Animation and Kinematics

Some Slides/Images adapted from Marschner and Shirley and David Levin

Animation and Kinematics

Agenda:

- Animation in Computer Graphics
- Forward Kinematics
- Skinning for Mesh Deformation
- Keyframe Animation + Splines
- Inverse Kinematics

“Core” Areas of Computer Graphics

Modeling/Geometry

Rendering

Animation

Animation Timeline

1908: Emile Cohl (1857-1938) France, makes his first film,

FANTASMAGORIE, arguably the first animated film.

1911: Winsor McCay (1867-1934) makes his first film, LITTLE NEMO. McCay, already famous for comic strips, used the film in his vaudeville act. His advice on animation:

Any idiot that wants to make a couple of thousand drawings for a hundred feet of film is welcome to join the club.

1928: Walter Disney (1901-1966) working at the Kansas City Slide Company creates Mickey Mouse.

1974: First Computer animated film “Faim” from NFB nominated for an Oscar.

Animation Principles

- Squash & Stretch
- Timing
- Ease-In & Ease-Out
- Arcs
- Anticipation
- Follow-through & Secondary Motion
- Overlapping Action & Asymmetry
- Exaggeration
- Staging
- Appeal
- Straight-Ahead vs. Pose-to-Pose

Case Study: Squash and Stretch

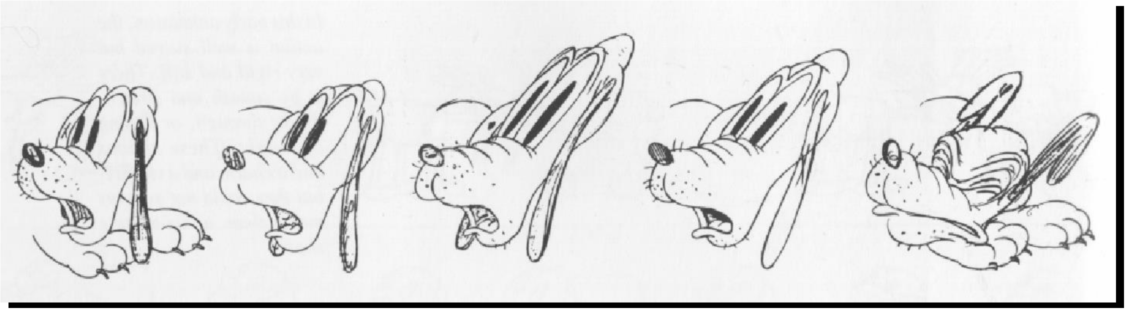
Rigid objects look robotic: deformations make motion natural

Accounts for physics of deformation

- Think squishy ball…
- Communicates to viewer what the object is made of, how heavy it is, …
- Usually large deformations conserve volume: if you squash one dimension, stretch in another to keep mass constant

Also accounts for persistence of vision

- Fast moving objects leave an elongated streak on our retinas



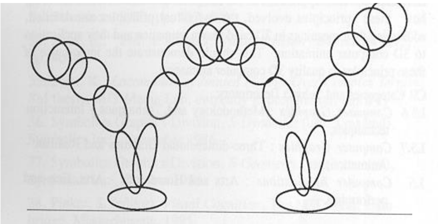


FIGURE 2. Squash & stretch in bouncing ball.

What can be animated?

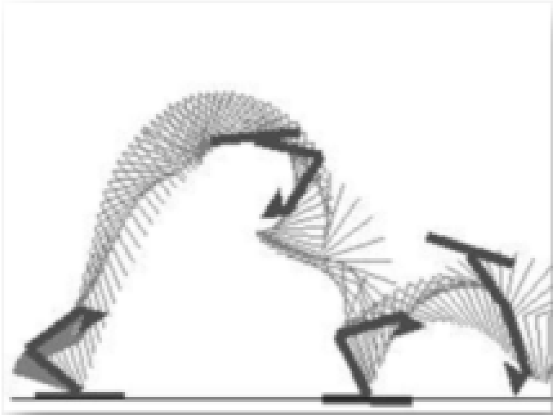
- Lights
- Camera
- **Jointed figures**
- **Skin/muscles**
- **Deformable objects**
- Clothing
- Wind/water/fire/smoke
- Hair

…any variable, Given the right time scale, almost anything…

Approaches to Animation

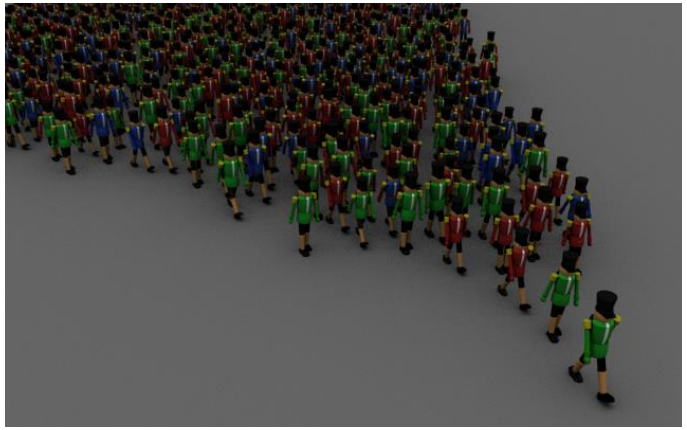
How does one make digital models move?

Keyframing



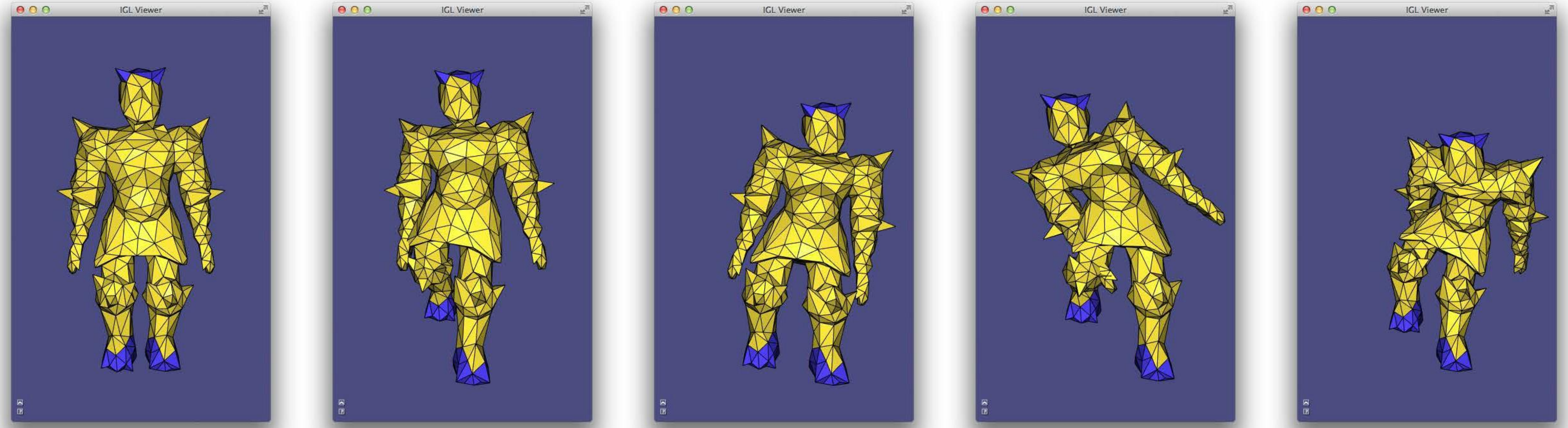
Physical simulation

Motion capture



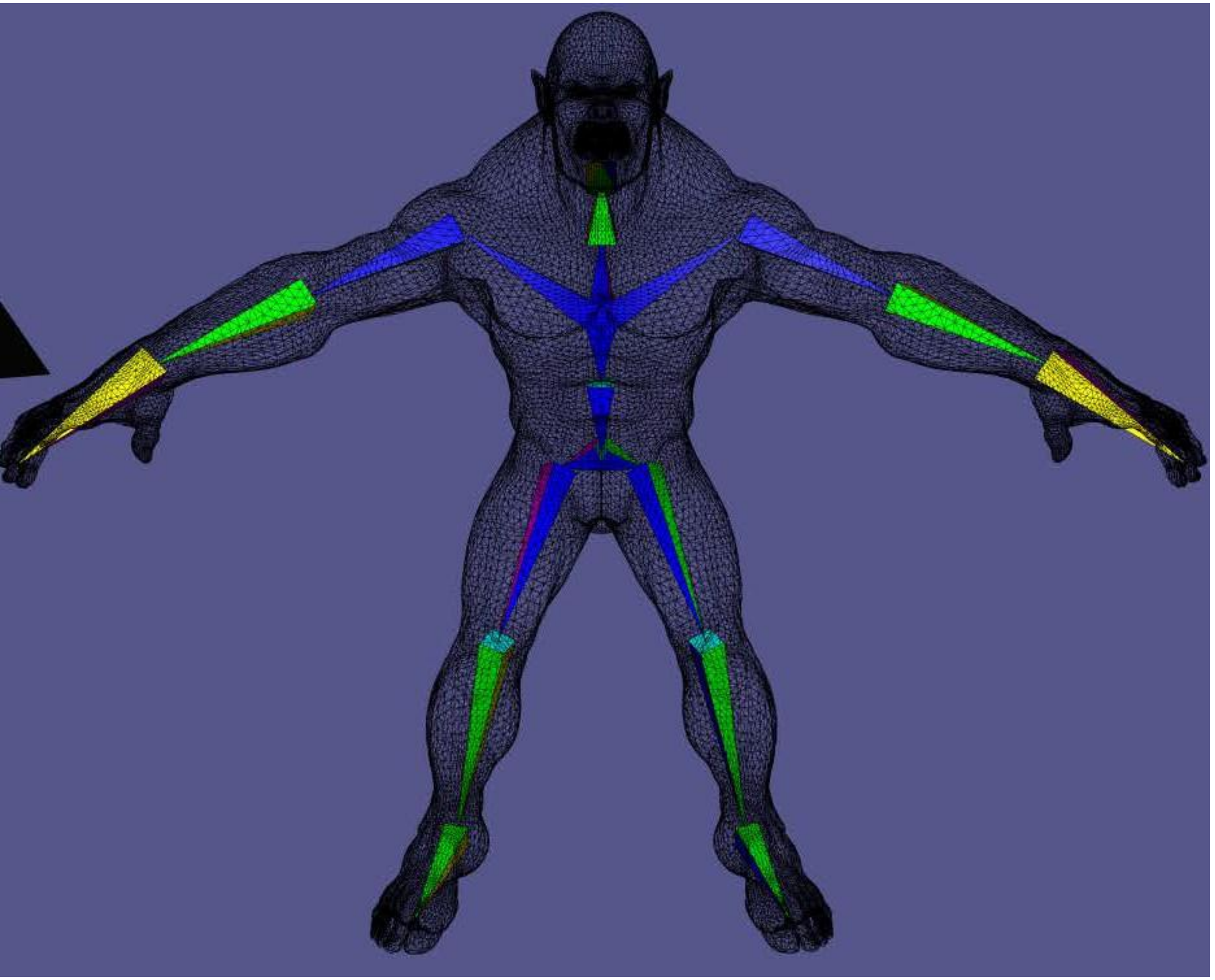
Behavior rules

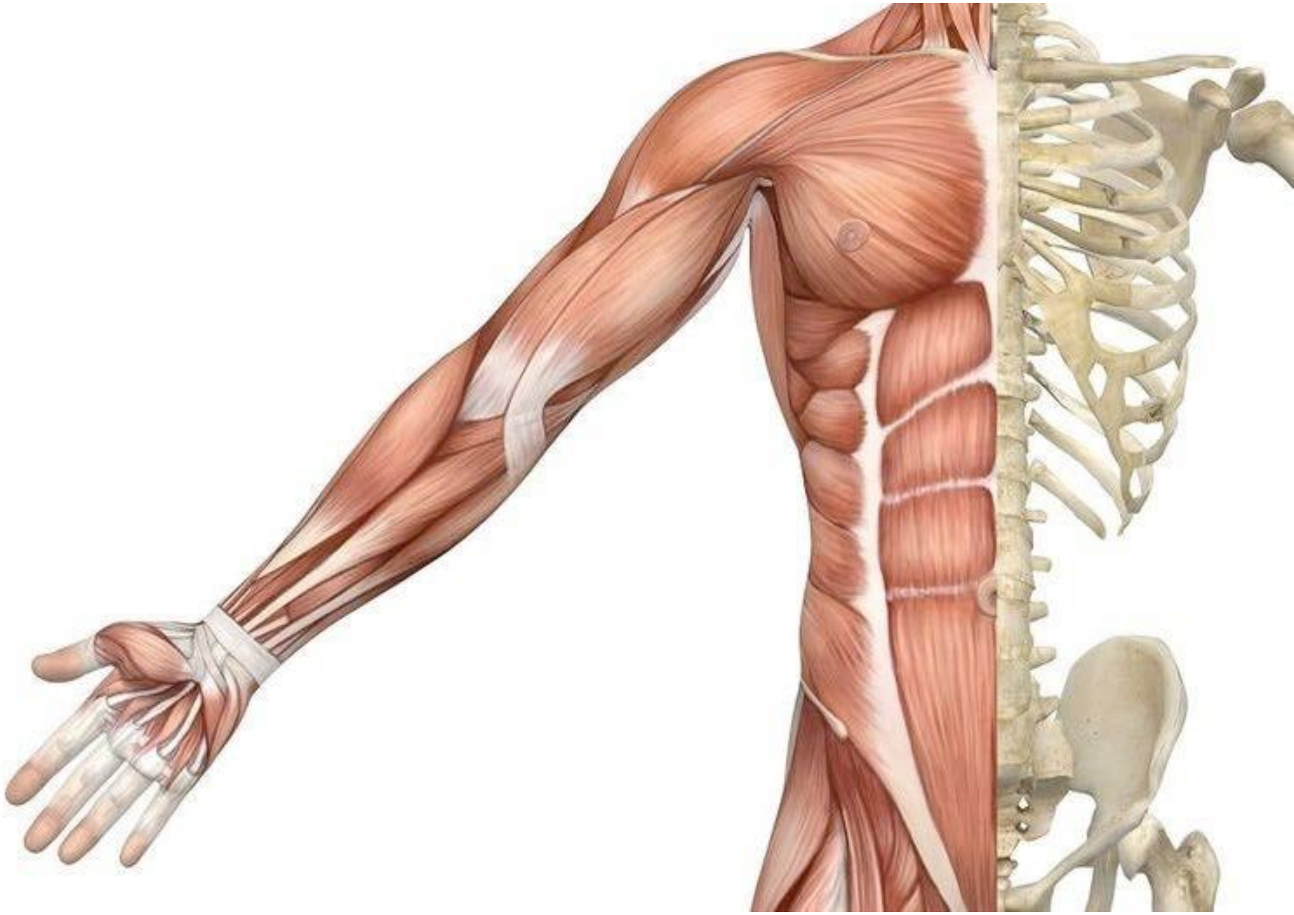
How do articulated characters move?
What to keyframe/simulate/animate?



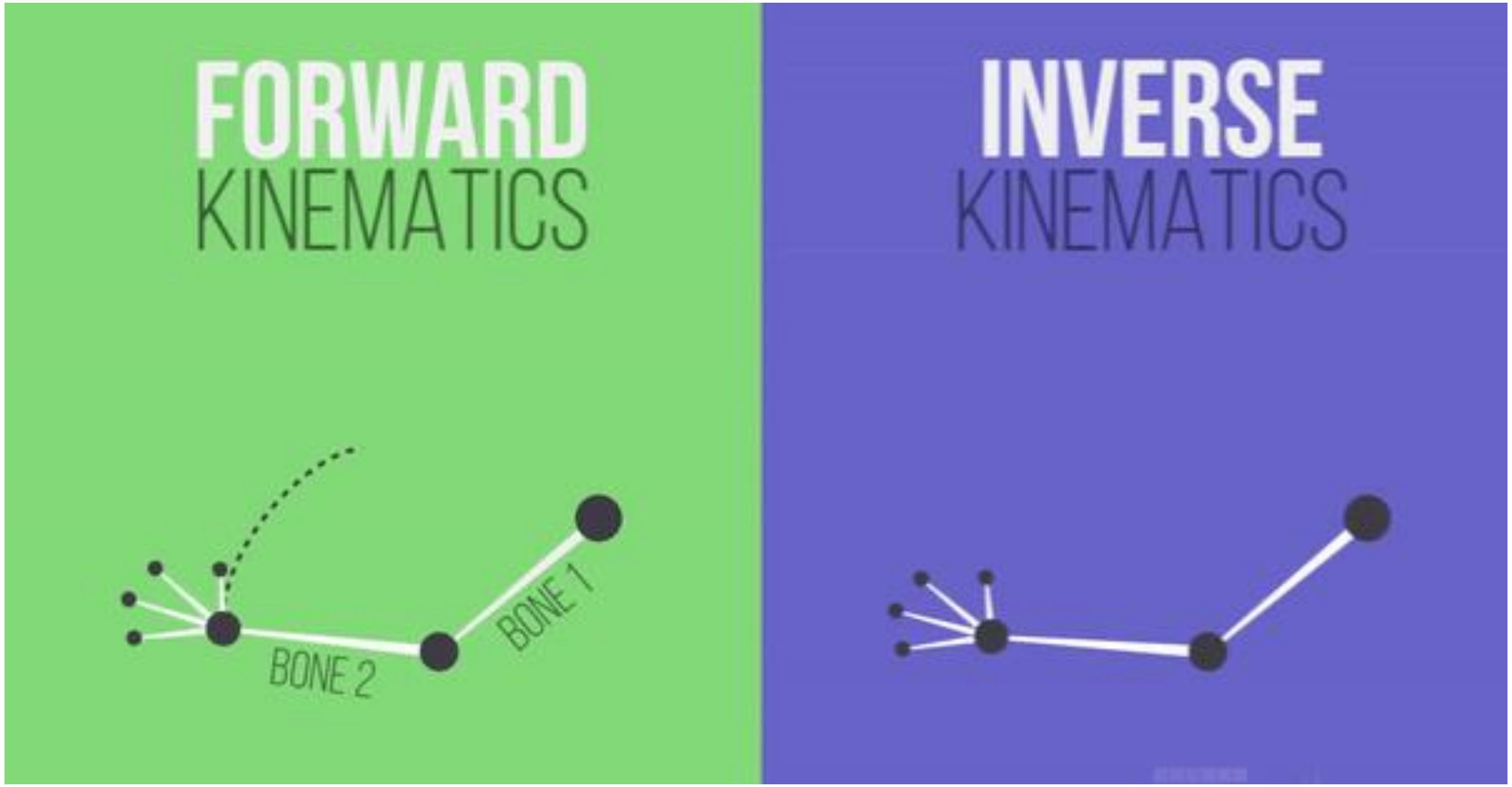
Per-Vertex?

Skeleton: hierarchy of bones/joints



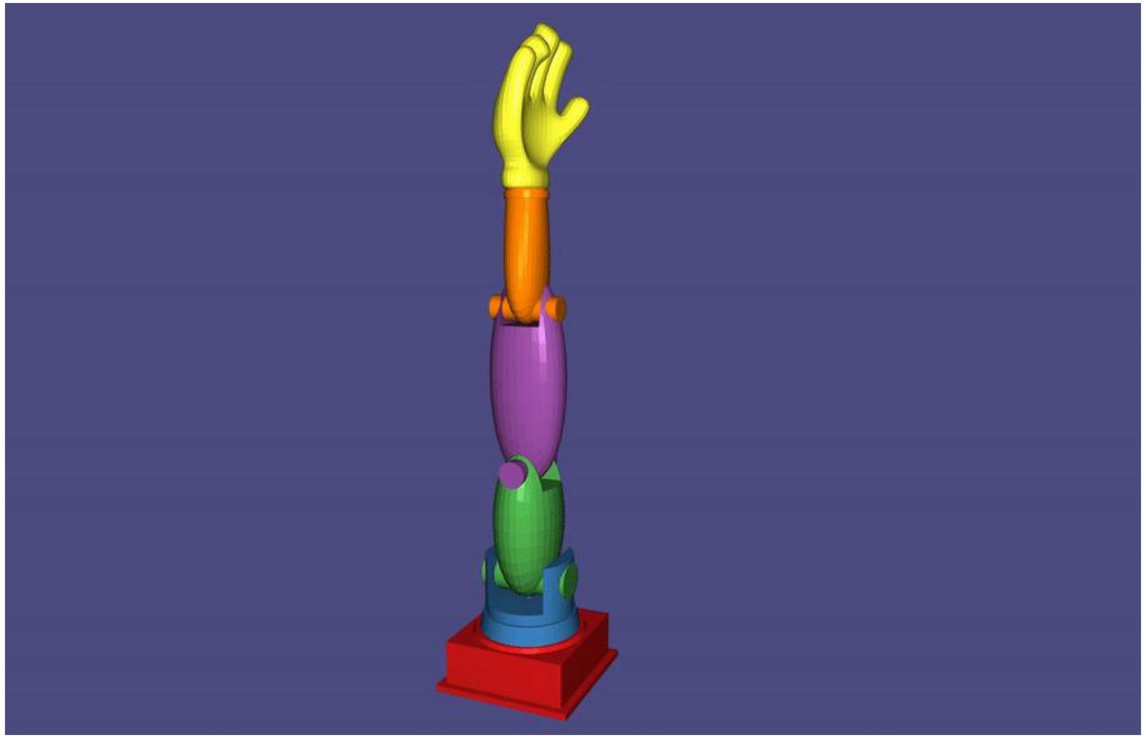


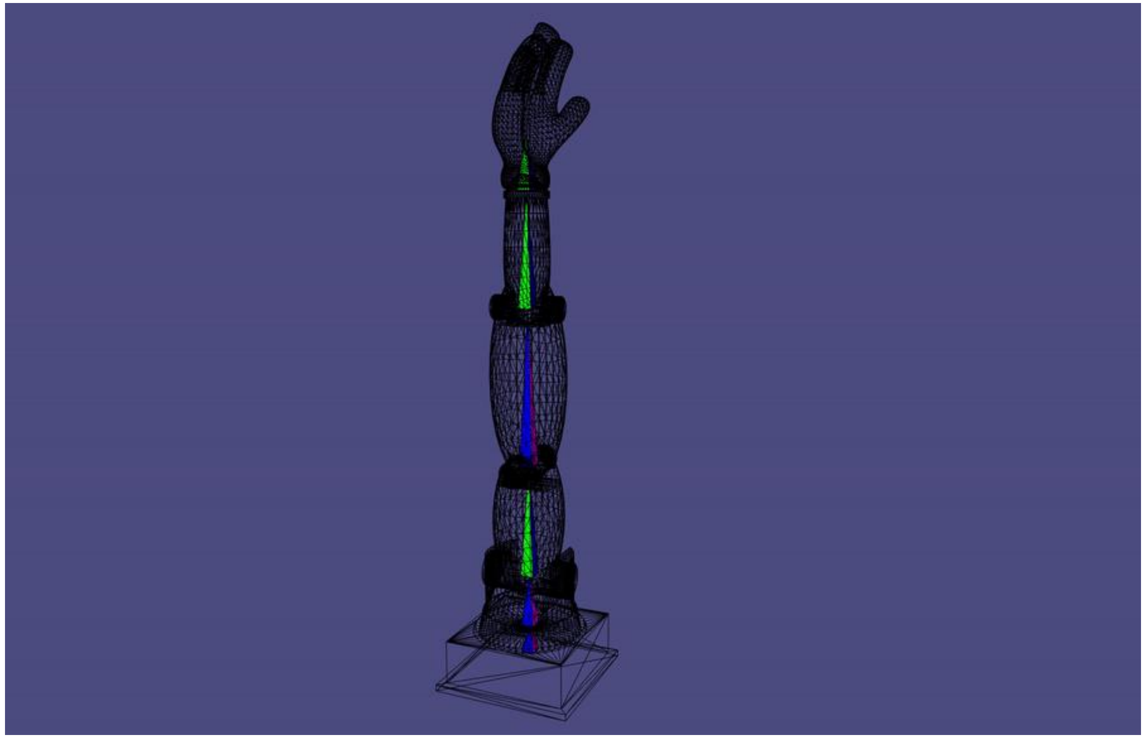
Posing a skeleton:
Forward Kinematics vs. Inverse Kinematics



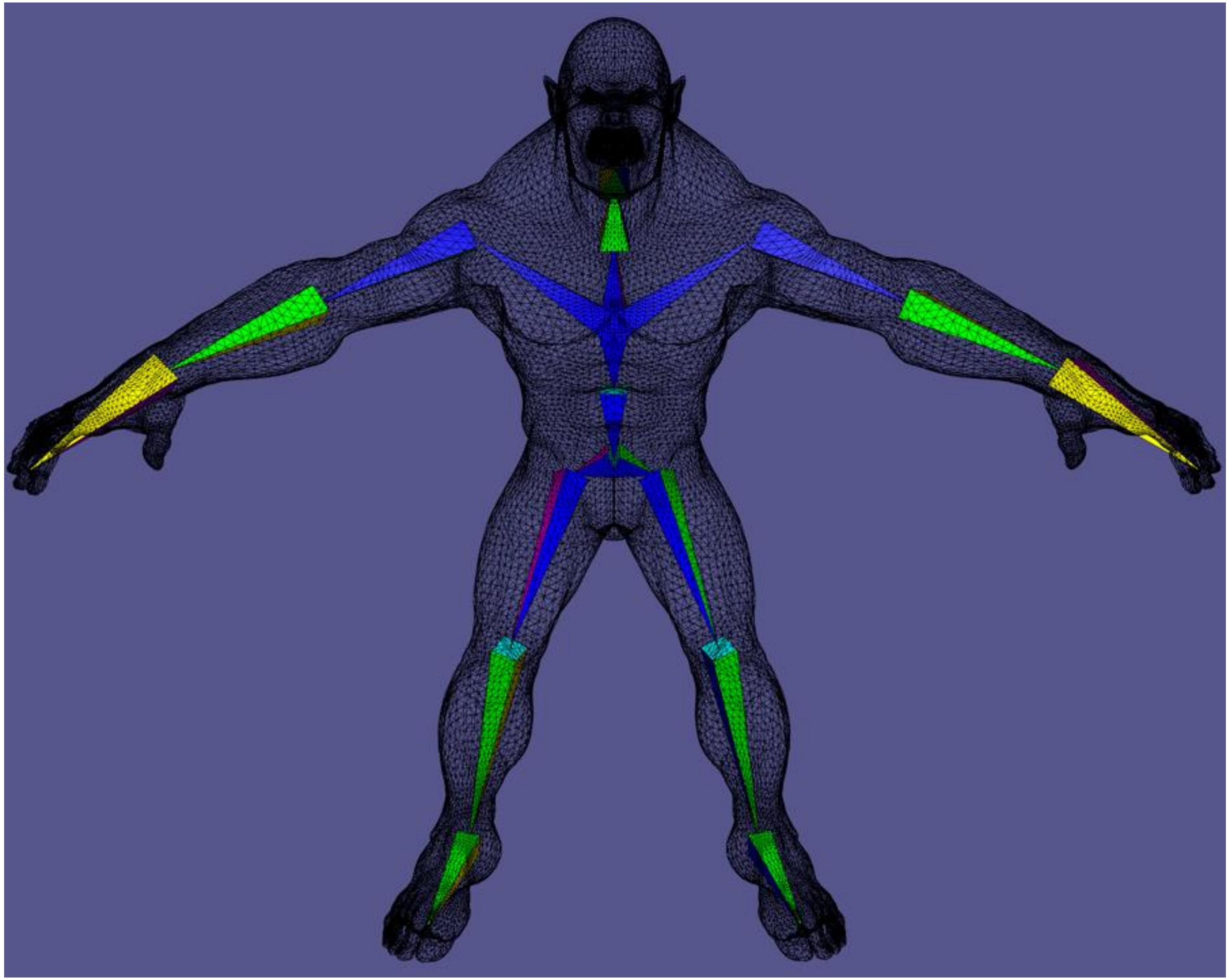
https://www.youtube.com/watch?v=0a9qIj7kwiA

Skeletons





Deforming the object: Skinning

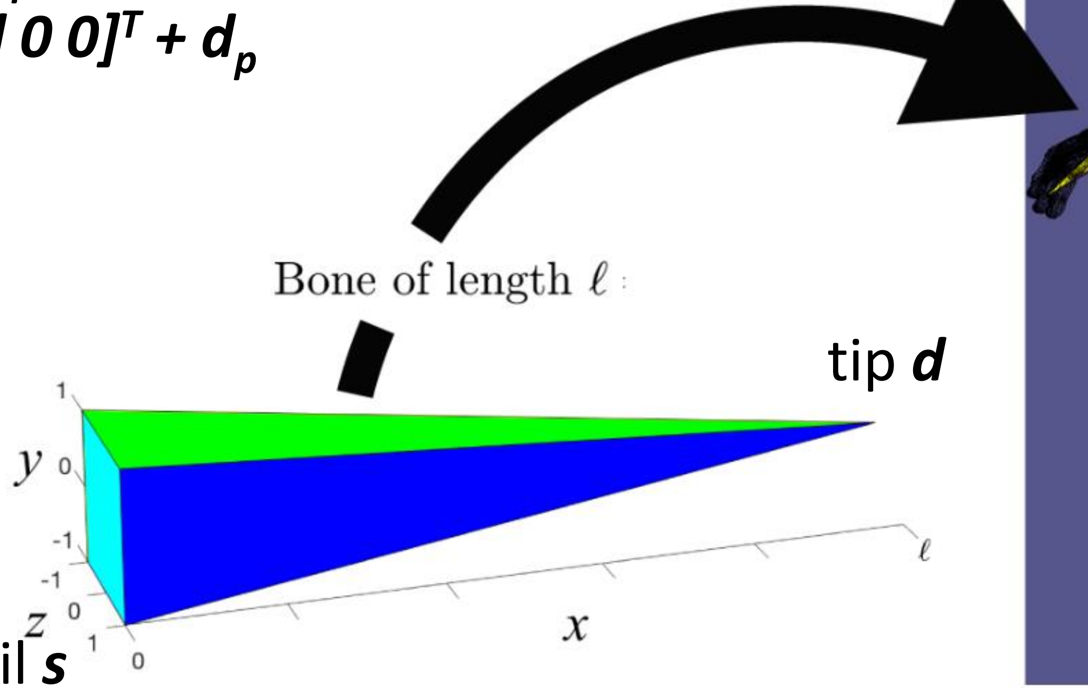


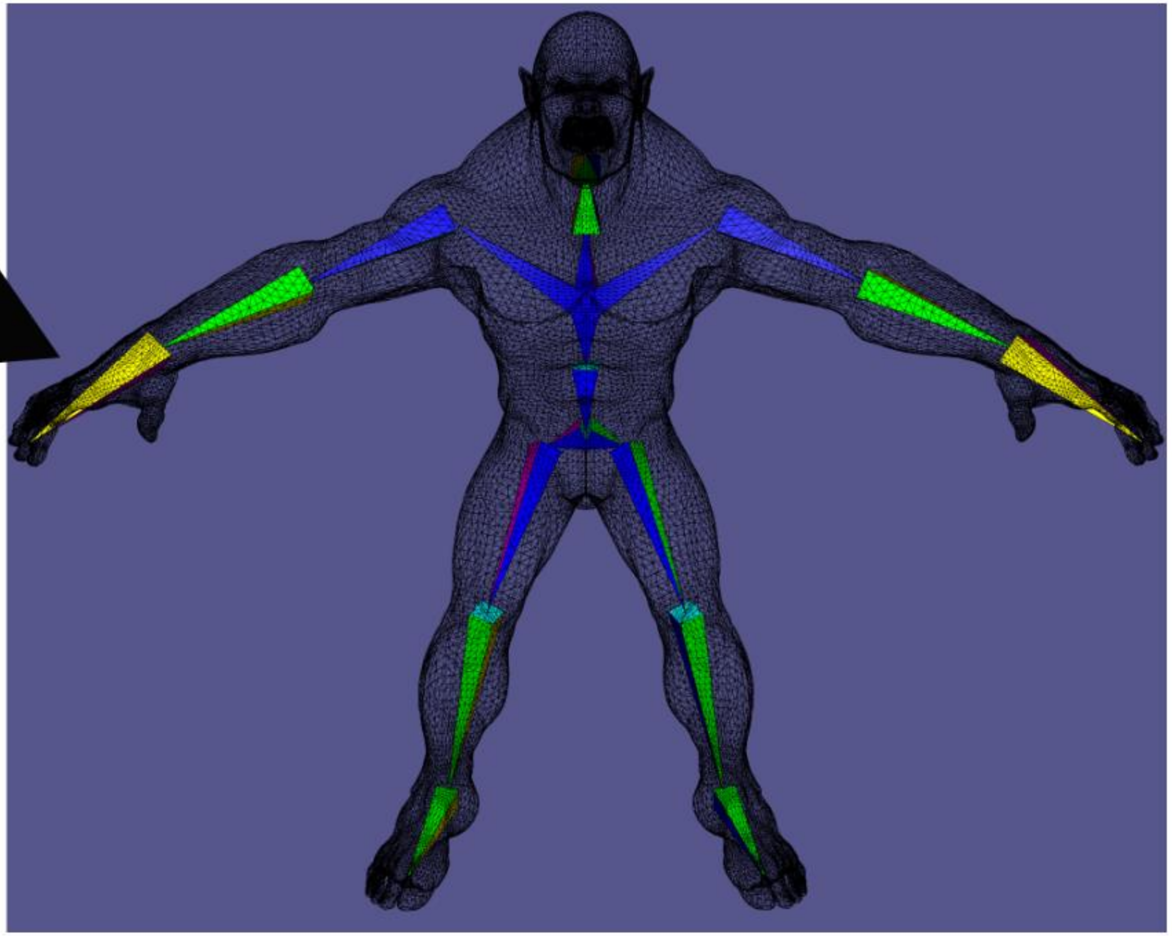
Skeletons: Rest Bone

$t = s = d_p$

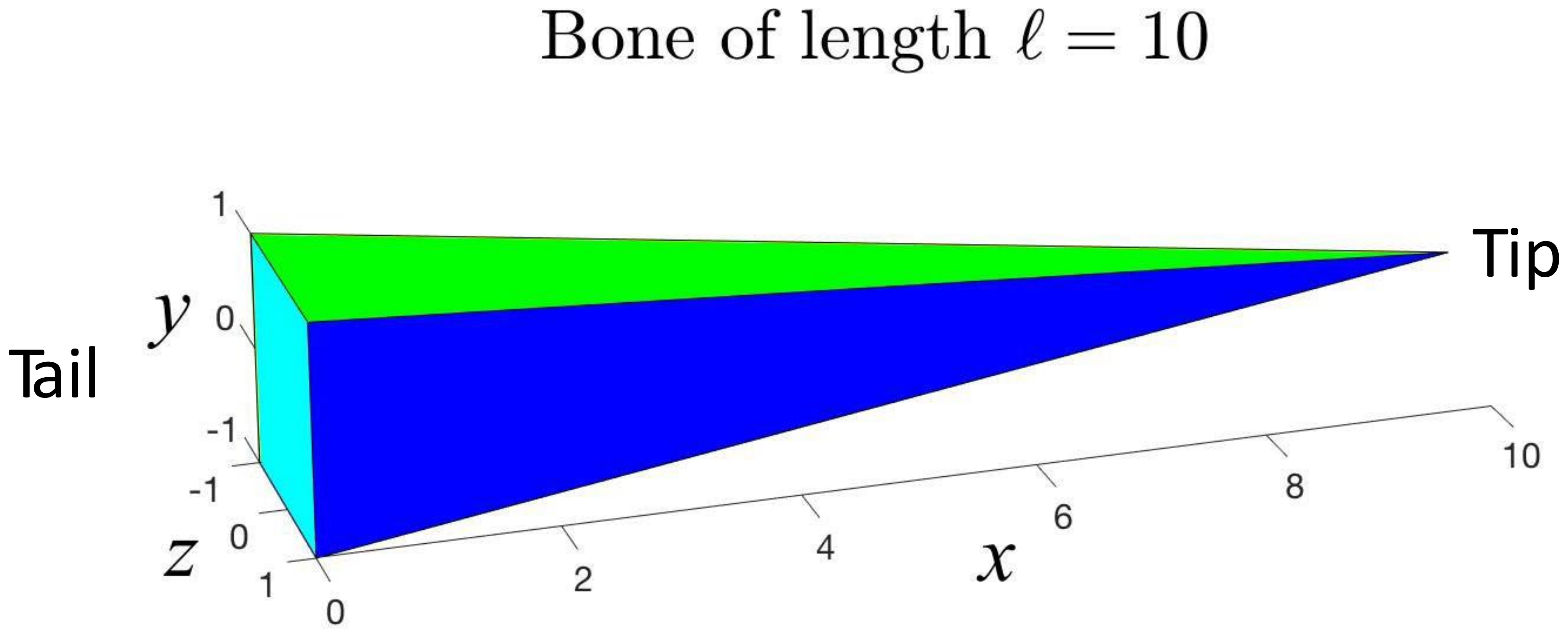
$d = R[l\ 0\ 0]^T + d_p$

$$\widehat{\mathbf{T}} = (\widehat{\mathbf{R}} \quad \hat{\mathbf{t}}) \in \mathbb{R}^{3\times4}$$

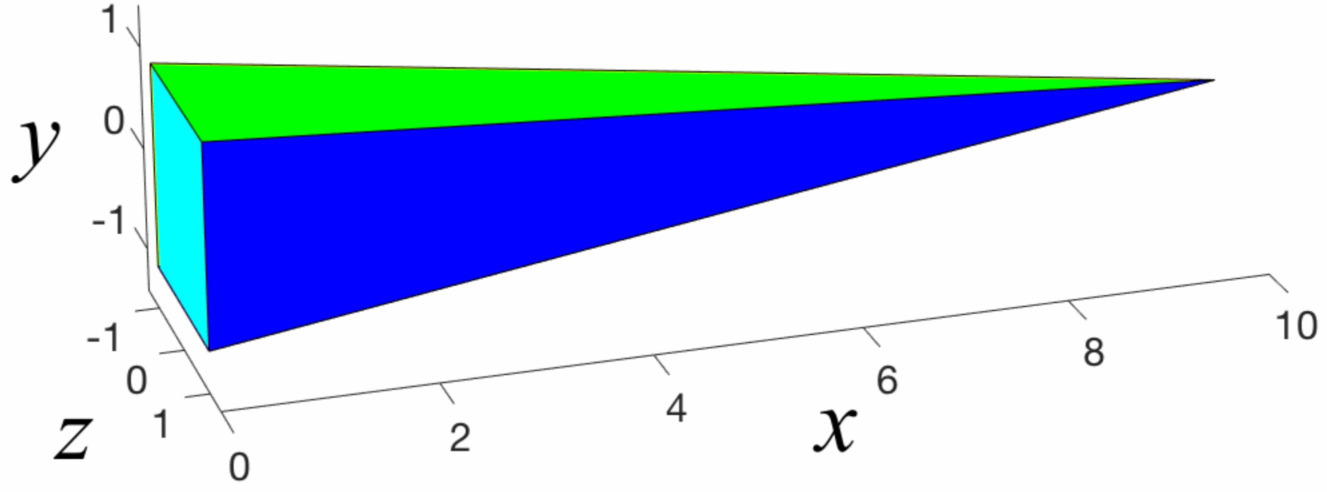




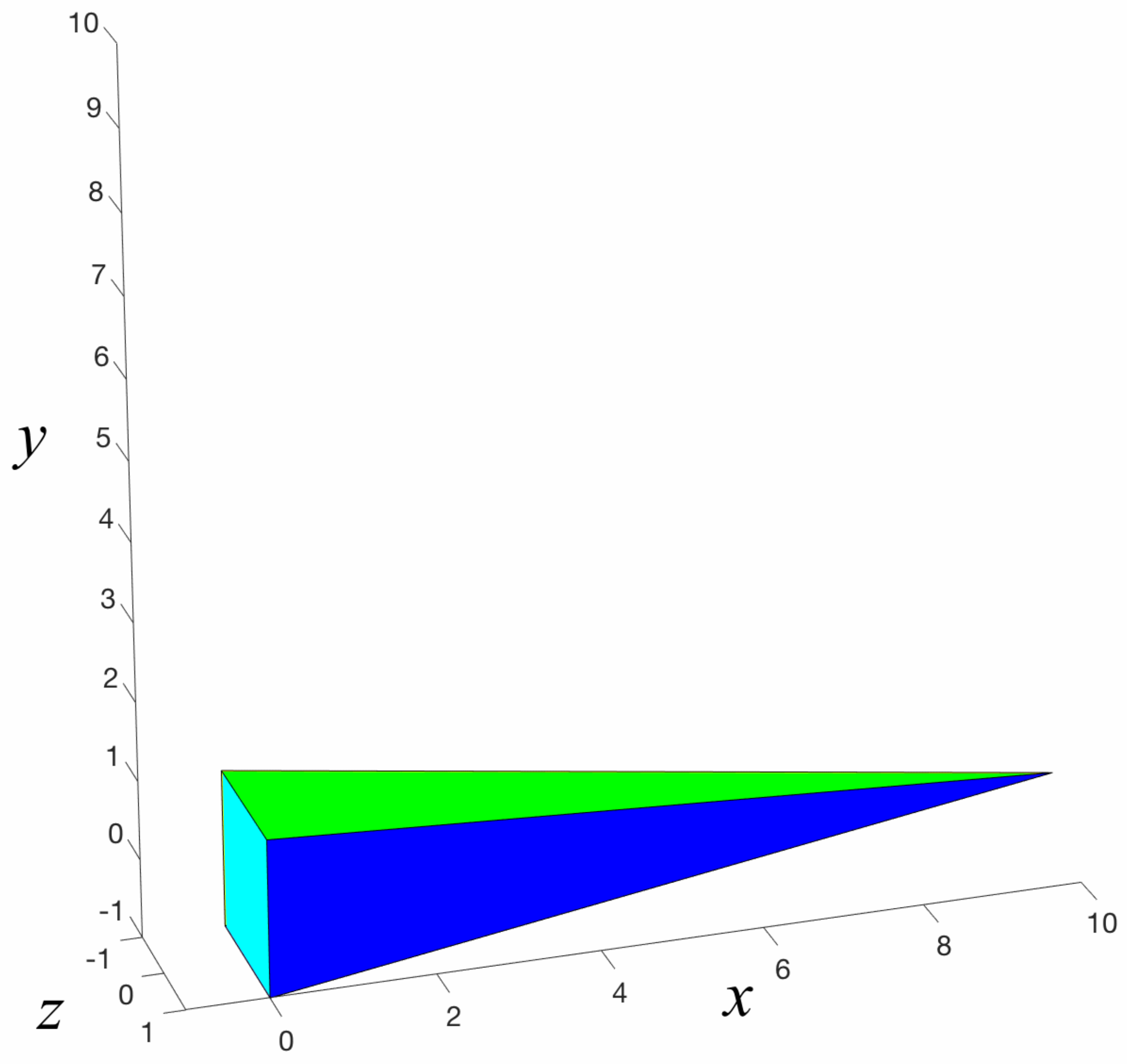
Skeletons: Specifying Rotations



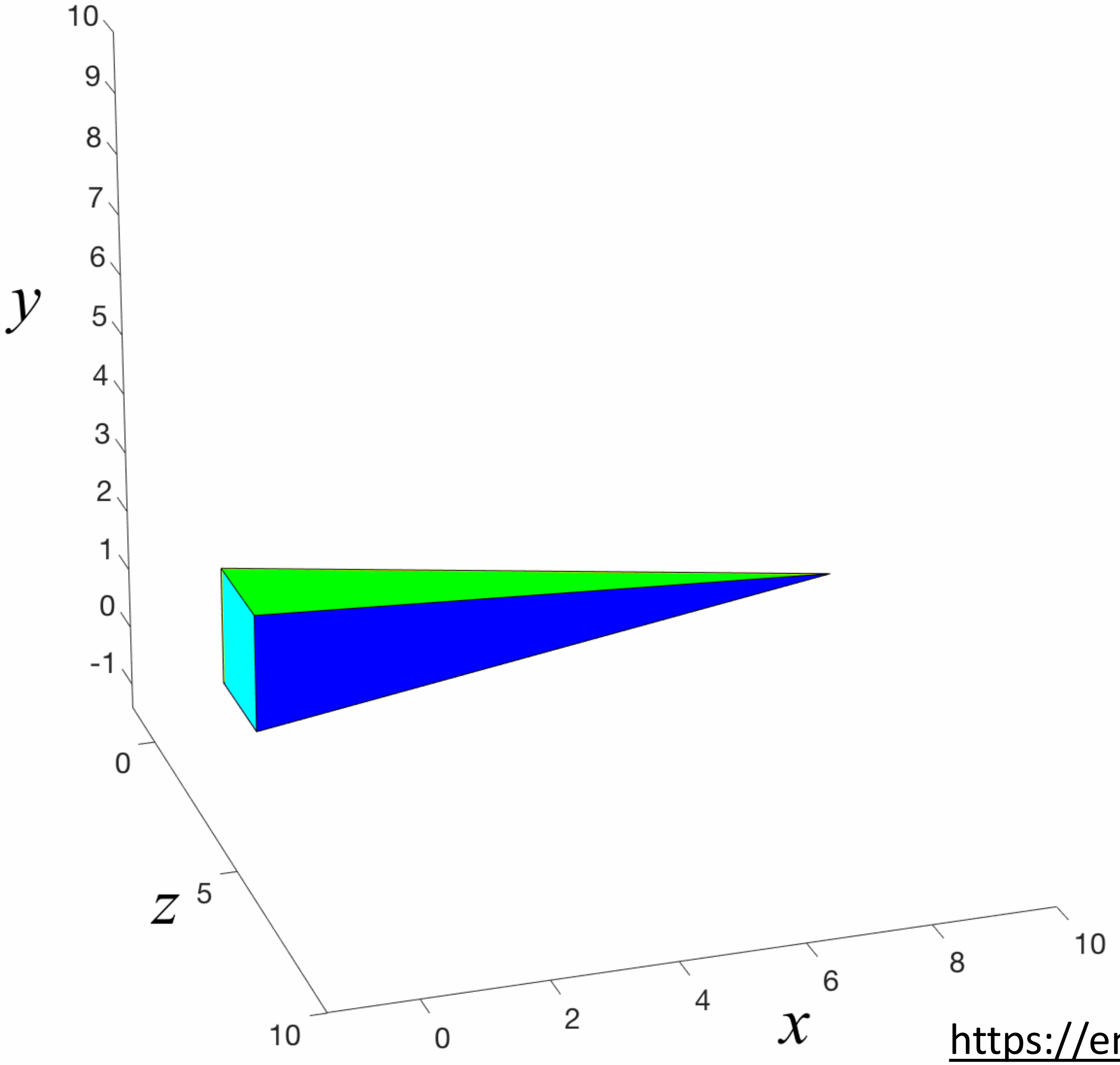
Twisting around x axis: $\theta_1 = 0^\circ$



Bending around z axis: $\theta_2 = 0^\circ$



Twist-bend-twist: $(\theta_1, \theta_2, \theta_3) = (0^\circ, 0^\circ, 0^\circ)$

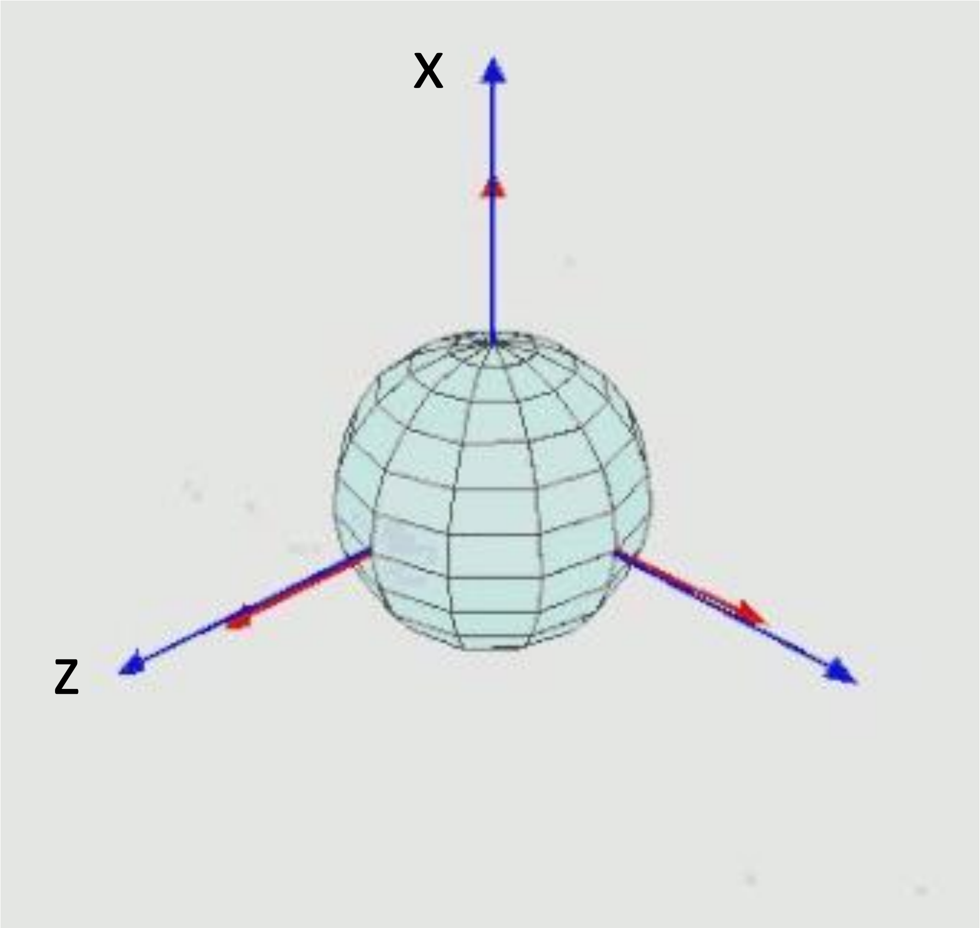


https://en.wikipedia.org/wiki/Euler_angles

https://mathworld.wolfram.com/EulerAngles.html

Euler Angle Rotations

$$R_x(\theta_3) * R_z(\theta_2) * R_x(\theta_1)$$



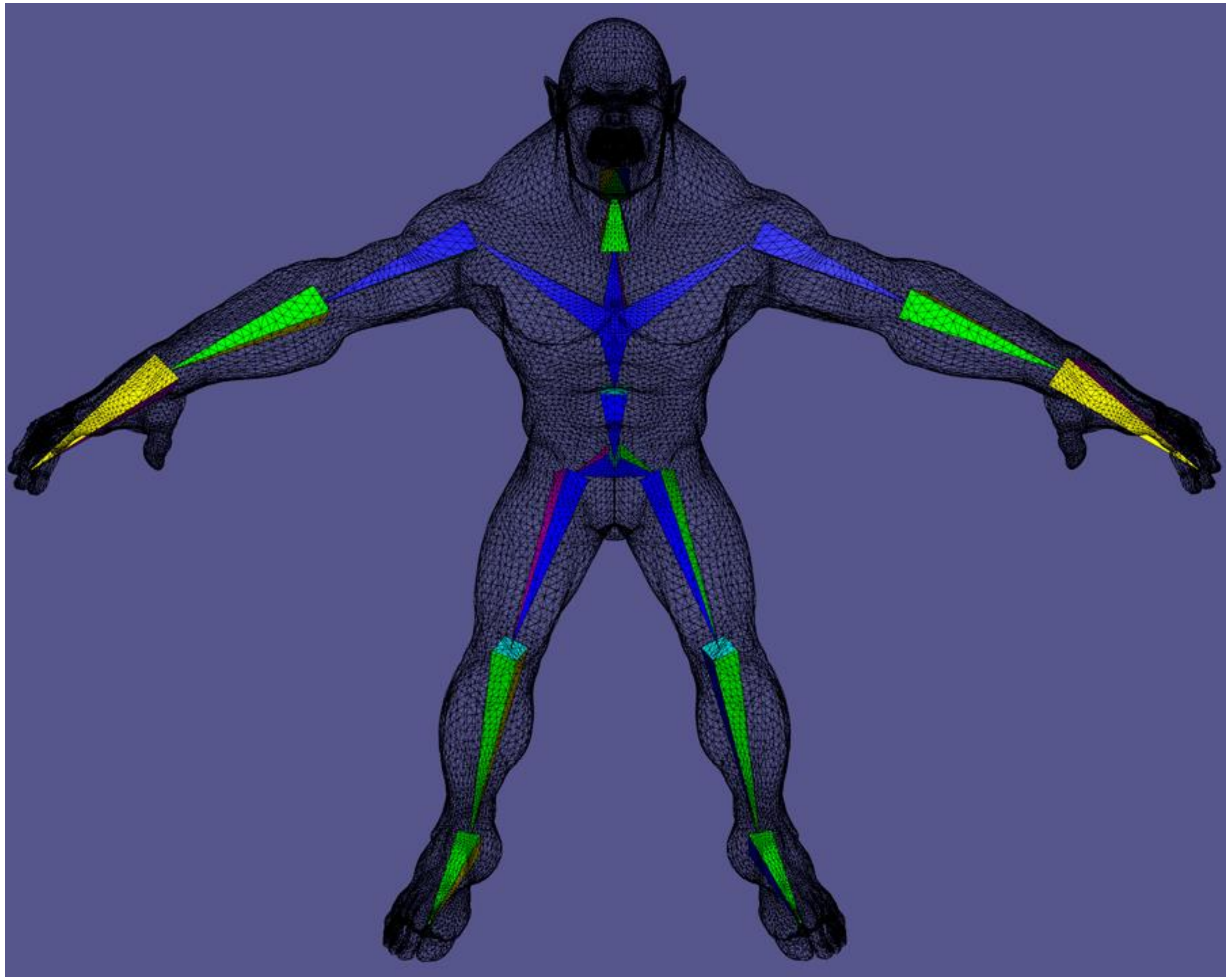
https://en.wikipedia.org/wiki/Euler_angles

https://mathworld.wolfram.com/EulerAngles.html

Skeletons: Forward Kinematics

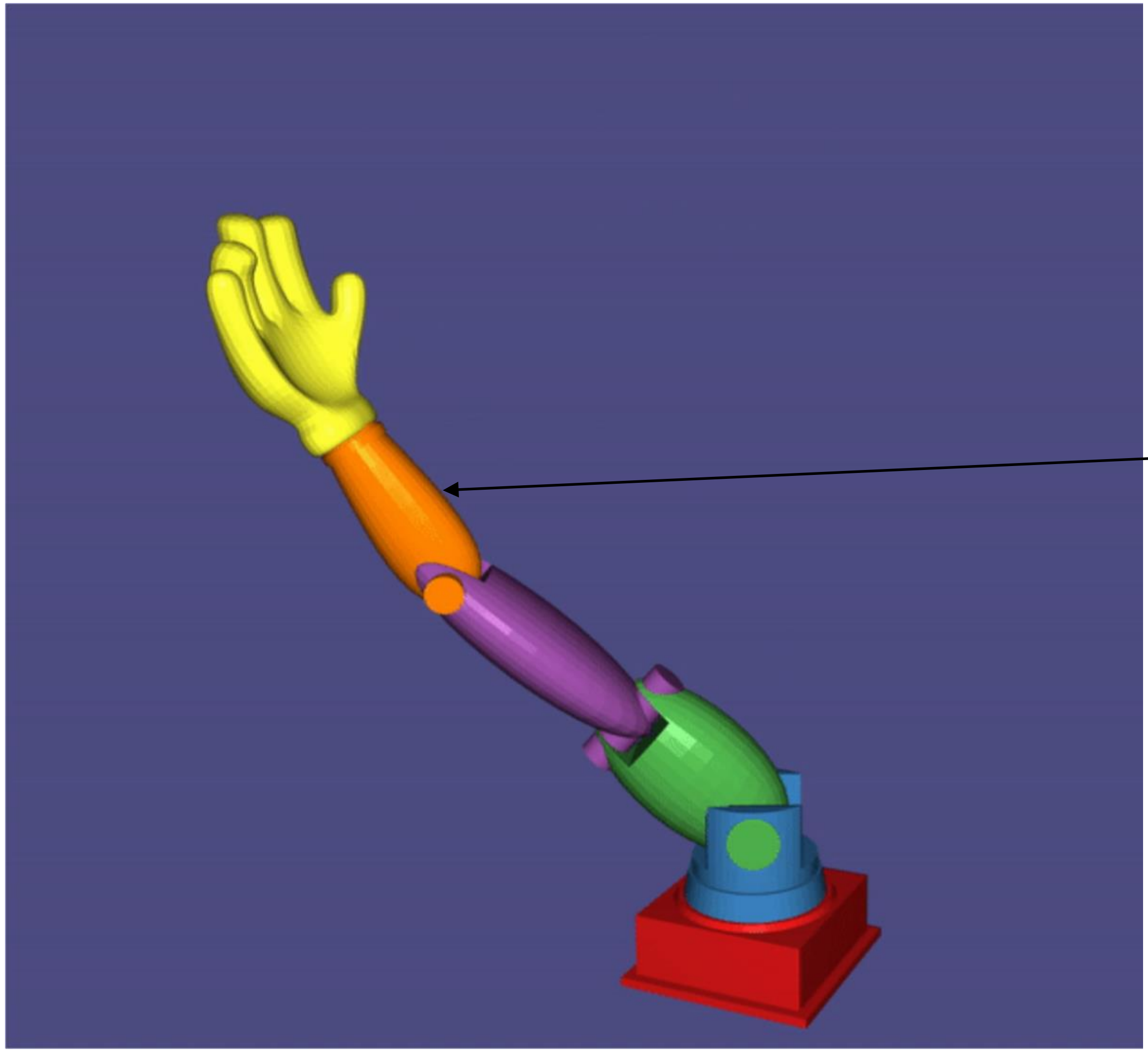
$$\mathbf{T}_i = \mathbf{T}_{p_i} \begin{pmatrix} \widehat{\mathbf{T}}_i \\ 0\,0\,0\,1 \end{pmatrix} \begin{pmatrix} & & 0 \\ \overline{\mathbf{R}}_i & & 0 \\ & & 0 \\ 0\,0\,0 & & 1 \end{pmatrix} \begin{pmatrix} \widehat{\mathbf{T}}_i \\ 0\,0\,0\,1 \end{pmatrix}^{-1}$$

$$\mathbf{T}_i = \mathbf{T}_{p_i} \widehat{\mathbf{T}}_i \begin{pmatrix} & 0 \\ \mathbf{R}_x(\theta_{i3}) & 0 \\ & 0 \\ 0\,0\,0 & 1 \end{pmatrix} \begin{pmatrix} & 0 \\ \mathbf{R}_z(\theta_{i2}) & 0 \\ & 0 \\ 0\,0\,0 & 1 \end{pmatrix} \begin{pmatrix} & 0 \\ \mathbf{R}_x(\theta_{i1}) & 0 \\ & 0 \\ 0\,0\,0 & 1 \end{pmatrix} \widehat{\mathbf{T}}_i^{-1}$$



Rigid “Skinning”

Idea: Attach each vertex to a single bone



$$\mathbf{v}_j = \mathrm{T}_i \hat{\mathbf{v}}_j$$

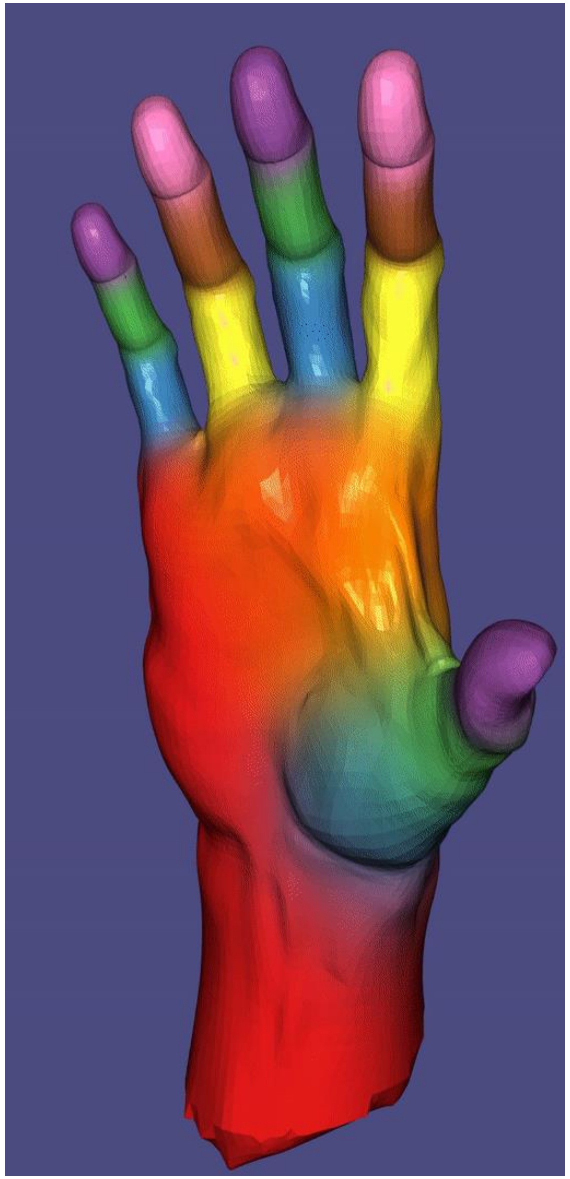
Deformable Skinning

Rigid Skinning is fine for mechanical things, but for smoother deformations we need to try something else

Rather than attach each vertex to a single bone, we attach each vertex to multiple bones and ***blend*** their transformations

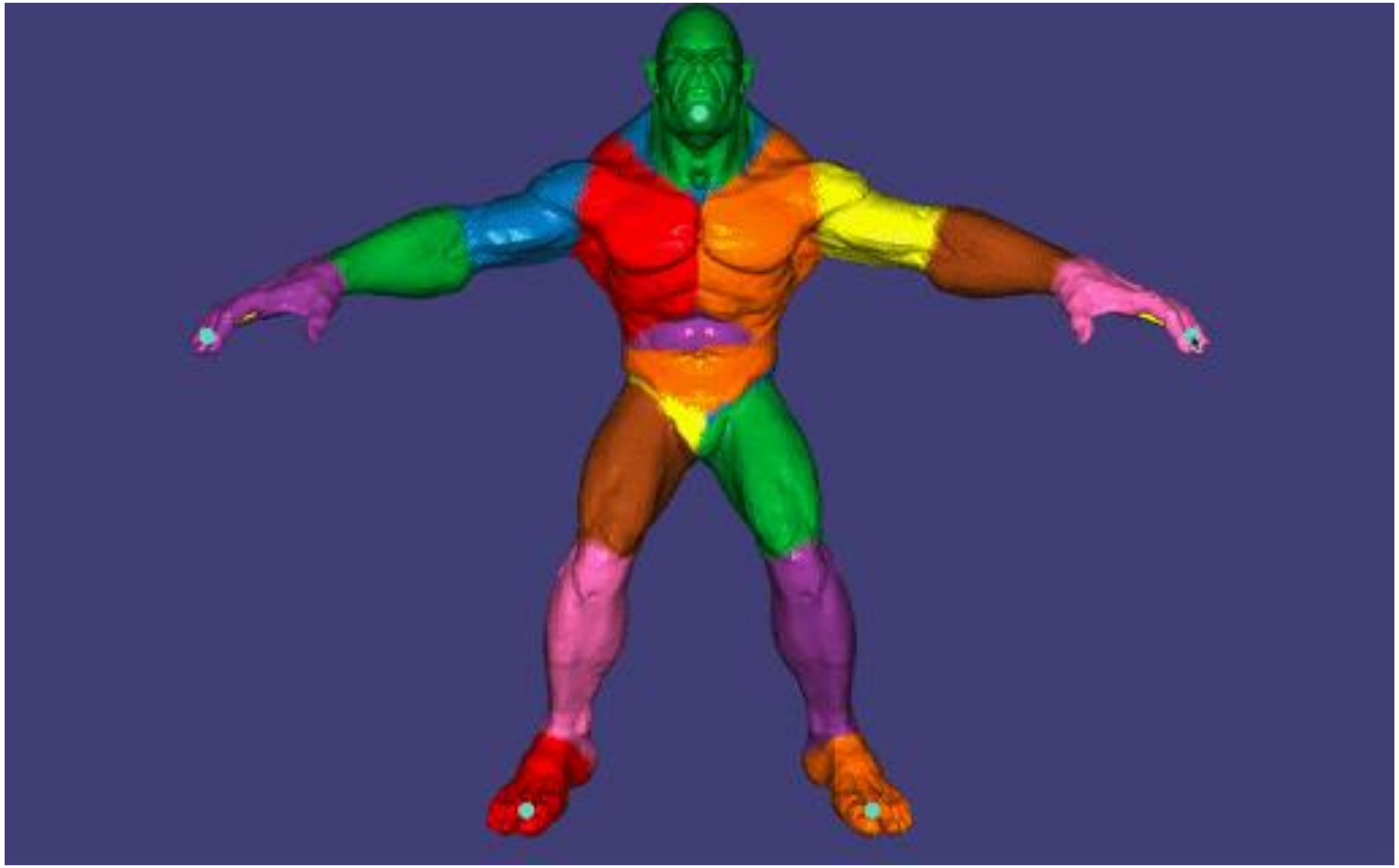
If this blending is linear in the transformations, we call it linear blend skinning.

Linear Blend Skinning



$$\mathbf{v}_j = \sum_{i=1}^{\#\text{bones}} w_{ij} \mathbf{T}_i \hat{\mathbf{v}}_j$$

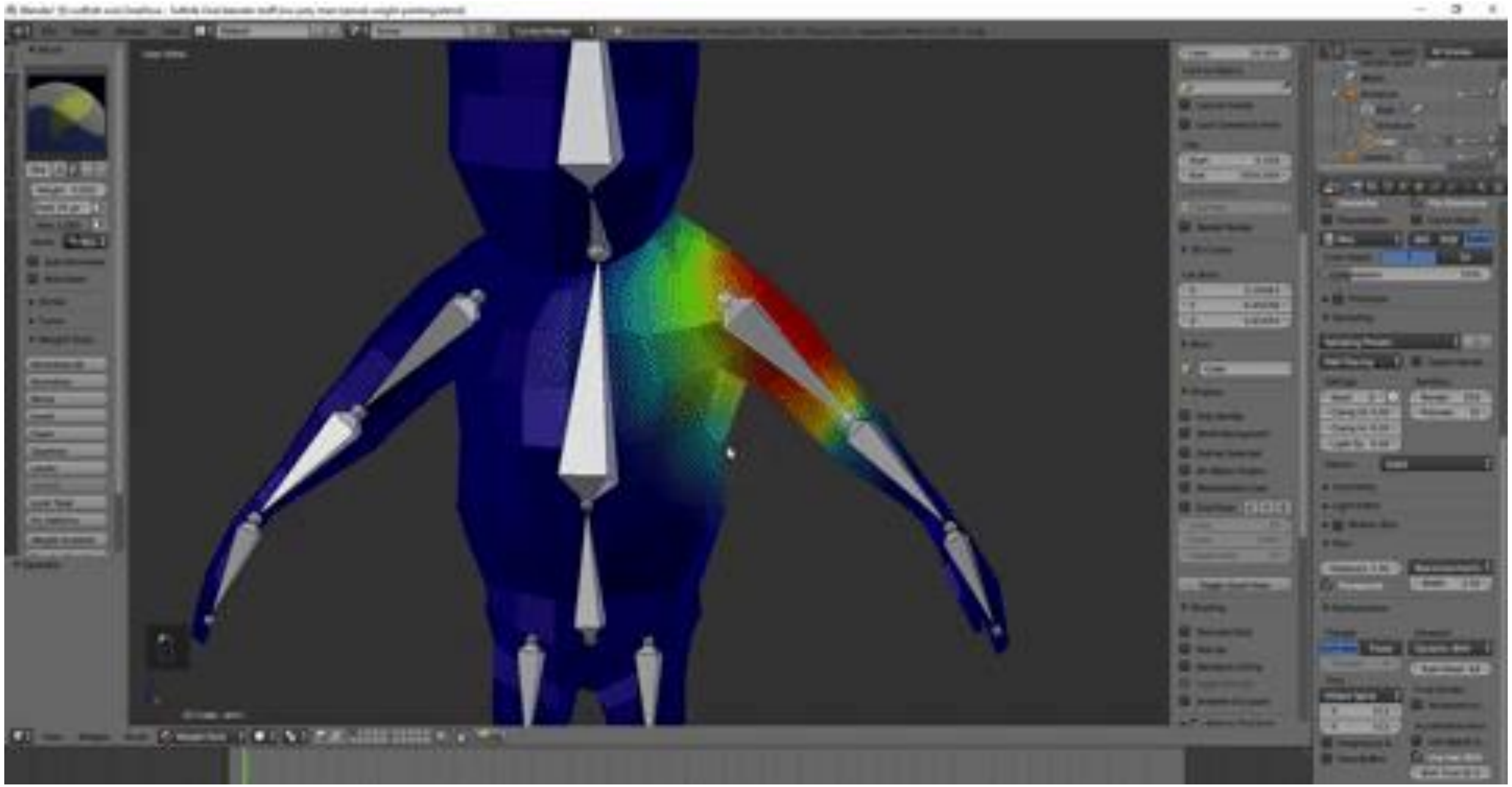
Rigid vs Linear Blend Skinning



Weight Painting

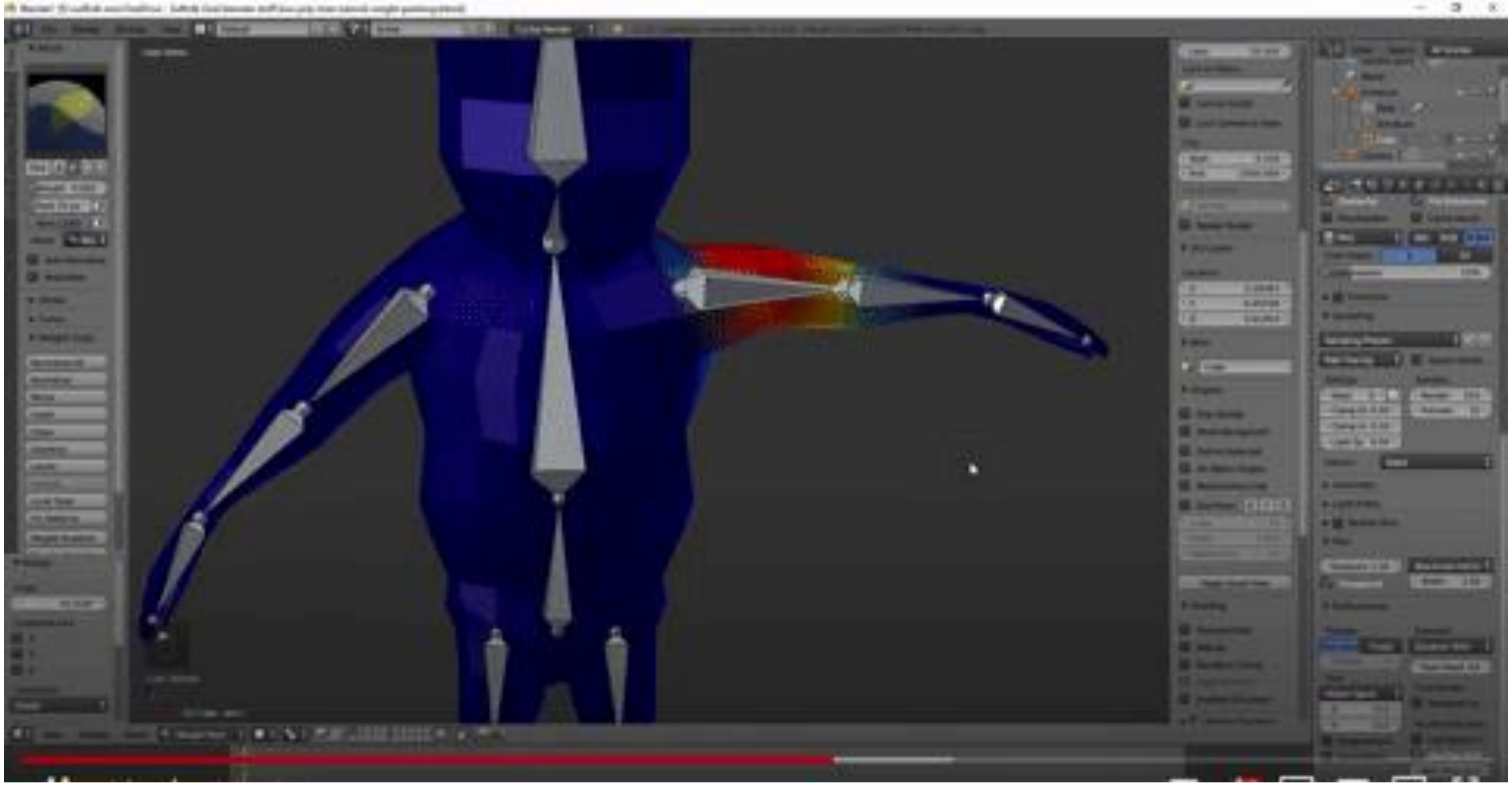
https://www.youtube.com/watch?v=Tl4qTgwQwYw

Weight Painting



https://www.youtube.com/watch?v=Tl4qTgwQwYw

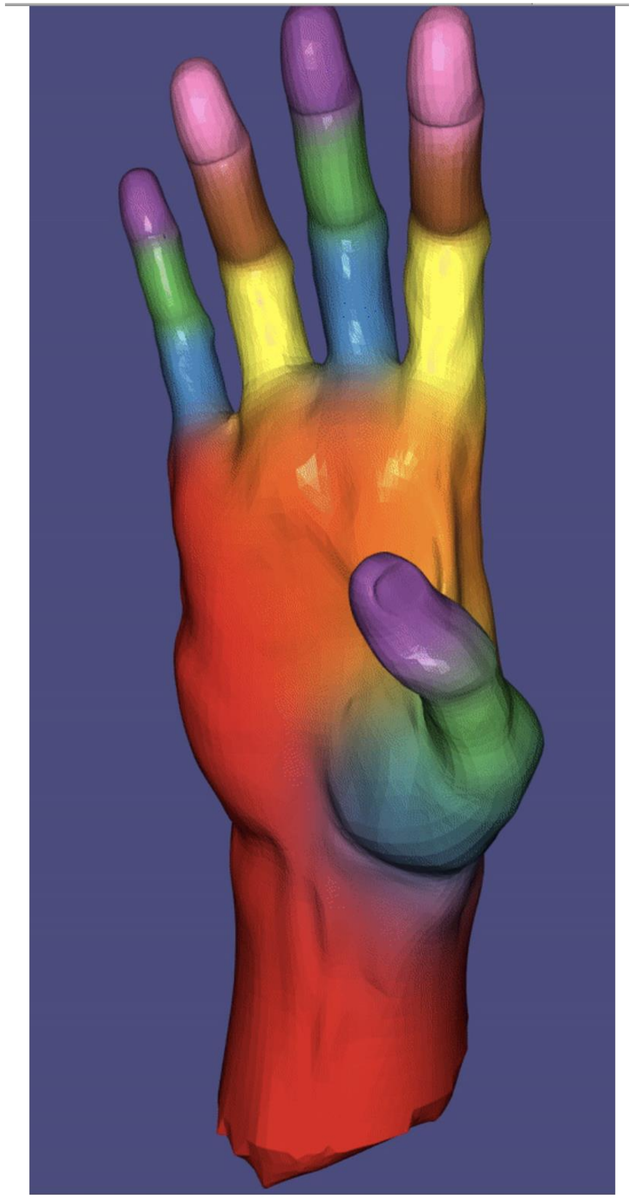
Weight Painting



https://www.youtube.com/watch?v=Tl4qTgwQwYw

Specifying Keyframes

Time = 0



Time = 10s

Poses are generated by specifying rotations of bones

Each pose can be represented as

$$\theta = \begin{pmatrix} \theta_{11} \\ \theta_{11} \\ \theta_{11} \\ \vdots \\ \theta_{n1} \\ \theta_{n2} \\ \theta_{n3} \end{pmatrix}$$

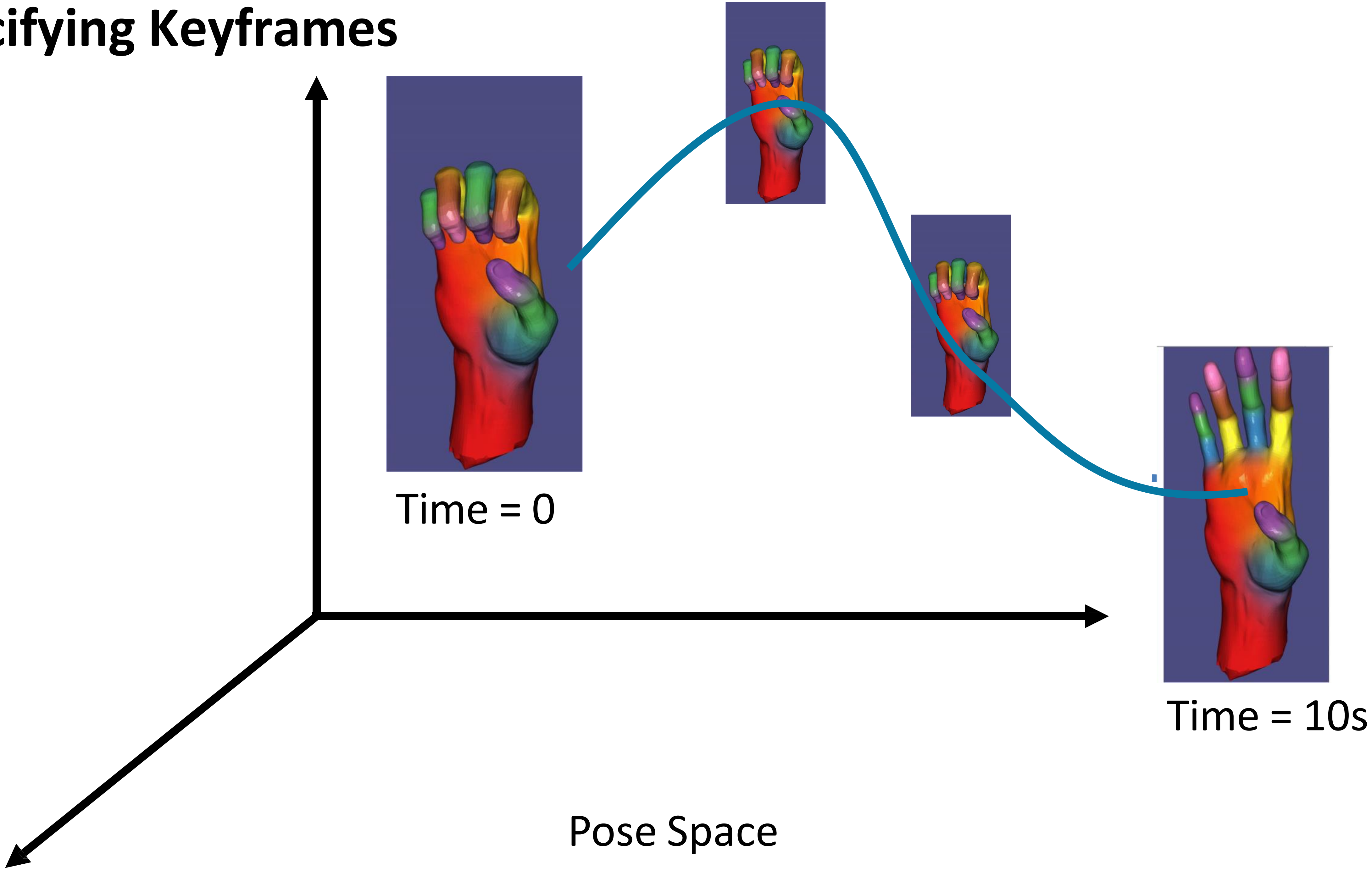
Specifying Keyframes

Time = 0

????????????

Time = 10s

Specifying Keyframes



Specifying Keyframes

https://en.wikipedia.org/wiki/Twelve_basic_principles_of_animation#Slow_in_and_slow_out
https://www.youtube.com/watch?v=fQBFsTqbKhY

Interpolating Keyframes

$$\theta = \mathbf{c}\left(t\right)$$

is a curve in the pose space

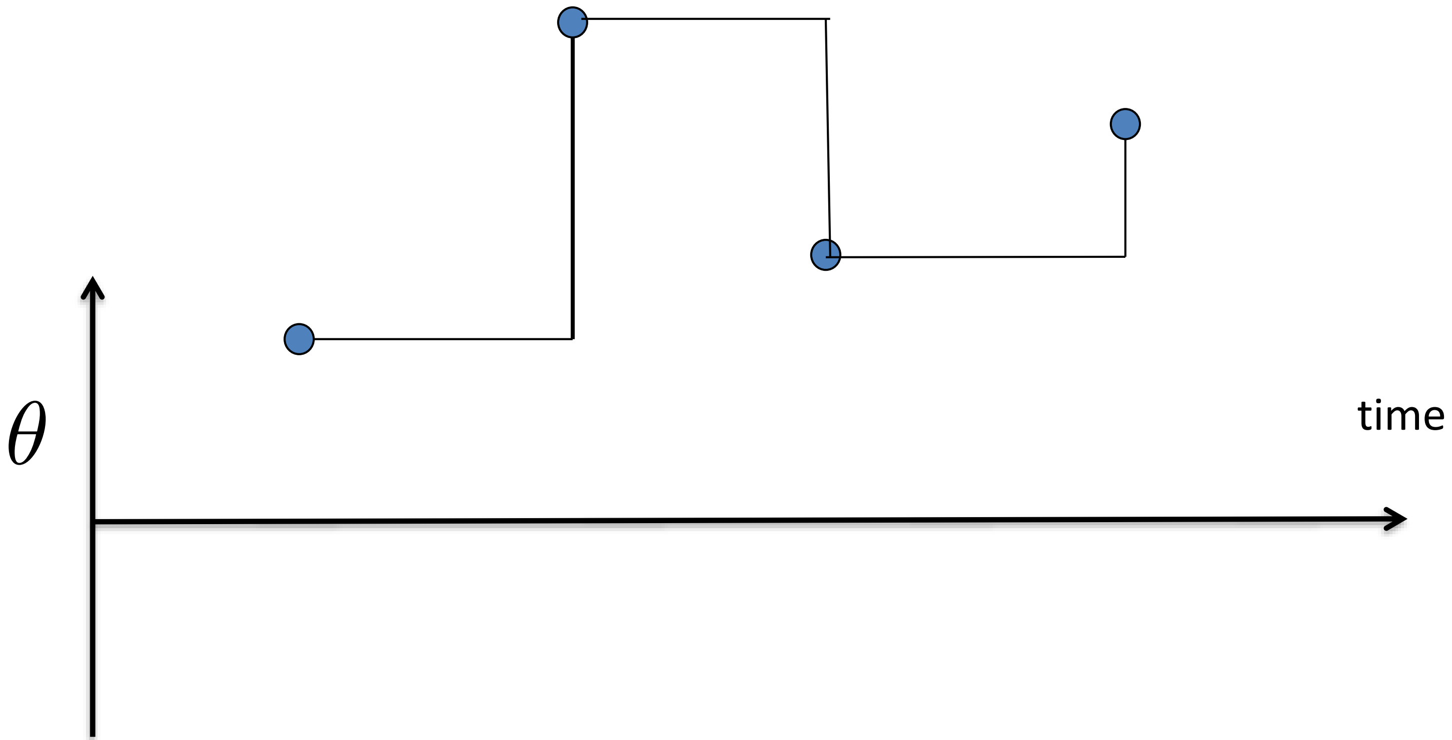
How could we construct such a curve from keyframes ?

Interpolating Keyframes

$$\theta = \mathbf{c}\left(t\right)$$

is a curve in the pose space

How could we construct such a curve from keyframes ?



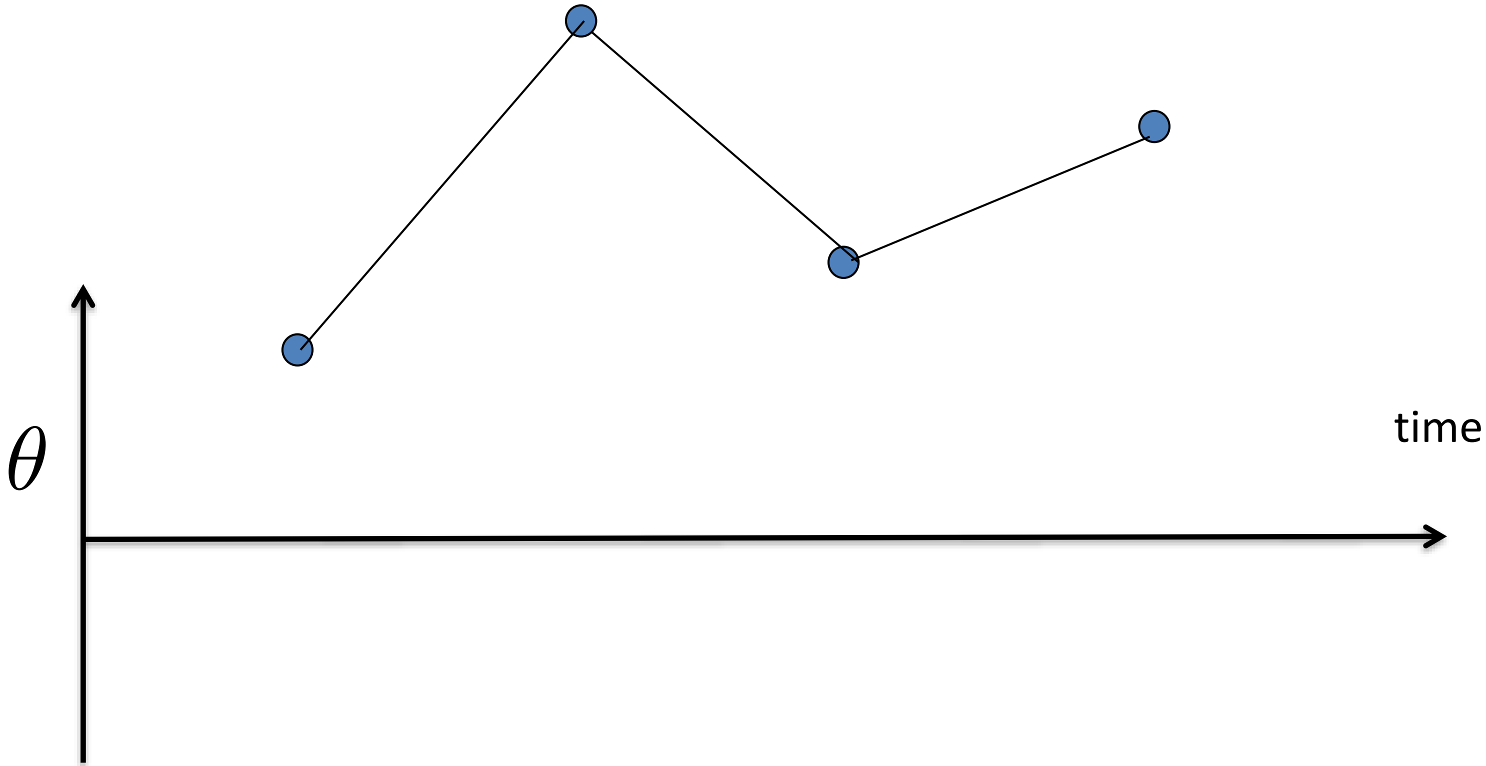
Hold

Interpolating Keyframes

$$\theta = \mathbf{c}\left(t\right)$$

is a curve in the pose space

How could we construct such a curve from keyframes ?



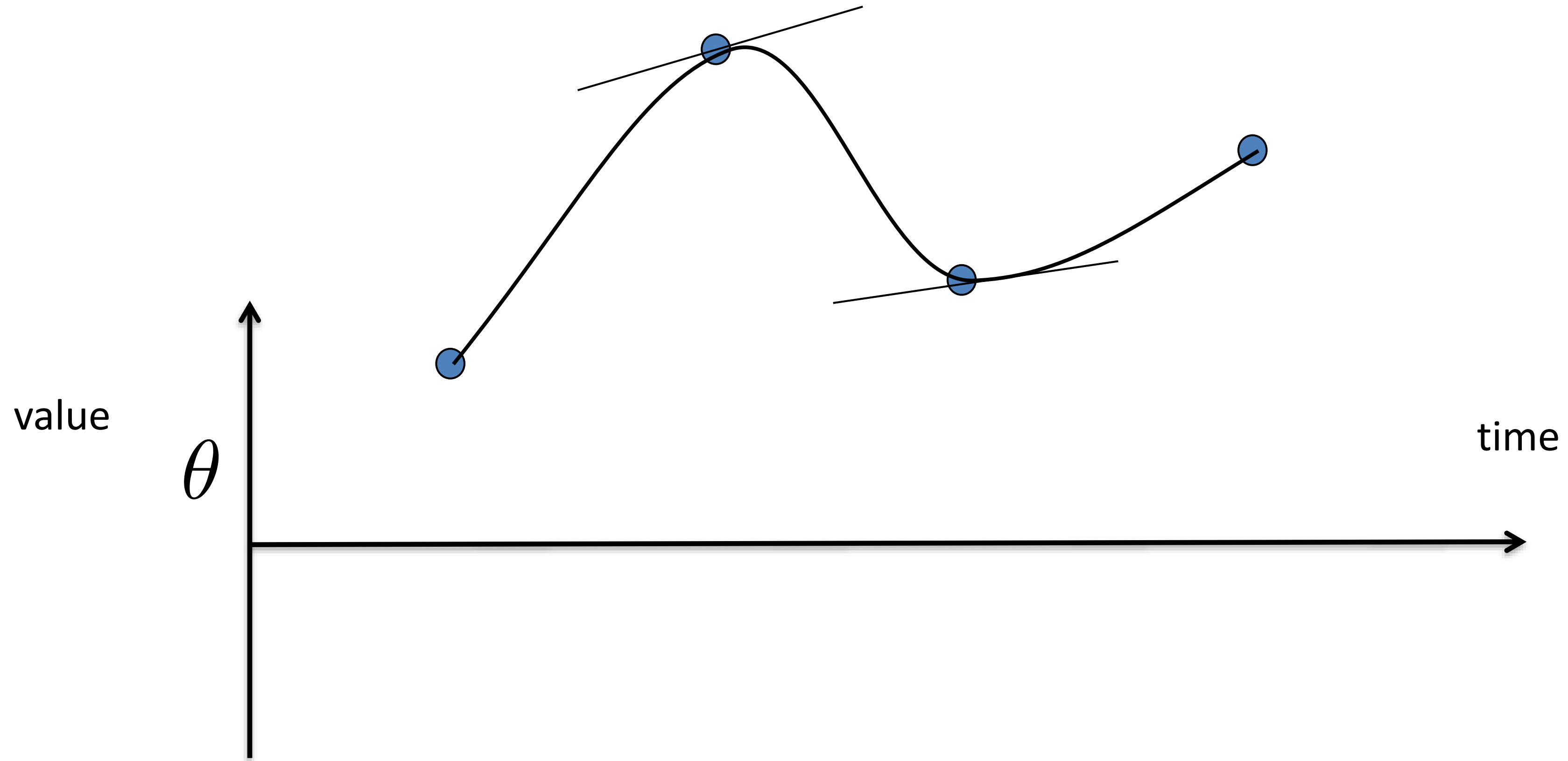
Linear

Interpolating Keyframes

$$\theta = \mathbf{c}\left(t\right)$$

is a curve in the pose space

How could we construct such a curve from keyframes ?



Spline

Catmull-Rom Spline

A **cubic** curve created by specifying the end points and the tangents of the curve.

$$\mathbf{c}(t) = at^3 + bt^2 + ct + d$$
$$\mathbf{c}'(t) = 3at^2 + 2bt + c$$

Catmull-Rom Spline

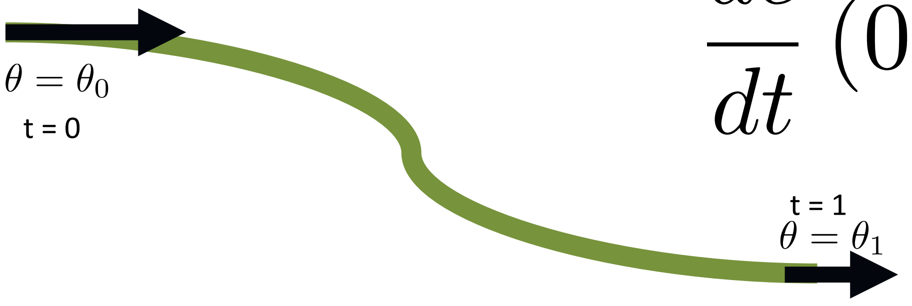
A **cubic** curve created by specifying the end points and the tangents of the curve.

$$c(0) = d$$

$$c(1) = a + b + c + d$$

$$\frac{dc}{dt}(1) = 3a + 2b + 1c$$

$$\frac{dc}{dt}(0) = 1c$$



Catmull-Rom Spline

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{a}^T \\ \mathbf{b}^T \\ \mathbf{c}^T \\ \mathbf{d}^T \end{pmatrix} = \begin{pmatrix} \theta_0^T \\ \theta_1^T \\ \mathbf{m}_0^T \\ \mathbf{m}_1^T \end{pmatrix}$$

Catmull-Rom Spline

After solving and rearranging we end up with

$$\mathbf{c}\,(t) = (2t^3 - 3t^2 + 1)\theta_0 + (t^3 - 2t^2 + t)\mathbf{m}_0 + (-2t^3 + 3t^2)\theta_1 + (t^3 - t^2)\mathbf{m}_1$$

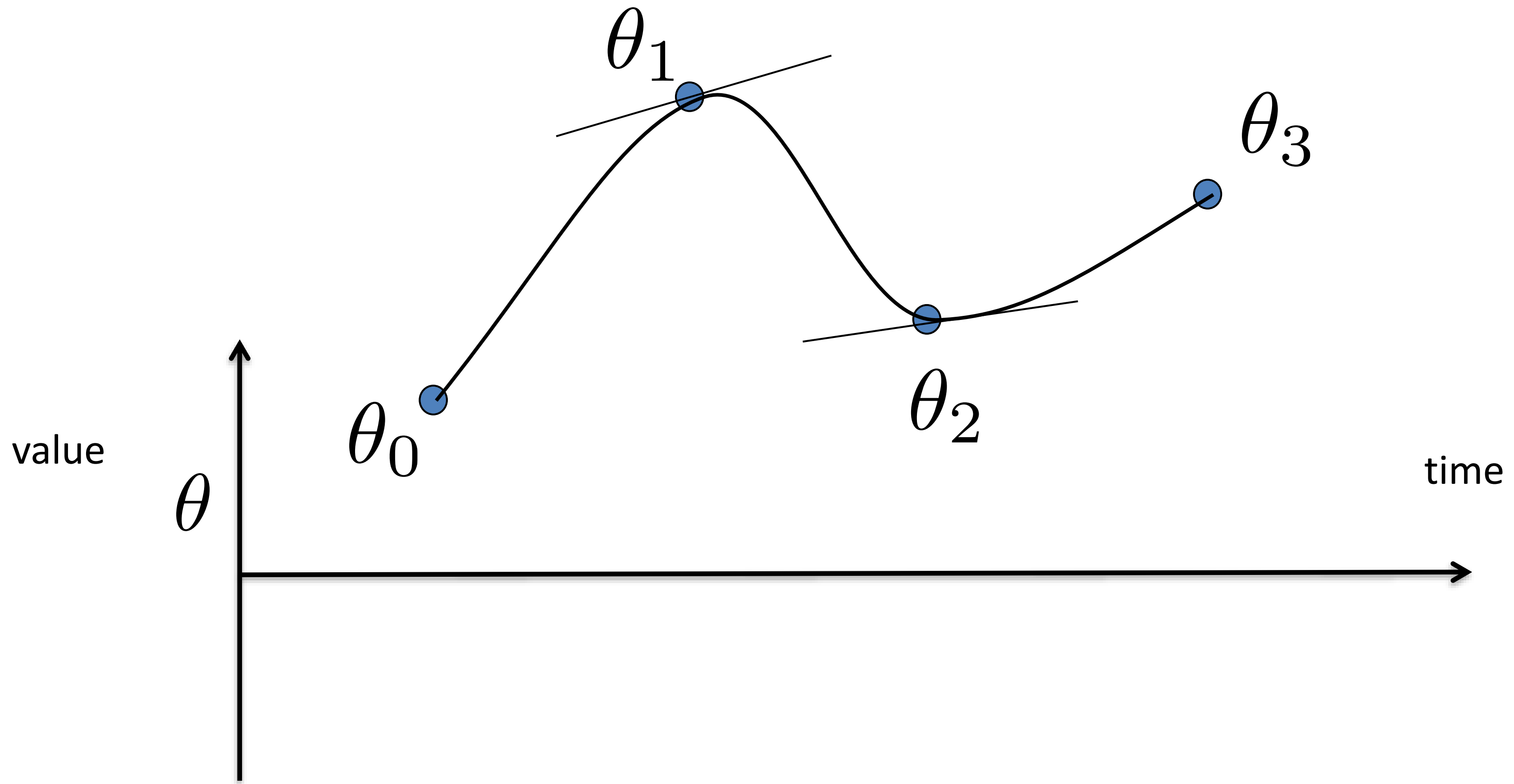
Remember this is for t = 0 to t = 1.

For arbitrary intervals substitute

$$t = (t' - t_0)/(t_1 - t_0)$$

Catmull-Rom Spline

Catmull-Rom chooses the tangents using "Finite Differences"

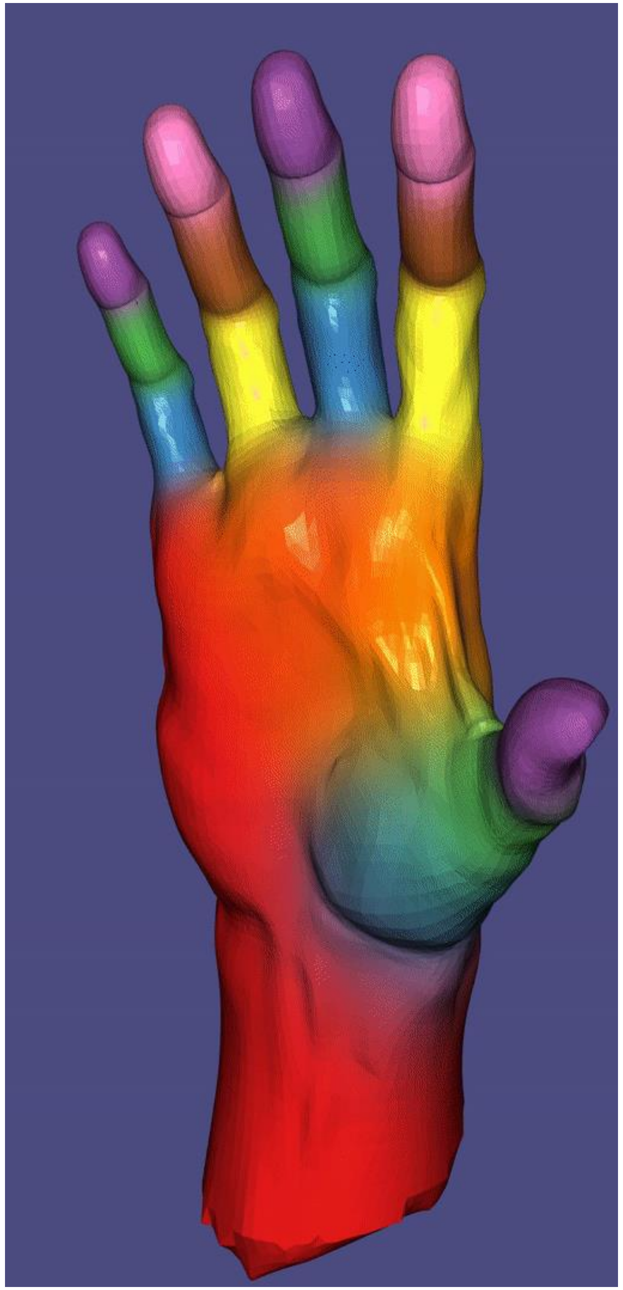


$$\mathbf{m}_k = \frac{\theta_{k+1} - \theta_k}{t_{k+1} - t_k}$$

Catmull-Rom Animation

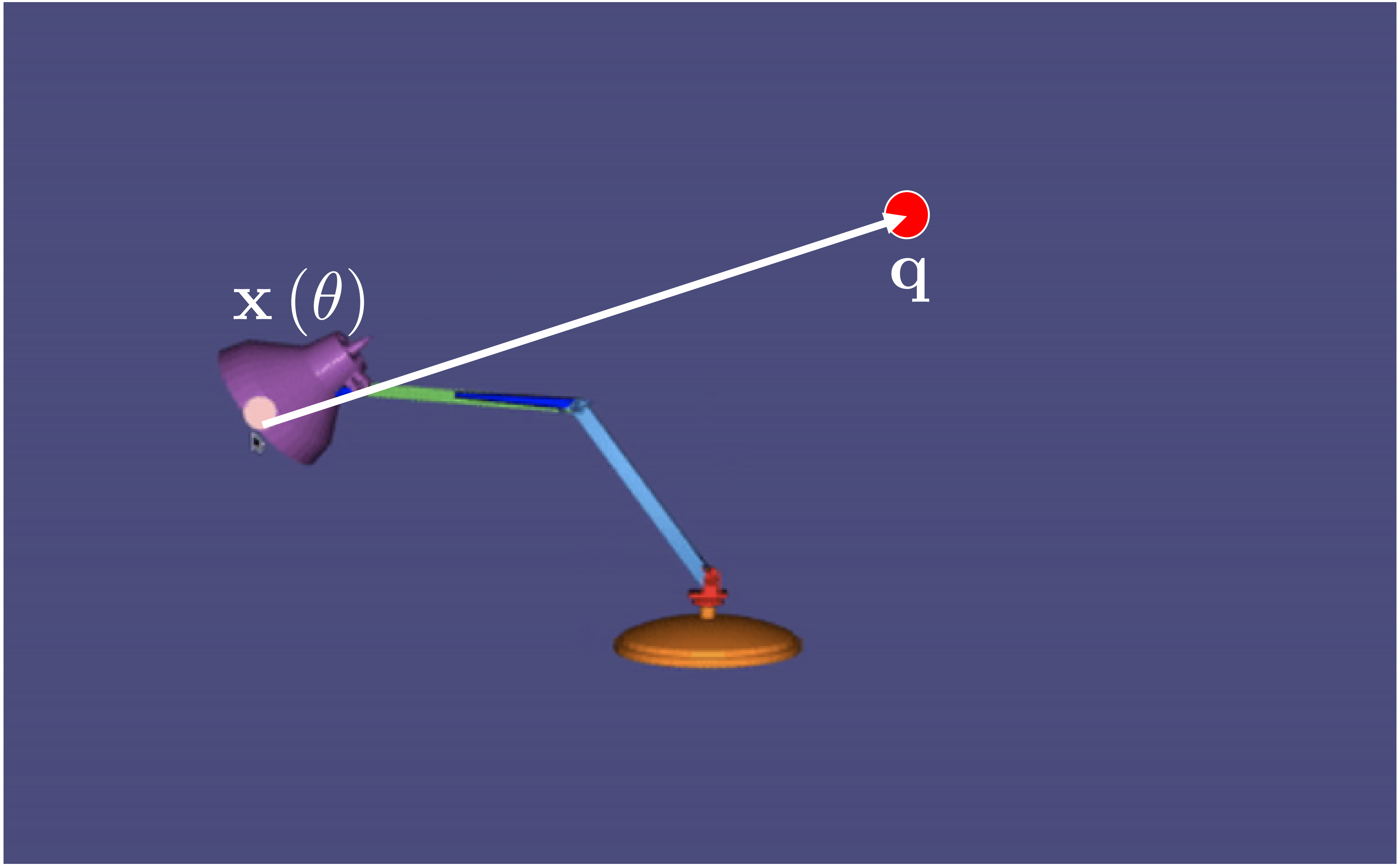
For each time, t, create a new θ(t) using your spline.

Re-pose your bones using θ, deform skin and draw…

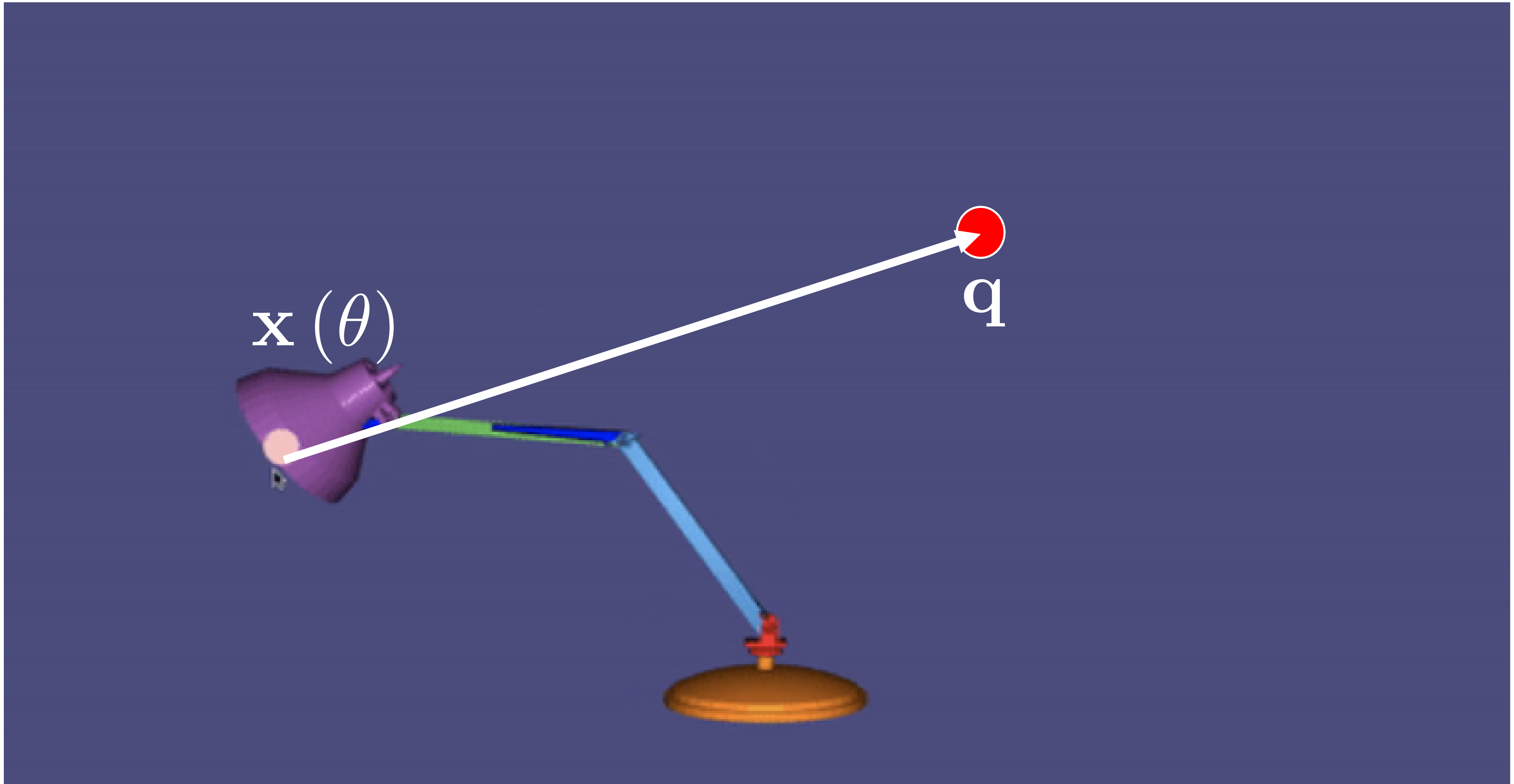


Inverse Kinematics

Inverse Kinematics



Inverse Kinematics

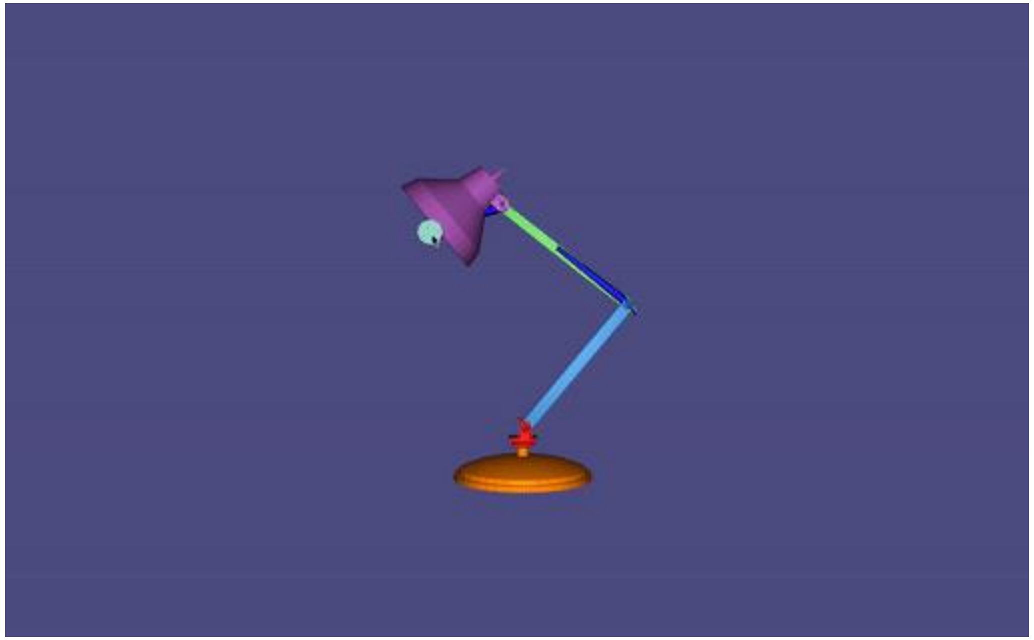


$$\theta^* = \arg\min \|\mathbf{x}(\theta) - \mathbf{q}\|_2^2$$

Skeletons: Inverse Kinematics

What is the pose (set of joint angles *a*) that lets us reach a given point (end-effector position).

$$\mathbf{a} = \begin{pmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \\ \theta_{21} \\ \theta_{22} \\ \theta_{23} \\ \vdots \\ \theta_{m1} \\ \theta_{m2} \\ \theta_{m3} \end{pmatrix}$$



What does it mean to reach (get as close as possible) to a point?

Skeletons: Inverse Kinematics

Closeness energy can be measured the squared distance between the pose tip x_b of some bone b and a desired goal location $q \in R^3$.

$$E(\mathbf{x}_b) = \|\mathbf{x}_b - \mathbf{q}\|^2.$$

Given pose vector a, the bone tip x_b is:

$$\mathbf{x}_b(\mathbf{a}) = \mathbf{T}_b \widehat{\mathbf{d}}_b$$

Now given any number of end-effectors $b_1,..b_k$:

$$\min_{\mathbf{a}} \underbrace{\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2}_{E(\mathbf{x}_b(\mathbf{a}))}$$

And we impose some joint angle limits: $\min\limits_{\mathbf{a}^{\min} \leq \mathbf{a} \leq \mathbf{a}^{\max}} E(\mathbf{x}_b(\mathbf{a}))$

Minimizing this energy is a non-linear least squares problem.

Inverse Kinematics: Energy

$$E(\mathbf{x}_b(\mathbf{a})) = \|\mathbf{x}_b(\mathbf{a}) - \mathbf{q}\|^2$$

the squared distance between the pose tip $\mathbf{x_b}$ of some bone **b**, and a desired goal location **q.**

Inverse Kinematics: Energy

$$E(\mathbf{x}_b(\mathbf{a})) = \|\mathbf{x}_b(\mathbf{a}) - \mathbf{q}\|^2$$

list of constrained end effectors $\rightarrow$ $b = \{b_1, b_2, \ldots, b_k\}$

$$\min_{\mathbf{a}} \quad \underbrace{\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2}_{E(\mathbf{x}_b(\mathbf{a}))}$$

Inverse Kinematics: Energy

Over all choices of Euler angles **a**, we want the angles that ensure all selected end effectors go to their prescribed locations, subject to angle constraints: $\min\limits_{\mathbf{a}^{\min} \leq \mathbf{a} \leq \mathbf{a}^{\max}} E(\mathbf{x}_b(\mathbf{a}))$

Position + rotation of bone $\mathbf{x}_{b_i}$ found using forward kinematics

Constrained end effector position

$$\min_{\mathbf{a}} \underbrace{\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \hat{\mathbf{x}}_{b_i}\|^2}_{E(\mathbf{x}_b(\mathbf{a}))}$$

Sum over all constrained end effectors

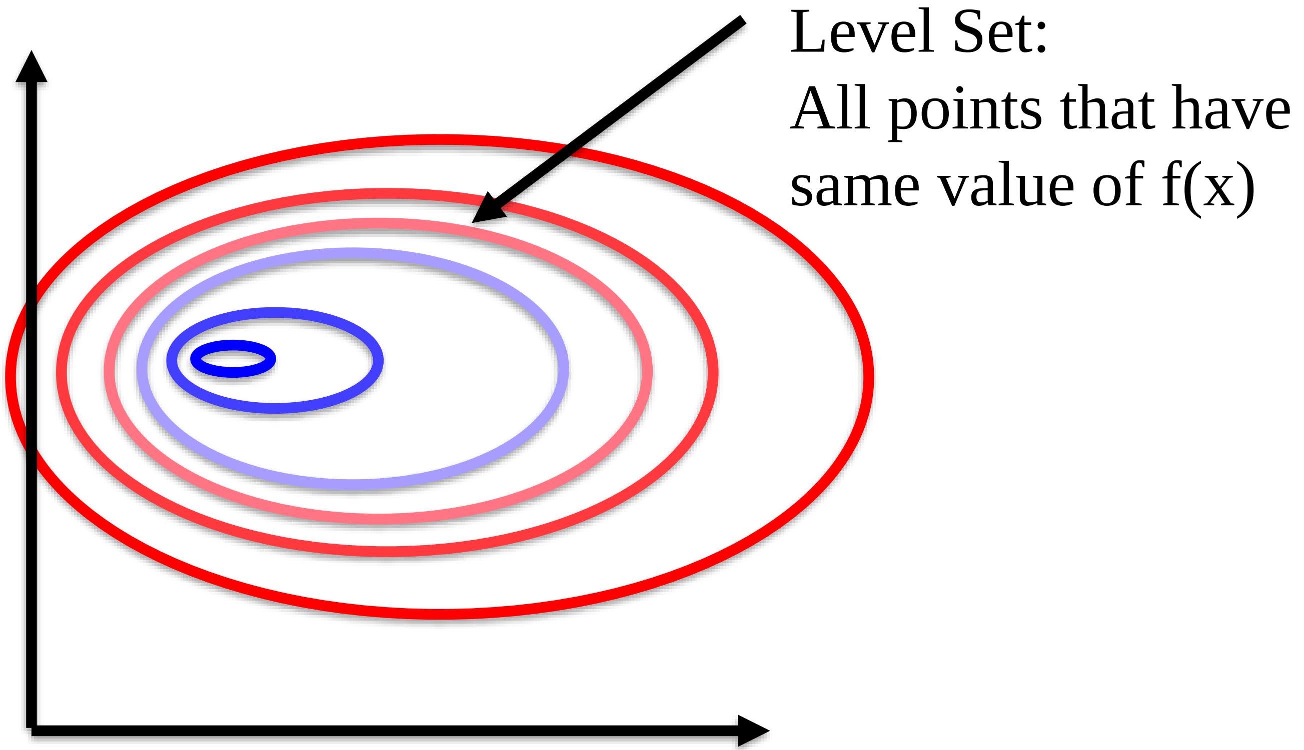
Gradient Descent

- Make a guess.
- Iteratively move in a direction lower energy.

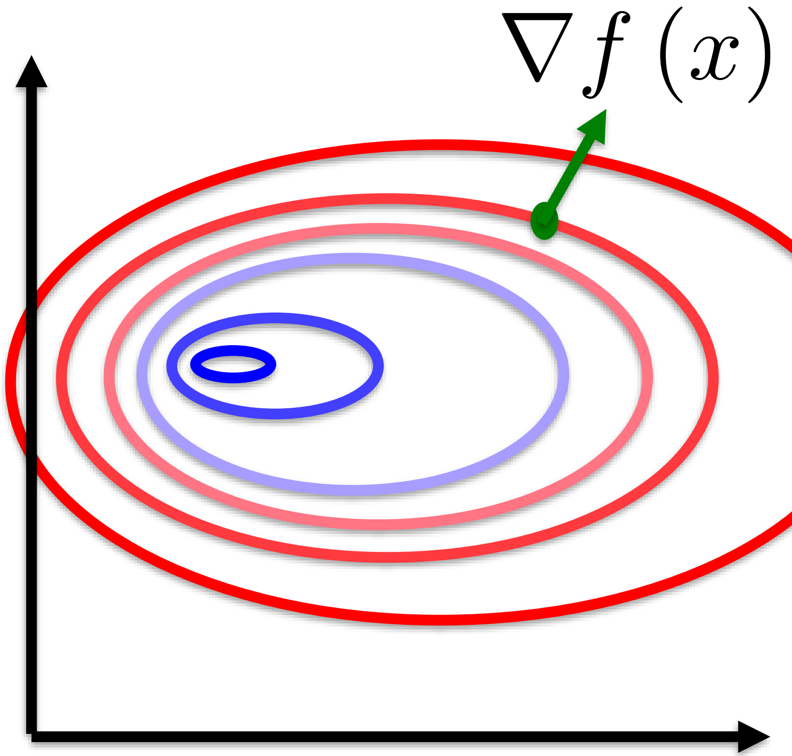
Recall that the gradient of a function points in direction of maximum ascent

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial \mathbf{x}_1} & \frac{\partial f}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f}{\partial \mathbf{x}_n} \end{pmatrix}$$

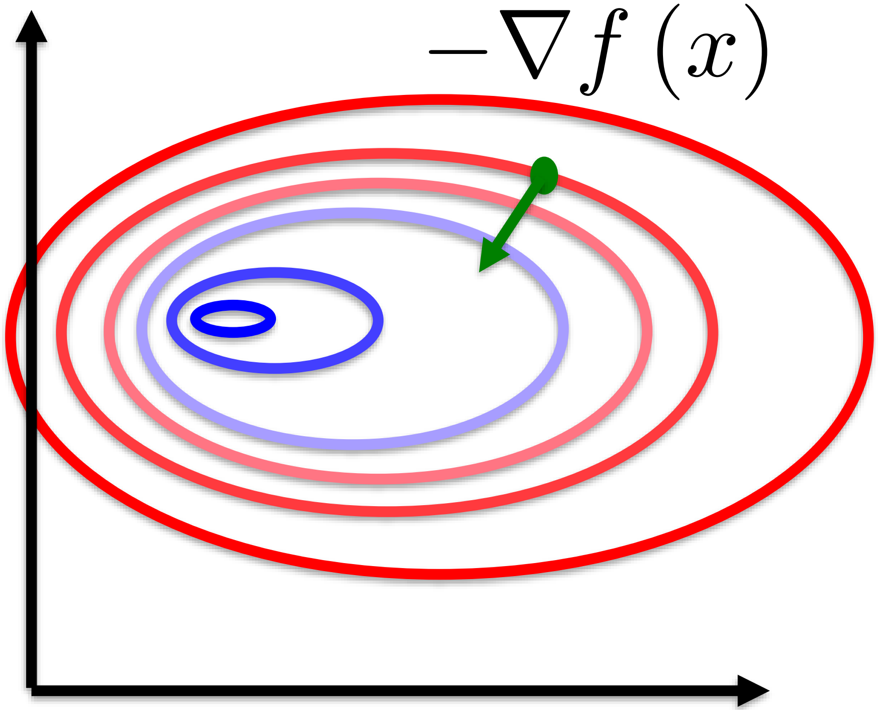
An Aside: Level Sets



Gradient Descent



Gradient Descent

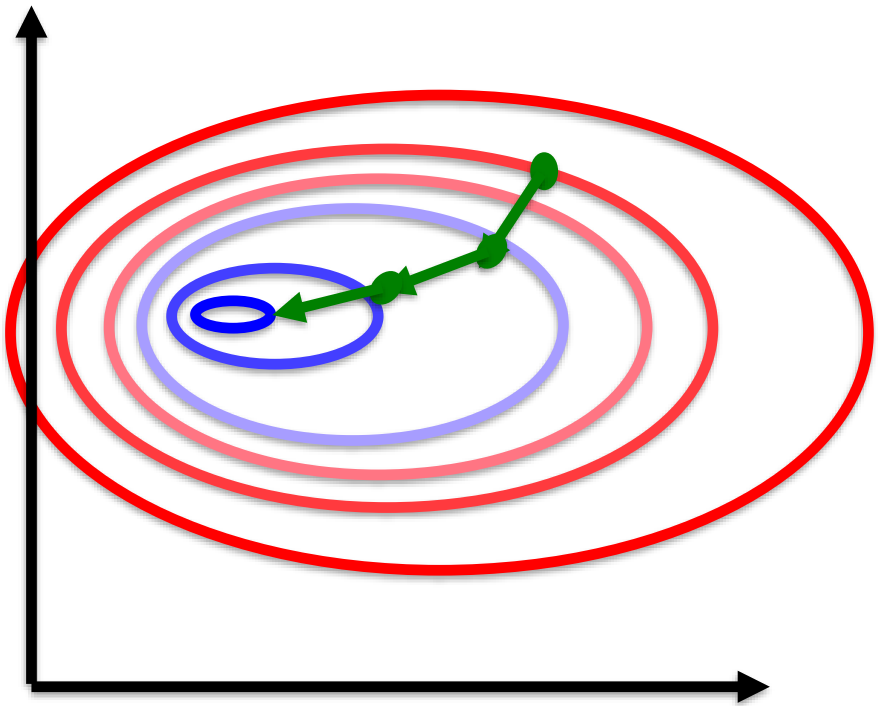


Projected Gradient Descent Algorithm

While not at an optimal point

- Compute the gradient at current point (x)
- Move to new point $x = x - h \nabla f(x)$
- Project *x* to satisfy any constraints like joint angle limits.
- Iterate to a minima.

Gradient Descent



Gradient Descent

So let's take a step in the *negative* gradient direction of the objective

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}} \right)^T$$

Gradient Descent

So let's take a step in the negative gradient direction of the objective

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}} \right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}} \right)^T \left(\frac{dE(\mathbf{x})}{d\mathbf{x}} \right)$$

Gradient Descent

So let's take a step in the negative gradient direction of the objective

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}} \right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \boxed{\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}} \right)^T} \left(\frac{dE(\mathbf{x})}{d\mathbf{x}} \right)$$

Gradient Descent: Kinematic Jacobian

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \boxed{\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}}\right)^T} \left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$

The change in tip positions **x** with respect to joint angles **a**

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \mathbf{J}^{\mathsf{T}} \left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$

Computed using finite differences

https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant

Line Search

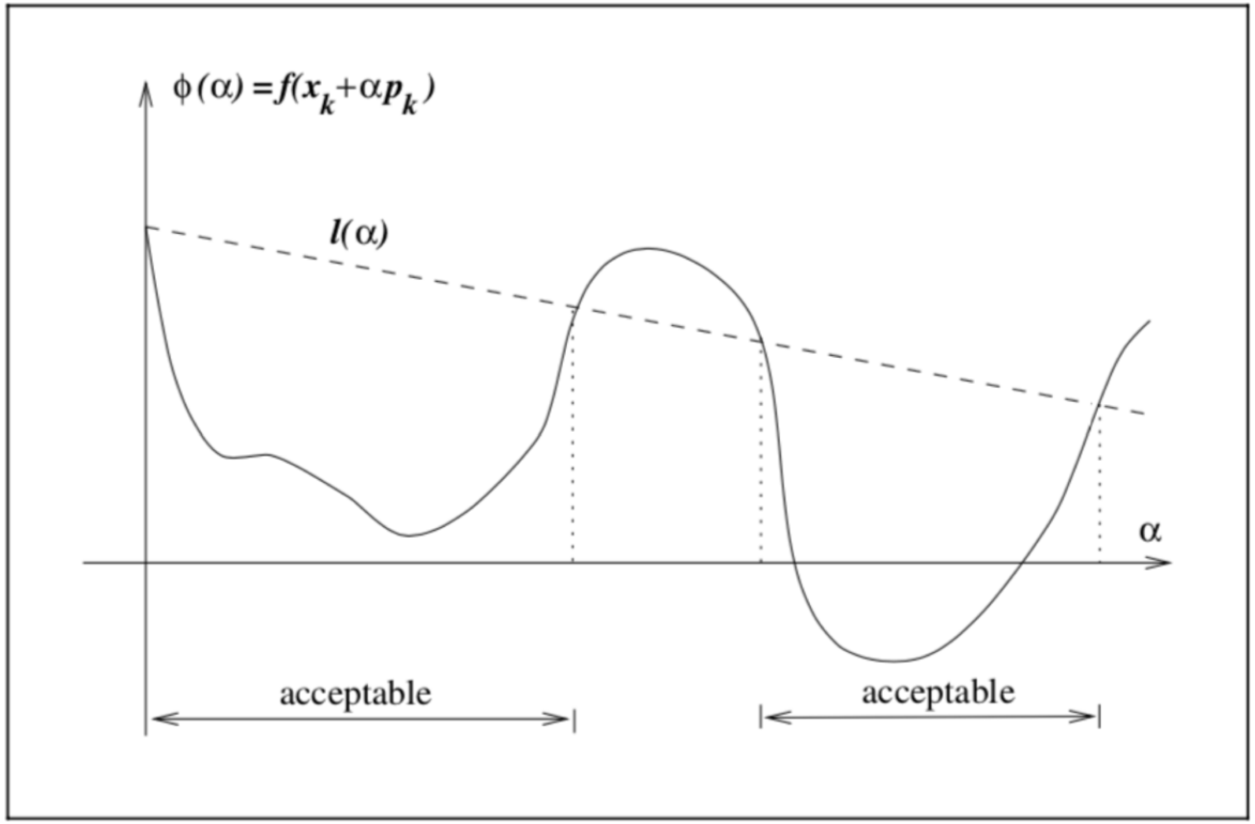
$$\mathbf{a} \leftarrow \mathbf{a} - \boxed{\sigma}\,\mathbf{J}^{\mathsf{T}} \left(\frac{dE(\mathbf{x})}{d\mathbf{x}} \right)$$

AKA we are moving in descent direction and then projecting:

$$\mathbf{a} \leftarrow \mathrm{proj}(\mathbf{a} + \Delta\mathbf{a})$$

Start with large σ and decrease by ½ until

$$E(\mathrm{proj}(\mathbf{a} + \sigma\Delta\mathbf{a})) < E(\mathbf{a})$$



https://en.wikipedia.org/wiki/Backtracking_line_search#Algorithm

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:
Start with an initial pose *a*, and move in direction of decrease in *E*, project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}}\right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}}\right)^T\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$ chain rule

$\frac{dE}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{a}|}$, $\frac{dE}{d\mathbf{x}} \in \mathbb{R}^{|\mathbf{x}|}$, and $\frac{d\mathbf{x}}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{x}|\times|\mathbf{a}|}$

$$\mathbf{J} = \frac{d\mathbf{x}}{d\mathbf{a}}.$$ also known as Jacobian measures the change in *x* for changes in joint angles *a*,

J can be computed using Finite Differences: $\mathbf{J}_{i,j} \approx \frac{\mathbf{x}_i(\mathbf{a}+h\delta_j)}{h}.$ $h = 10^{-7}$

$\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$ is gradient of $\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2$

Project to within limits: $\mathbf{a}_i \leftarrow \max[\mathbf{a}_i^{\min}, \min[\mathbf{a}_i^{\max}, \mathbf{a}_i]].$

Find a good step that lowers energy: $E(\mathrm{proj}(\mathbf{a} + \sigma\,\Delta\,\mathbf{a})) < E(\mathbf{a}).$

Next: Simulation, mass-spring systems