Animation

Movie and Game

Toy's story, Pixar/Disney, 1995

How NBA 2K Makes Basketball Players Look Real In Video Games
https://www.youtube.com/watch?v=TGTTkqDvfdw

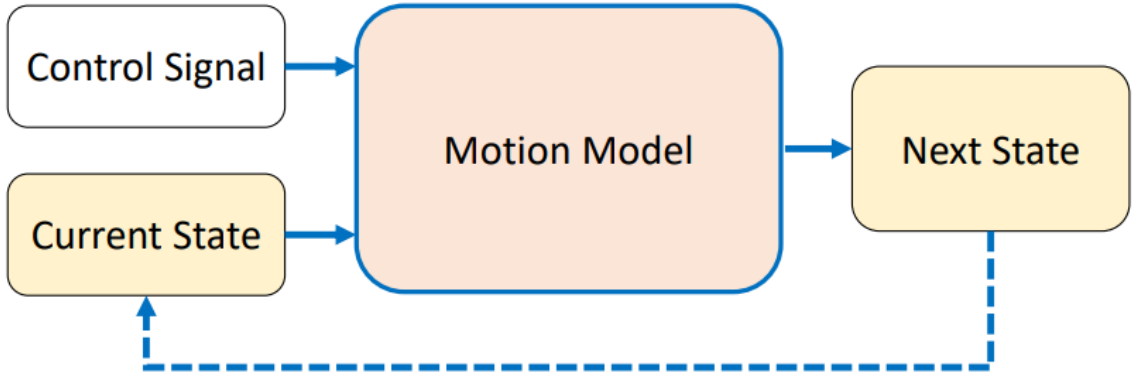
Game

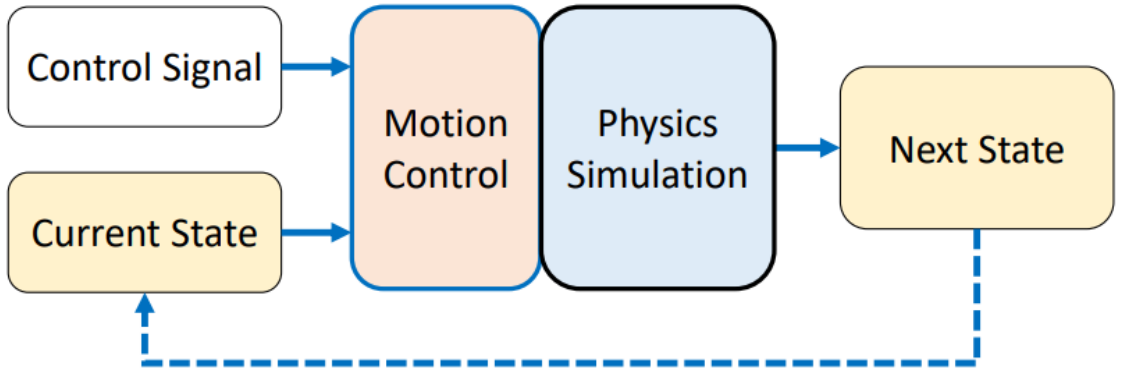
Elements of Animation

- Physical (force control)
- Behavioral (action control, physical and biophysical)
  - Action intention
  - Muscle activation
  - Emotion…
- Character Animation
  - Body (skeleton)
  - Face

Character Animation Methods

- Kinematics:
  - Keyframe Animation, Inverse Kinematics
  - Motion Capture, Motion Retargeting
  - Motion Graphs, Motion Matching
- Data driven & Learning-based Methods:
  - Generative Models
- Dynamics:
  - Physics-based motion control





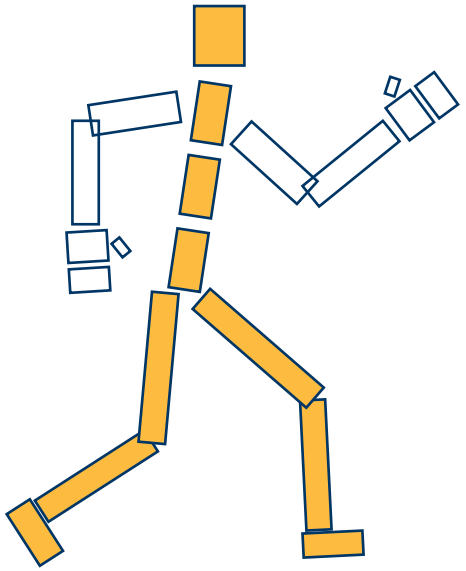
Skeleton Animation Basics

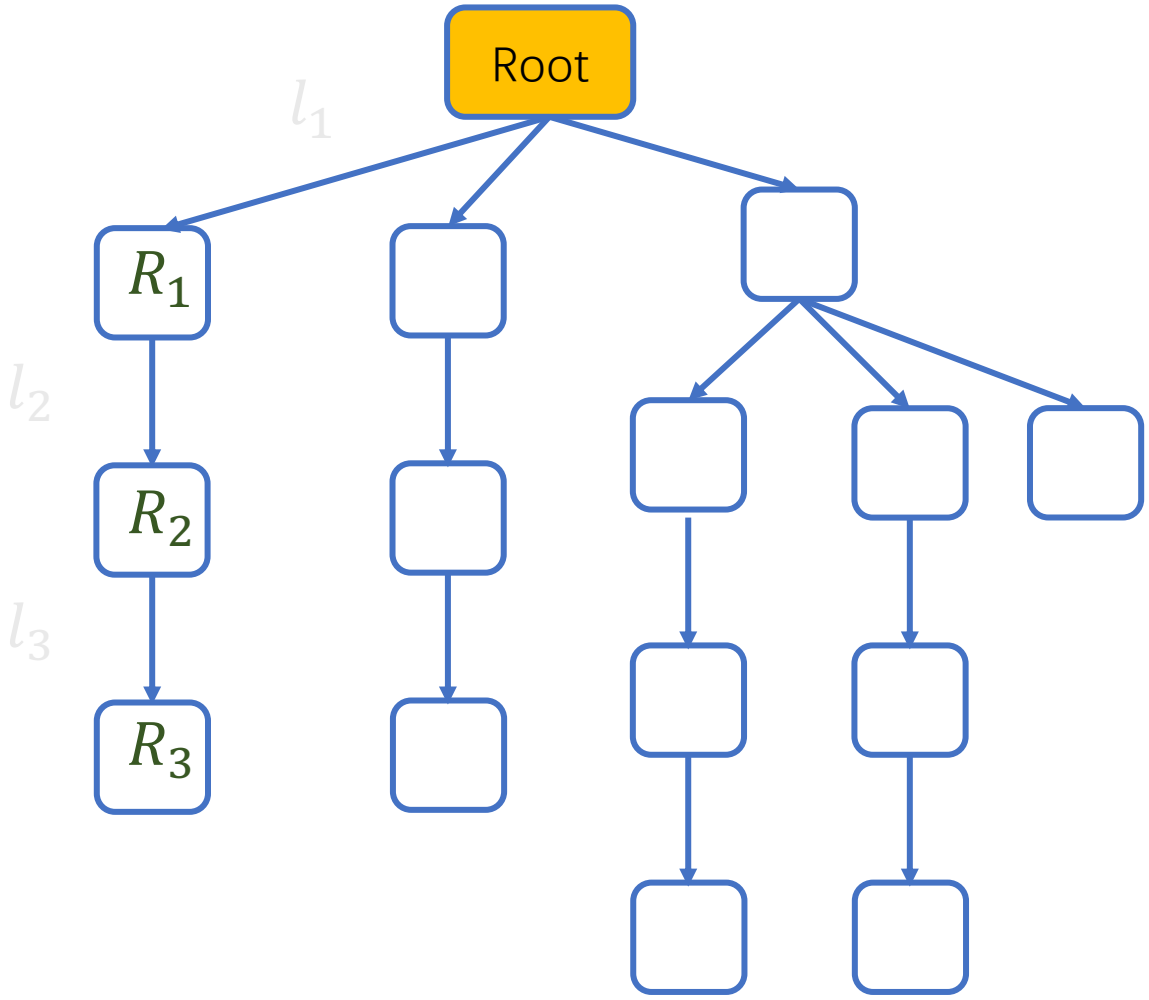
Skeleton Animation

https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road
https://zhuanlan.zhihu.com/p/136971426

Representation of skeleton system

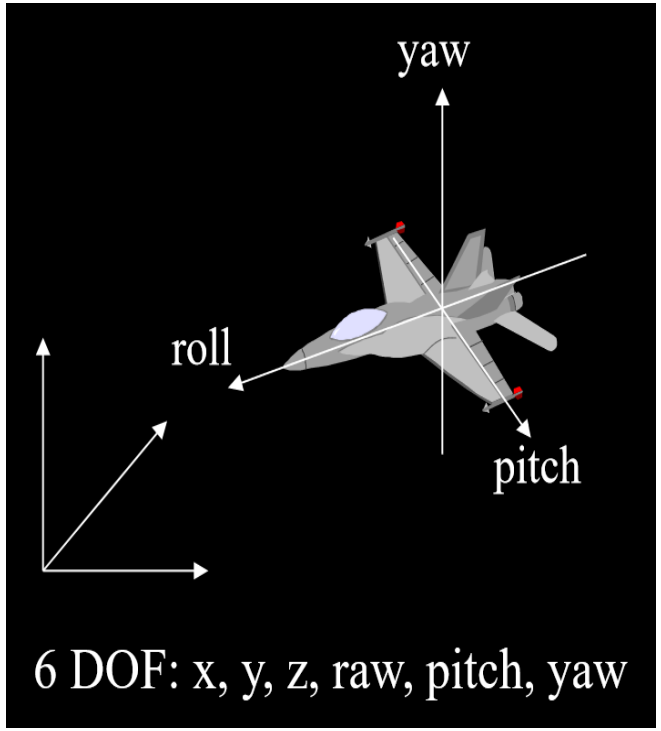
- Use **Joint** to link **Bones**
- Tree structure
  - Nodes represents **Joints**
  - Edges represents **Bones**



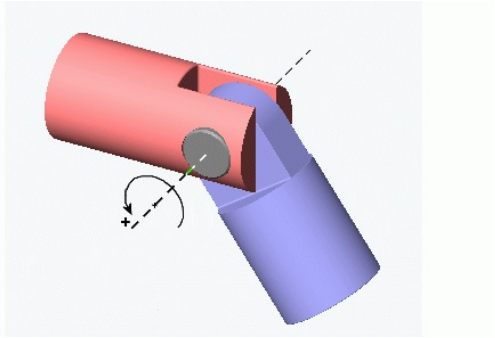


Joint model

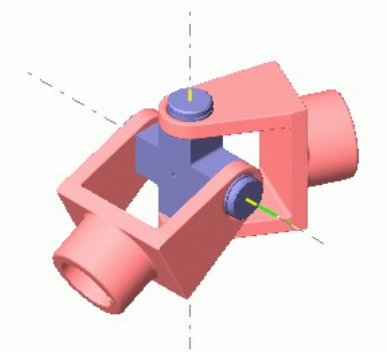
- Degree of freedom, DoF:
  - the least number of parameters to assign the motion of an object



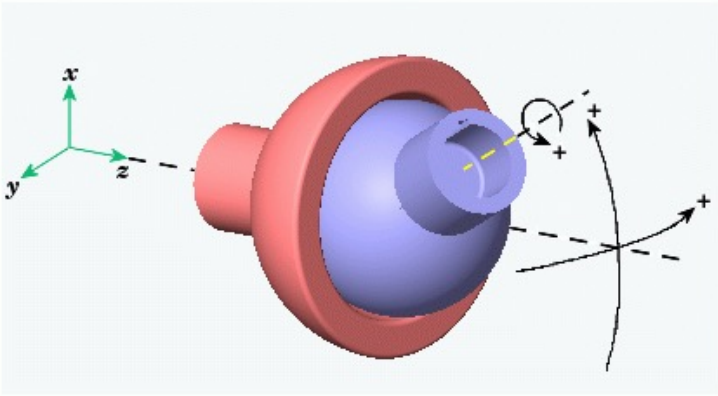
Joint model



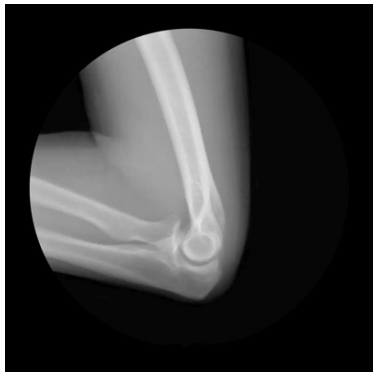
Hinge joint
Revolute joint
1 DoF



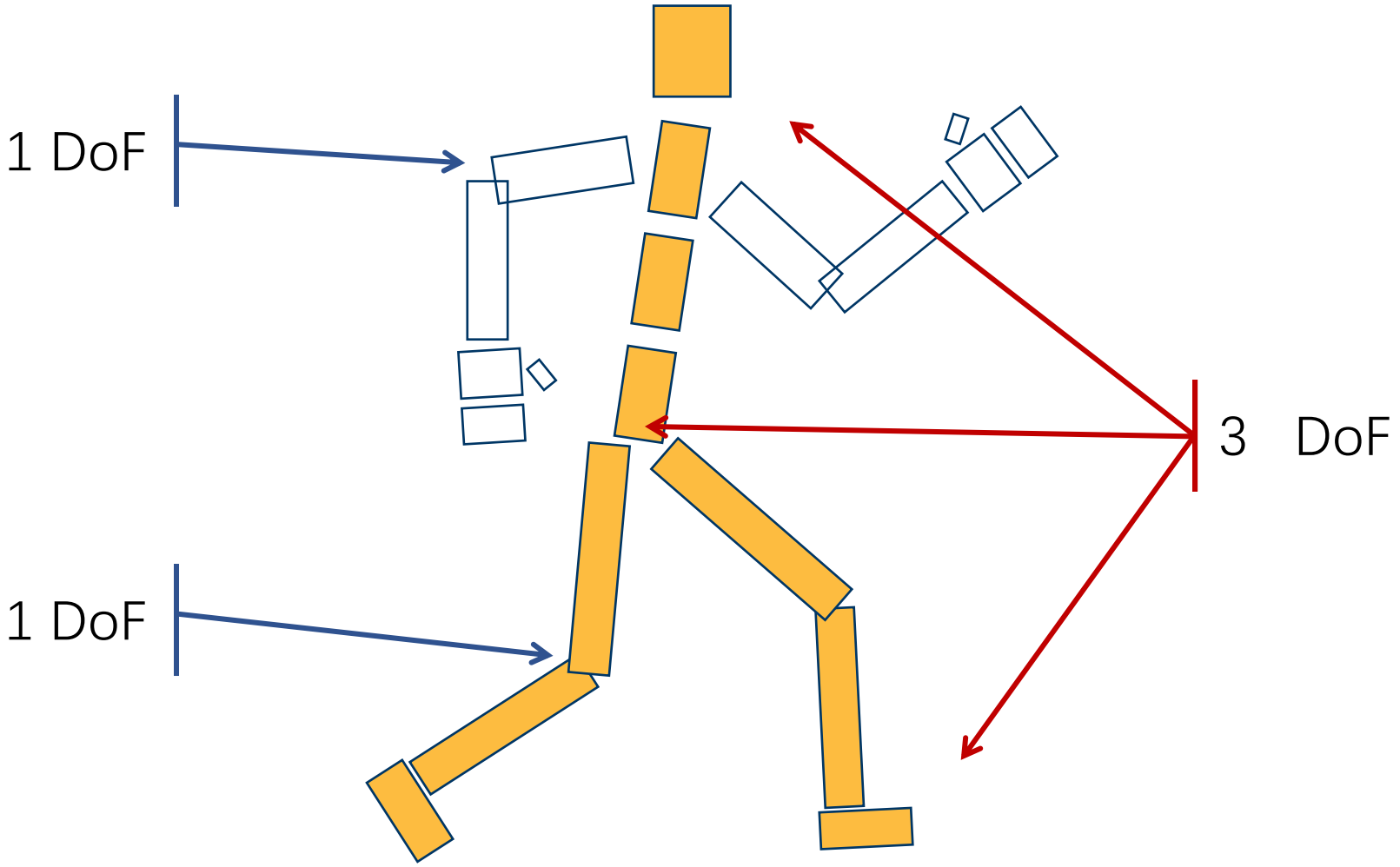
Universal joint
2 DoF



Ball-and-socket
3 DoF



Skeleton System

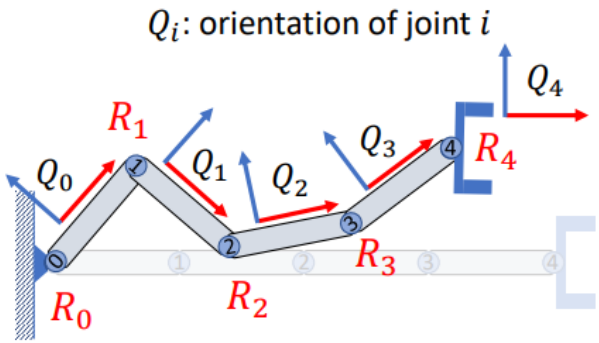


Motion data

- Pose can be represent as

$$(\boldsymbol{t}_0, R_0\,,\, R_1, R_2, \dots\dots)$$

root	internal joints
Global position	Local rotations



$$Q_0 = R_0$$

$$Q_1 = R_0R_1$$

$$Q_2 = R_0R_1R_2$$

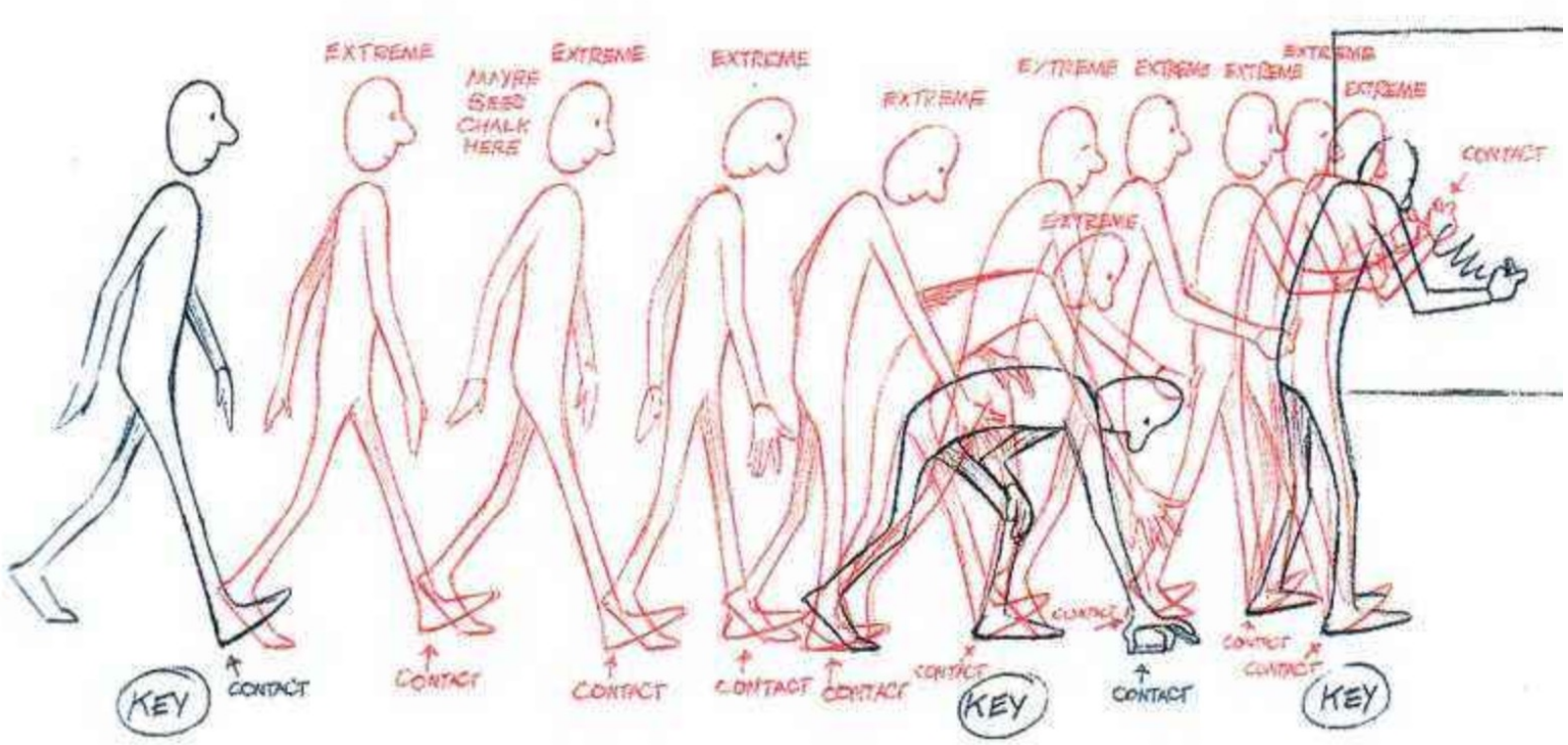
$$Q_3 = R_0R_1R_2R_3$$

$$Q_4 = R_0R_1R_2R_3R_4$$

- A sequence of local rotations(axis angle, Euler angle): rotation matrix, or quaternion
- To show the pose, you need to change it into global rotations -> orientation Q
- Forward Kinematic: Compute the position of the end-effector from specified values for the joint parameters

Keyframe Animation and Inverse Kinematics

Keyframe Animation

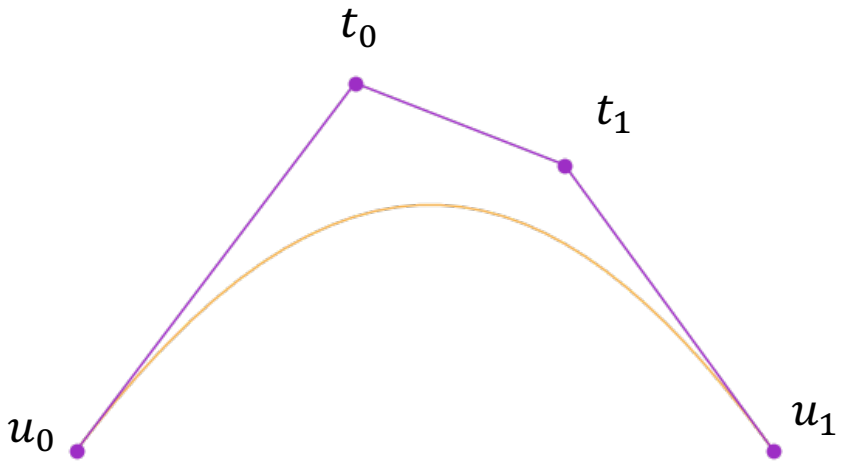


How to generate key frame?

How to generate inbetweening?

Keyframe Interpolation

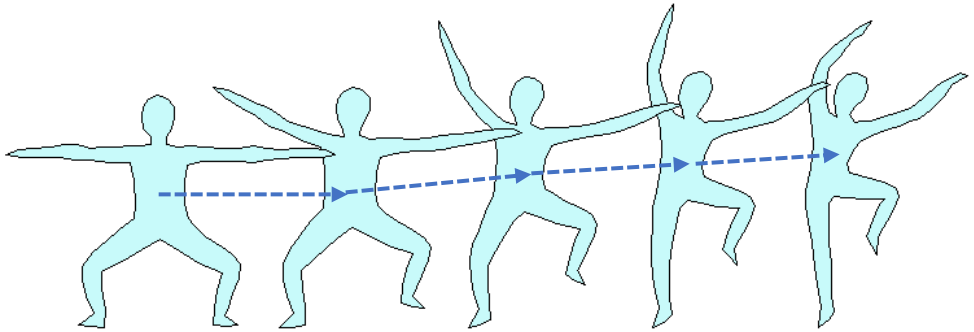
- We have learned linear interpolation(lerp) and Beizer interpolation

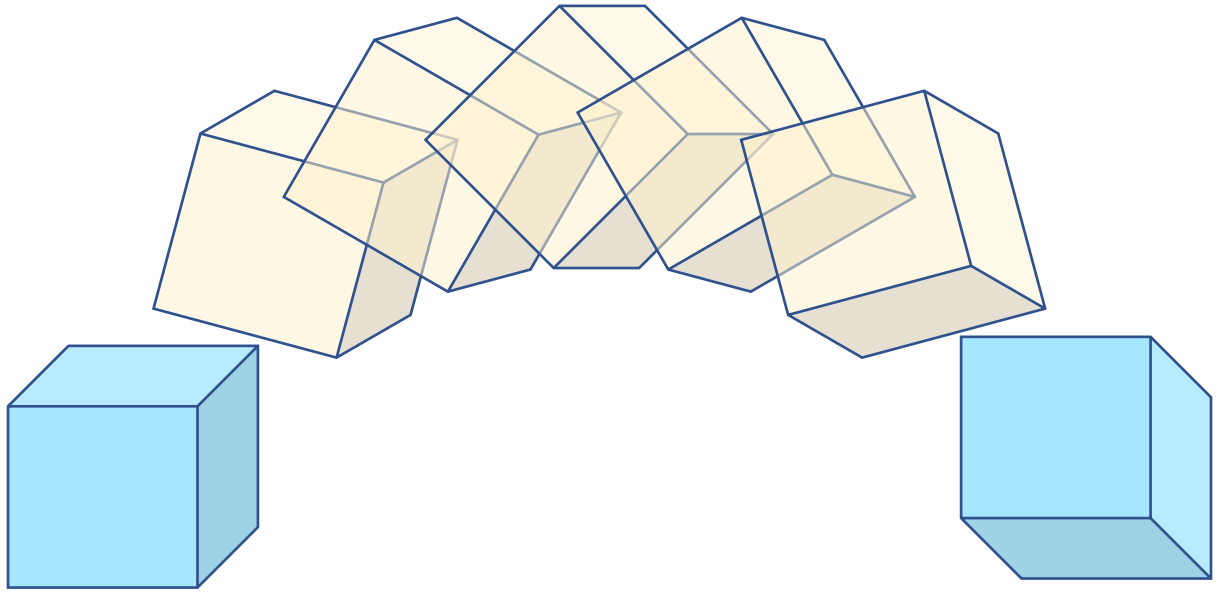


Keyframe Interpolation

- We can interpolate position with lerp
- What about rotation?

  Interpolate with angle

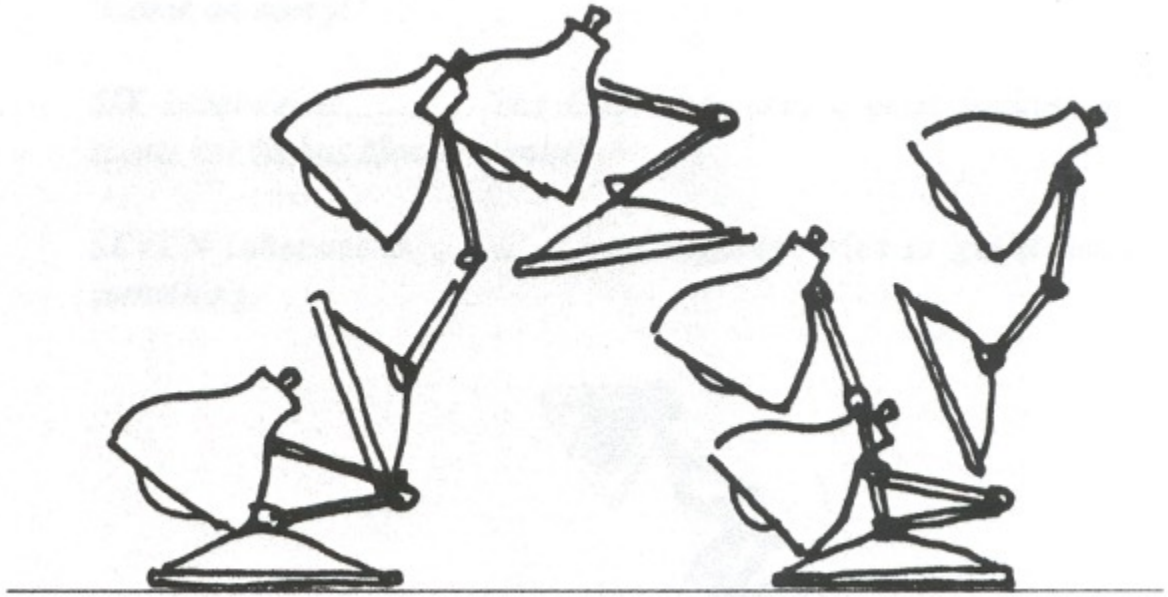


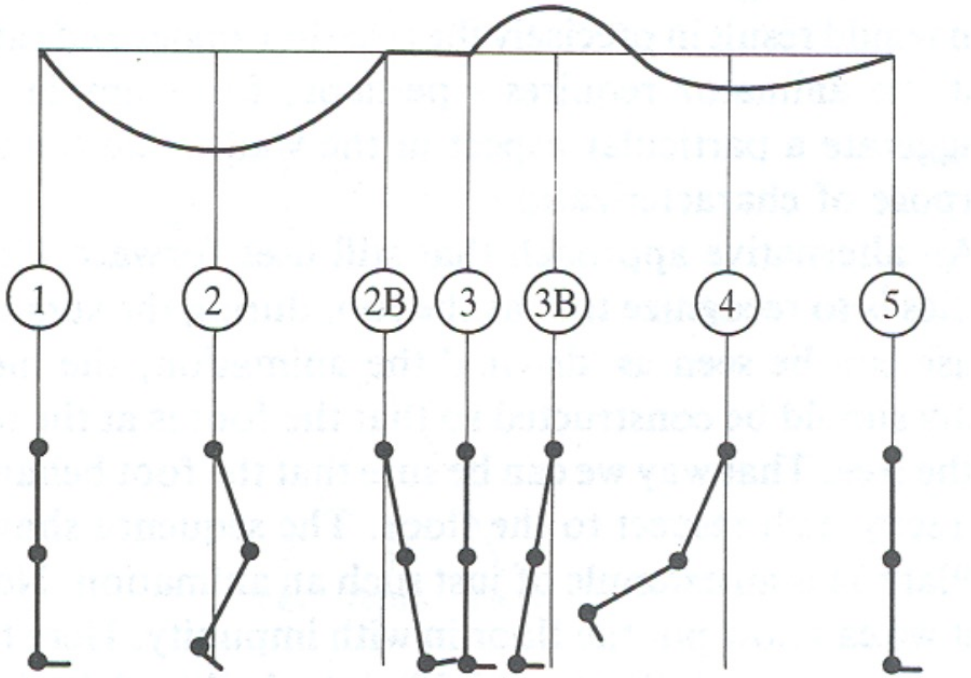


R_0 R_1

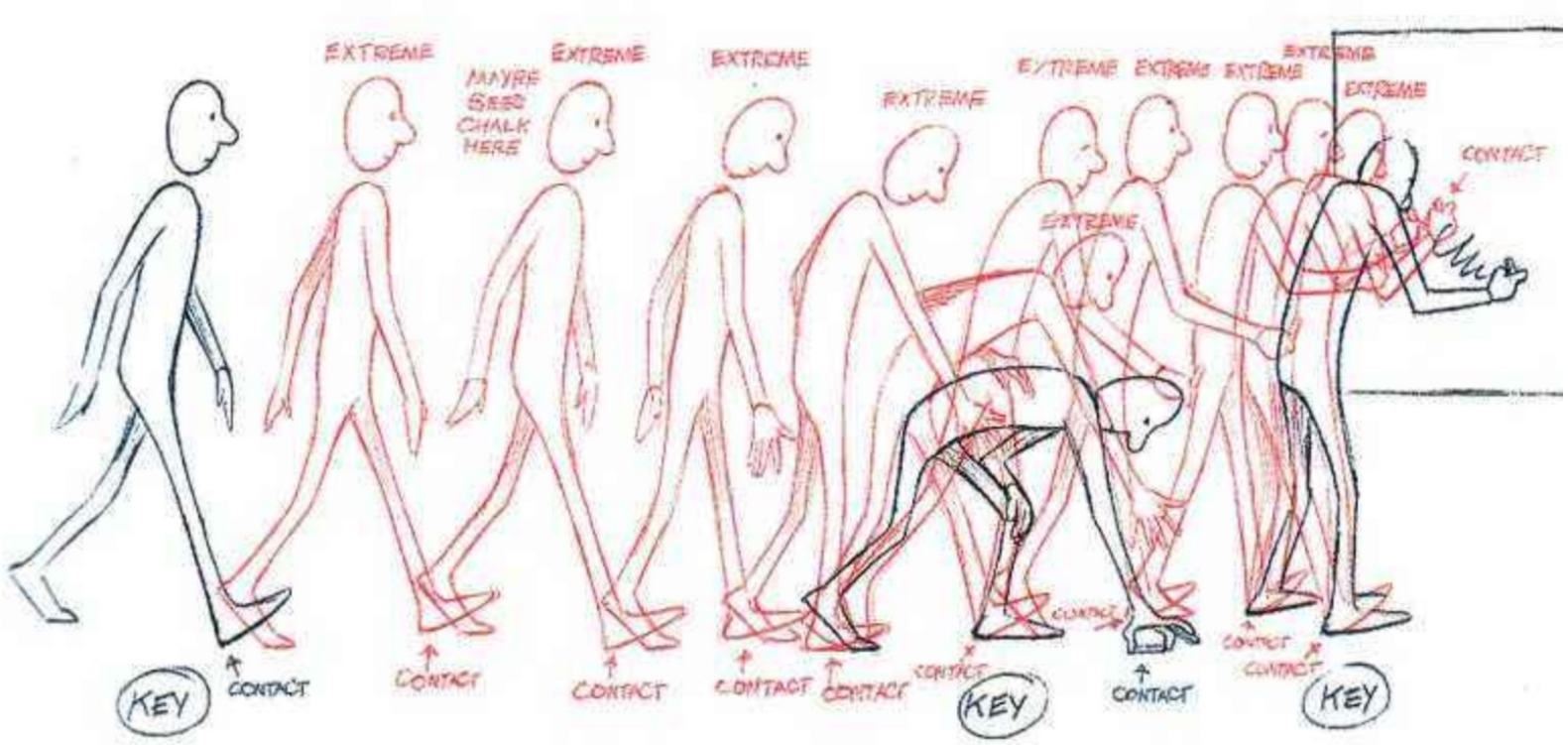
Keyframe Interpolation

- Examples, Pixar's Lamp and a Walk Cycle





Keyframe Animation



How to generate key frame?

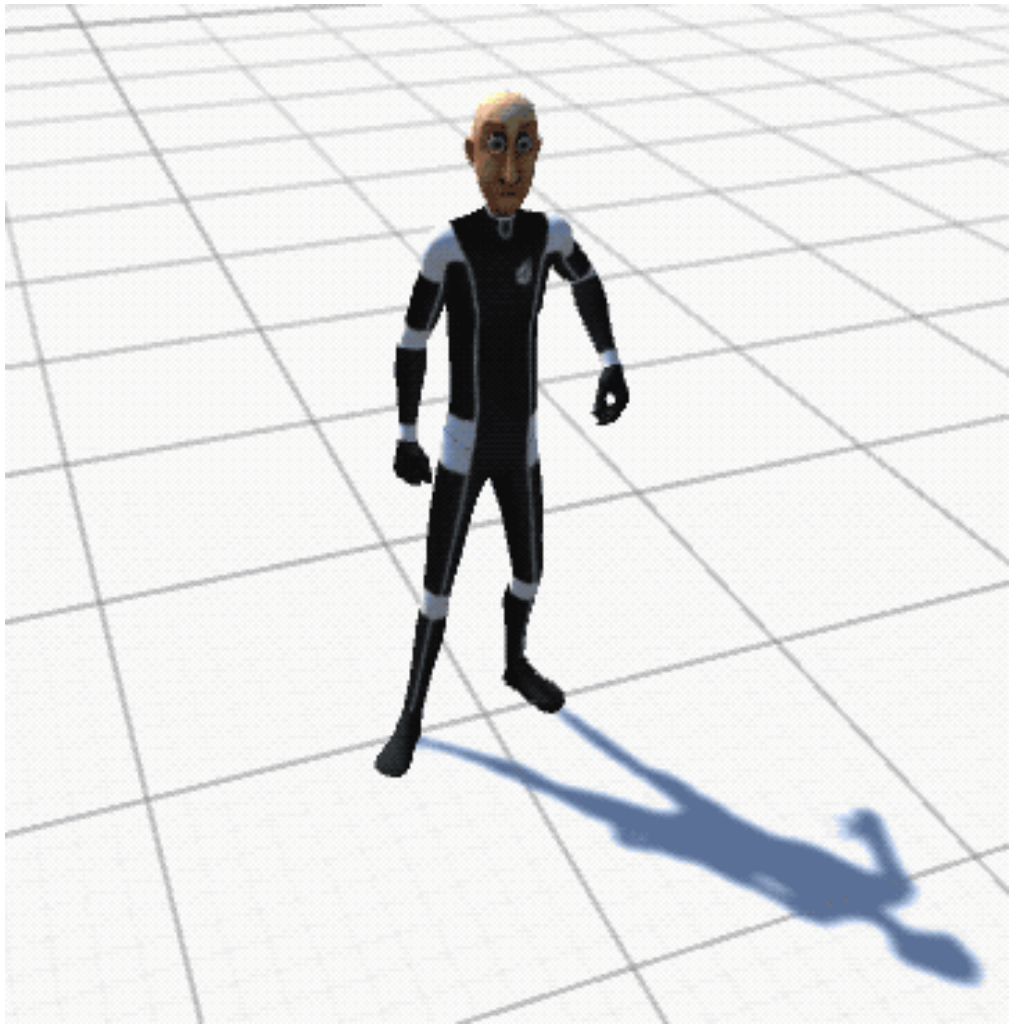
How to generate inbetweening?

Inverse Kinematics (IK)

Definition of Inverse Kinematics (IK)

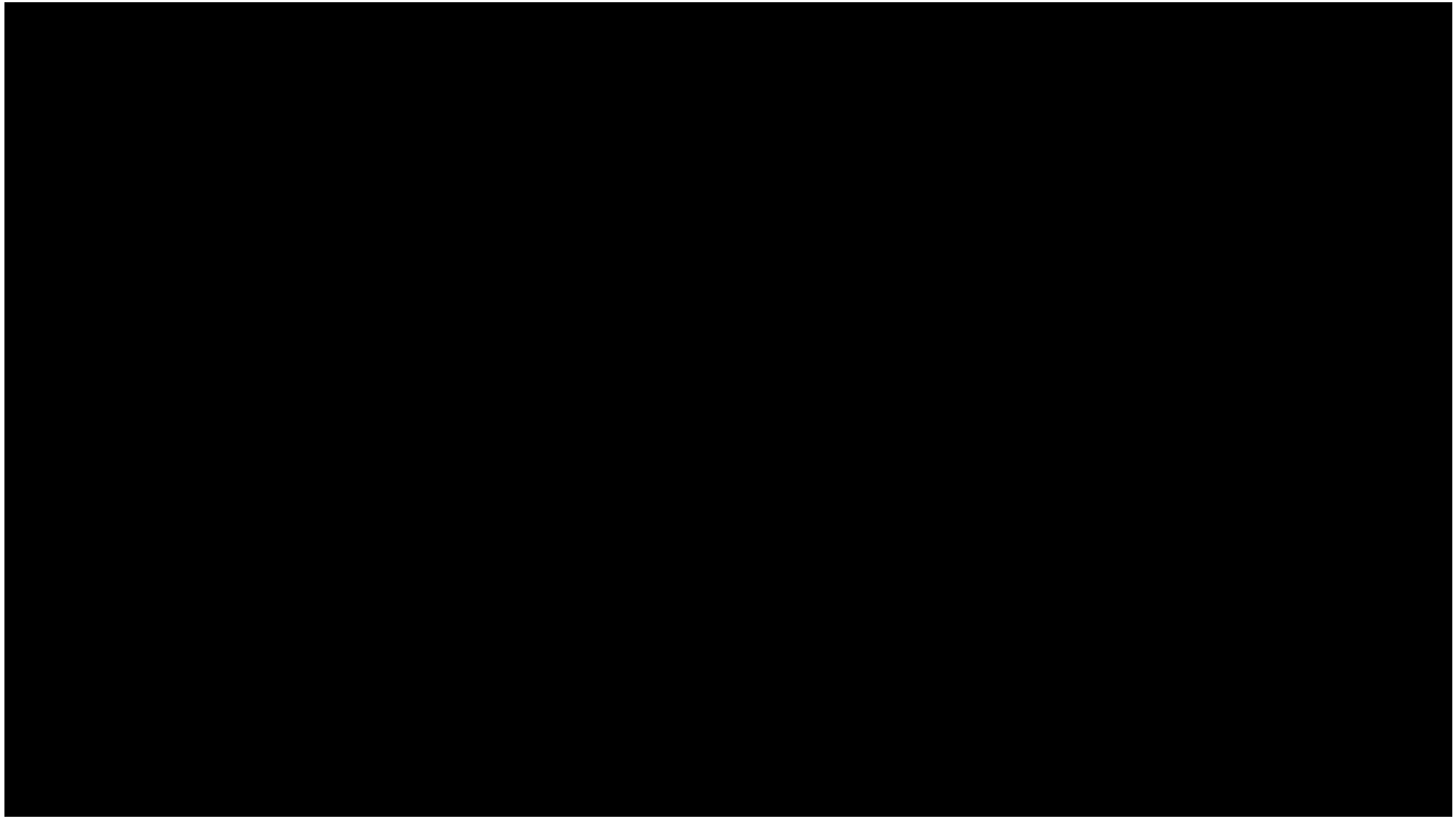
- Forward Kinematics (FK)
- Given the joint rotation, calculate the global information such as the position and orientation of the end effectors
- Inverse Kinematics (IK)
- Given target position of the end effector, the corresponding joint rotation is calculated to make the end limb reach the target position

Application of IK



- Virtual character attitude control
- Manipulator control
- etc

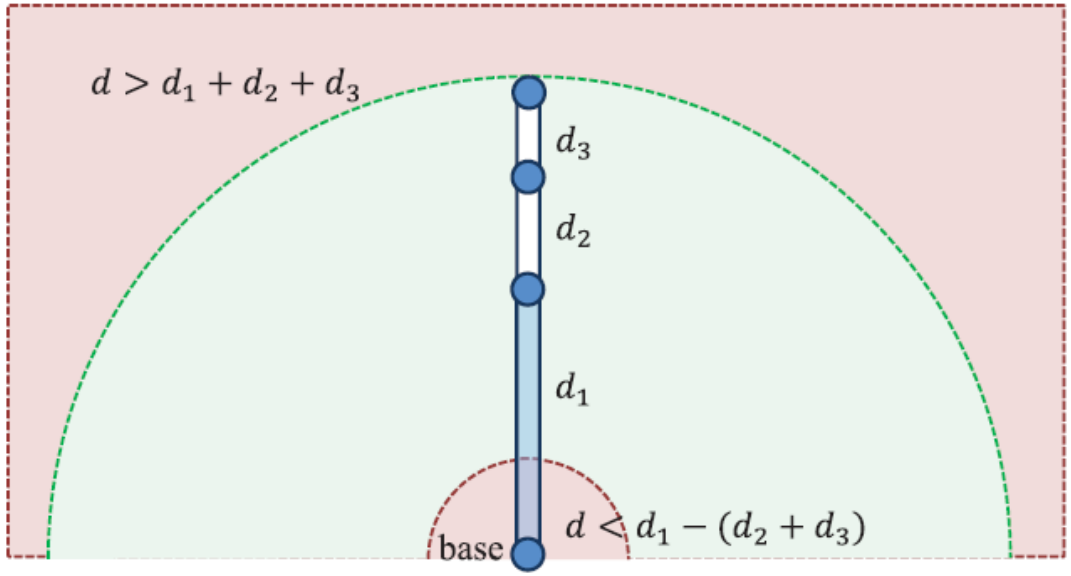
Application of IK



Human Body Rig in Blender
https://www.youtube.com/watch?v=MAM7mF2v7dE

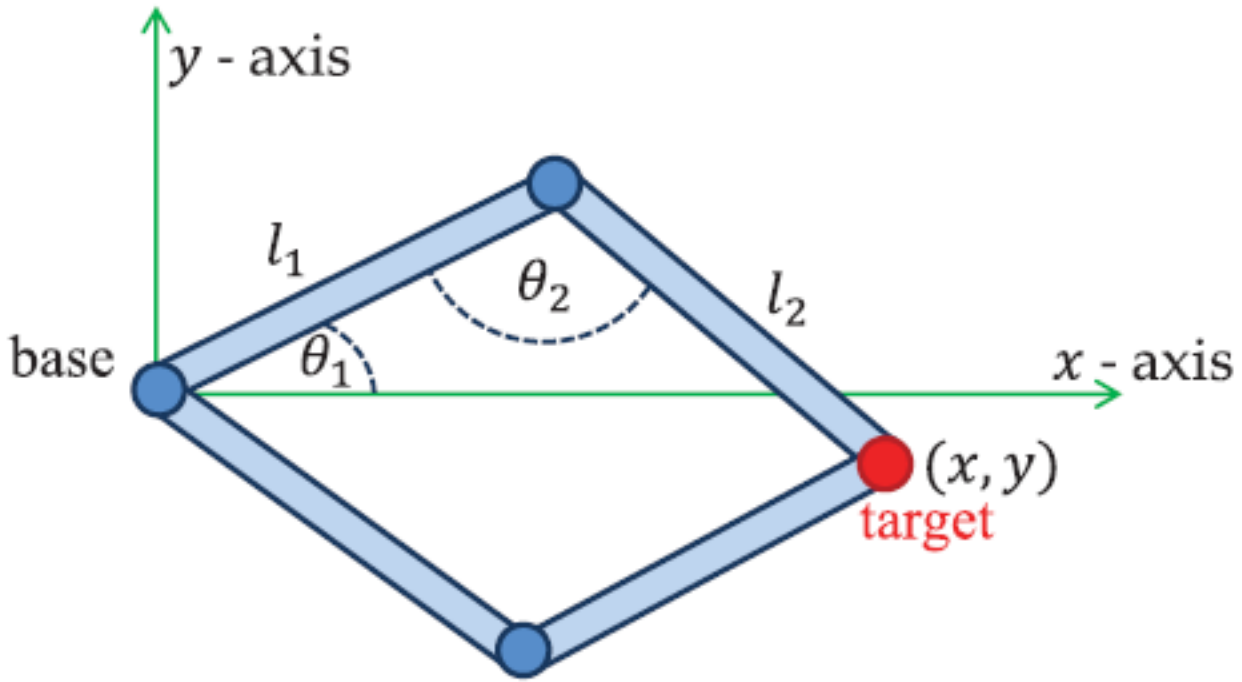
Feasible Region of IK

- Workspace of manipulator
- Beyond this range, the manipulator cannot reach
- IK has no solution at this time



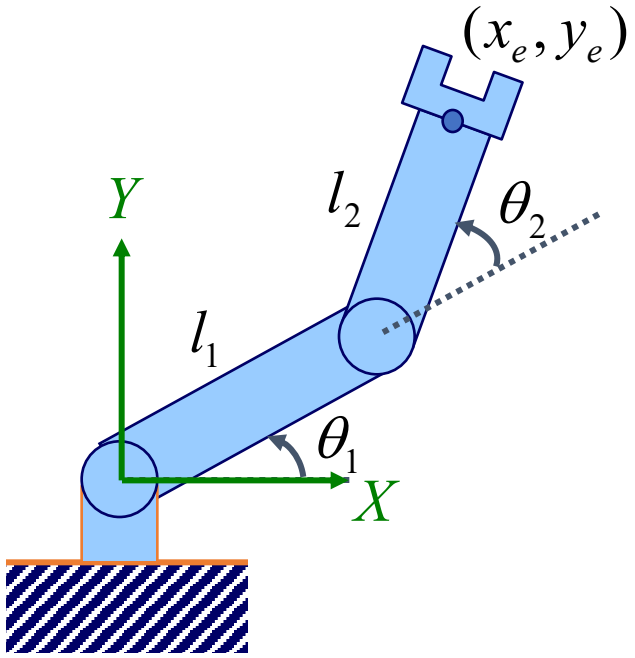
Multiple Solutions of IK

- It is possible that several different rotation modes can reach the same position
- It's best to be smooth and natural



Two Link IK

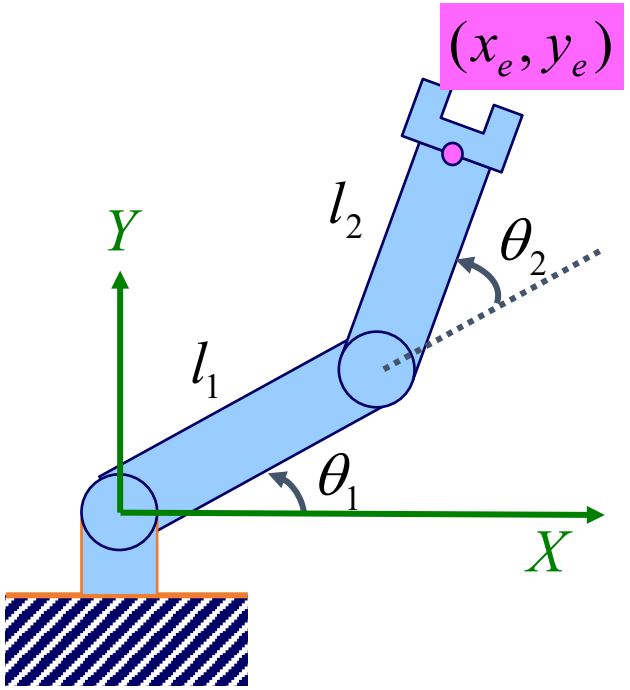
- A mechanical arm with only two joints
- Forward (FK) problem: give the joint angle and find the position of the end point



$$x_e = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y_e = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

Two Link IK

- A mechanical arm with only two joints
- Inverse (FK) problem: give the position of the end point and calculate the joint angle



$\theta_1 = ??$

$\theta_2 = ??$

$$\cos(\theta_r) = \frac{x_e}{\sqrt{x_e^2 + y_e^2}}$$

$$\theta_r = \cos^{-1}\left(\frac{x_e}{\sqrt{x_e^2 + y_e^2}}\right)$$

$$\cos(\theta_r - \theta_1) = \frac{l_1^2 + x_e^2 + y_e^2 - l_2^2}{2l_1\sqrt{x_e^2 + y_e^2}}$$

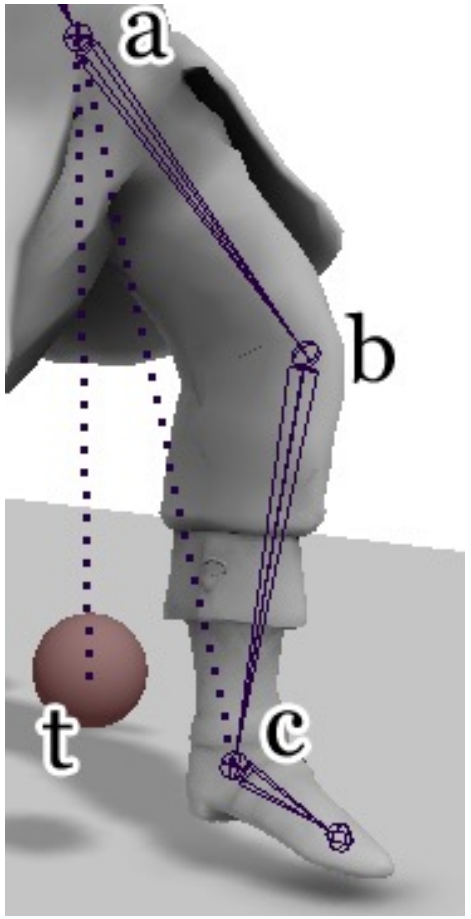
$$\theta_1 = \theta_r - \cos^{-1}\left(\frac{l_1^2 + x_e^2 + y_e^2 - l_2^2}{2l_1\sqrt{x_e^2 + y_e^2}}\right)$$

$$\cos(\pi - \theta_2) = \frac{l_1^2 + l_2^2 - x_e^2 - y_e^2}{2l_1 l_2}$$

$$\theta_2 = \pi - \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x_e^2 - y_e^2}{2l_1 l_2}\right)$$

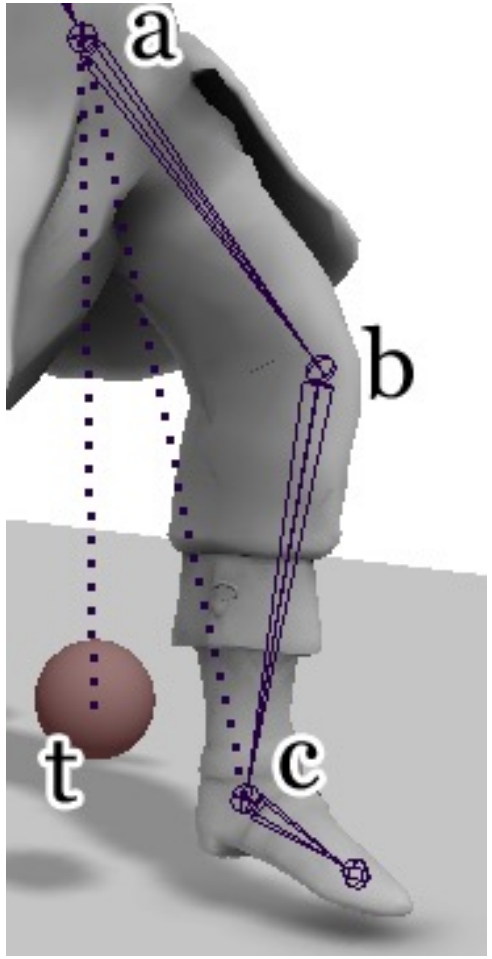
Two Link IK in 3D

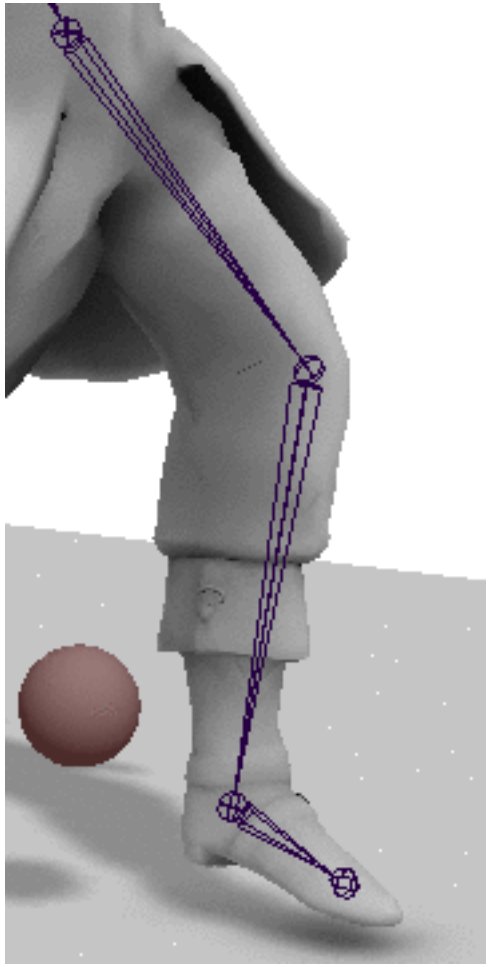
- Problem description:
- Rotate the joint a, b
- c can reach the target position t
- Have analytical solution



http://theorangeduck.com/page/simple-two-joint

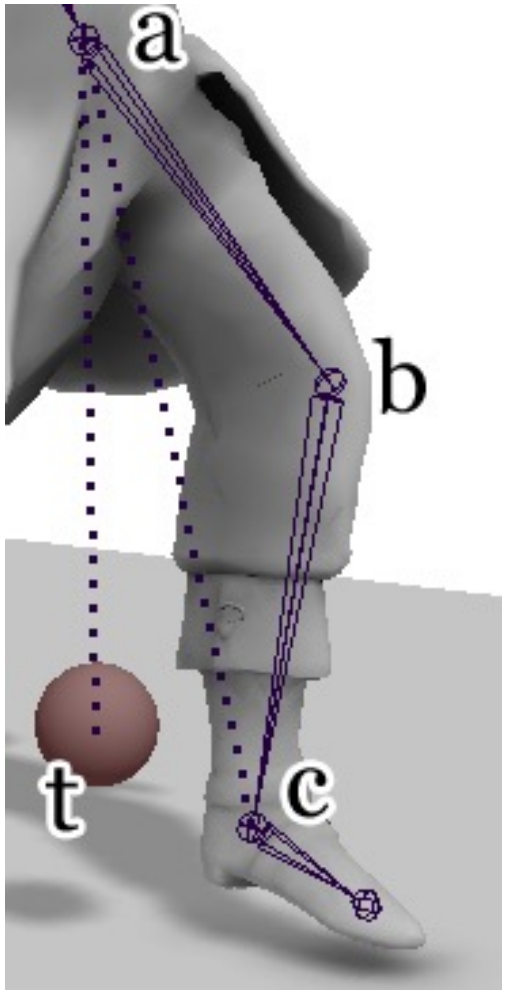
Two Link IK in 3D

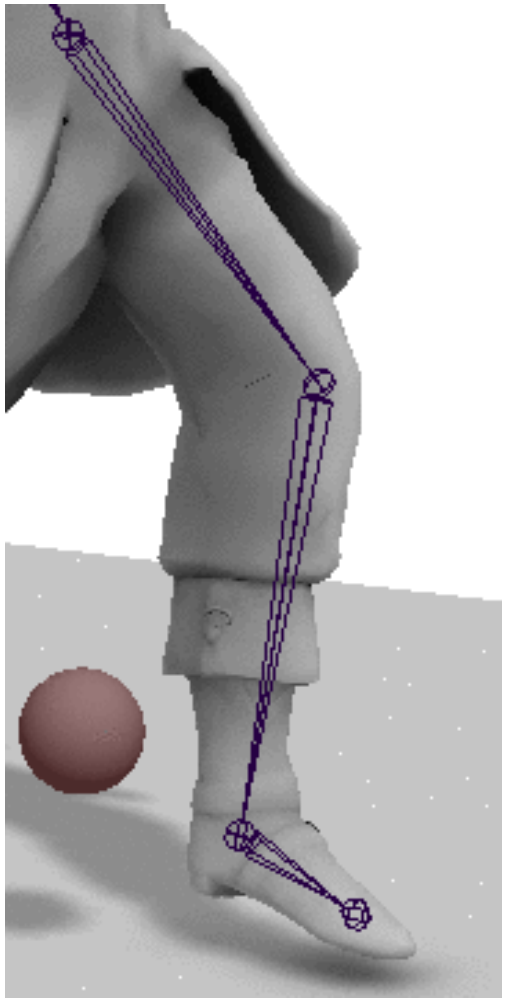


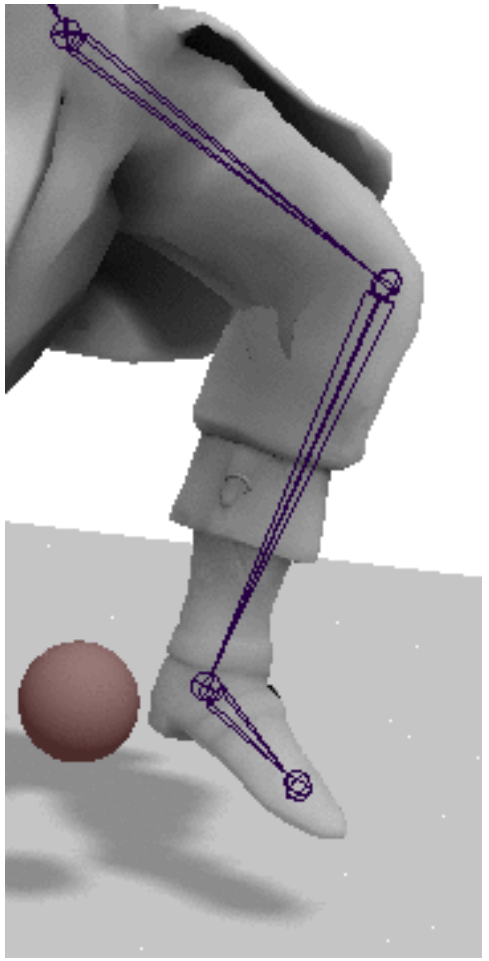


Rotate knee joint, so that $ac = at$

Two Link IK in 3D



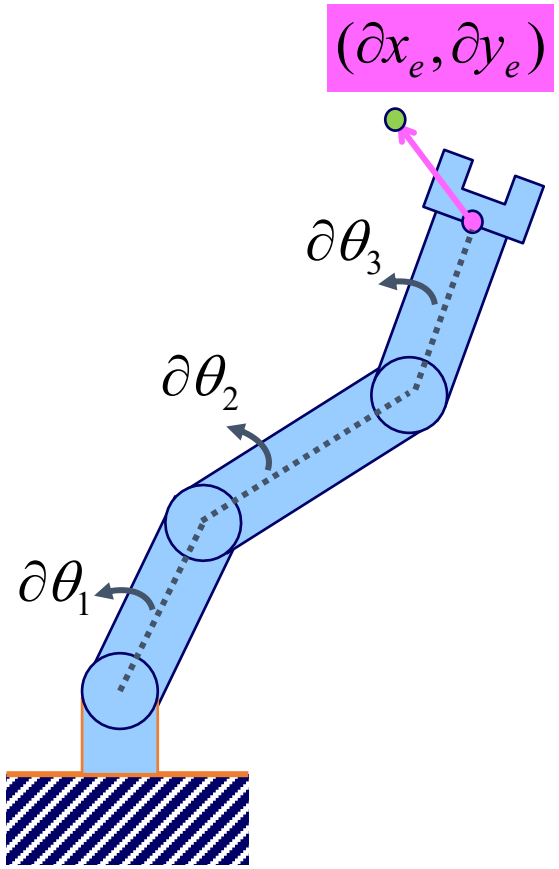




Rotate the hip joint so that c coincides with the position of target t

Numerical solutions of general problems

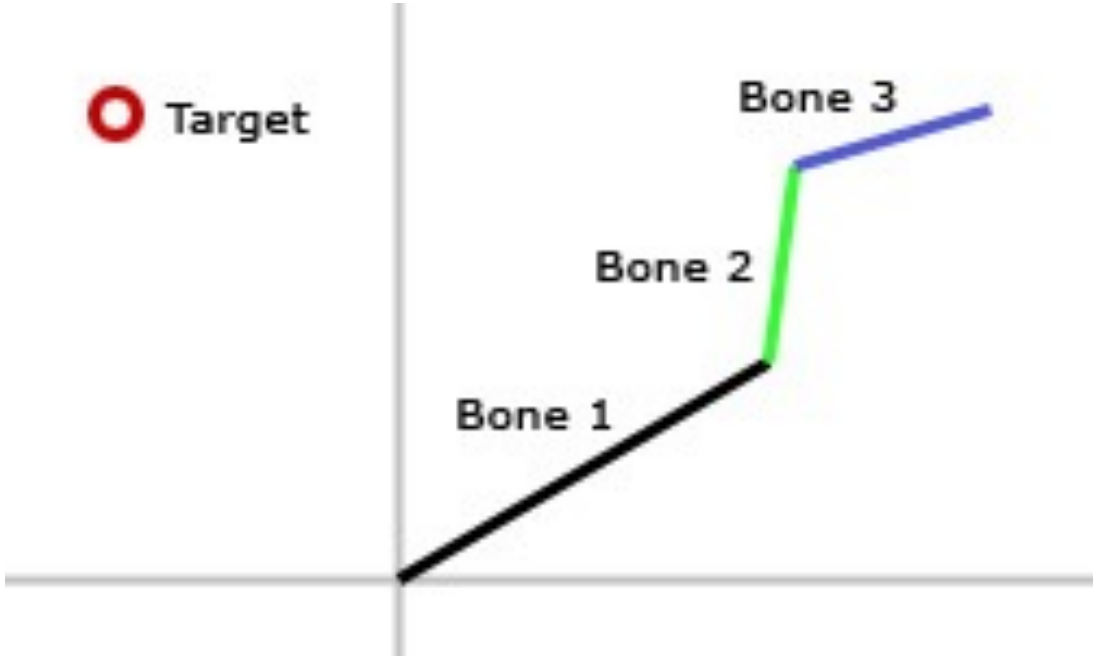
- Generally, there is no analytical solution for IK problems with joint number > 2
- Solving IK problems numerically
- Basic idea: iteration
- Starting from the initial state
  - Loop：
    - Calculate / guess the update of joint angle
    - Update the joint angle and calculate the error



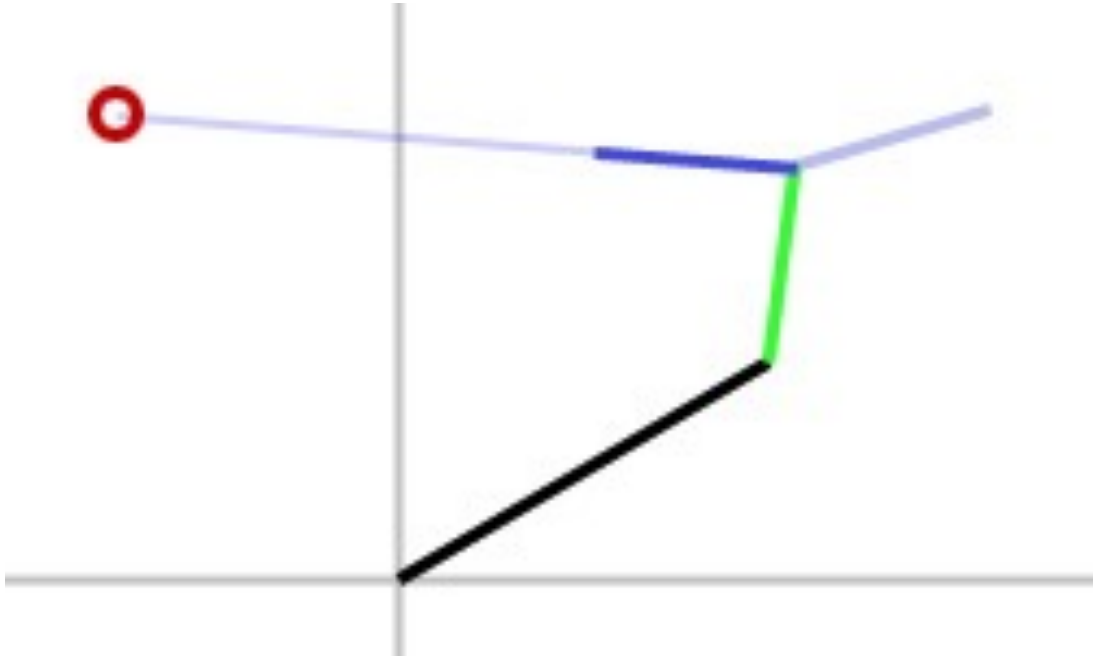
Cyclic Coordinate Descent (CCD)

- Common IK Algorithm
- starting from the end of the chain, point the line between each joint and the end to the target in turn
- cycle several times until the end of the chain coincides with the target

Cyclic Coordinate Descent (CCD)



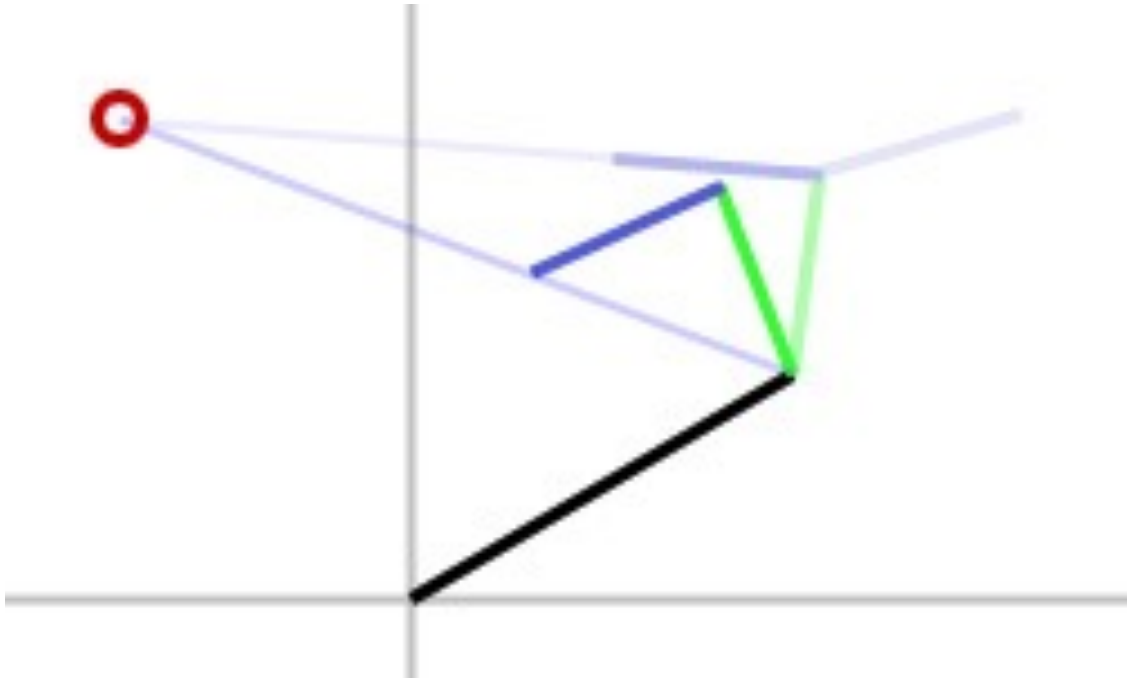
Initial State



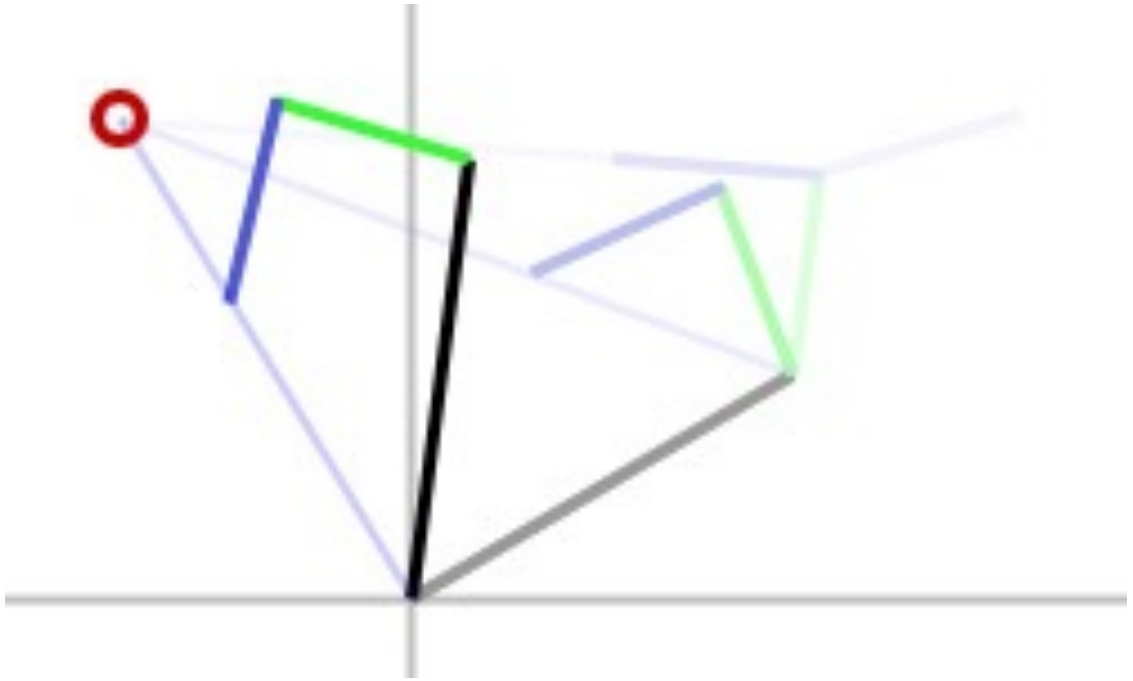
Let the connection between the blue joint and the end point to the target

https://www.ryanjuckett.com/cyclic-coordinate-descent-in-2d/

CCD IK

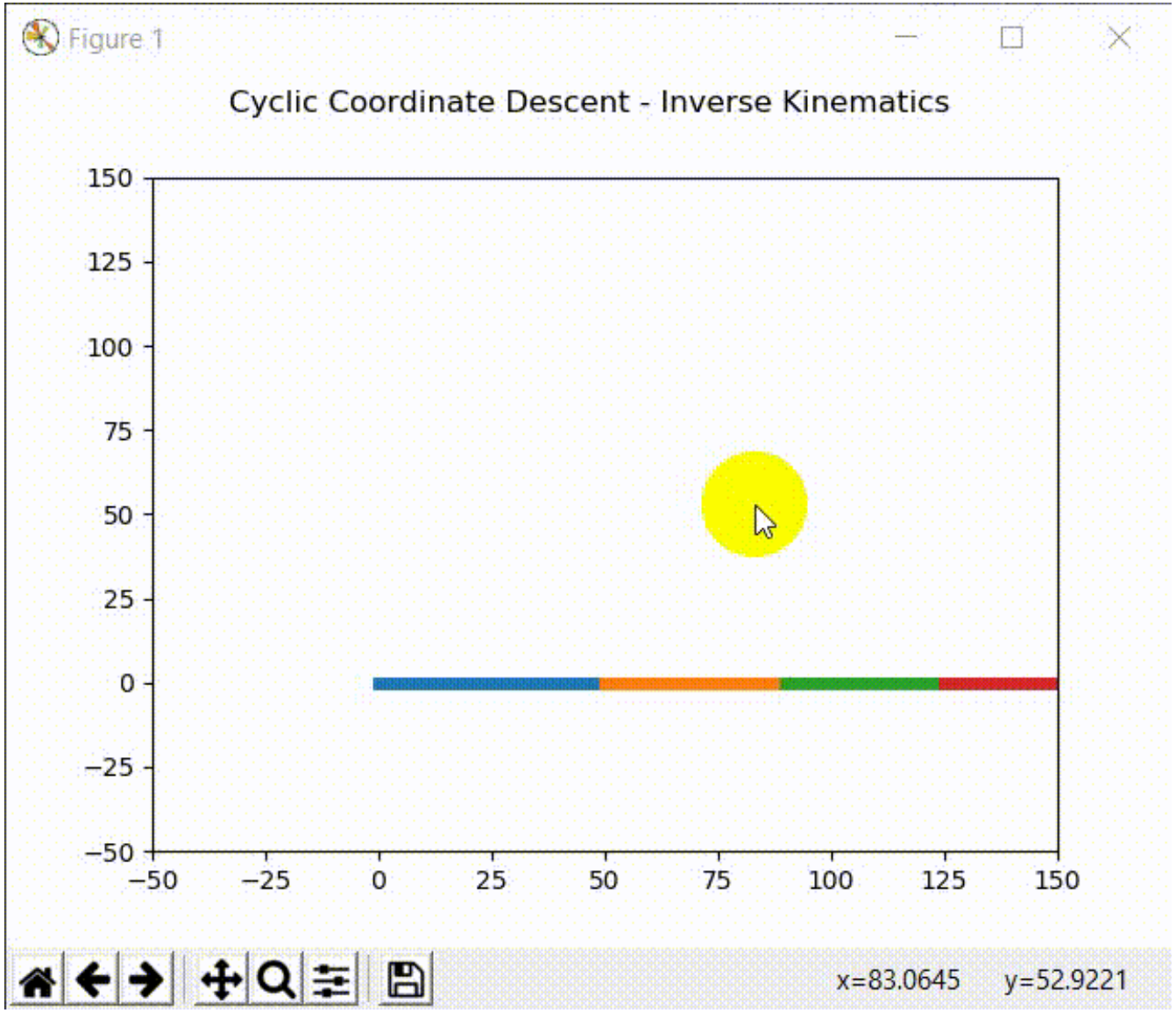


Let the connection between the green joint and the end point to the target

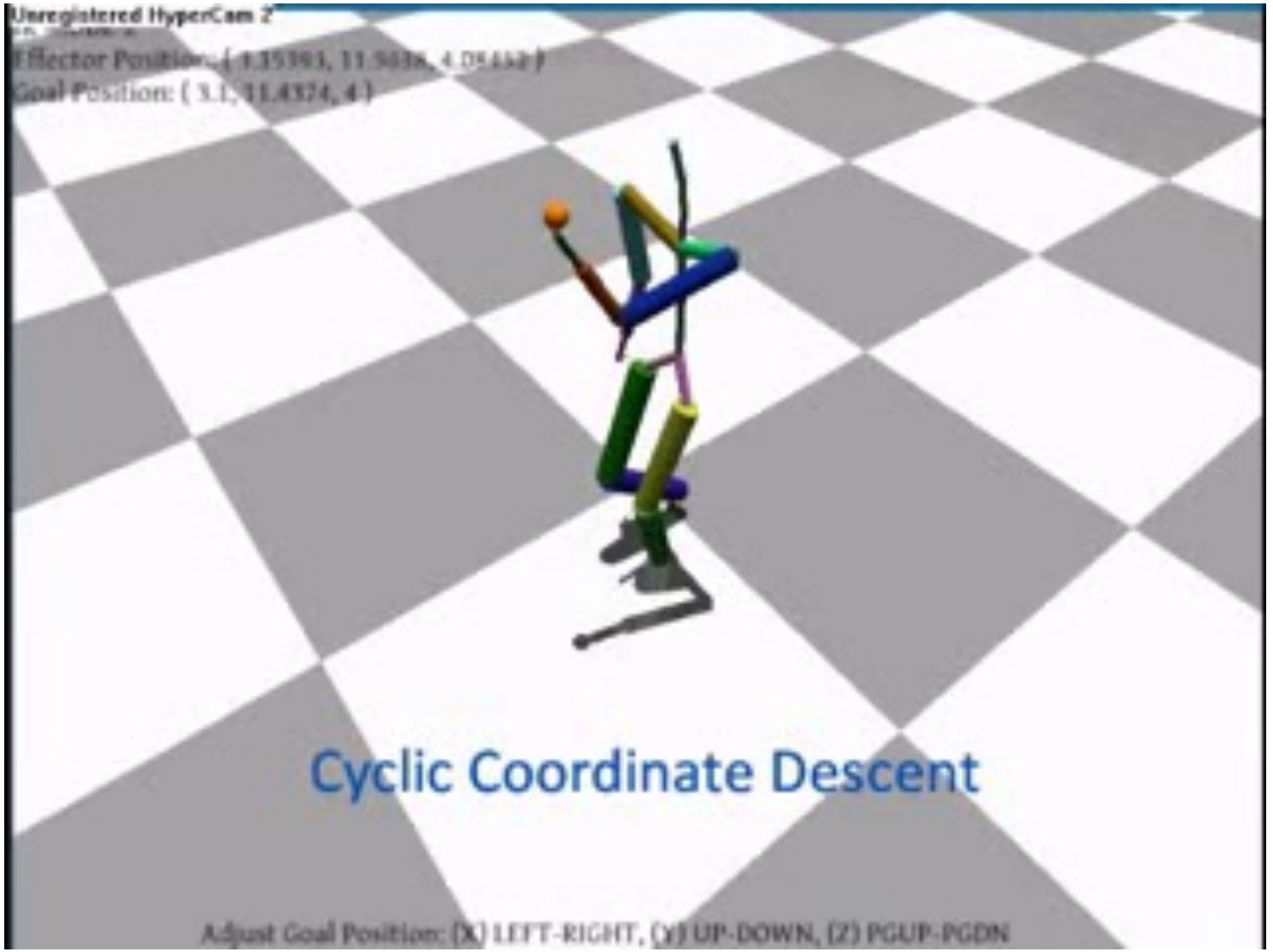


Let the line connecting the black joint and the end point to the target

Simple Demo for CCD IK



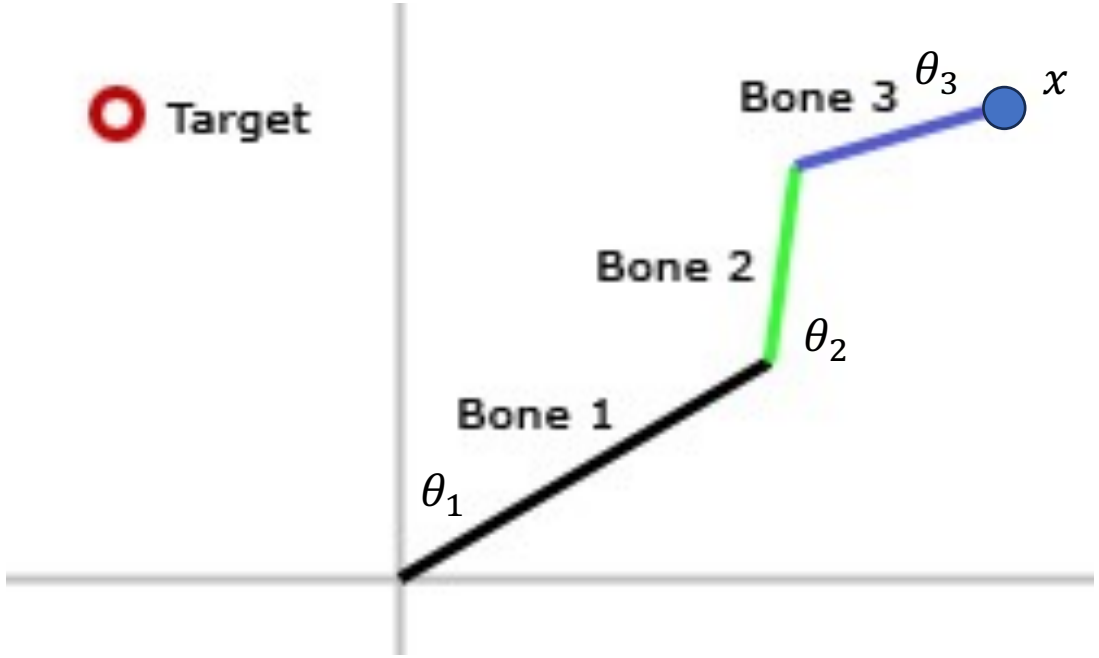
Demo for CCD IK

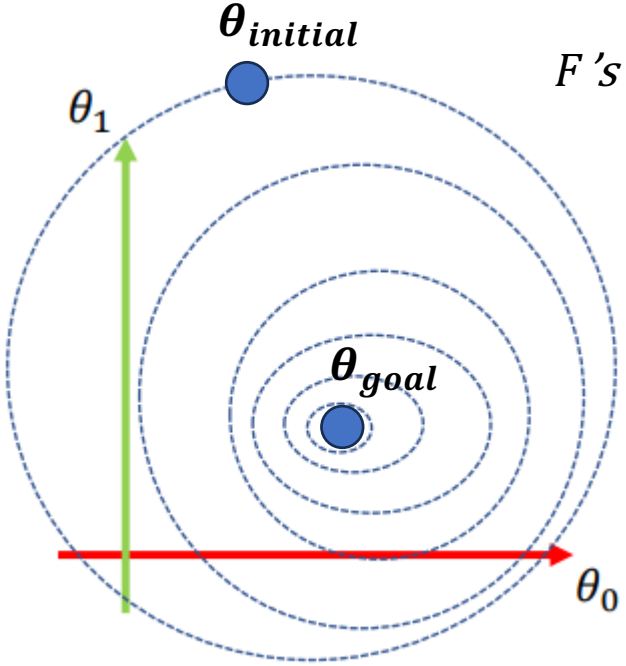


CCD IK

- Simple & Fast
- More end effector movement
- It may converge slowly or even not
- Tracking continuous targets may be unstable

IK, an Optimization problem





F's iso-surfaces, for e.g,

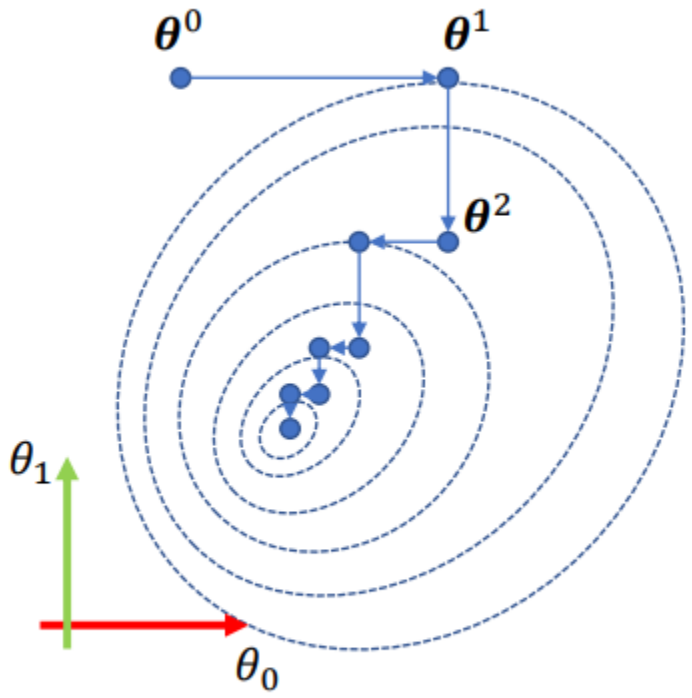
$f(\boldsymbol{\theta}) = 6$
$f(\boldsymbol{\theta}) = 5$
$f(\boldsymbol{\theta}) = 4$
$f(\boldsymbol{\theta}) = 3$
$f(\boldsymbol{\theta}) = 2$
$f(\boldsymbol{\theta}) = 1$

$$\boldsymbol{x} = f(\boldsymbol{\theta})$$

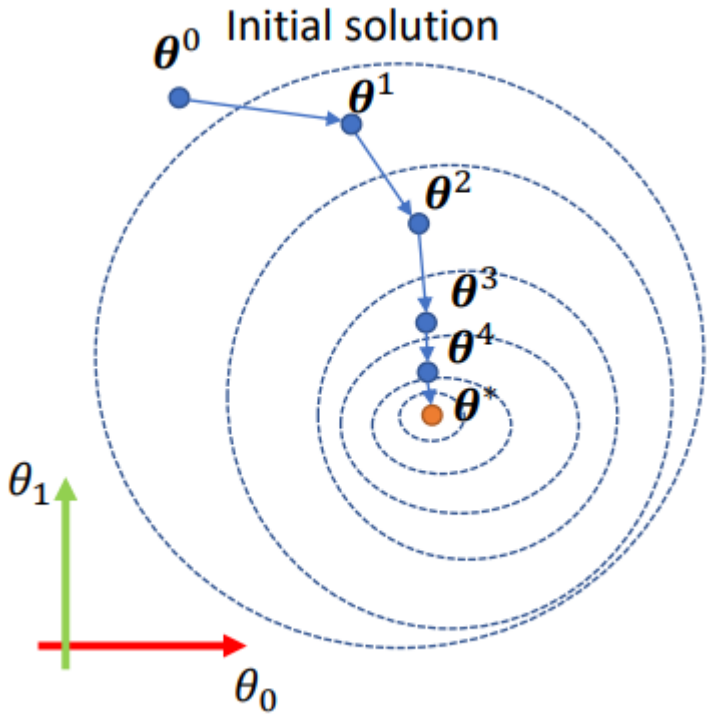
Goal: $\boldsymbol{x} = \boldsymbol{x_p}$

Optimization: $\min_{\boldsymbol{\theta}} \frac{1}{2} \left\| \mathrm{f}(\boldsymbol{\theta}) - \boldsymbol{x}_p \right\|_2^2 = \min_{\boldsymbol{\theta}} F$

Cyclic Coordinate Descent and Jacobian IK



CCD IK

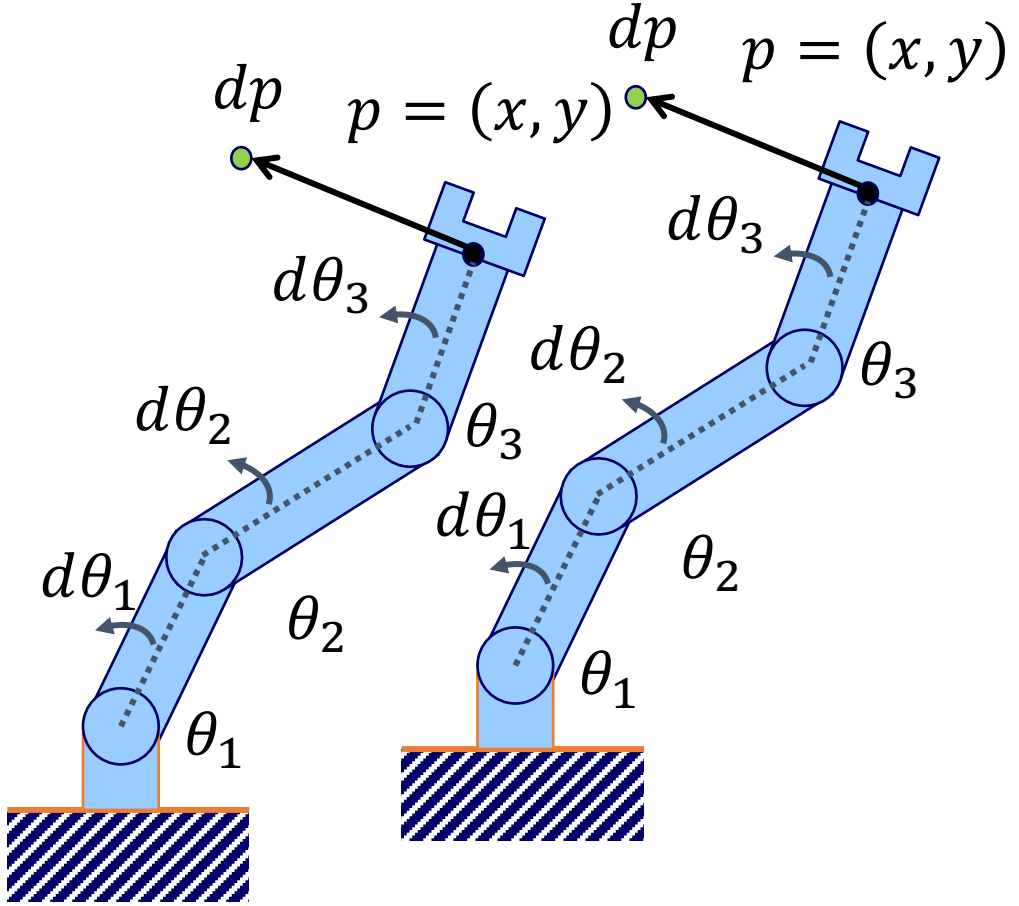


Jacobian IK

Jacobian IK

- The position of the end limb can be written as a function of the rotation angle
- $p = f(\theta_0, \theta_1, \theta_2)$
- Take the derivative on both sides

$$\boldsymbol{dp} = \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} \frac{\partial f_x}{\partial \theta_1} & \frac{\partial f_x}{\partial \theta_2} & \frac{\partial f_x}{\partial \theta_3} \\ \frac{\partial f_y}{\partial \theta_1} & \frac{\partial f_y}{\partial \theta_2} & \frac{\partial f_y}{\partial \theta_3} \end{pmatrix} \begin{pmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \end{pmatrix} = \mathbf{J}\mathbf{d}\boldsymbol{\theta}$$

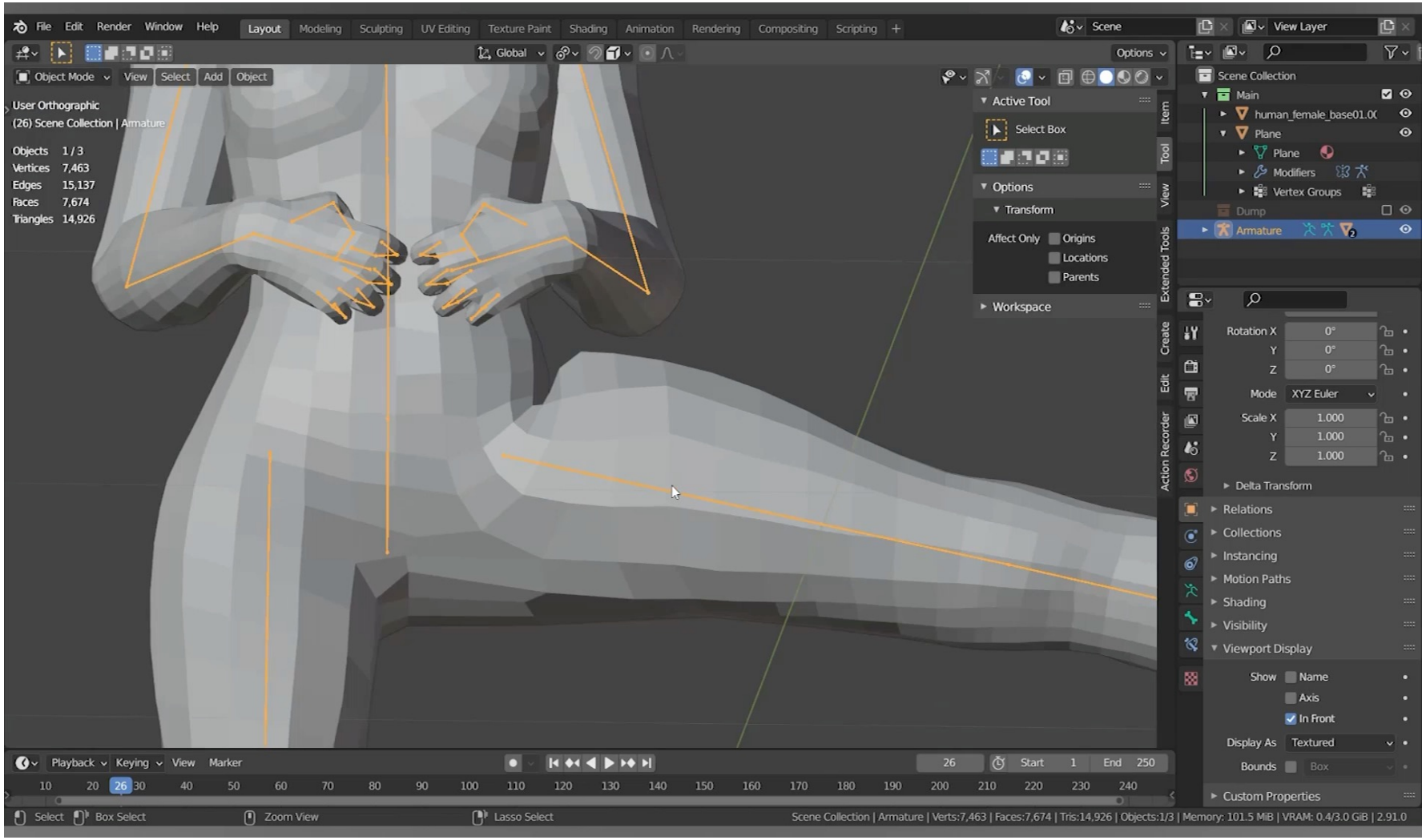


How to Compute Jacobian Matrix?

- Analytical derivation
  - Automatic derivation tool: PyTorch, TensorFlow
- Finite difference
- Geometric approach

Rigging and Skinning/Binding

Rigging and Skinning



Skinning/Binding

Human Body Rig in Blender
https://www.youtube.com/watch?v=MAM7mF2v7dE

Skinning/Binding

Face Rig
https://www.youtube.com/watch?v=ueYtM2KprqY