_2 & -l_3 \sin\theta_3 \\ l_1 \cos\theta_1 & l_2 \cos\theta_2 & l_3 \cos\theta_3 \end{bmatrix}$$

You can imagine how computing Jacobian gets ugly when there are multiple joints!

Computing Jacobian geometrically

- Instead of computing Jacobian analytically, we can take a geometric approach to compute it

- Let's say we are just concerned with the end effector position $\mathbf{p}$ for now.
  - This also implies that the Jacobian will be an $3\times N$ matrix where N is the number of DOFs
  - For each DOF of a joint, we analyze how $\mathbf{p}$ would change if the DOF changes

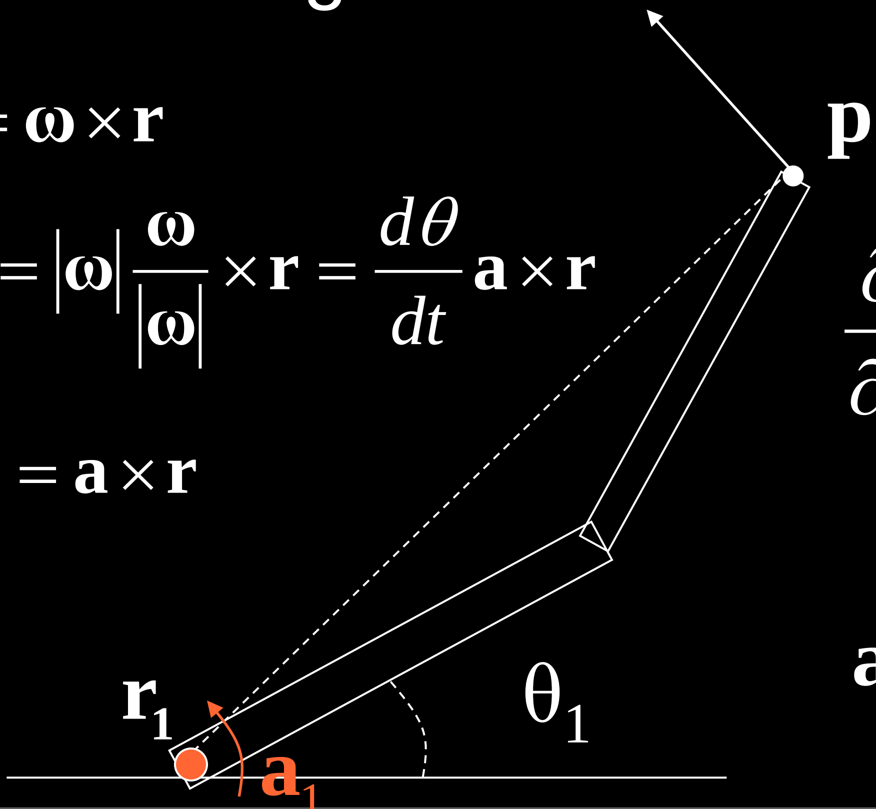
Rotational DOFs

- Let's consider a 1-DOF rotational joint first
- We want to know how the global position **p** will change if we rotate around the axis.

$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r}$$

$$\frac{d\mathbf{p}}{dt} = |\boldsymbol{\omega}| \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \times \mathbf{r} = \frac{d\theta}{dt} \mathbf{a} \times \mathbf{r}$$

$$\frac{d\mathbf{p}}{d\theta} = \mathbf{a} \times \mathbf{r}$$

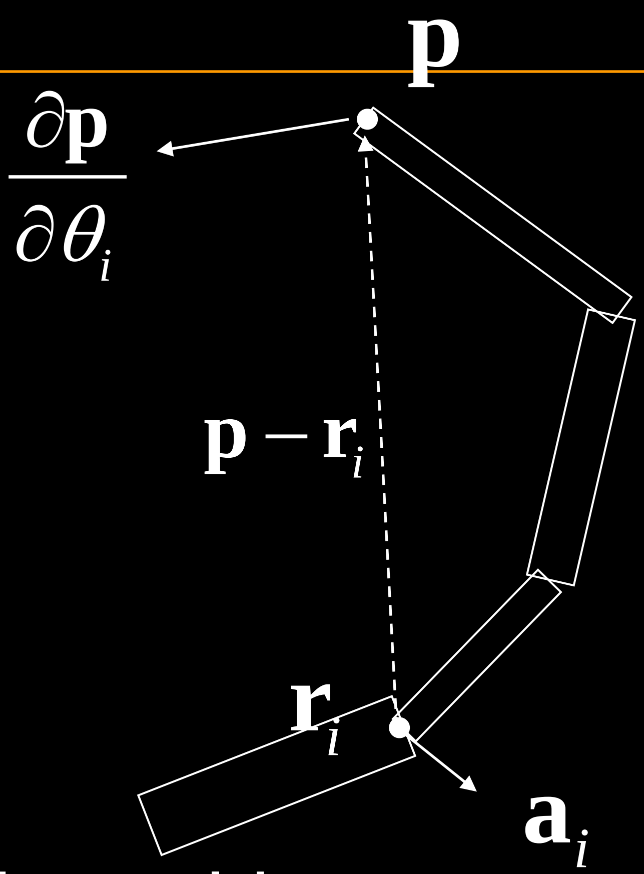


$$\frac{\partial \mathbf{p}}{\partial \theta_1} = \begin{bmatrix} \dfrac{\partial p_x}{\partial \theta_1} \\ \dfrac{\partial p_y}{\partial \theta_1} \end{bmatrix} = \mathbf{a}_1 \times (\mathbf{p} - \mathbf{r}_1)$$

unit-length rotation axis vector

$$\mathbf{a}_1 = \frac{\boldsymbol{\omega}_1}{|\boldsymbol{\omega}_1|}$$

Rotational DOFs



$$\frac{\partial \mathbf{p}}{\partial \theta_i} = \mathbf{a}_i \times (\mathbf{p} - \mathbf{r}_i)$$

$\mathbf{a}_i$: unit length rotation axis in world space

$\mathbf{r}_i$: position of joint pivot in world space

p: end effector position in world space

3-DOF Rotational Joints

- Once we have each axis in world space, each one will get a column in the Jacobian matrix
- At this point, it is essentially handled as three 1-DOF joints, so we can use the same formula for computing the derivative as we did earlier:

$$\frac{\partial \mathbf{p}}{\partial \theta_i} = \mathbf{a}_i \times \left(\mathbf{p} - \mathbf{r}_i\right)$$

Iterative IK Using Inverse Jacobian

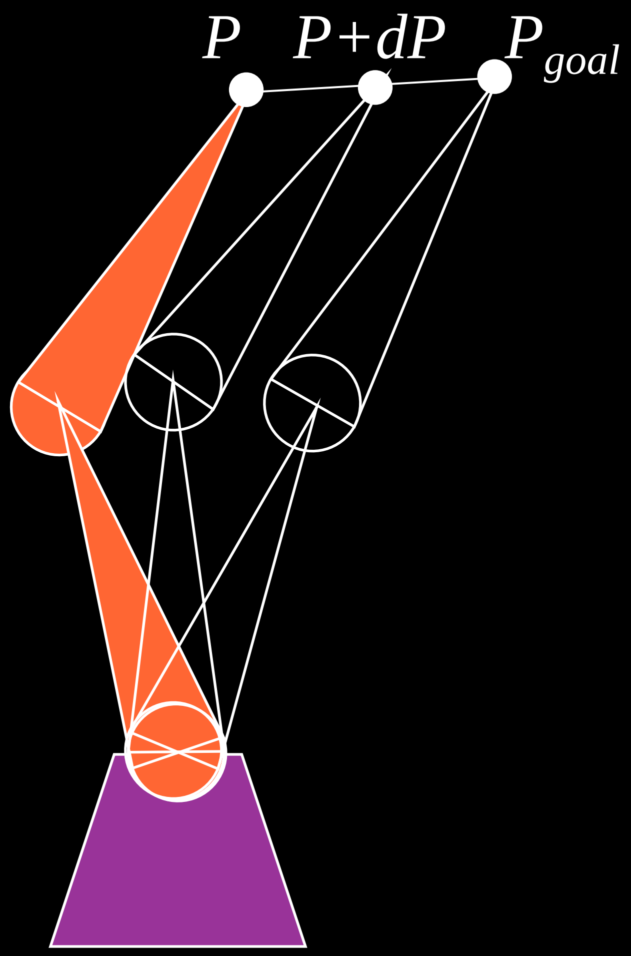
$$\theta = f^{-1}(P)$$

$$V = J(\theta)\dot{\theta}$$

$$\dot{\theta} = J^{-1}(\theta)V$$

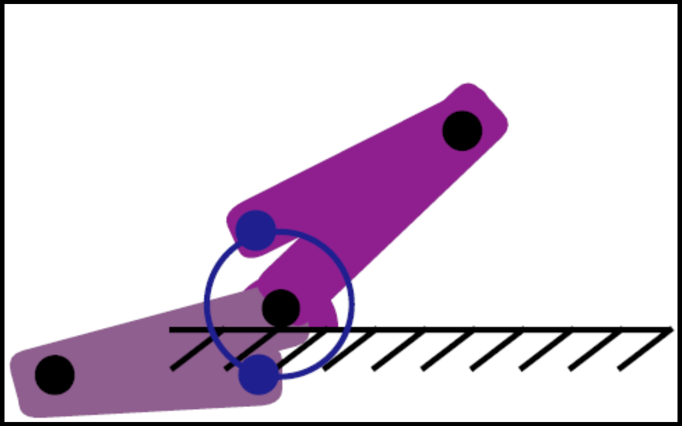
$$\theta_{k+1} = \theta_k + \Delta t J^{-1}(\theta_k)V$$

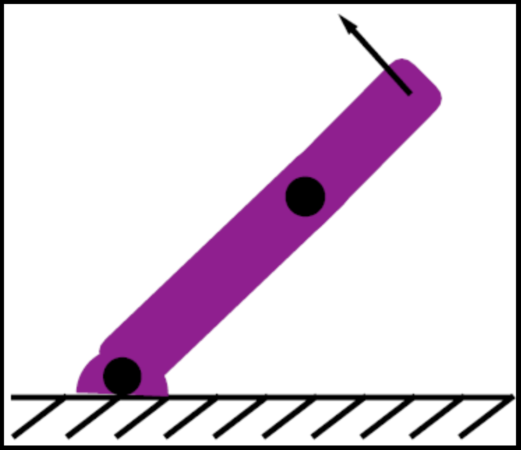
- Linearize about θ_k locally
- Small increments



Jacobian may not be invertible!

- Non-square matrix
  - pseudo inverse
- Singularity
  - causes infinite joint velocities
  - occurs when any $\dot{\theta}$ cannot achieve V that is not perpendicular to the arm





Remedy to Singularity Problem

- Add redundancy
  - add more joints to the original joint chain (more DOFs are added)
  - Jacobian matrix is not square
- Use pseudo inverse of Jacobian!

Pseudo Inverse of the Jacobian

$$V = J\dot{\theta}$$

$$J^T V = J^T J\dot{\theta}$$

$$(J^T J)^{-1} J^T V = (J^T J)^{-1} J^T J\dot{\theta}$$

$$J^+ = (J^T J)^{-1} J^T$$

$$J^+ V = \dot{\theta}$$

Adding more control to IK

- Pseudo inverse computes one of many possible solutions that minimize joint angle velocities
- IK using pseudo inverse Jacobian may not provide natural poses
- A control term can be added to the pseudo inverse Jacobian solution
- The control term should not add anything to the velocities, that is

$$J\dot{\theta} = V$$

control term

$$\dot{\theta} = J^{+}V + \boxed{(J^{+}J - I)z}$$

Control Term Adds Zero Linear Velocities

A solution of this form → $C = (J^{+}J - I)z$

When put into this formula → $V = JC$

Like this → $V = J(J^{+}J - I)z$

After some manipulation, you can show that …

$$V = (JJ^{+}J - J)z$$

$$V = (J - J)z$$

$$V = 0z$$

…it doesn't affect the desired configuration

$$V = 0$$

But it can be used to bias
The solution vector

Null space

- The control term *C* is in the null space of *J*

$$C = (J^{+}J - I)z$$

- The null space of *J* is the set of vectors which have no influence on the constraints

$$\theta \in nullspace(J) \Leftrightarrow J\theta = 0$$

Utility of Null Space

- The null space can be used to reach secondary goals

$$\dot{\theta} = J^{+}V + (J^{+}J - I)z$$

$$\min_{z} G(\theta)$$

- Or to find natural pose / control stiffness of joints

$$G(\theta) = \sum_{i} \alpha_i (\theta_{natural}(i) - \theta(i))^2$$

Optimization-based Method

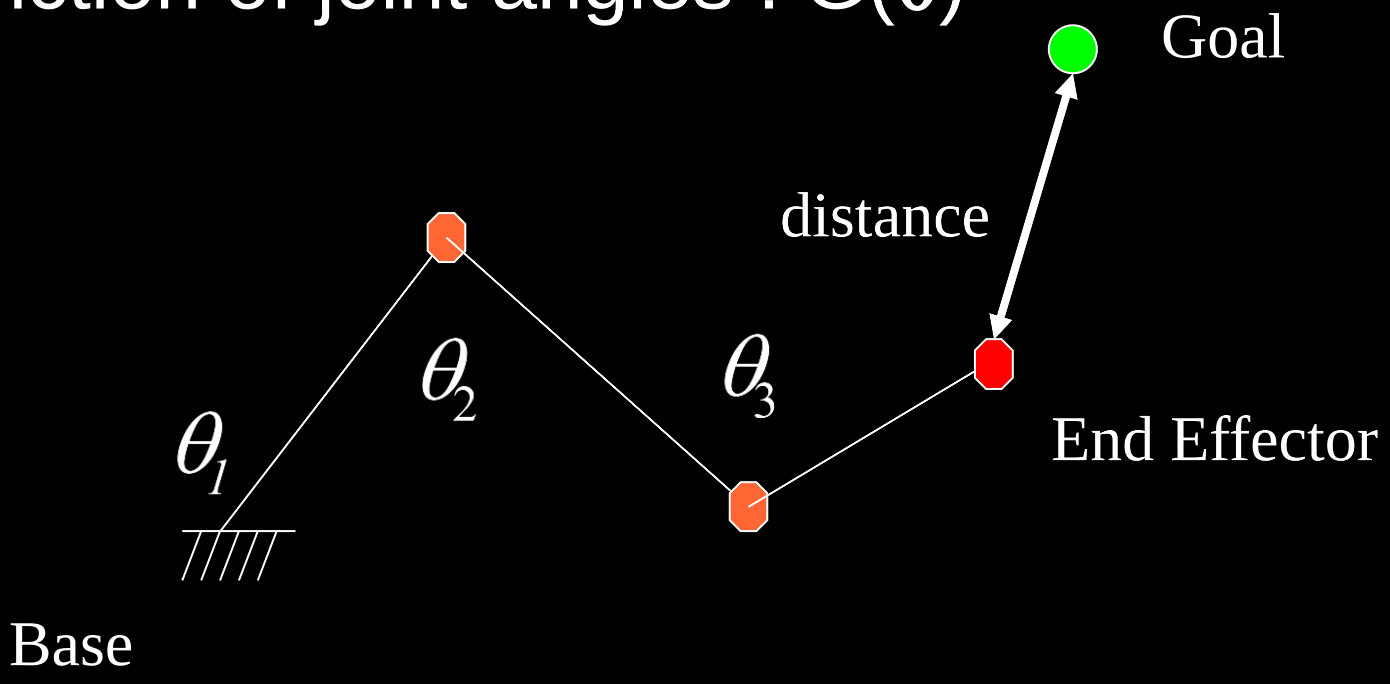
- Formulate IK as an nonlinear optimization problem
  - Example

$$\text{minimize } x^2(y+1)+\sin(x+y)$$
$$\text{subject to } \mathrm{x} \geq 0, \mathrm{y} \geq 0$$

  - Objective function
  - Constraint
  - Iterative algorithm
- Nonlinear programming method by Zhao & Badler, TOG 1994

Objective Function

- “distance” from the end effector to the goal position/orientation
- Function of joint angles : G(θ)



Objective Function

Position Goal

$$\left\|\mathbf{p}_\mathbf{g} - \mathbf{p}_\mathbf{e}\right\|^2$$

Orientation Goal

$$\left\|\mathbf{r}_\mathbf{x}^\mathbf{g} - \mathbf{r}_\mathbf{x}^\mathbf{e}\right\|^2 + \left\|\mathbf{r}_\mathbf{y}^\mathbf{g} - \mathbf{r}_\mathbf{y}^\mathbf{e}\right\|^2$$

Position/Orientation Goal

$$w\left\|\mathbf{p}_\mathbf{g} - \mathbf{p}_\mathbf{e}\right\|^2 + (1-w)\left(\left\|\mathbf{r}_\mathbf{x}^\mathbf{g} - \mathbf{r}_\mathbf{x}^\mathbf{e}\right\|^2 + \left\|\mathbf{r}_\mathbf{y}^\mathbf{g} - \mathbf{r}_\mathbf{y}^\mathbf{e}\right\|^2\right)$$

weighted sum

Nonlinear Optimization

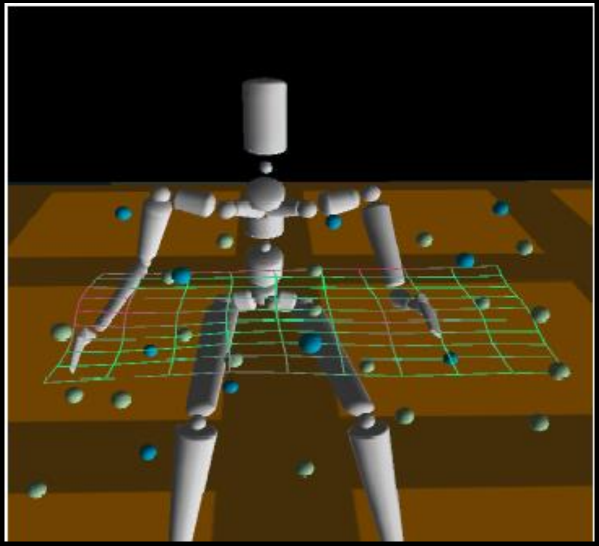
- Constrained nonlinear optimization problem

$$\begin{cases} \text{minimize} \quad G(\boldsymbol{\theta}) \\ \text{subject to} \begin{cases} \mathbf{a}^T\boldsymbol{\theta} = \mathbf{b}_1 & \text{limb coordination} \\ \mathbf{a}^T\boldsymbol{\theta} \le \mathbf{b}_2 & \text{joint limits} \end{cases} \end{cases}$$

- Solution
  - standard numerical techniques
  - MATLAB or other optimization packages
  - usually a local minimum
  - depends on initial condition

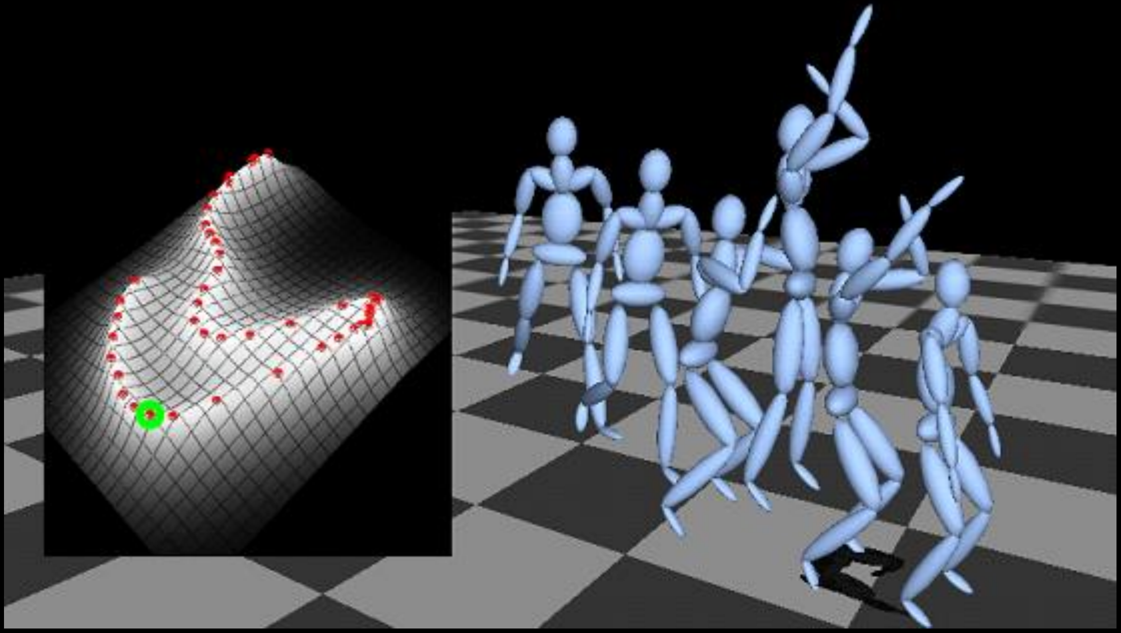
Example-based Method

- Utilize motion database to assist IK solving
- IK using interpolation
  - Rose et al., “Artist-directed IK using radial basis function interpolation,” Eurographics’01



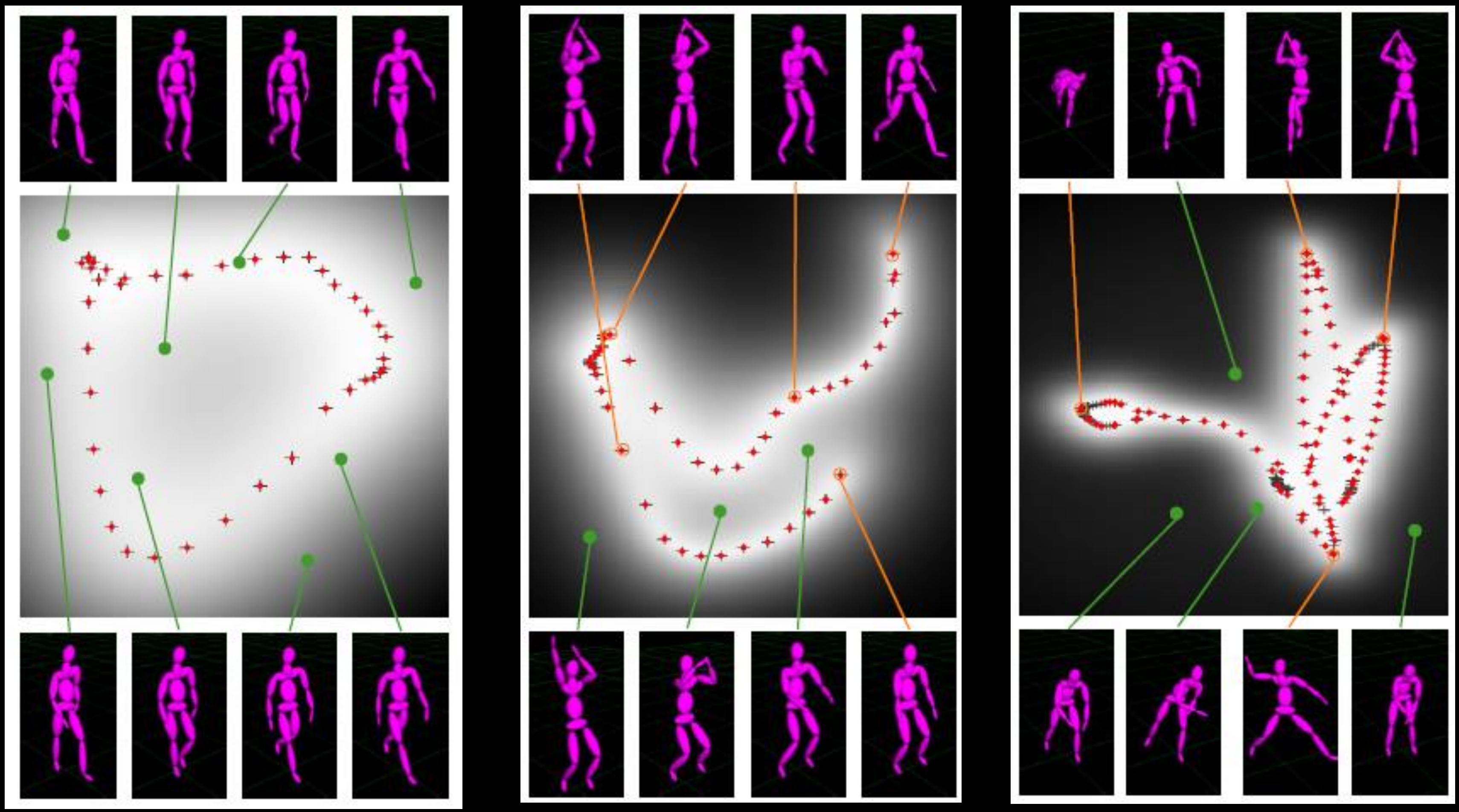
Example-based Method (cont.)

- IK using constructed statistical model
  - Grochow et al., “Style-based inverse kinematics,” SIGGRAPH’04
  - Provide the most likely pose based on given constraints



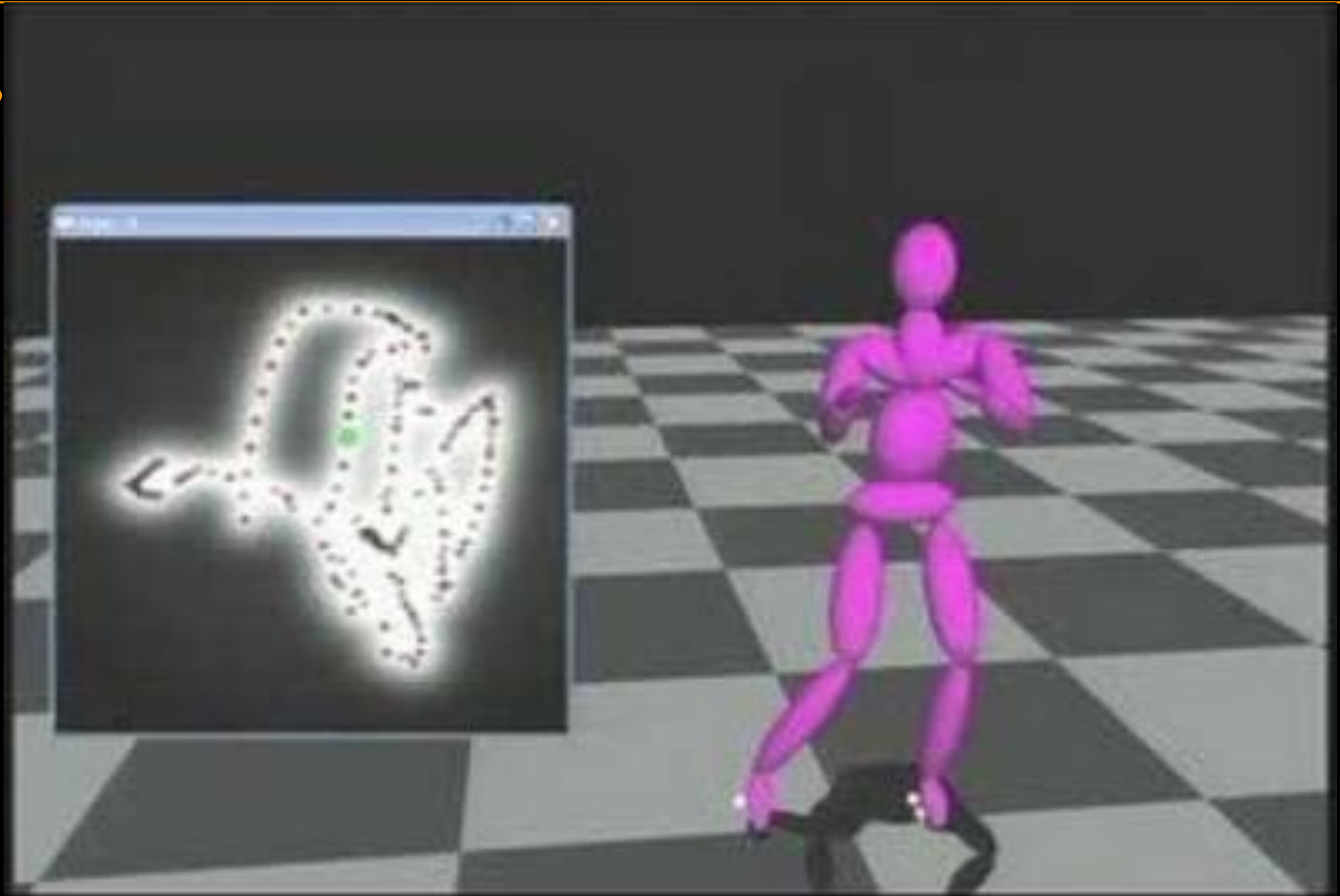
Example-based Method (cont.)

- Constructed pose space (training, extrapolated)



Videos

•



References

- Zhao and Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," ACM TOG 1994.
- Rose et al., “Artist-directed IK using radial basis function interpolation,” Eurographics’01
- Keith Grochow, Steven L. Martin, Aaron Hertzmann and Zoran Popovic , "Style-based inverse kinematics," SIGGRAPH’04.
- Aristidou et al., “Inverse Kinematics Techniques in Computer Graphics: A Survey,” EG 2018