Computer Animation

National Cheng Kung University, Fall 2017

Instructor: Shih-Chin Weng 翁士欽 (Style.me)

What is Animation?

$$shape = f(time)$$

TED-Ed: Animation basics: The art of timing and spacing

12 PRINCIPLES OF ANIMATION

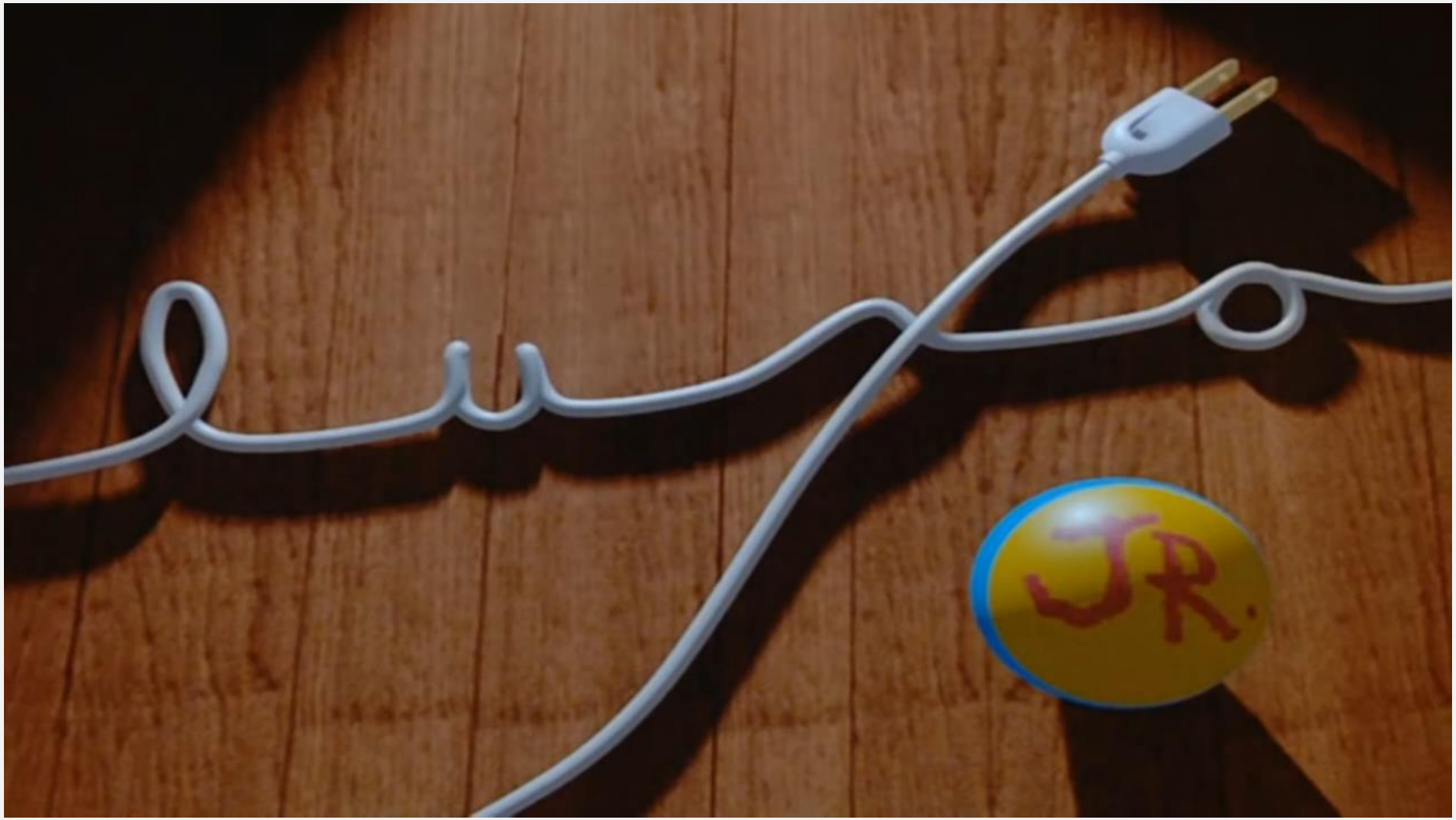
12 principles of animation by aCreativeAgency

Principles of Animation

1. *Squash & Stretch*
2. *Anticipation*
3. *Arcs*
4. *Ease In & Ease Out*
5. *Appeal*
6. *Timing*
7. *Solid Drawing*
8. *Exaggeration*
9. *Pose To Pose*
10. *Staging*
11. *Secondary Motion*
12. *Following Through*

https://en.wikipedia.org/wiki/12_basic_principles_of_animation

Luxo Jr., PIXAR 1986



Mariachi Plaza Clip, Coco, PIXAR 2017

https://www.youtube.com/watch?v=bFqeYK96JXc

Taxonomy of Animation Techniques

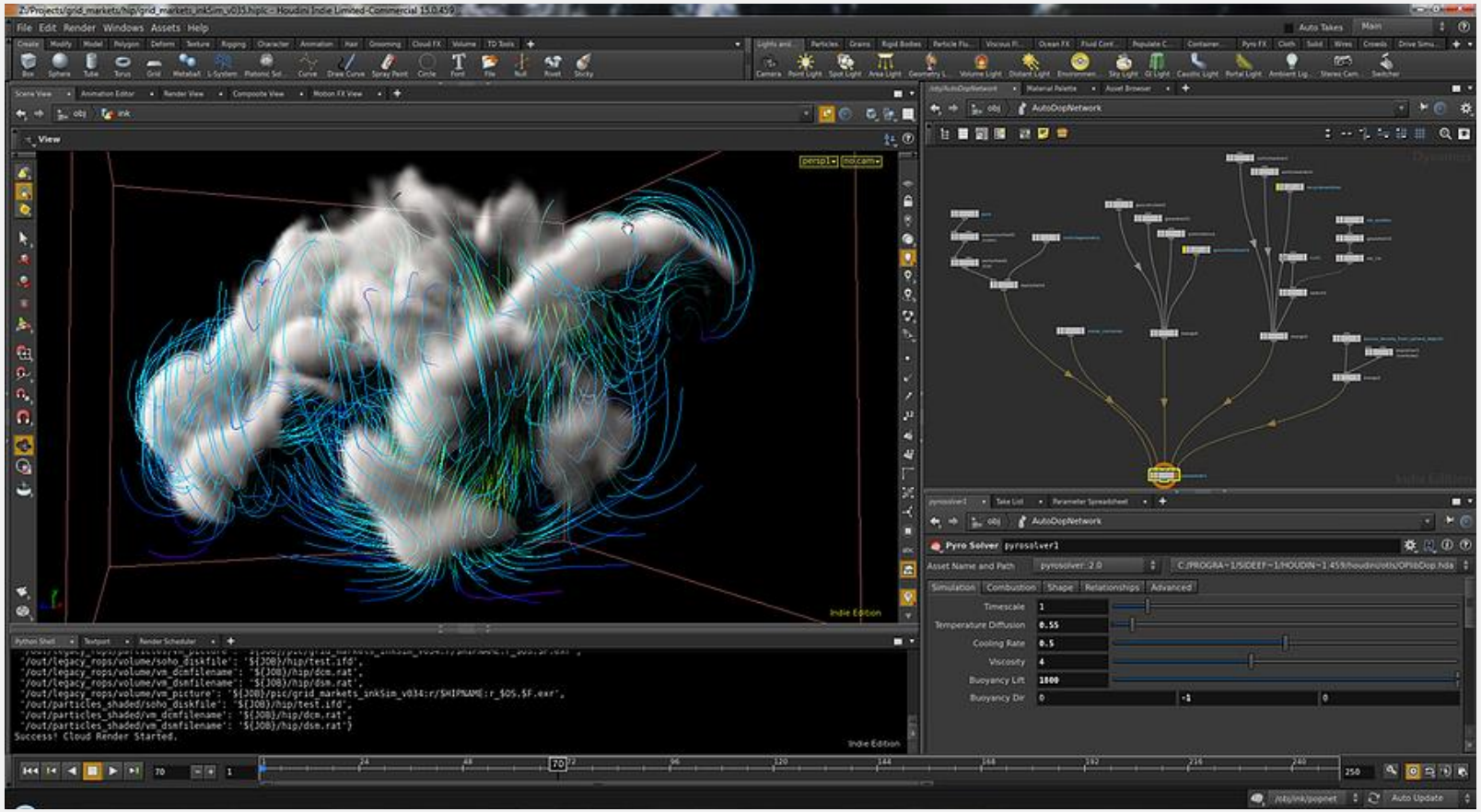
- Keyframe animation directed by artists

Taxonomy of Animation Techniques

- Data-driven (motion capture, performance capture)

Taxonomy of Animation Techniques

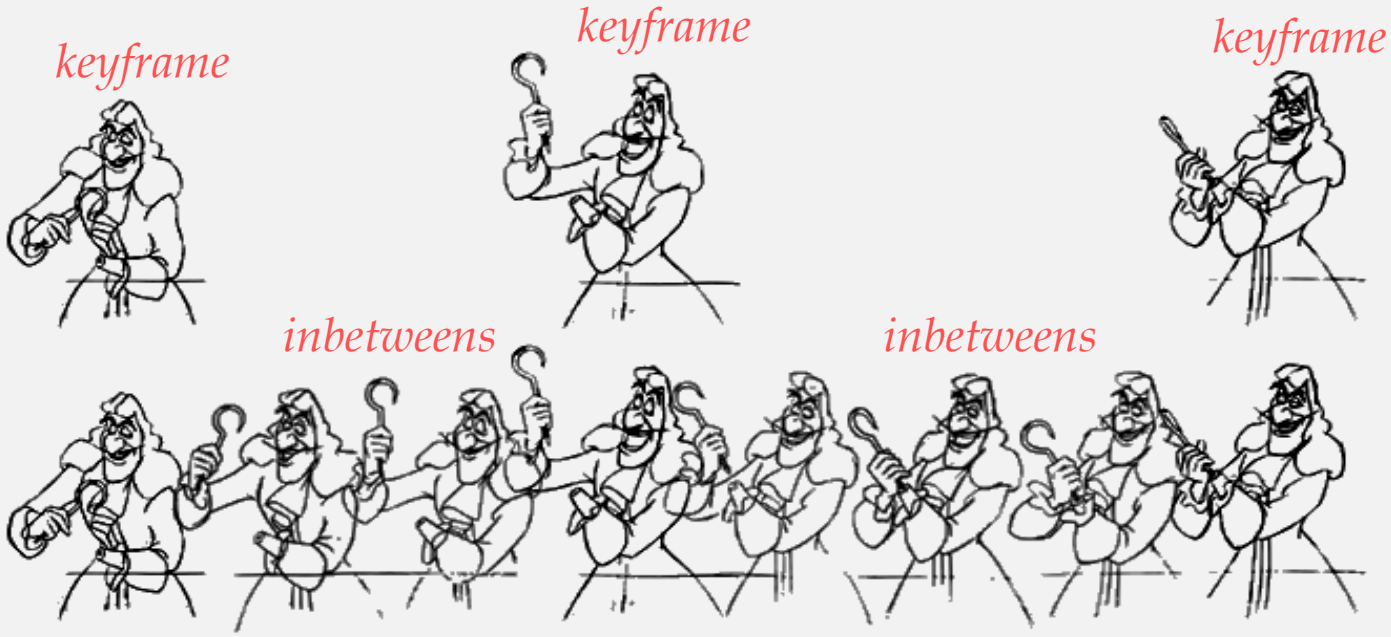
- Procedural (e.g., physically based simulation)



https://www.gridmarkets.com/phill-mayer

Keyframe Animation

- Animator specifies keyframes, software generate the frames in-between
- ***Interpolation*** is the major operation in
  - time-variant **transformations**
  - pose-to-pose **deformation**



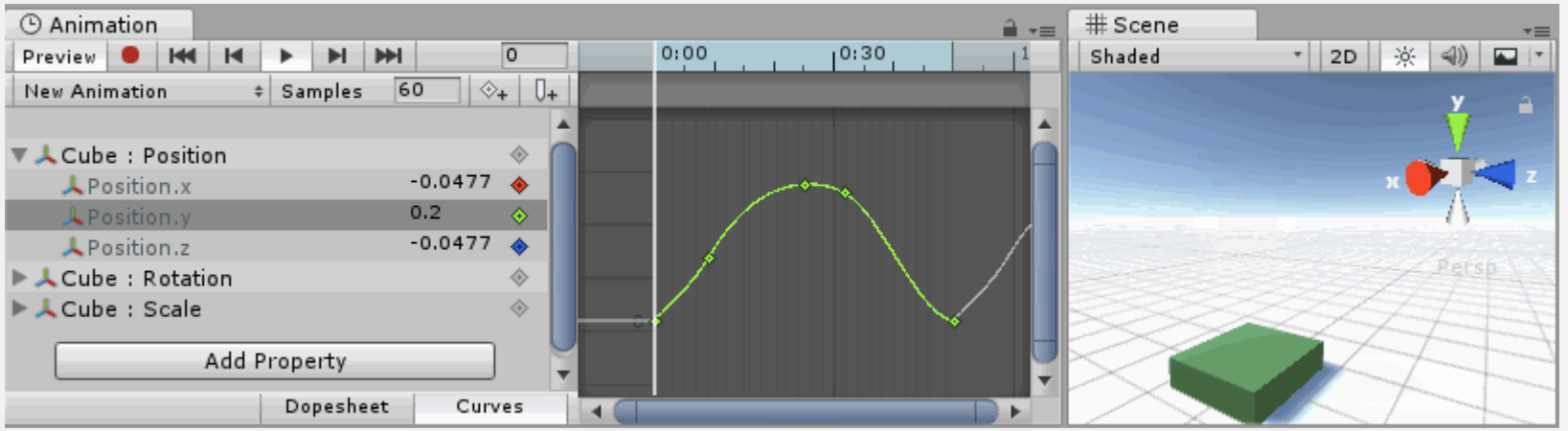
(c) Walt Disney Company, from "The Illusion of Life"

Keyframe Animation (Cont.)

- Key ideas of implementation
  - Create high-level controls for adjusting geometry
  - Interpolate these controls between keyframes over time
- Some animation principles can be modeled with physical laws
  - e.g. squash & stretch, ease in/out, following through, etc.
- Few additional requirements
  - Physically plausible (e.g. volume preservation)
  - Art-directibility

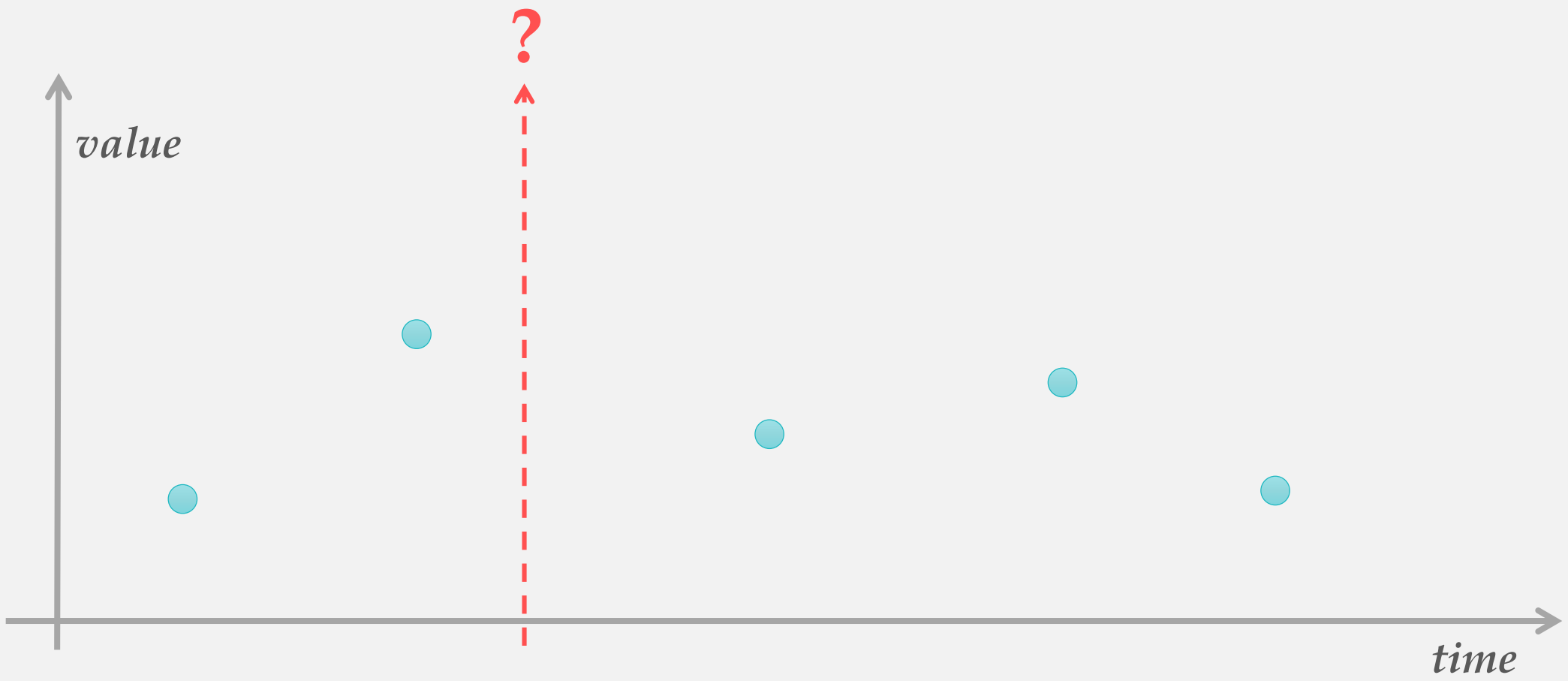
In-between Interpolations

- What do we interpolate exactly?
  - Shape (e.g. position, normal vector, etc.)
  - Transformations (i.e. translation, rotation, scale/sheer)
  - High-level controls (e.g. open mouth, bend elbow, etc.)
- How could we specify the interpolating behaviors?
  - Smoothness
  - Changes of speed



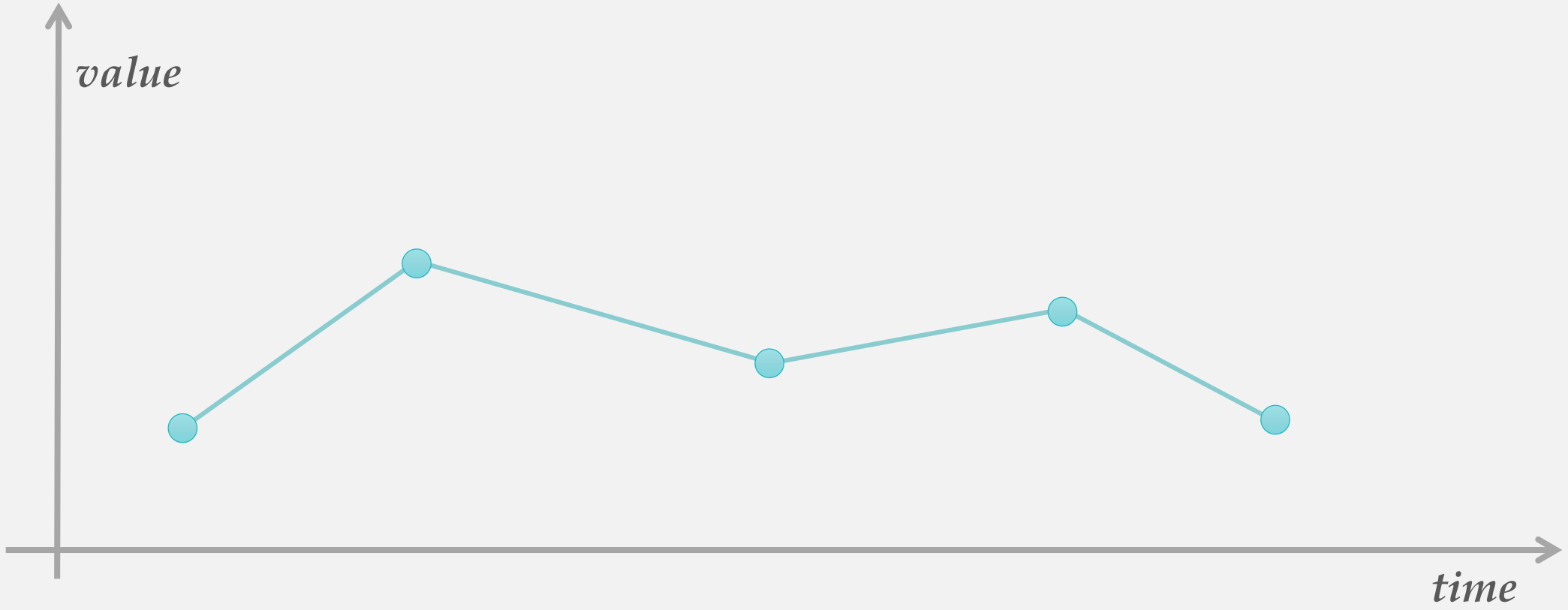
https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html

Data Interpolation in 1D



Piecewise Linear

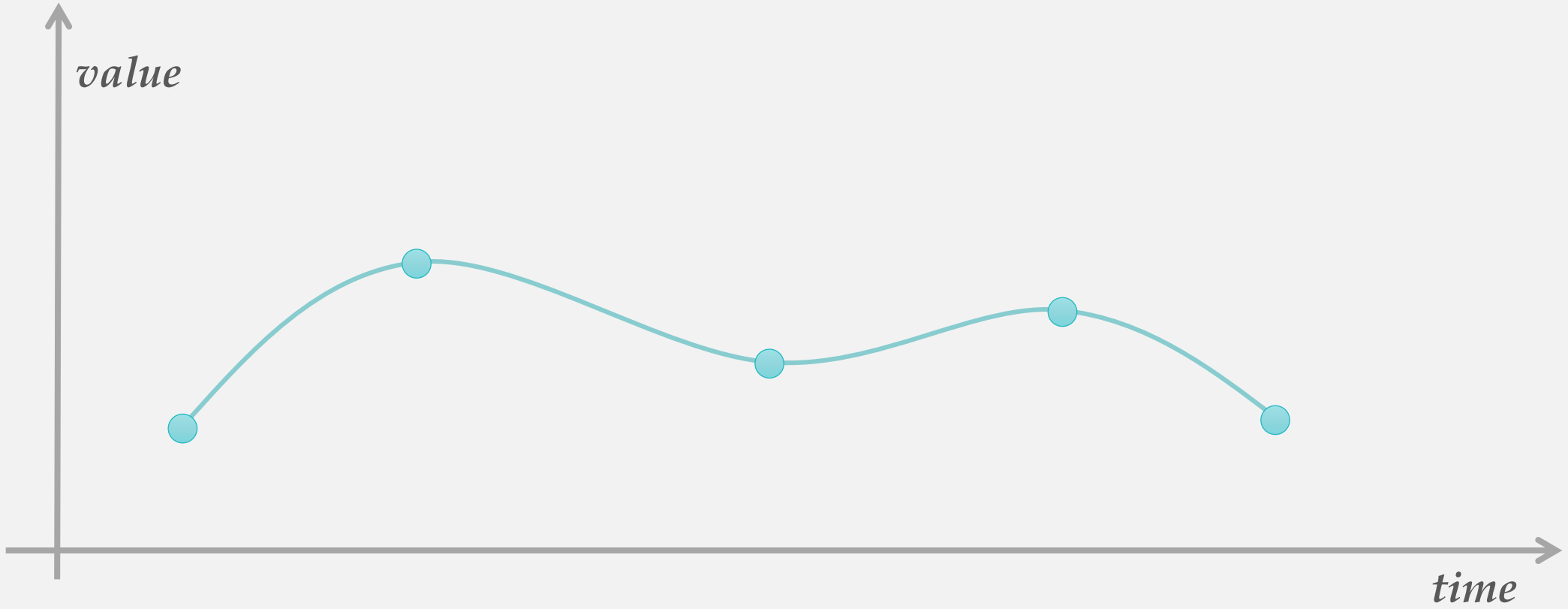
- Simple, but not smooth!



Piecewise Polynomial

- Represent data with polynomial function

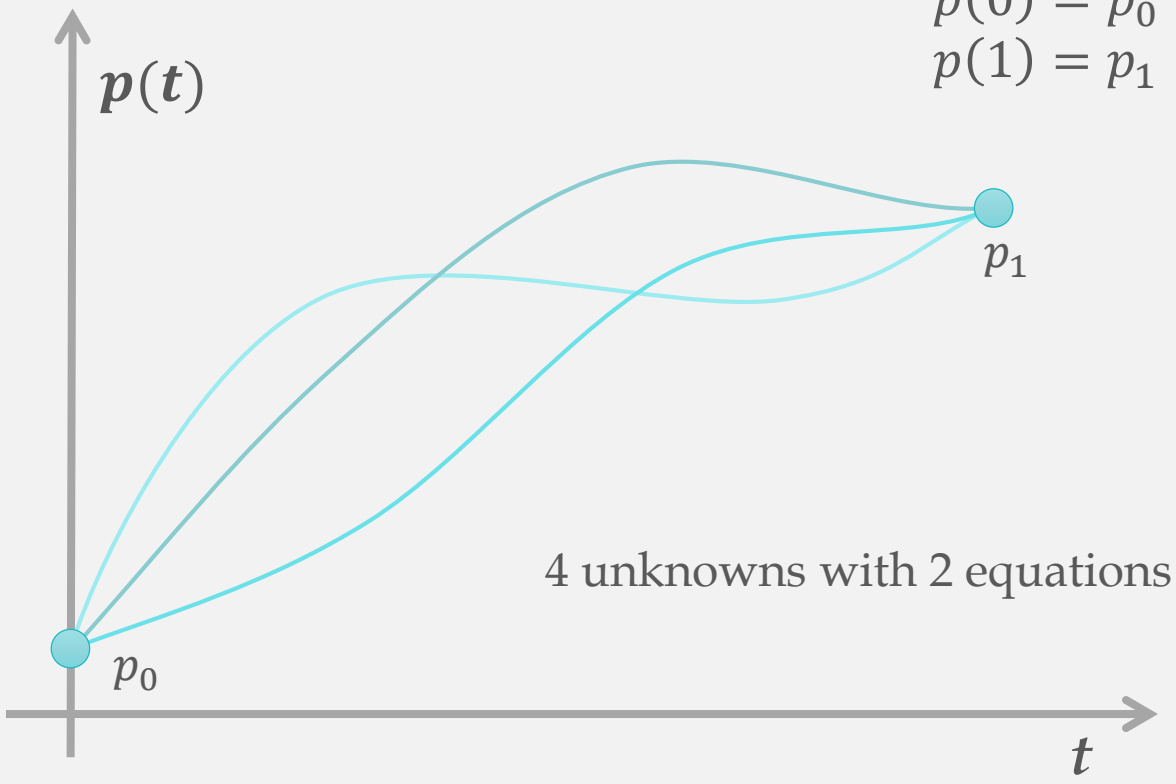
$$p(x) = a_0 + a_1 x + \cdots + a_n x^n, \qquad a_0, \ldots, a_n \in \mathbb{R}$$



Fitting Cubic Polynomial to End Points

$$p(t) = at^3 + bt^2 + ct + d, \qquad t \in [0,1]$$

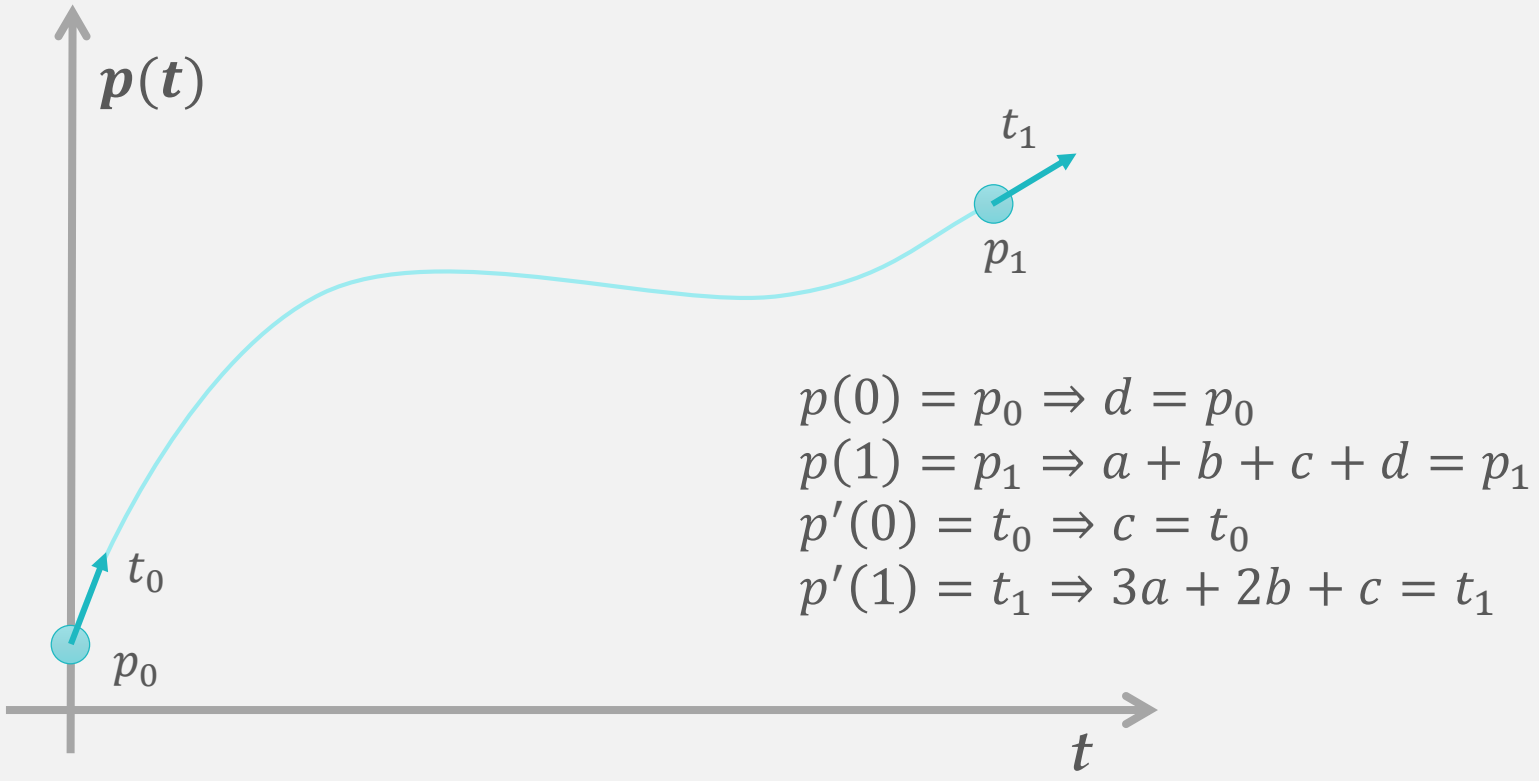
$$p(0) = p_0 \Rightarrow d = p_0$$
$$p(1) = p_1 \Rightarrow a + b + c + d = p_1$$



4 unknowns with 2 equations ⇒ no unique solution!

Fitting Cubic Polynomial to End Points and Tangents

- Provide extra tangents of endpoints



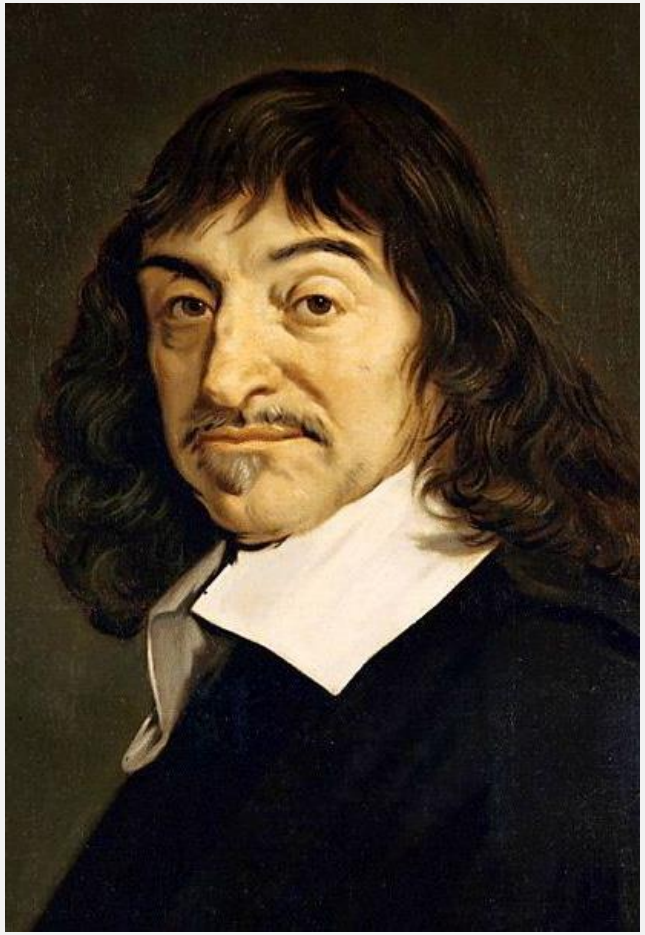
$$p(0) = p_0 \Rightarrow d = p_0$$
$$p(1) = p_1 \Rightarrow a + b + c + d = p_1$$
$$p'(0) = t_0 \Rightarrow c = t_0$$
$$p'(1) = t_1 \Rightarrow 3a + 2b + c = t_1$$

Interpolation with Parametric Curves

- Cubic Bezier: 4 positions
- Catmull-Rom: 2 positions, 2 tangents (derived from nearby CVs)
- Hermit Curve: 2 (position + tangent)
  - Tangents are specified at each CV
- What are *'good'* curves for interpolation?
  - Local control: each CV only affects neighboring segments
  - Smoothness, degree of continuity
    - C^0: matches position
    - C^1: matches tangent vector and its magnitude
    - C^2: matches curvature

Cartesian Unit Vectors

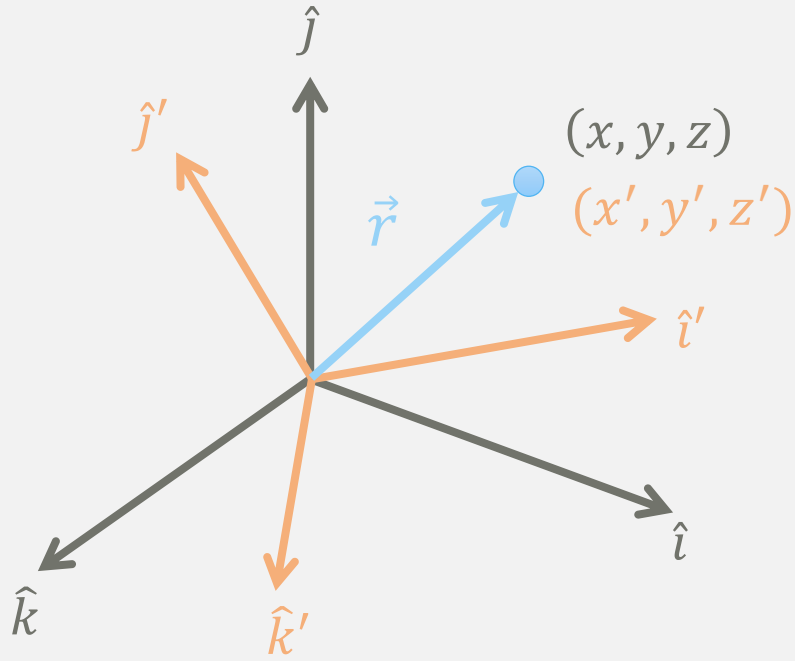
- $\hat{\imath}, \hat{\jmath}, \hat{k}$
  - Coordinate axes
  - Orthonormal
  - Can be drawn at any location, not just at origin
    - Invariant at different locations
- Vector components
  - Projections of the vector onto the coordinate axes



René Descartes (1596-1650)

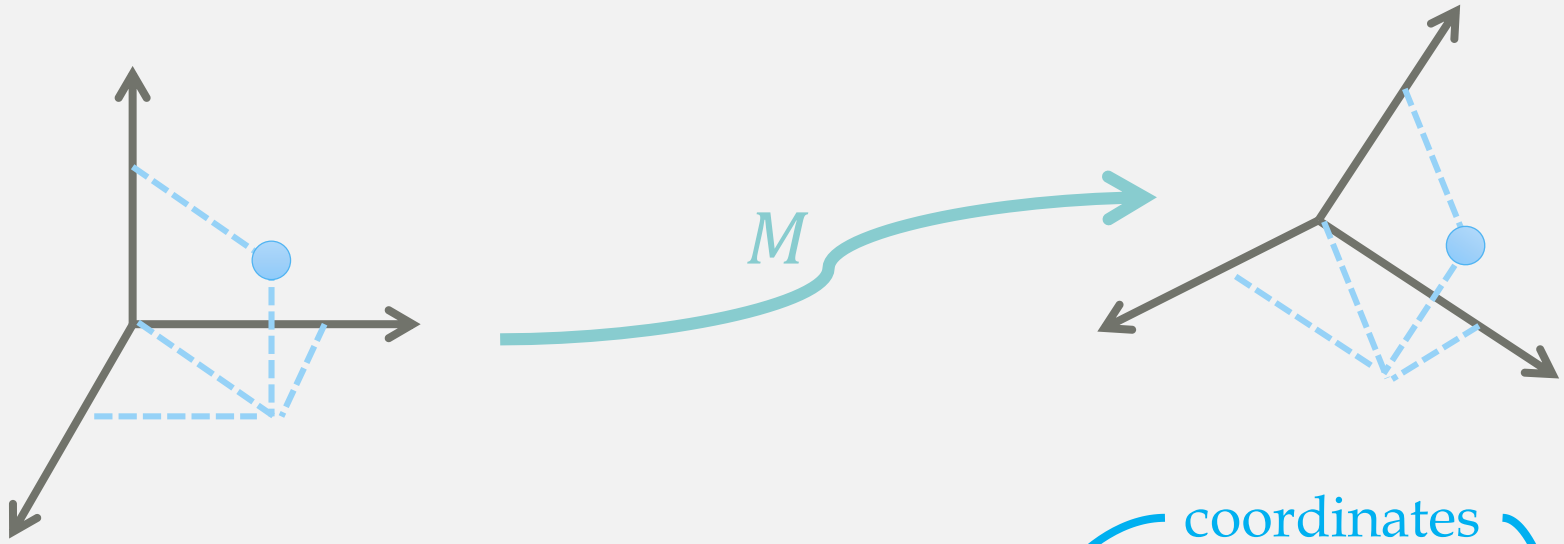
Change Axes in Cartesian Coordinate

- ***Geometric information = coordinates + unit basis***
  - Coordinates are meaningless without unit basis
- $\vec{r}$: displacement vector
- $\vec{r} = x\hat{\imath} + y\hat{\jmath} + z\hat{k} = x'\hat{\imath}' + y'\hat{\jmath}' + z'\hat{k}'$
- Two types of transformations
  - Coordinate-system transformations
    - Transform basis vector
    - Vector is the same, but components change
  - Transform vector in *the same* coordinate
    - Vector is different from original one



$\vec{r}$ is fixed!
But its components change!!

Orientation = Rotation



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t_1 & n_1 & b_1 \\ t_2 & n_2 & b_2 \\ t_3 & n_3 & b_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} x + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} y + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} z$$

coordinates

unit basis

orthogonal matrix: $R^T R = I$

Group

- A family of transformations forms a group
- A set G together with a binary operation $\circ$ defined on elements of G is called a group, if it satisfies the axioms of ***closure, identity, inverse and associativity***

Closure

$$g_1, g_2 \in G \rightarrow g_1 \circ g_2 \in G$$

Inverse

$$\forall g \; \exists g^{-1} \in G: \; g \circ g^{-1} = g^{-1} \circ g = e$$

Identity

$$\exists e \in G: \; g \circ e = e \circ g = g$$

Associativity

$$g_1, g_2, g_3 \in G, \quad g_1 \circ (g_2 \circ g_3) = (g_1 \circ g_2) \circ g_3$$

Introduction to Higher Mathematics - Lecture 16: Group Theory
SIG'14: Mathematical Basics of Motion and Deformation in Computer Graphics

Two Special Groups in 3D

- ***SO: Special Orthogonal group***

$$\mathrm{SO}(3) = \left\{\mathrm{R} \in \mathbb{R}^{3\times 3} : \mathrm{RR}^{\mathrm{T}} = \mathrm{I}, \det R = +1\right\}$$

  - 3D rotations centered at the origin

- ***SE: Special Euclidean group***

$$\mathrm{SE}(3) = \{(p, R) : p \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times \mathrm{SO}(3)$$

  - Rigid motion: 3D rotations with translations
    - preserve *distance* and *orientation*

Interpolating Rotation Matrices

90°CW around z-axis *90°CCW around z-axis*

$$0.5\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.5\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Interpolating Rotation Matrices

90°CW around z-axis *90°CCW around z-axis*

$$0.5\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.5\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & ? & 0 \\ 0 & ? & 0 \\ 0 & ? & 1 \end{bmatrix}$$

Oops!! This is NOT a rotation matrix!!

Rotation matrix is a group with multiplication **NOT** addition

Rotation Representations

- Rotation matrix
- Euler Angles
- Axis-angle
- Quaternion
- and many more…

Rotations

A project by @BERKELEY.EDU dynamics

http://rotations.berkeley.edu

After seeing this site, I just realized I didn't know much about rotations at all…

Euler's Rotation Theorem in 3D Space

- Any sequence of rotations about a fixed point is equivalent to a **single** rotation by a given angle θ about a fixed axis
- Any two ***orthonormal*** coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes
- Any two *Cartesian coordinate systems* with a *common* origin are related by a rotation about some fixed axis (orientation = rotation)

Leonhard Euler (1707-1783)

Euler's rotation theorem @Wikipedia

Euler Angles

- $R(\alpha, \beta, \gamma) = R_z(\gamma)R_y(\beta)R_x(\alpha)$
  - Product of 3 rotations around local axes
  - Rotation order is ***important***!
    - Ex. XYZ, ZXY, YZX, etc.

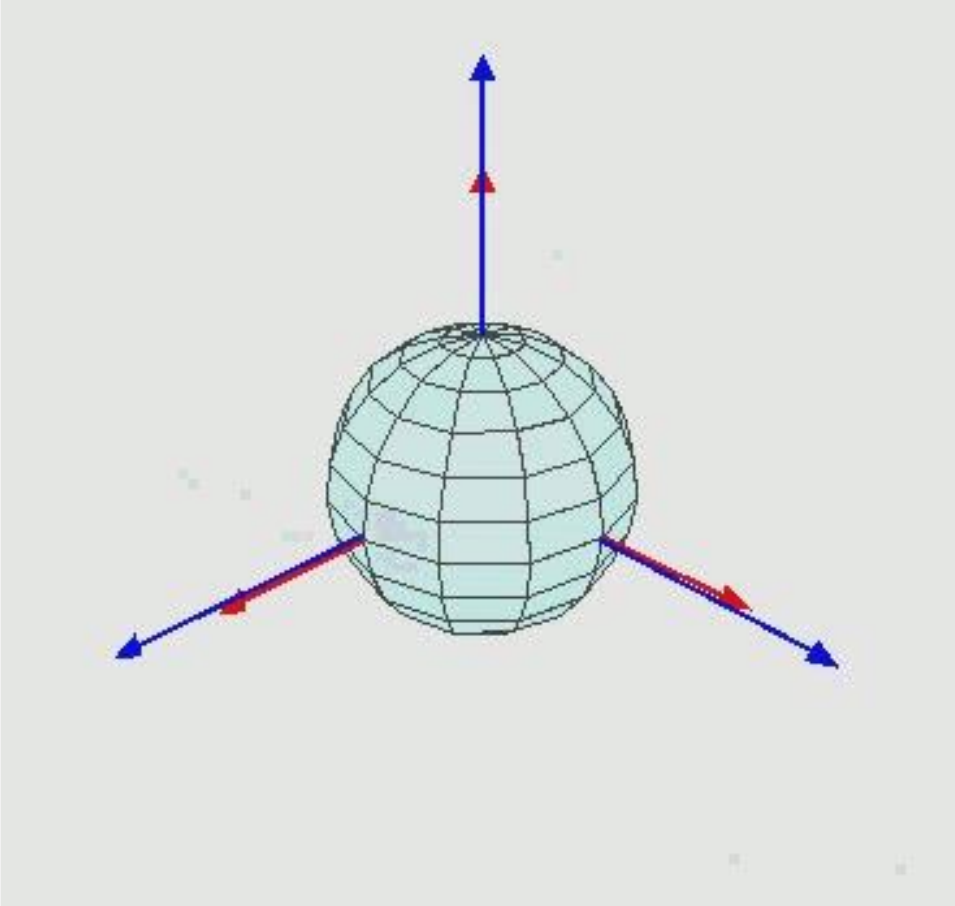
✓ Intuitive control

✓ Smallest representation possible

x Non-unique representation for a given orientation

x Hard to interpolate

x Gimbal lock



https://en.wikipedia.org/wiki/Euler_angles#/media/File:Euler2a.gif

Degrees of Freedom (DOFs)

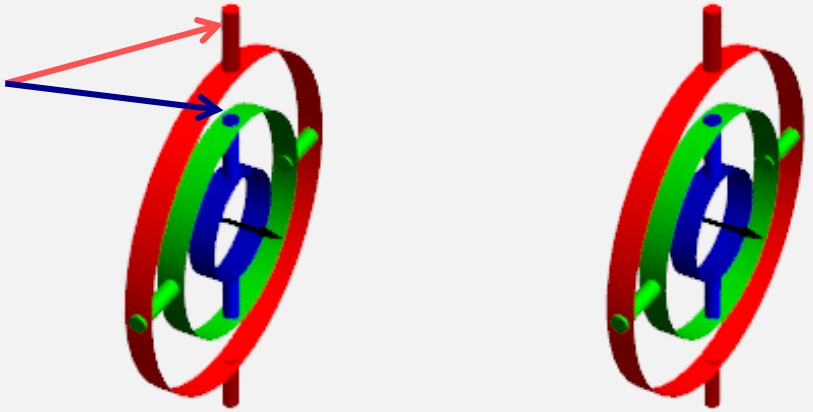
- A variable describing a particular axis or dimension of movement
- 3D Rotation: 3 DOFs
  - Axis-angle: axis θ, ϕ and rotation radius α
  - Euler angle: α, β, γ
  - Rotation matrix has 9 DOFs: 6 of them are redundant for rotation!
- Rigid body transformation in 3D: 6 DOFs
  - 3 for translation and 3 for rotation

Gimbal Lock

- When the second rotation value is $\pm\pi/2$, one degree of freedom (DOF) would be lost
- Can we use any specific rotation order to avoid this?

  Not possible!! ☹

red-axis is aligned with blue-axis!!



Euler (gimbal lock) Explained by The Guerrilla CG Project

https://www.allaboutcircuits.com/technical-articles/dont-get-lost-in-deep-space-understanding-quaternions/

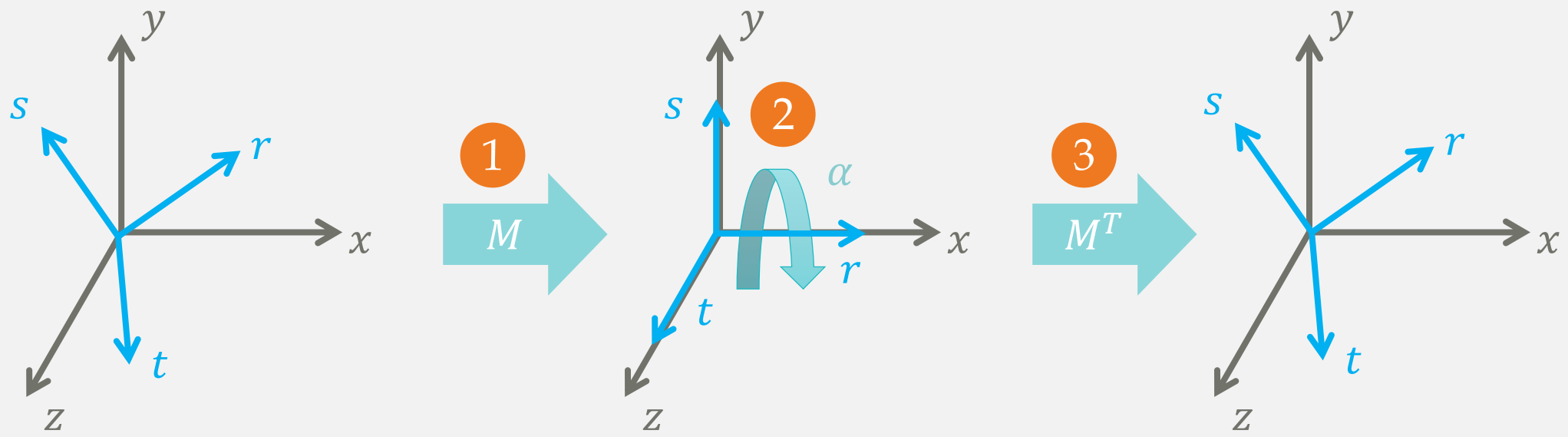
Singularity

When you go east at the **North Pole**, you are still at **the same** position!!

- A continuous subspace of the parameter space, where
  - all elements correspond to ***the same*** rotation
  - any movement within the subspace produces ***no*** change in rotation
- **NEVER** be eliminated in any 3-dimensional representation of SO(3)
  - That's why do we need quaternion!

Rotate About an Arbitrary Axis

1. Change to new frame
2. Rotate α radians around
3. Transform back to standard basis

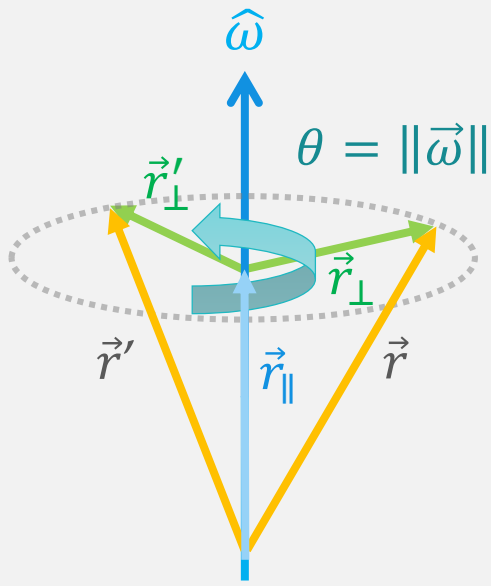


Axis-Angle

- Specify rotation axis $\hat{\omega}$, and rotation angle $\|\vec{\omega}\|$

$$\vec{r}'_{\perp} = \cos\theta\, \vec{r}_{\perp} + \sin\theta\, (\hat{\omega} \times \vec{r})$$

$$\vec{r}_{\perp} = \vec{r} - \vec{r}_{\parallel} = \vec{r} - (\vec{r} \cdot \hat{\omega})\hat{\omega}$$



$$\vec{r}' = \vec{r}'_{\perp} + \vec{r}_{\parallel}$$
$$= \cos\theta\, (\vec{r} - (\vec{r} \cdot \hat{\omega})\hat{\omega}) + \sin\theta\, (\hat{\omega} \times \vec{r}) + (\vec{r} \cdot \hat{\omega})\hat{\omega}$$
$$= \cos\theta\ \vec{r} + \sin\theta\, (\hat{\omega} \times \vec{r}) + (1 - \cos\theta)\big((\vec{r} \cdot \hat{\omega})\hat{\omega}\big)$$

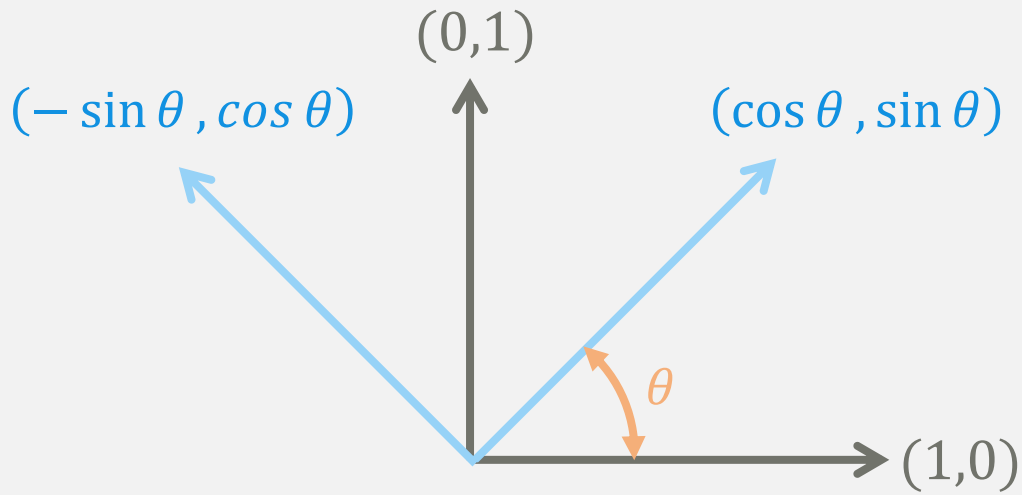
Axis-Angle (Cont.)

- ✓ Good for communication with geometry insight
- – Compact representation
  - The rotation angle could be embedded into the magnitude of vector (rotation axis)
- – Interpolate axis and angle separately?
- – Multiple representations for identity rotation (with a period of 2π)
- x Not used for composing rotations
  - Need to convert to matrix form back and forth

2D Rotation in Complex Plane

$$(x' + y'i) = e^{i\theta}(x + yi)$$
$$where\ e^{i\theta} = \cos\theta + i\sin\theta$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Is it possible to extend this concept to 3D?

Quaternion

$$\boldsymbol{i^2 = j^2 = k^2 = ijk = -1}$$

$$\begin{aligned} ijk &= -1 \\ i(ijk) &= i(-1) \\ -jk &= -i \\ jk &= i \end{aligned}$$

$$ij = k, \quad ji = -k, \qquad jk = i, \quad kj = -i, \qquad ki = j, \quad ik = -j$$

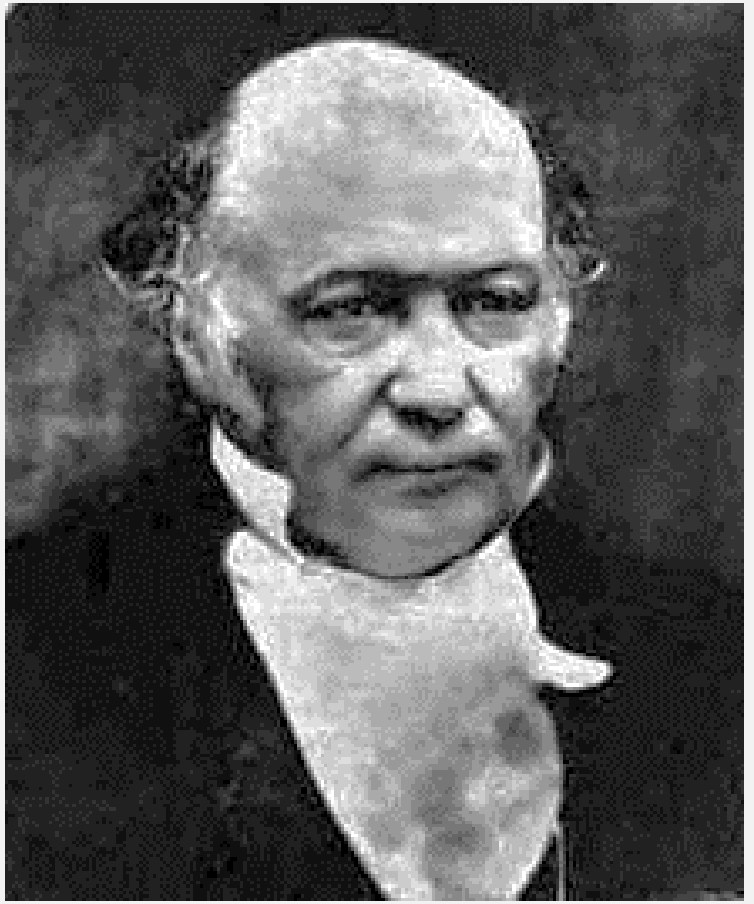
- Can be represented in several ways:

$$q = (w, x, y, z)$$

$$q = w + xi + yj + zk$$

$$q = w + \boldsymbol{v}$$

scalar part *vector part*



William Rowan Hamilton (1805–1865)

Hamilton Product

$$i^2 = j^2 = k^2 = ijk = -1$$
$$ij = k, \quad jk = i, \quad ki = j$$
$$ji = -k, \quad kj = -i, \quad ik = -j$$

$$q_0 * q_1 = (w_0 + x_0 i + y_0 j + z_0 k) * (w_1 + x_1 i + y_1 j + z_1 k)$$

$$= w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1$$
$$+ (w_0 x_1 + x_0 w_1 + y_0 z_1 - z_0 y_1) i$$
$$+ (w_0 y_1 + y_0 w_1 - x_0 z_1 + z_0 x_1) j$$
$$+ (w_0 z_1 + z_0 w_1 + x_0 y_1 - y_0 x_1) k$$

$$= w_0 w_1 - \boldsymbol{v_0} \cdot \boldsymbol{v_1} + w_0 \boldsymbol{v_1} + w_1 \boldsymbol{v_0} + \boldsymbol{v_0} \times \boldsymbol{v_1}$$

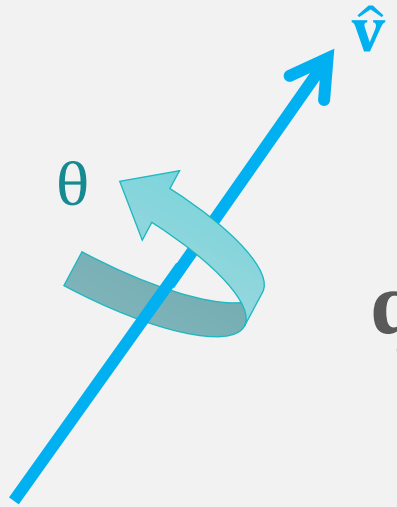
non-commutative!!

$$q_1 * q_0 \neq q_0 * q_1$$

Quaternion (Cont.)

- Linearity
  - $\boldsymbol{q_0} + \boldsymbol{q_1} = (w_0 + w_1, \boldsymbol{v_0} + \boldsymbol{v_1})$
  - $\alpha \boldsymbol{q} = \boldsymbol{q}\alpha = (\alpha w, \alpha \boldsymbol{v})$
- Identity: $\mathbf{q}_\mathrm{I} = (1, 0, 0, 0)^\mathrm{T}$
- Conjugate: $q^* = (w, -\boldsymbol{v})$
  - $(\boldsymbol{q}^*)^* = \boldsymbol{q}$
  - $(\boldsymbol{pq})^* = \boldsymbol{q}^*\boldsymbol{p}^*$
  - $(\boldsymbol{p} + \boldsymbol{q})^* = \boldsymbol{p}^* + \boldsymbol{q}^*$
- Norm
  - $N(\boldsymbol{q}) = \boldsymbol{q}\boldsymbol{q}^* = \boldsymbol{q}^*\boldsymbol{q} = w^2 + x^2 + y^2 + z^2$
  - $N(\boldsymbol{q_0}\boldsymbol{q_1}) = N(\boldsymbol{q_0})N(\boldsymbol{q_1})$
  - $N(\boldsymbol{q}^*) = N(\boldsymbol{q})$
- Inverse: $\boldsymbol{q^{-1}} = \frac{\boldsymbol{q}^*}{N(\boldsymbol{q})}$
  - $\mathbf{q} \circ \mathbf{q}^{-1} = \mathbf{q}^{-1} \circ \mathbf{q} = (1, 0, 0, 0)^\mathrm{T}$
  - $(\boldsymbol{q_0}\boldsymbol{q_1})^{-1} = \boldsymbol{q_1^{-1}}\boldsymbol{q_0^{-1}}$
- Difference: $\mathbf{q_0}\mathbf{q_d} = \boldsymbol{q_1} \Rightarrow \boldsymbol{q_d} = \boldsymbol{q_0^{-1}}\boldsymbol{q_1}$

Unit Quaternion



*Unit quaternion q and –q represent the **same** orientation*

$$\mathbf{q} = (w, x, y, z)^T = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{v}}\right]^T$$

why $\frac{1}{2}$ *?*

Vector Rotation

$$\mathrm{p}' = \mathrm{rotate}(\boldsymbol{p}) = \mathrm{q} * \tilde{p} * \mathrm{q}^{-1}$$

- Rotate a vector $\mathbf{p} \in \mathbb{R}^3$ by an **unit** quaternion $\mathrm{q} \in \mathcal{S}^3$
- $\tilde{p} = (0, \boldsymbol{p})^{\mathrm{T}}$ extended with a zero scalar component
- *rotate()* function would strip off the scalar part of quaternion

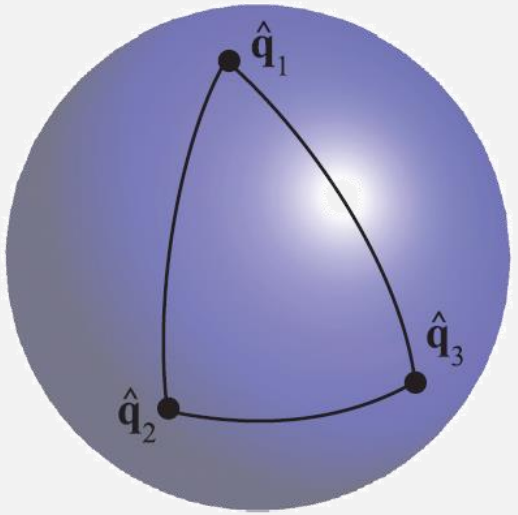


Figure from Real-time Rendering, 3/e

Quaternion – Why $\theta/2$??

Recall: $\boldsymbol{q_0 q_1} = w_0 w_1 - \boldsymbol{v_0} \cdot \boldsymbol{v_1} + w_0 \boldsymbol{v_1} + w_1 \boldsymbol{v_0} + \boldsymbol{v_0} \times \boldsymbol{v_1}$

suppose $\boldsymbol{q}$ *is an* ***unit*** *quaternion*

$$
\begin{aligned}
qpq^{-1} &= (w + t\hat{v})\vec{p}(w + t\hat{v})^{-1} \\
&= (-t\hat{v} \cdot \vec{p} + w\vec{p} + t\hat{v} \times \vec{p})(w - t\hat{v}) \\
&= -wt\hat{v} \cdot \vec{p} + (w\vec{p} + t\hat{v} \times \vec{p}) \cdot t\hat{v} + w(w\vec{p} + t\hat{v} \times \vec{p}) \\
&\quad + (t\hat{v} \cdot \vec{p})t\hat{v} - (w\vec{p} + t\hat{v} \times \vec{p}) \times t\hat{v} \\
&= w^2\vec{p} + 2wt\hat{v} \times \vec{p} + t^2(\hat{v} \cdot \vec{p})\hat{v} - t^2\hat{v} \times \vec{p} \times \hat{v} \\
&= \left(w^2 - t^2\right)\vec{p} + 2wt\hat{v} \times \vec{p} + 2t^2(\vec{p} \cdot \hat{v})\hat{v}
\end{aligned}
$$

Quaternion – Why $\theta/2$??

Recall: $\boldsymbol{q_0 q_1} = w_0 w_1 - \boldsymbol{v_0} \cdot \boldsymbol{v_1} + w_0 \boldsymbol{v_1} + w_1 \boldsymbol{v_0} + \boldsymbol{v_0} \times \boldsymbol{v_1}$

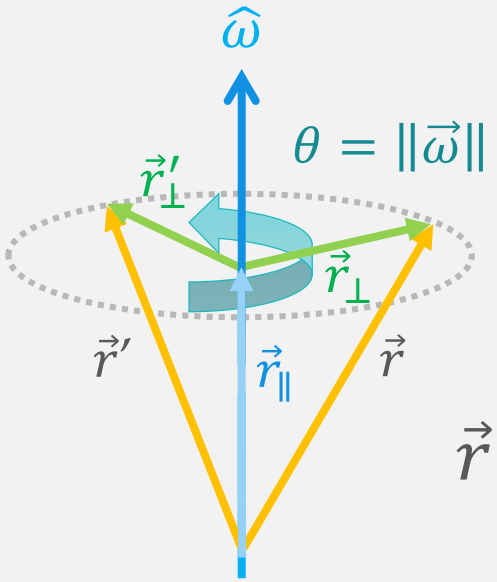
suppose $\boldsymbol{q}$ *is an* ***unit*** *quaternion*

$$
\begin{aligned}
qpq^{-1} &= (w + t\hat{v})\vec{p}(w + t\hat{v})^{-1} \\
&= (-t\hat{v} \cdot \vec{p} + w\vec{p} + t\hat{v} \times \vec{p})(w - t\hat{v}) \\
&= -wt\hat{v} \cdot \vec{p} + (w\vec{p} + t\hat{v} \times \vec{p}) \cdot t\hat{v} + w(w\vec{p} + t\hat{v} \times \vec{p}) \\
&\quad + (t\hat{v} \cdot \vec{p})t\hat{v} - (w\vec{p} + t\hat{v} \times \vec{p}) \times t\hat{v} \\
&= w^2\vec{p} + 2wt\hat{v} \times \vec{p} + t^2(\hat{v} \cdot \vec{p})\hat{v} - t^2\hat{v} \times \vec{p} \times \hat{v} \\
&= \left(w^2 - t^2\right)\vec{p} + 2wt\hat{v} \times \vec{p} + 2t^2(\vec{p} \cdot \hat{v})\hat{v}
\end{aligned}
$$

Look familiar??

Recap

Axis-Angle Rotation



$$\vec{r}' = \cos\theta\ \vec{r} + \sin\theta\,(\hat{\omega} \times \vec{r}) + (1 - \cos\theta)\big((\vec{r}\ \cdot \hat{\omega})\hat{\omega}\big)$$

Quaternion – Why $\theta/2$?? (Cont.)

$$
\begin{aligned}
qpq^{-1} &= (w + t\hat{v})\vec{p}(w + t\hat{v})^{-1} \\
&= (-t\hat{v}\cdot\vec{p} + w\vec{p} + t\hat{v}\times\vec{p})(w - t\hat{v}) \\
&= -wt\hat{v}\cdot\vec{p} + (w\vec{p} + t\hat{v}\times\vec{p})\cdot t\hat{v} + w(w\vec{p} + t\hat{v}\times\vec{p}) \\
&\quad +(t\hat{v}\cdot\vec{p})t\hat{v} - (w\vec{p} + t\hat{v}\times\vec{p})\times t\hat{v} \\
&= w^2\vec{p} + 2wt\hat{v}\times\vec{p} + t^2(\hat{v}\cdot\vec{p})\hat{v} - t^2\hat{v}\times\vec{p}\times\hat{v} \\
&= \left(w^2 - t^2\right)\vec{p} + 2wt\hat{v}\times\vec{p} + 2t^2(\vec{p}\cdot\hat{v})\hat{v}
\end{aligned}
$$

$$
\vec{r}' = \cos\theta\ \vec{r} + \sin\theta\,(\hat{\omega}\times\vec{r}) + (1 - \cos\theta)\left((\vec{r}\cdot\hat{\omega})\hat{\omega}\right)
$$

Quaternion – Why $\theta/2$?? (Cont.)

$$w^2 - t^2 = \cos\theta$$
$$2wt = \sin\theta$$
$$2t^2 = 1 - \cos\theta \Rightarrow t = \sin\frac{\theta}{2} \Rightarrow w = \cos\frac{\theta}{2}$$

where $2\sin\theta\cos\theta = \sin 2\theta$

therefore **unit** quaternion represents a rotation

$\boldsymbol{q} = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{v}}\right) \leftrightarrow$ rotate θ around axis $\hat{\mathbf{v}}$

Quaternion – Why $\theta/2$?? (Cont.)

$$w^2 - t^2 = \cos\theta$$
$$2wt = \sin\theta$$
$$2t^2 = 1 - \cos\theta$$

Read More

1. Quaternions, Ken Shoemake.
2. Game Physics 2/e, Ch10, David H. Eberly.

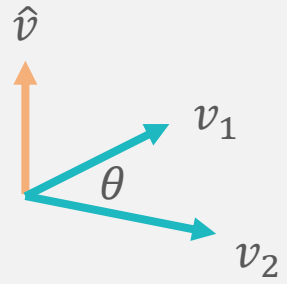
$\cos\theta = \sin 2\theta$

therefore **unit** quaternion represents a rotation

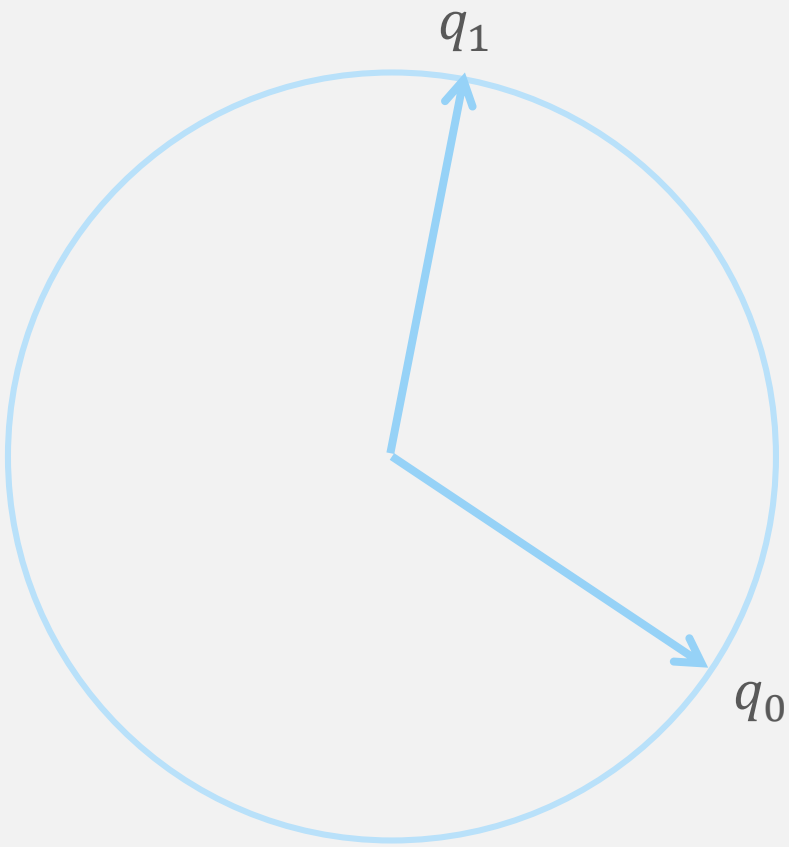
$\boldsymbol{q} = (\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{v}}) \leftrightarrow$ rotate θ around axis $\hat{\mathbf{v}}$

Quaternion qpq^{-1}

- Concatenation
  - $q_1 \cdot (q_0 \cdot p \cdot q_0^{-1}) \cdot q_1^{-1} = (q_1 \cdot q_0) \cdot p \cdot (q_1 \cdot q_0)^{-1}$
- Any non-zero real multiple of q gives the same action
  - $(sq)p(sq)^{-1} = (sq)p(q^{-1}s^{-1}) = qpq^{-1}ss^{-1} = qpq^{-1}$
- Create rotation quaternion: given v_1, want to rotate to v_2
  1. $\hat{v} = v_1 \times v_2,\ \theta = \text{acos}(v_1 \cdot v_2)$
  2. $q = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{v}\right), p = (0, v_1)$
  3. $p' = qpq^*$

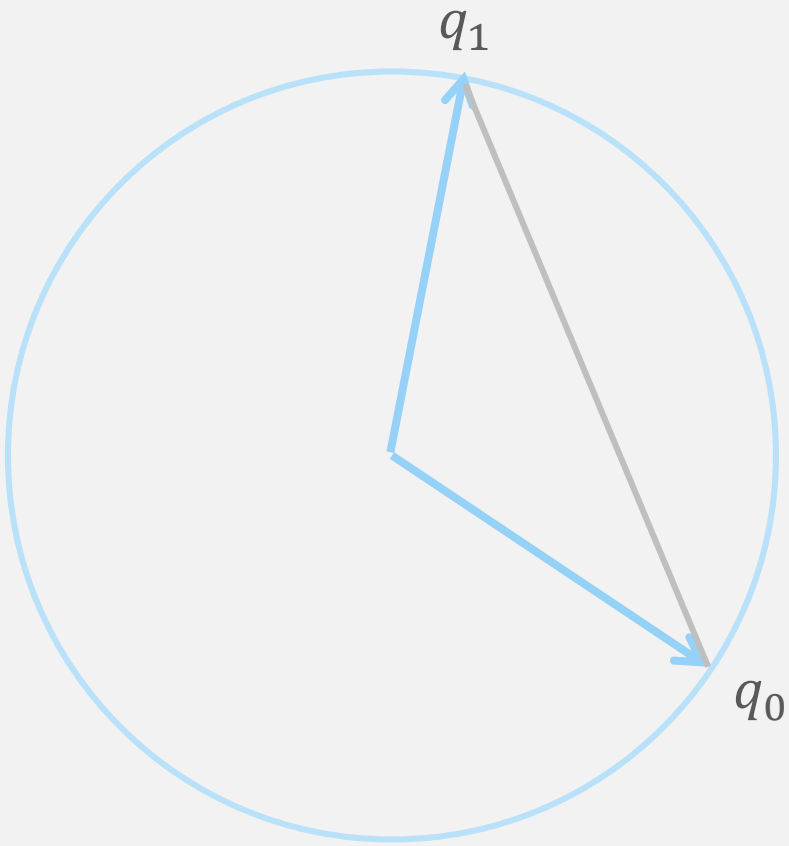


Linear Quaternion Interpolations (LQP)



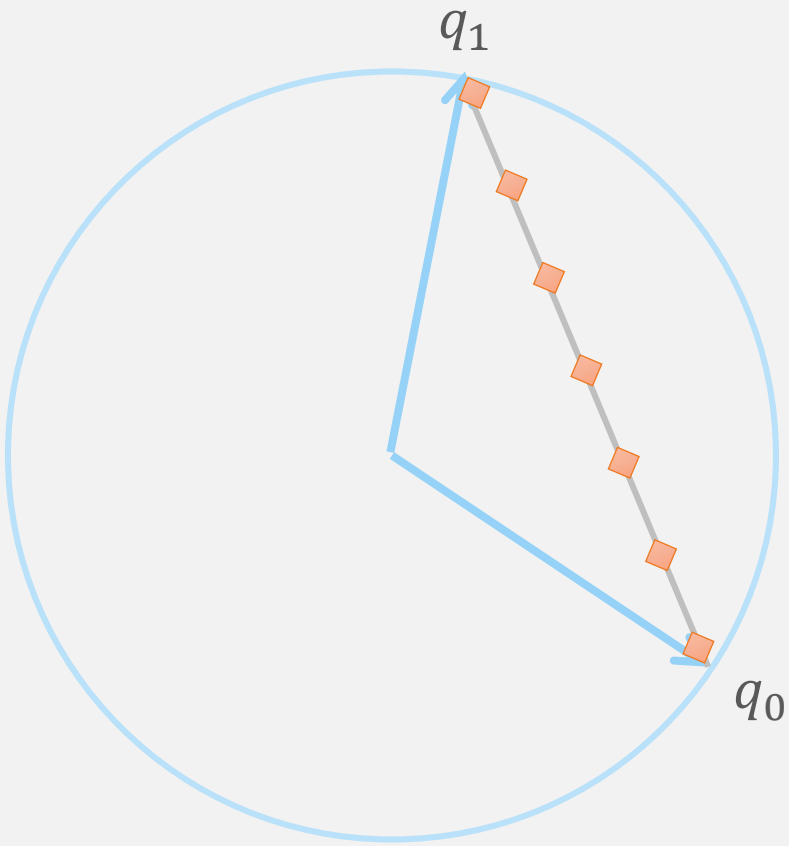
$$\frac{(1-\alpha)q_0 + \alpha q_1}{\|(1-\alpha)q_0 + \alpha q_1\|}$$

Linear Quaternion Interpolations (LQP)



$$\frac{(1-\alpha)q_0 + \alpha q_1}{\|(1-\alpha)q_0 + \alpha q_1\|}$$

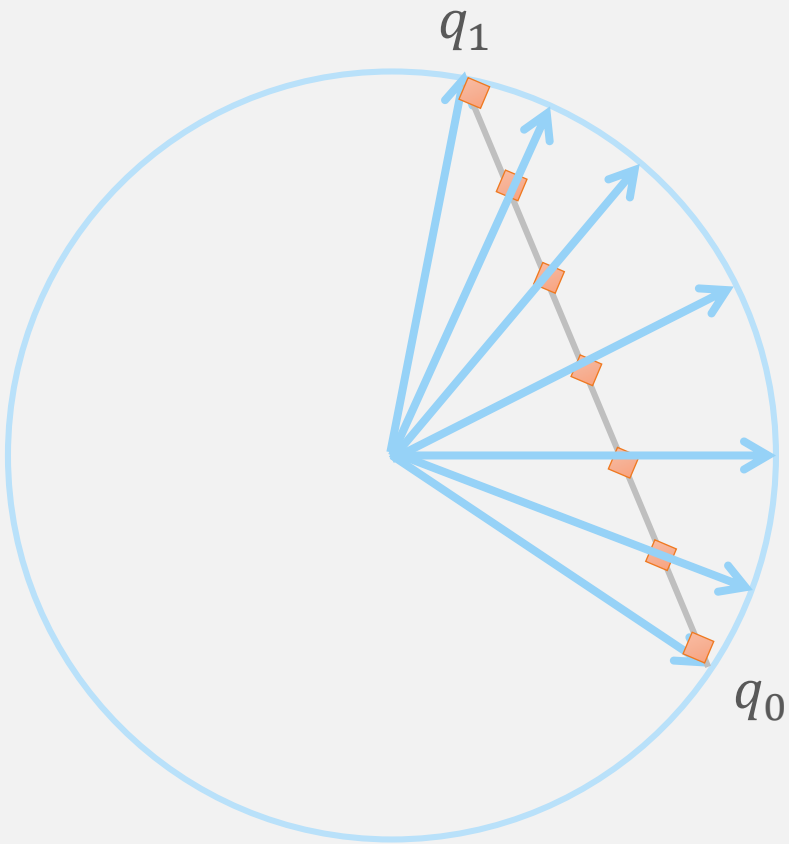
Linear Quaternion Interpolations (LQP)



$$(1-\alpha)q_0 + \alpha q_1$$

$$\frac{(1-\alpha)q_0 + \alpha q_1}{\|(1-\alpha)q_0 + \alpha q_1\|}$$

Linear Quaternion Interpolations (LQP)

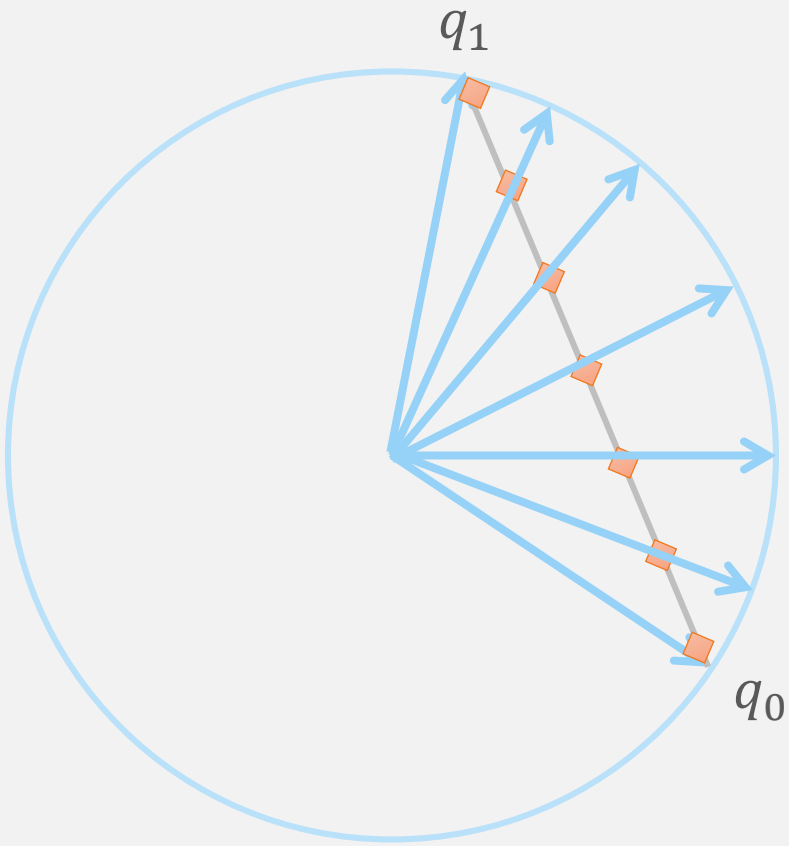


$$(1-\alpha)q_0 + \alpha q_1$$

Need normalization… ↓↓

$$\frac{(1-\alpha)q_0 + \alpha q_1}{\|(1-\alpha)q_0 + \alpha q_1\|}$$

Linear Quaternion Interpolations (LQP)



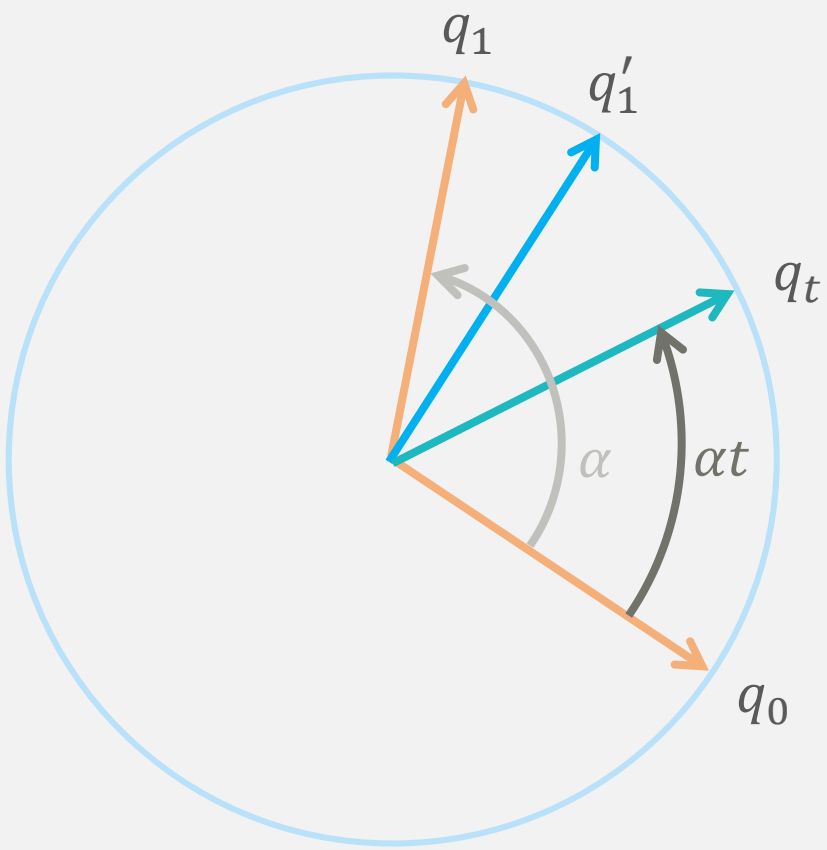
$$(1-\alpha)q_0 + \alpha q_1$$

Need normalization… ↓↓

$$\frac{(1-\alpha)q_0 + \alpha q_1}{\|(1-\alpha)q_0 + \alpha q_1\|}$$

The angular speed is **NOT** constant!!

Spherical Linear Interpolation (SLERP)



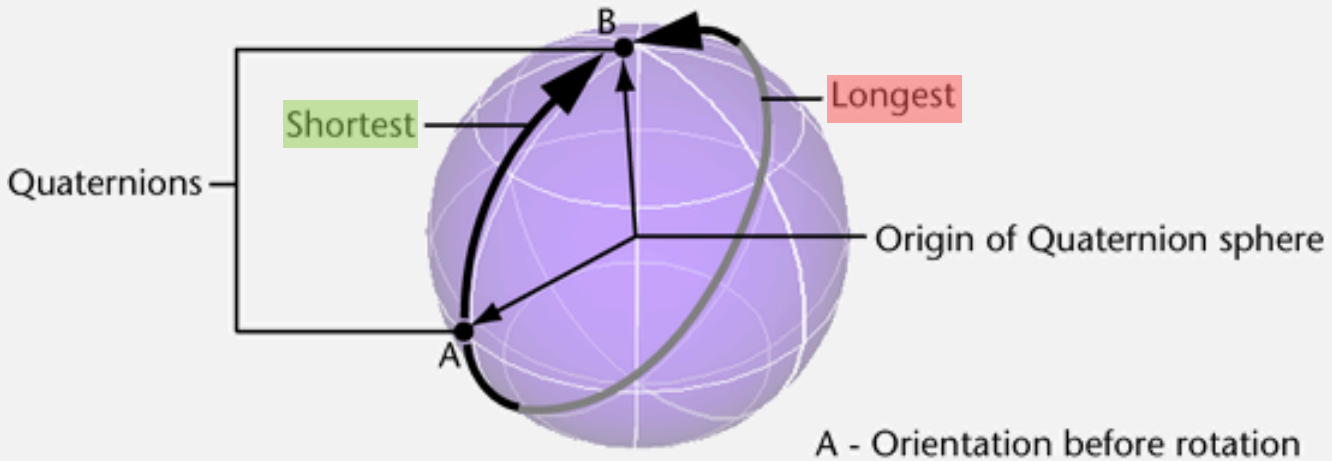
$$q_t = (\cos \alpha t) q_0 + (\sin \alpha t) q_1'$$

$$q_1' = \frac{q_1 - \cos \alpha \, q_0}{\sin \alpha}$$

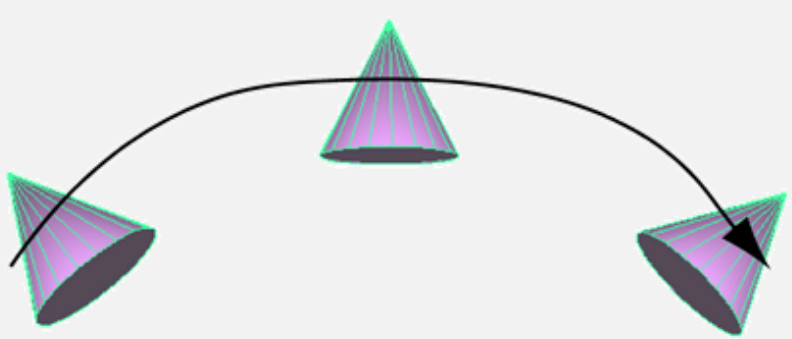
$$q_t = \frac{\sin[(1-t)\alpha]}{\sin \alpha} \mathrm{q}_0 + \frac{\sin \alpha t}{\sin \alpha} \mathrm{q}_1$$

Numerical error as $\alpha \to 0$, use lerp instead!

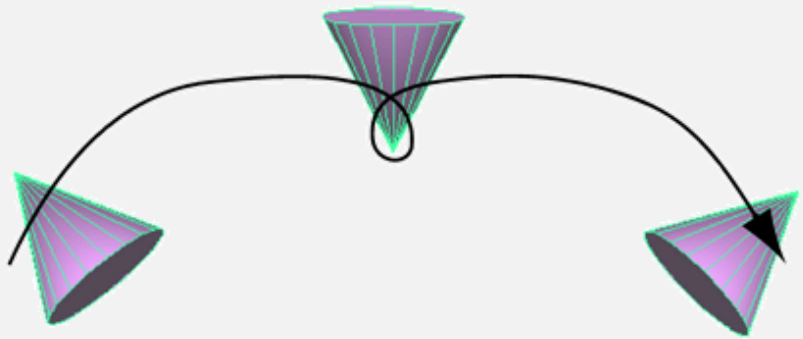
Twist in Longest Rotation Interpolation



Quaternion rotation interpolation



Quaternion Shortest



Quaternion Longest

Animated rotation in Maya

Why Quaternion?

- ✓ Smooth interpolation with SLERP
- ✓ Without singularity (gimbal lock)
- ✓ Compact representation (only 4 numbers)
- ✓ Fast concatenation and inversion of angular displacements
- ✓ Fast conversion from/to matrix representation
- – Not intuitive

Rotations

- Do not commute
- SO(3) is a 3D non-Euclidean space
- Parameterization considerations
  - compute position/orientation from parameters
  - compute derivatives of position/orientation w.r.t. parameters
  - ability to integrate ODEs in parameter space
  - ability to interpolate smoothly between key-frames
  - ability to combine rotations
  - existences of an inverse operation

Character Animation

Shape Interpolation

- Now we know how to interpolate 1D values and transformations, how about the pose-to-pose interpolation?
  - Rigid body: the relative orientations between all vertices keep constant
  - Soft body: deformable object
- How do we create different poses for each keyframe?
  - Per-vertex animation?
  - Hierarchical geometry interpolation
- How do we smoothly interpolate pose-to-pose geometry data?

Character Rigging

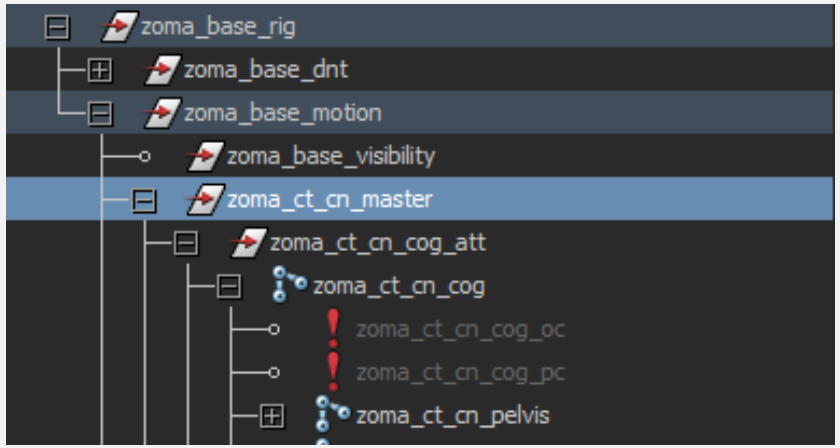
- Provide high-level controls which parameterize the meaningful deformations
- The quality of rigging has a large influence on the quality of animation
- Care should be taken about the computation performance
  - Efficient rigs reduce the time of geometry adjustment
- Types of rigging
  - Anatomical rigging
  - Skeletal rigging
  - Blend shapes

Anatomical Rigging from Weta Digital

https://www.youtube.com/watch?v=YncZtLaZ6kQ

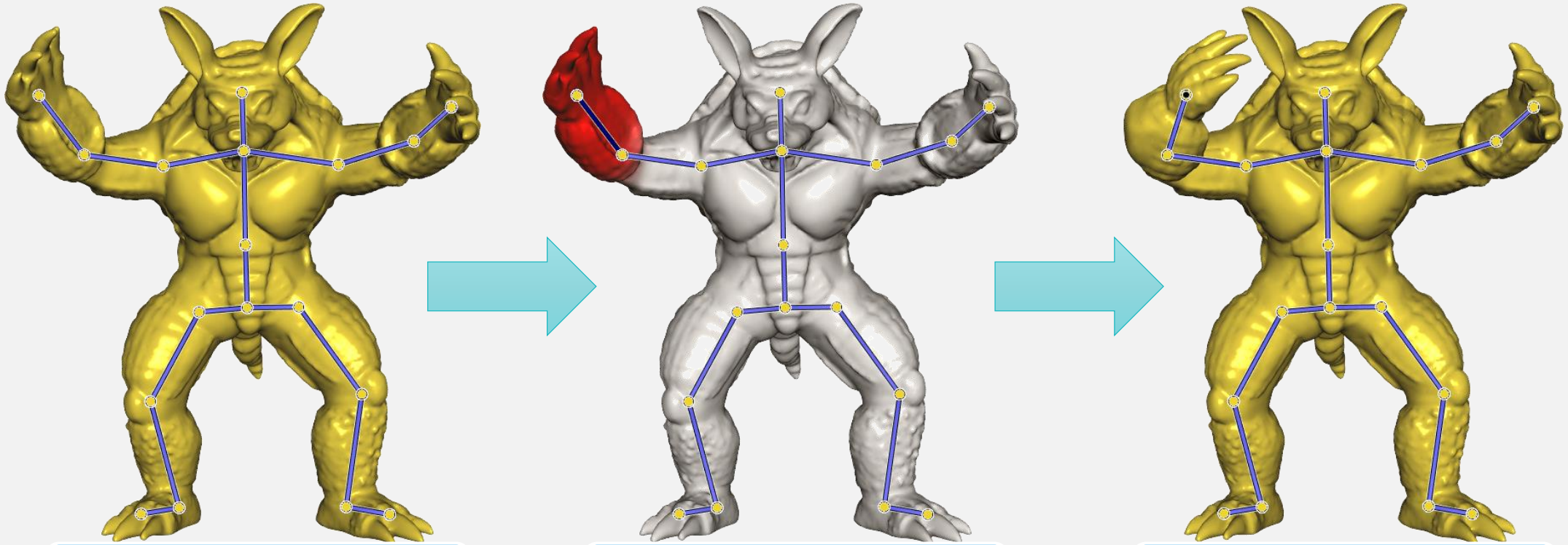
Skeletal Rigging

- Hierarchy of affine transformations
  - Changes to parent frame affect all descendent bones
  - Joints: local coordinate frames
  - Bones: vectors between consecutive pairs of joints (without loops)
- A joint allows relative movement within the skeleton
  - essentially 4x4 matrix transformation
  - can also be rotational, translational, or some non-realistic types



Hotel Transylvania / Zombie Rig from SONY Pictures Animation

Skeletal Animation Workflow



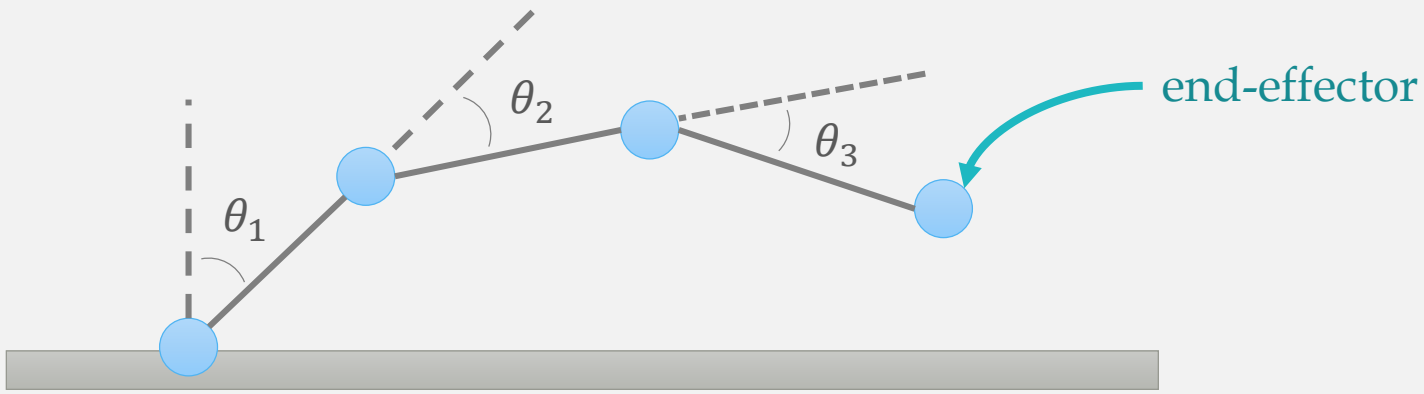
Animating Process in Disney

https://www.youtube.com/watch?v=z-4c3JSLVww

Revisit

Degrees of Freedom (DOFs)

- A variable describing a particular axis or dimension of movement
- Rigid body transformation has 6 DOFs
  - 3 for position and 3 for rotation
- **Pose:** a vector of N numbers that maps to a set of DOFs in the skeleton
- **Animation:** changing parameters of DOFs over time



Forward Kinematics

Hotel Transylvania / Zombie Rig from SONY Pictures Animation

Forward Kinematics (Cont.)

- Compute geometry properties from corresponding DOFs
- Assume $n + 1$ joints: $J_0, J_1, \ldots, J_n$
- Each joint corresponds to a local frame
  - The frame of j-th joint J_j is expressed w.r.t. parent joint $J_{p(j)}$

$$M_j = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_j & T_j \\ 0 & 1 \end{pmatrix}$$

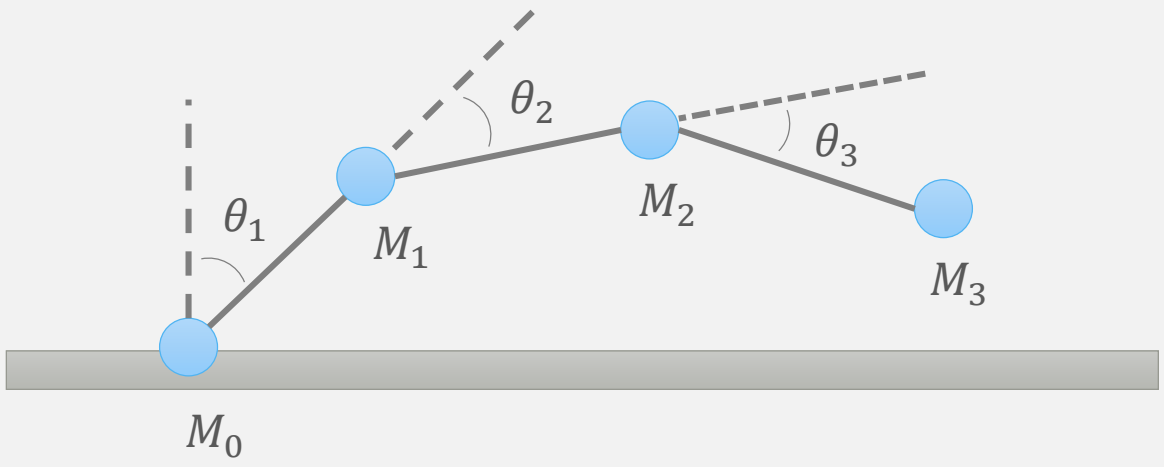
  - M_j contains orientation R_j and translation T_j
    - T_j typically comes from bind pose set by rigger
    - R_j usually set by animator

Forward Kinematics (Cont.)

- The transformation from joint j to world space is

$$M_j^w = M_0 \dots M_{p(p(j))} M_{p(j)} M_j$$

- Each joint rotation has up to 3 DOFs
- $M_3^w = M_0 M_1 M_2 M_3 = R(\theta_1) T_1 R(\theta_2) T_2 R(\theta_3) T_3$



Inverse Kinematics

Hotel Transylvania / Zombie Rig from SONY Pictures Animation

Inverse Kinematics (Cont.)

- Determine parameters of DOFs from the positions (and orientations) of end-effectors

$$\Theta = f^{-1}(x)$$

- $\Delta x \approx \frac{dJ}{d\Theta}\Delta\Theta = J(f,\Theta)\Delta\Theta = \mathrm{J}\Delta\Theta \Rightarrow \Delta\Theta = J^{*}\Delta x$
  - $\Delta x = \beta(g - x),\ where\ 0 < \beta \leq 1$
  - $J^{*} = \left(J^{T}J\right)^{-1}J^{T}$

$$J(f,\Theta) = \frac{\partial f}{\partial \Theta} = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_N} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial \theta_1} & \frac{\partial f_M}{\partial \theta_1} & \cdots & \frac{\partial f_M}{\partial \theta_N} \end{bmatrix}$$

- Might have many or no solutions

Inverse Kinematics (Cont.)

- Determine parameters of DOFs from the positions (and orientations) of end-effectors

$$\Theta = f^{-1}(x)$$

- $\Delta x \approx \frac{dJ}{d\Theta}\Delta\Theta = J(f,\Theta)\Delta\Theta = \mathrm{J}\Delta\Theta \Rightarrow \Delta\Theta = J^{*}\Delta x$
  - $\Delta x = \beta(g - x)$, *where* 0 <

Read More

Inverse Kinematics And Geometric Constraints for Articulated Figure Manipulation, Chris Welman, 1993.

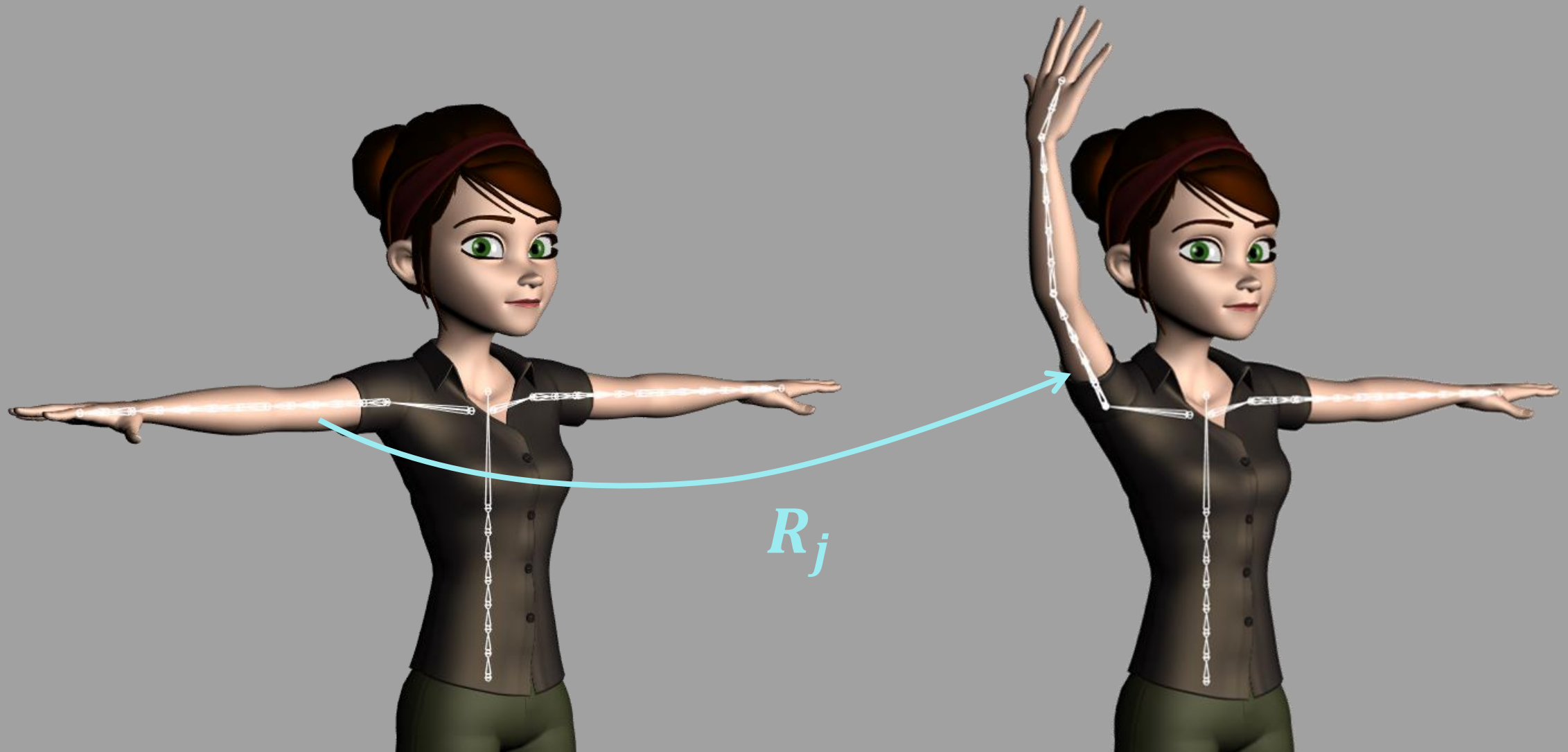
$$\partial\Theta - \begin{bmatrix} \frac{\partial f_1}{\partial\theta_1} & \frac{\partial f_1}{\partial\theta_2} & \cdots & \frac{\partial f_1}{\partial\theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial\theta_1} & \frac{\partial f_M}{\partial\theta_1} & \cdots & \frac{\partial f_M}{\partial\theta_N} \end{bmatrix}$$

- Might have many or no solutions

Now we know how to specify the pose with FK/IK

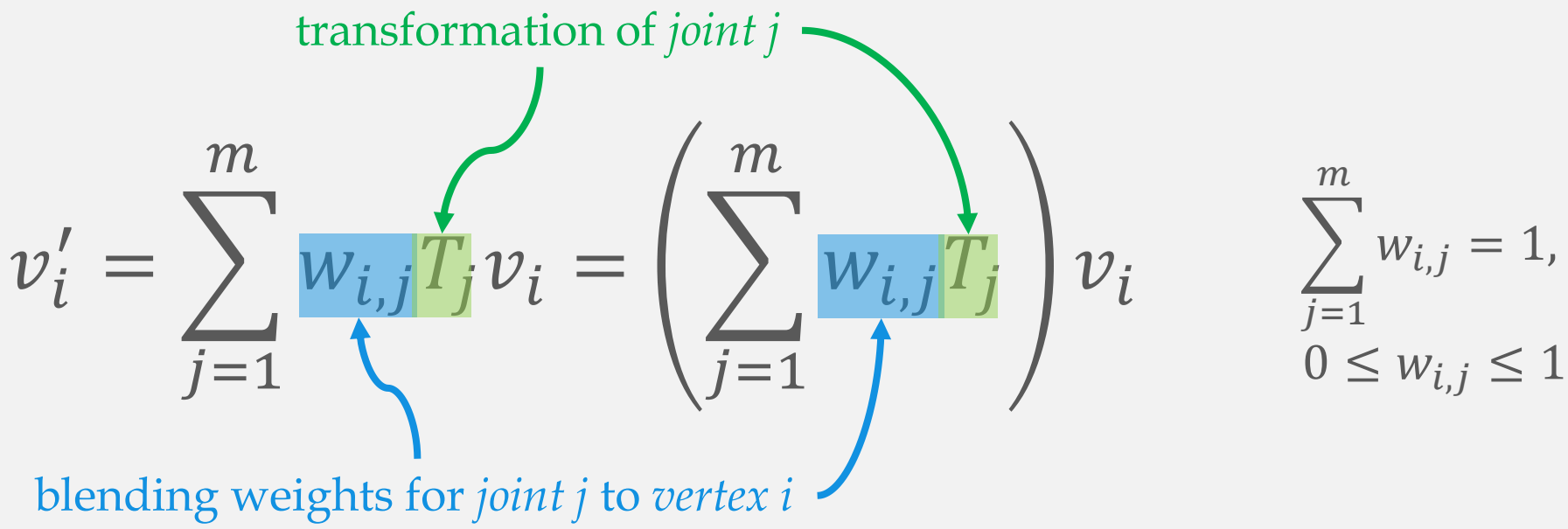
How do we derive the surface from a skeletal pose?

Linear Blend Skinning (LBS)



Linear Blend Skinning (Cont.)

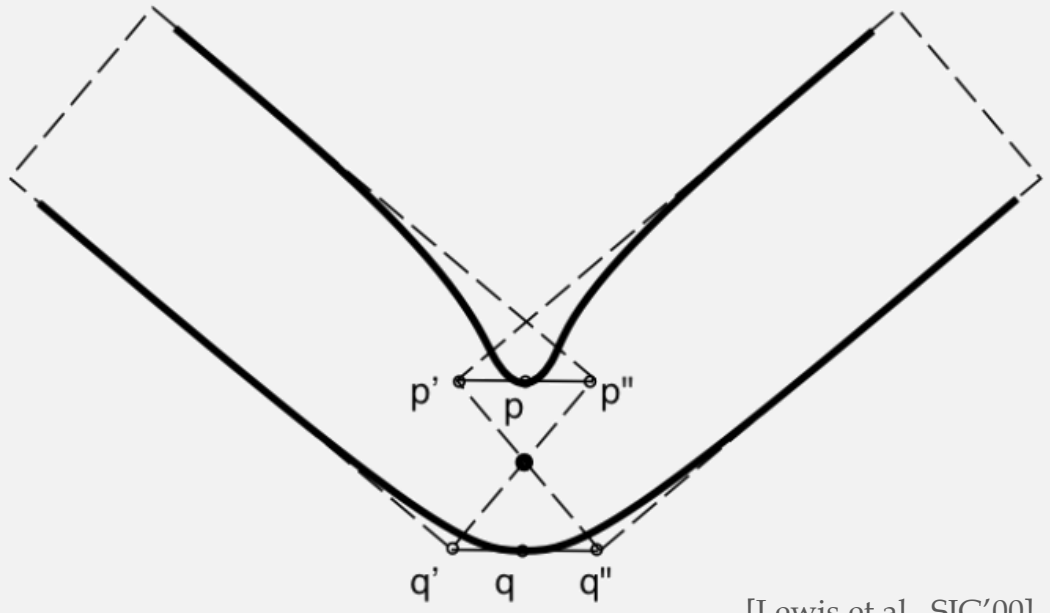
- Rigid binding: each vertex is only affected by *one* joint
- Smooth binding: each vertex is affected by *multiple* joints (usually < 4)



Wait! Direct matrix interpolation?

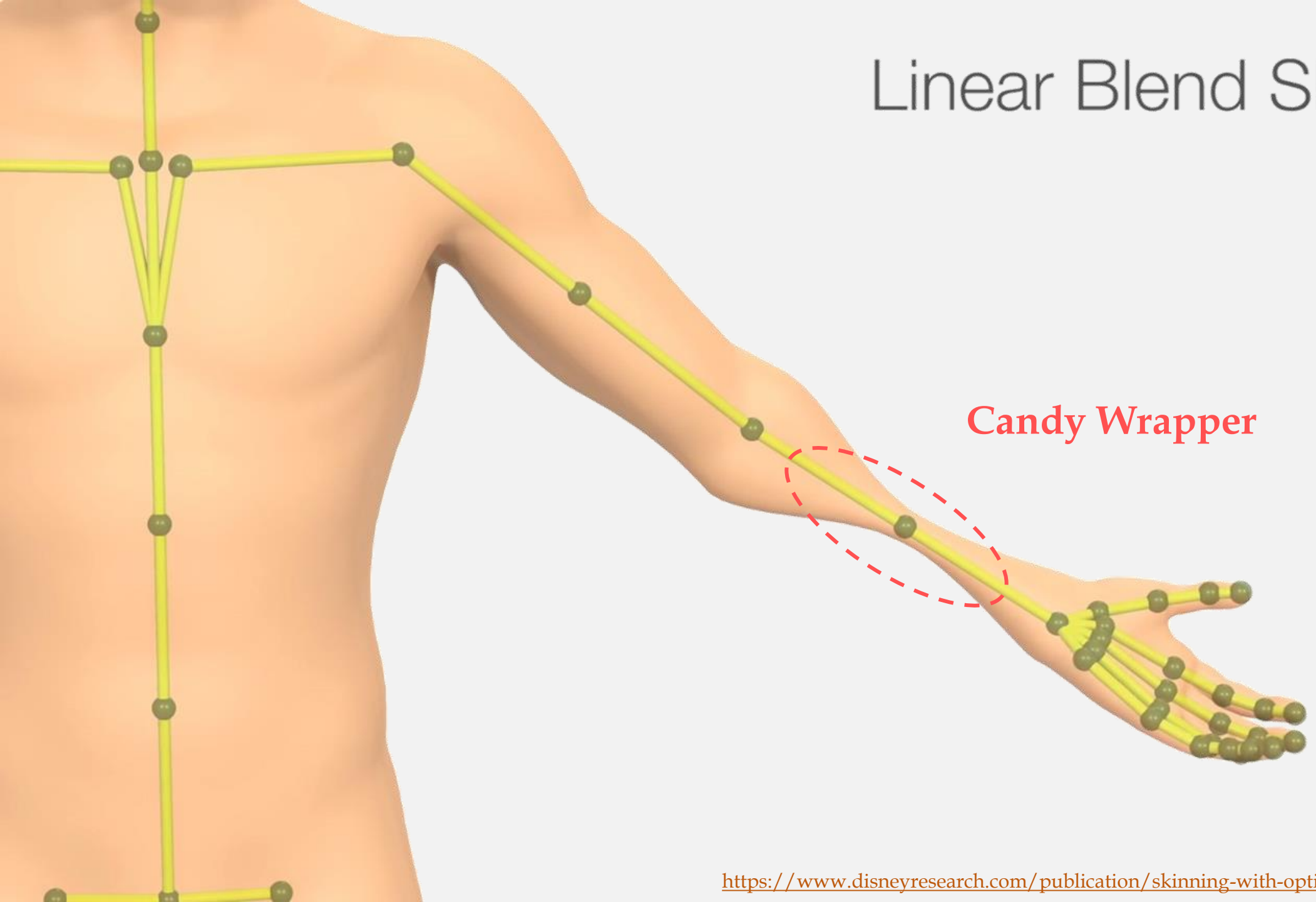
Linear interpolated rotation matrices is **NOT** a rotation matrix!!

$$v_i' = \left(\sum_{j=1}^{m} w_{i,j} T_j \right) v_i$$



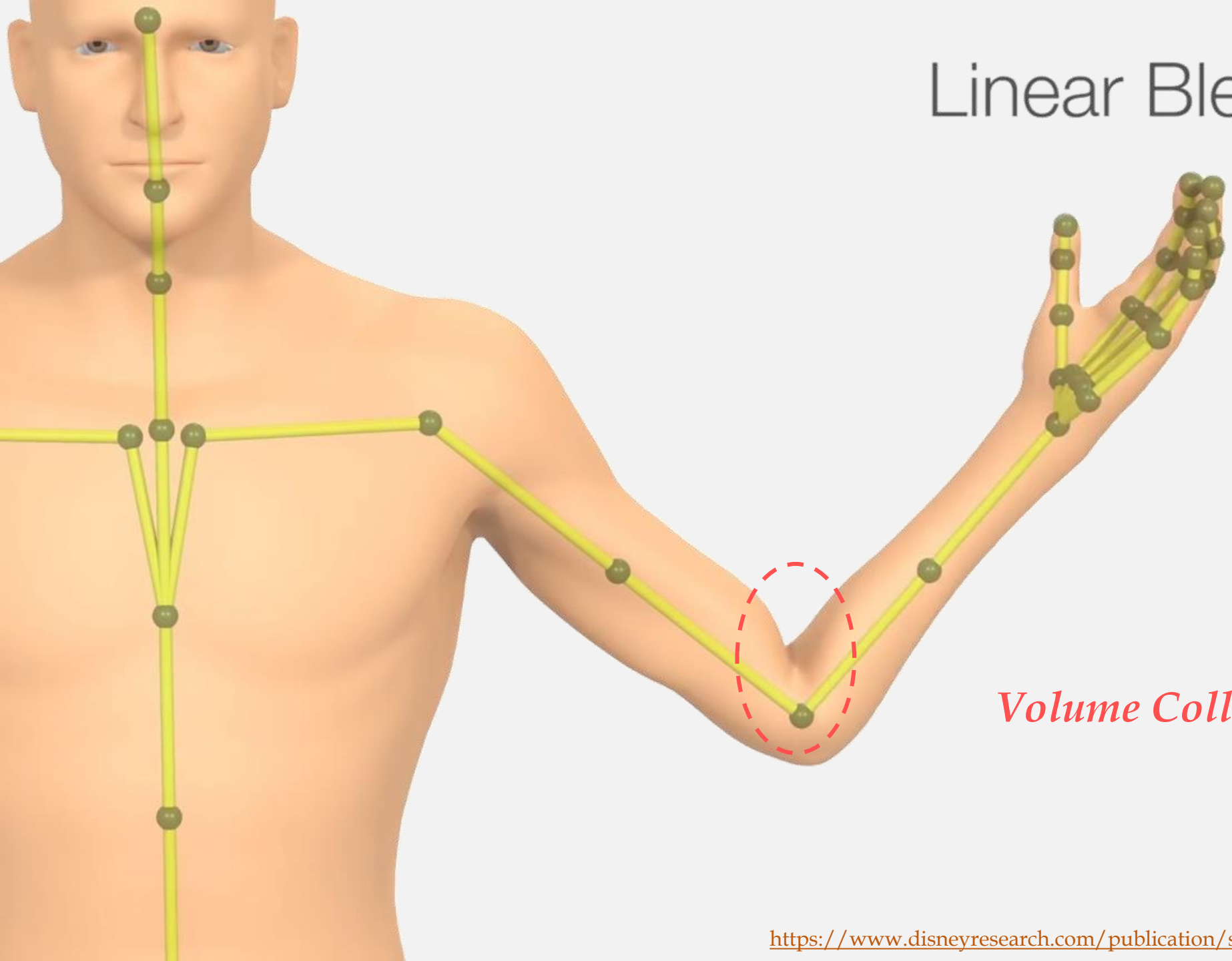
[Lewis et al., SIG'00]

Linear Blend Skinning (LBS)



https://www.disneyresearch.com/publication/skinning-with-optimized-cors/

Linear Blend Skinning (LBS)



Volume Collapse

https://www.disneyresearch.com/publication/skinning-with-optimized-cors/

Heat Weight Transfer

- Weight painting is a laborious task in the past
- Use *heat equation* to spread weights from each bone on the surface

heat

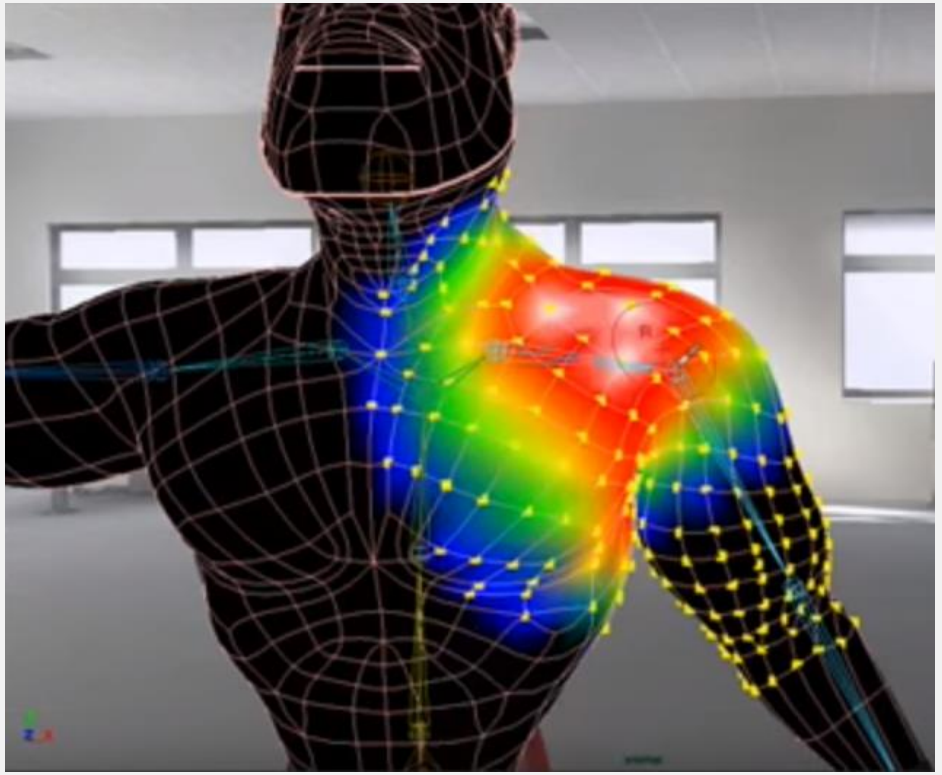
$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u, \qquad \alpha > 0$$

(thermal diffusivity)

$$\frac{\partial w^i}{\partial t} = L_C w^i + H\left(p^i - w^i\right) = 0$$

discrete Laplacian on mesh

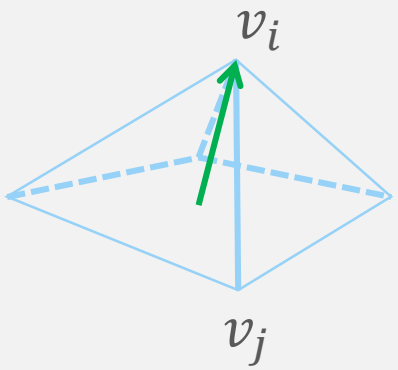
initial weights

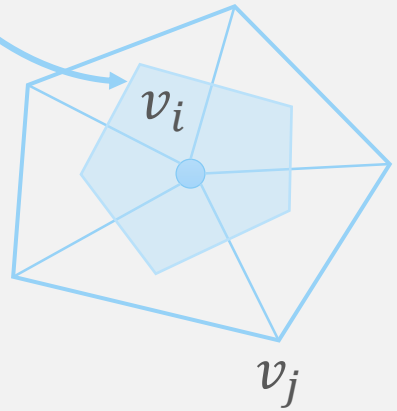


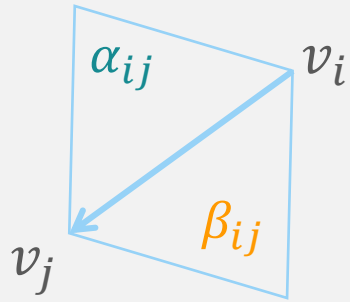
Discrete Laplace-Beltrami

Measures the difference between
the value of the function at that point and
the average of the values at surrounding points

$$L_C(v_i) = \frac{1}{2A(v_i)} \sum_v (\cot \alpha_{ij} + \cot \beta_{ij})(v_j - v_i)$$





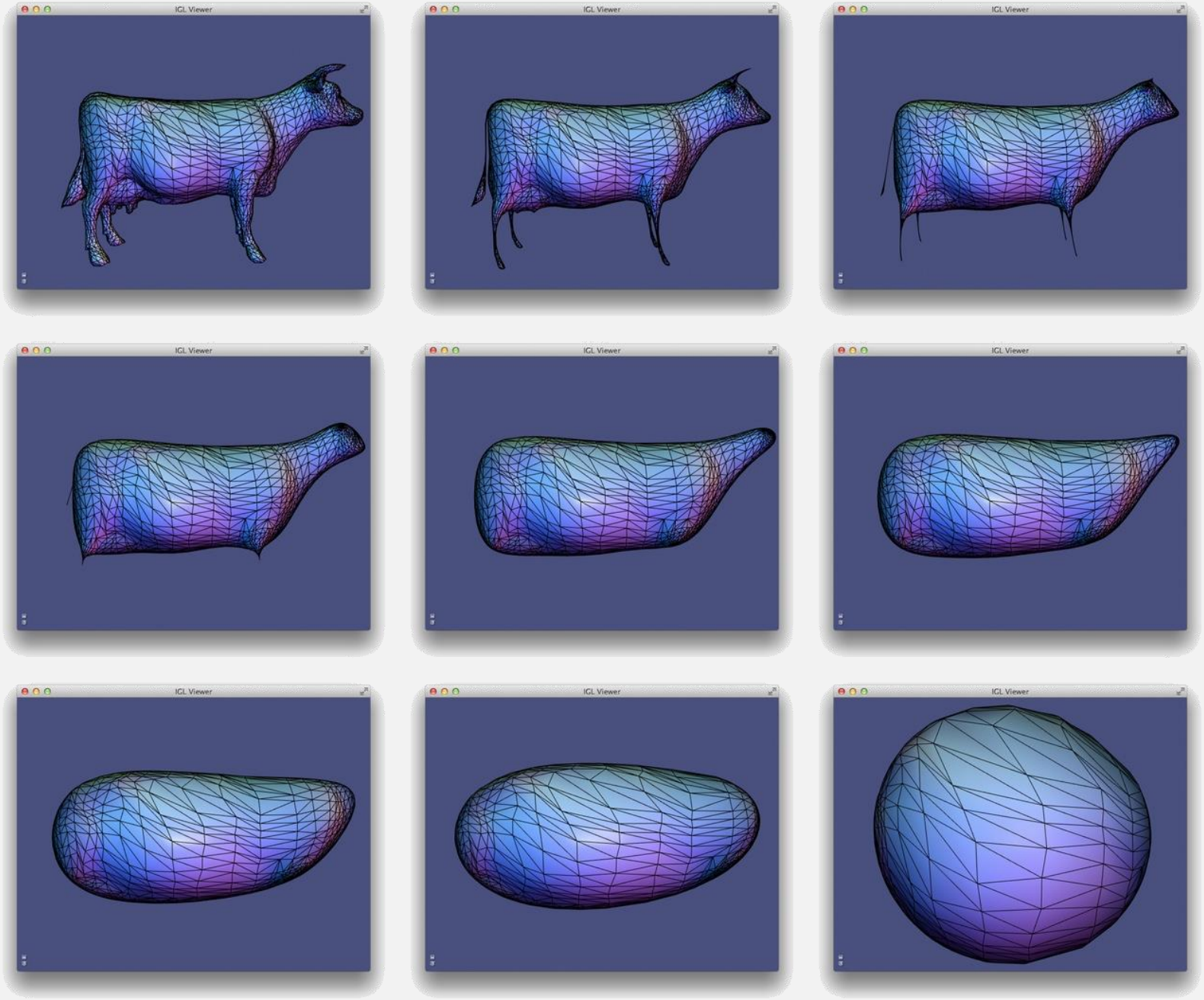


Mesh Smoothing

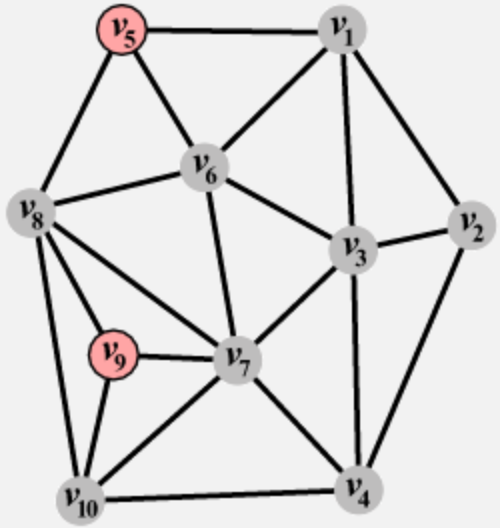
$$V' = L_C(V) + V$$

$$L_C(V) = -H\boldsymbol{n}$$

mean curvature



http://libigl.github.io/libigl/tutorial/tutorial.html



The mesh

$$\begin{bmatrix}
4 & -1 & -1 & & -1 & -1 & & & & \\
-1 & 3 & -1 & -1 & & & & & & \\
-1 & -1 & 5 & -1 & & -1 & -1 & & & \\
 & -1 & -1 & 4 & & & -1 & & & -1 \\
-1 & & & & 3 & -1 & & -1 & & \\
-1 & & -1 & & & 4 & -1 & -1 & & \\
 & & -1 & -1 & & -1 & 6 & -1 & -1 & -1 \\
 & & & & -1 & -1 & -1 & 6 & -1 & -1 \\
 & & & & & & -1 & -1 & 3 & -1 \\
 & & & -1 & & & -1 & -1 & -1 & 4
\end{bmatrix}$$

The symmetric Laplacian L_s

$$\begin{bmatrix}
4 & -1 & -1 & & -1 & & & & \\
-1 & 3 & -1 & -1 & & & & & \\
-1 & -1 & 5 & -1 & & -1 & & & \\
 & -1 & -1 & 4 & & & & & -1 \\
 & & & & & & & & \\
-1 & & -1 & & 4 & -1 & -1 & & \\
 & & -1 & -1 & -1 & 6 & -1 & & \\
 & & & & -1 & -1 & 3 & & -1 \\
 & & & -1 & & -1 & -1 & & 4
\end{bmatrix}$$

Invertible Laplacian

$$\begin{bmatrix}
4 & -1 & -1 & & -1 & & & & \\
-1 & 3 & -1 & -1 & & & & & \\
-1 & -1 & 5 & -1 & & -1 & -1 & & \\
 & -1 & -1 & 4 & & & -1 & & -1 \\
-1 & & & & 3 & -1 & & -1 & \\
-1 & & -1 & & & 4 & -1 & -1 & \\
 & & -1 & -1 & & -1 & 6 & -1 & -1 \\
 & & & & -1 & -1 & -1 & 6 & -1 \\
 & & & & & & -1 & -1 & 3 \\
 & & & -1 & & & -1 & -1 & -1 \\
 & & & & & 1 & & & \\
 & & & & & & & & 1
\end{bmatrix}$$

2-anchor $\tilde{L}$

[Sorkine et al., EG'05]

Now, we have the surface driven by skeletons.

Are there any other ways to manipulate the surface?

How do we enrich shape details at different pose?

Deformation

$$shape = f(space)$$

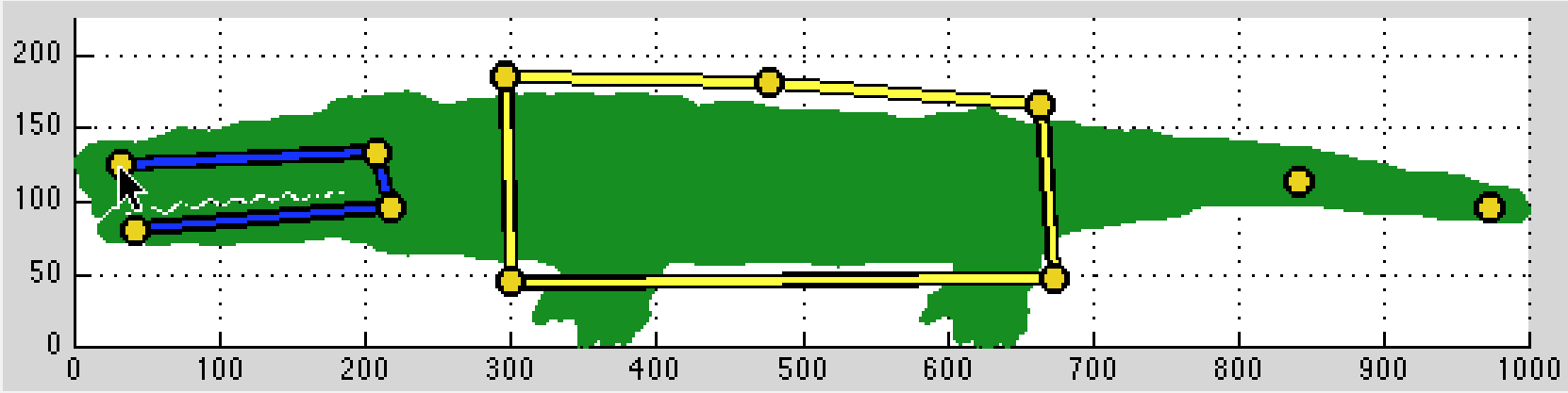
space/volume/free-form deformer

$$shape = f(shape)$$

surface deformer

Deformer

- Change the position of vertices
  - Vertices in, vertices out
  - Topology is unchanged
- Users manipulate the shape via handles such as
  - curve
  - cage
  - proxy mesh
  - etc.



[Jacobson et al., SIG'11]

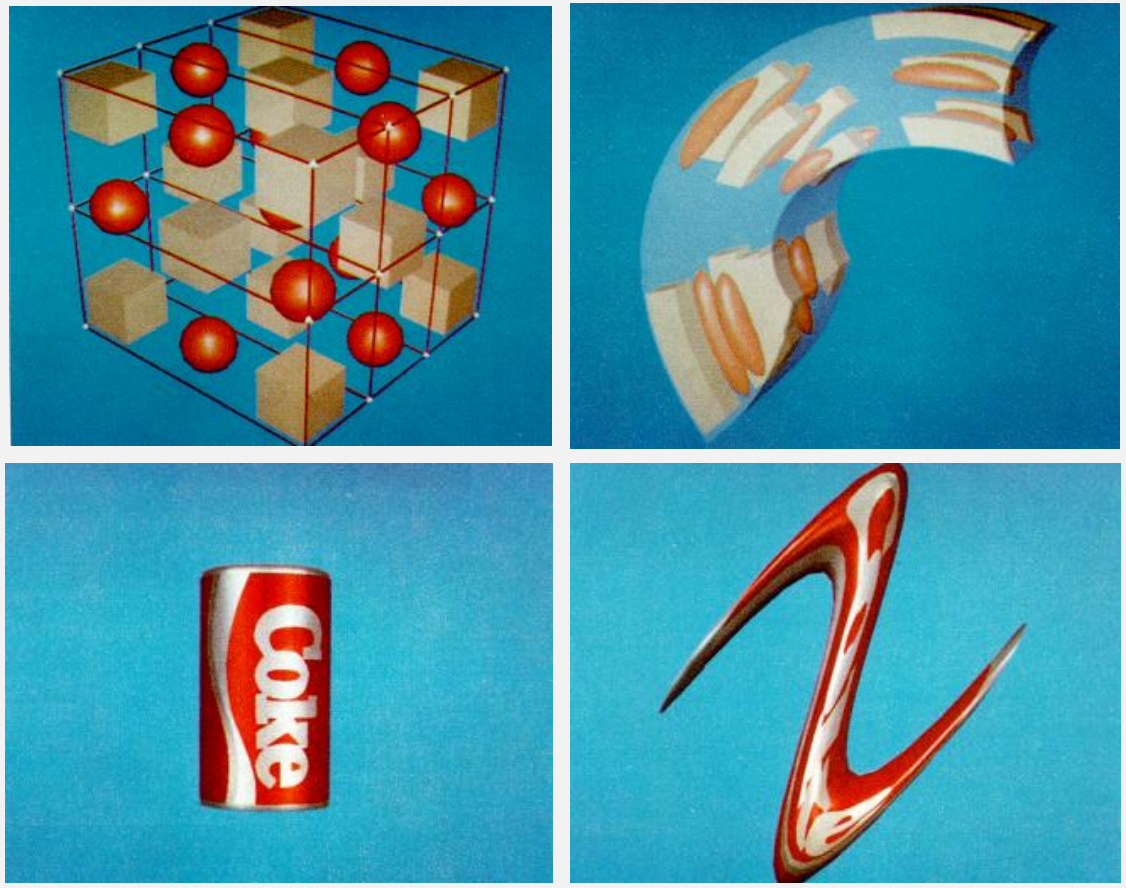
Why Deformer?

- Manipulate mesh for aesthetic purposes
  - Squash, stretch, collision, etc.
- Character posing for animation
- Fake dynamics
  - Secondary animation by using procedural
- Simulation post-fix?
  - I think it would be great for production

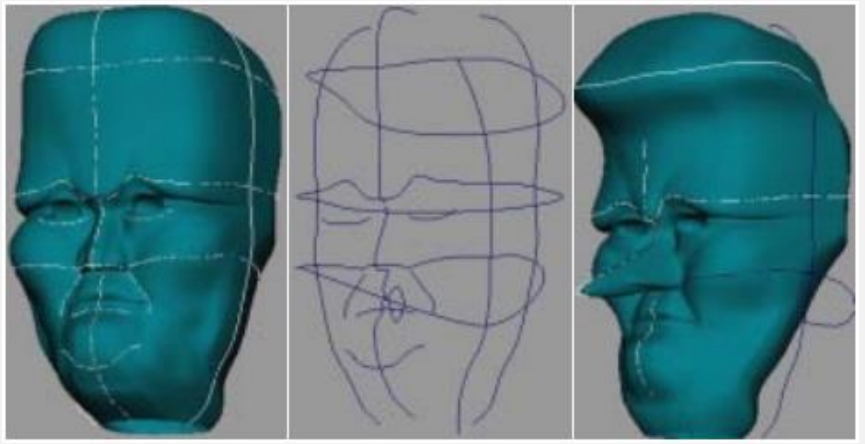
Deformer Requirements

- Sufficiently fast & robust
- Easy to setup and control
- Aesthetically pleasing
  - Physically plausible
  - Preserve local details or volume
- Large scale deformation (optional)

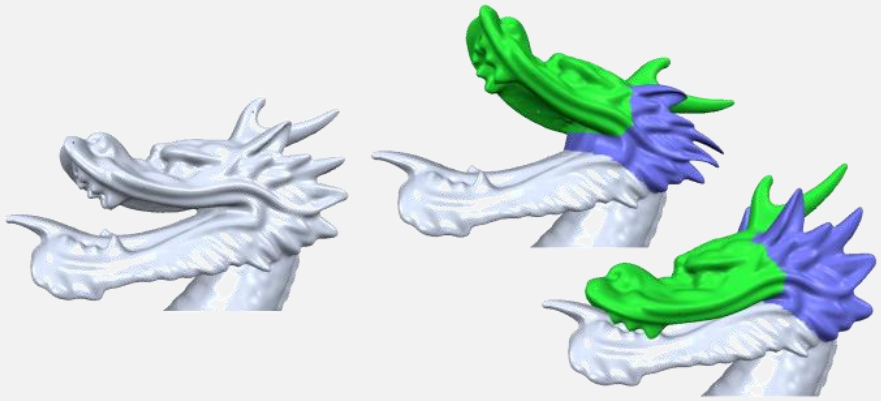
Space Deformation: $shape = f(space)$



[Sederberg and Parry, SIG'86]

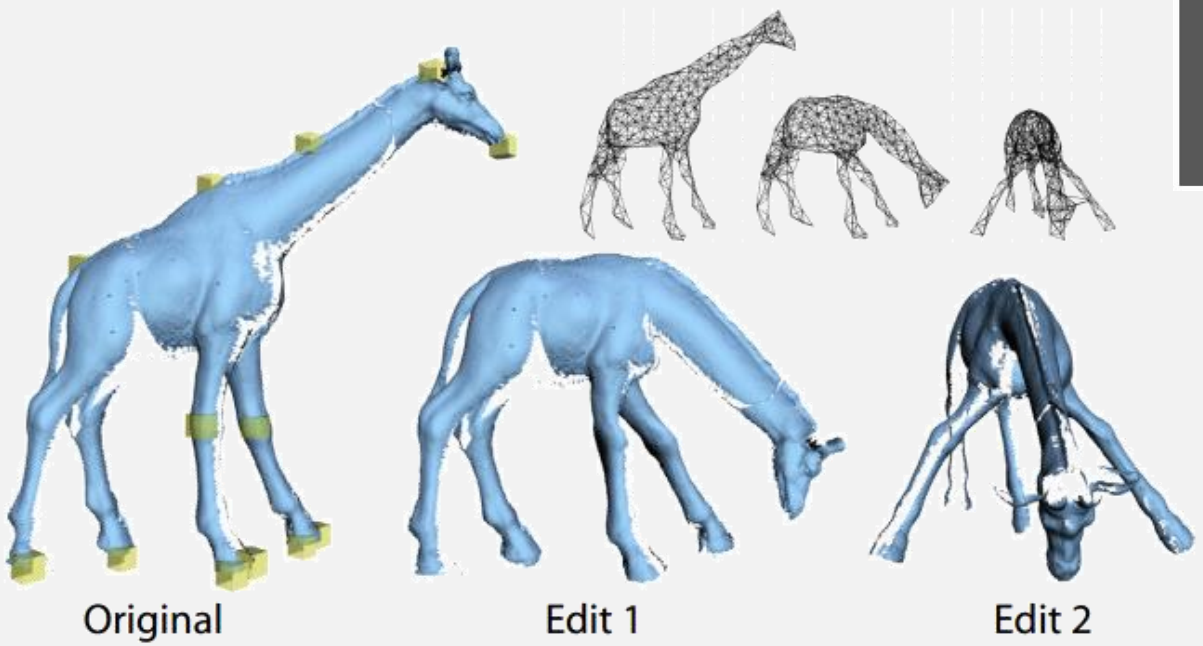


[Singh and Fiume 98]



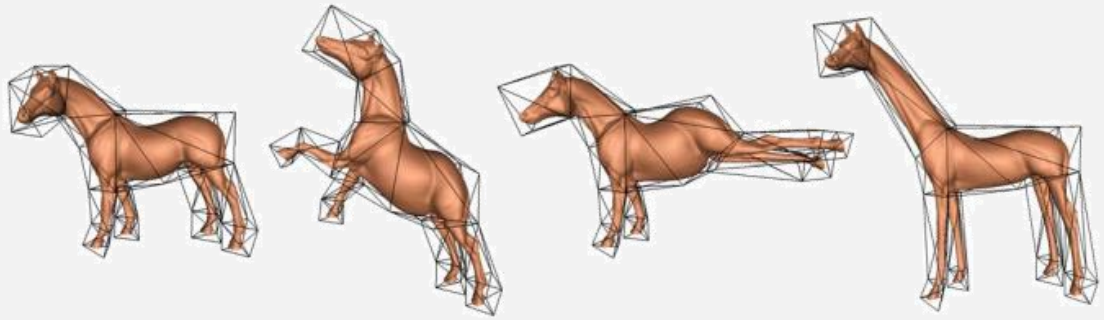
[Botsch and Kobbelt, EG'05]

Space Deformation: $shape = f(space)$



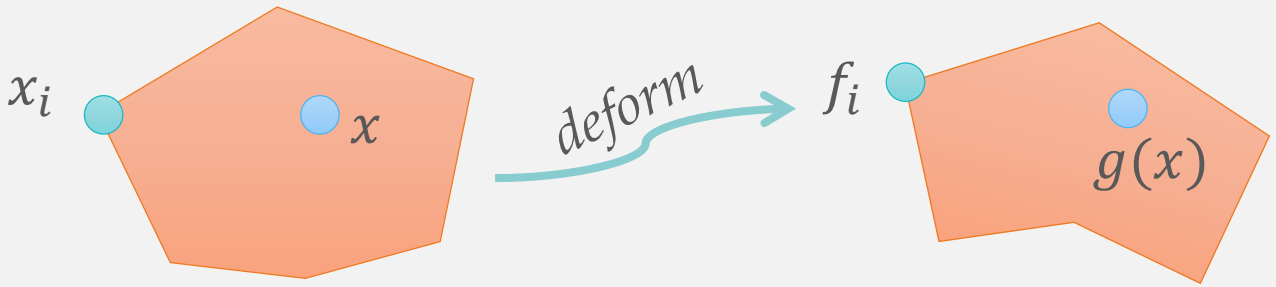
[Summer et al., SIG'07]

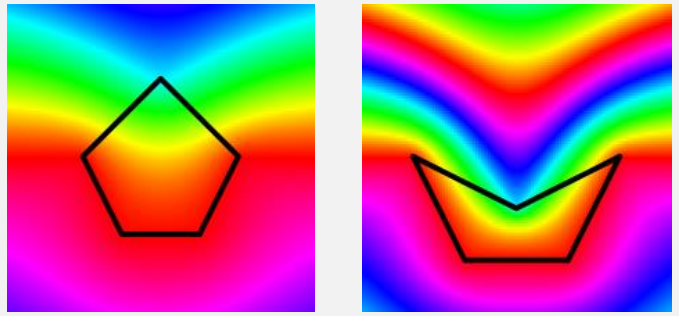
[Joshi et al., SIG'07]



[Ju et al., SIG'05]

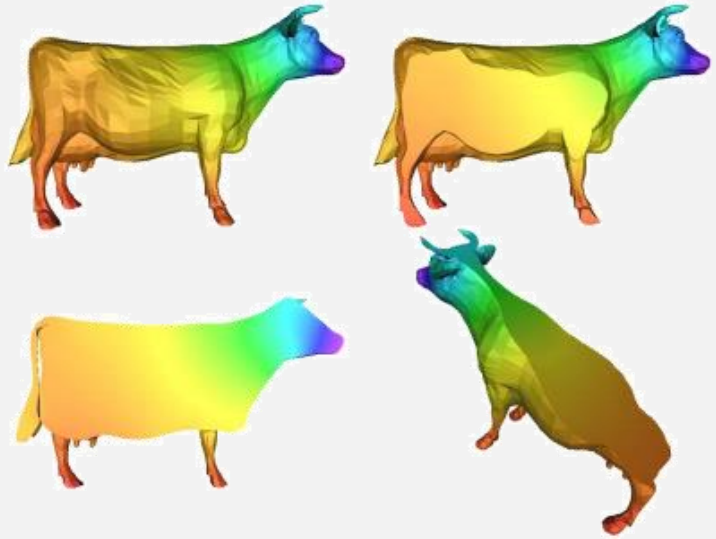
Coordinate Mapping





$$g(x) = \sum_{i=1}^{n} w_i(x) f_i$$

should be smooth!!

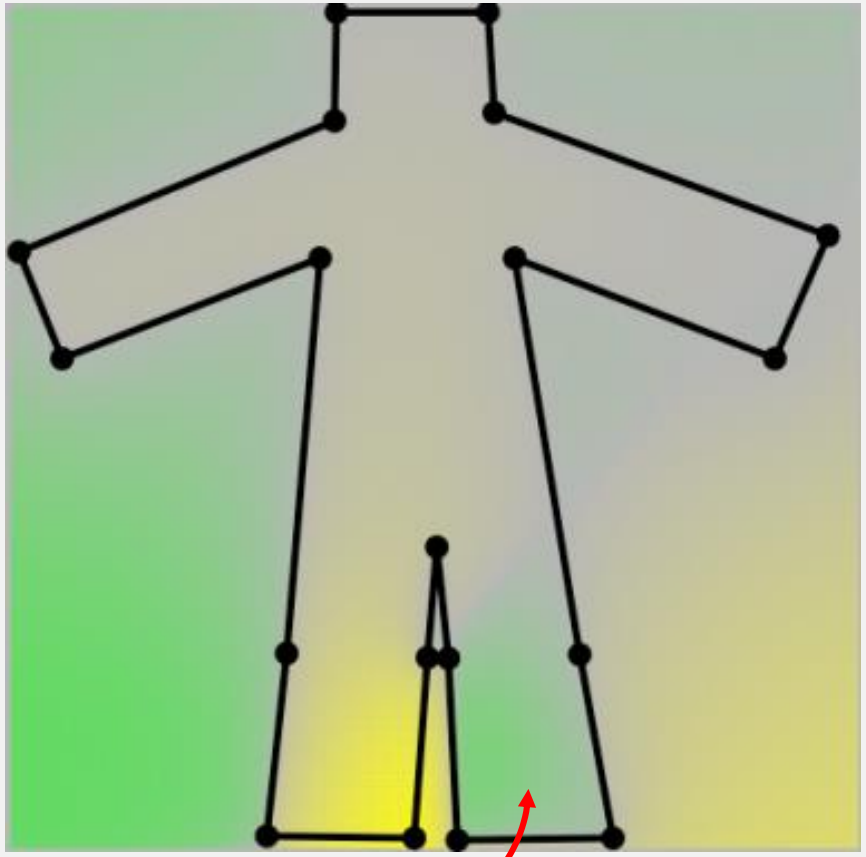


How do we compute the weights inside?
⇒ **Generalized Barycentric Coordinates**

[Ju et al. '05]

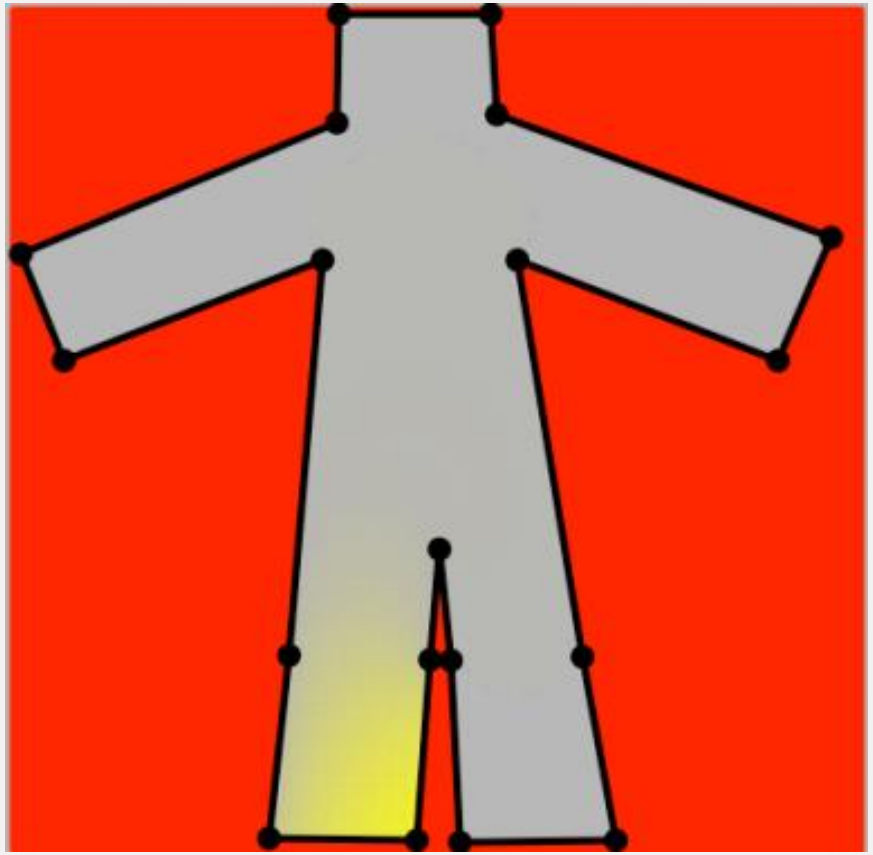
Coordinate Mapping (Cont.)

Mean Value Coordinate



negative weights!!

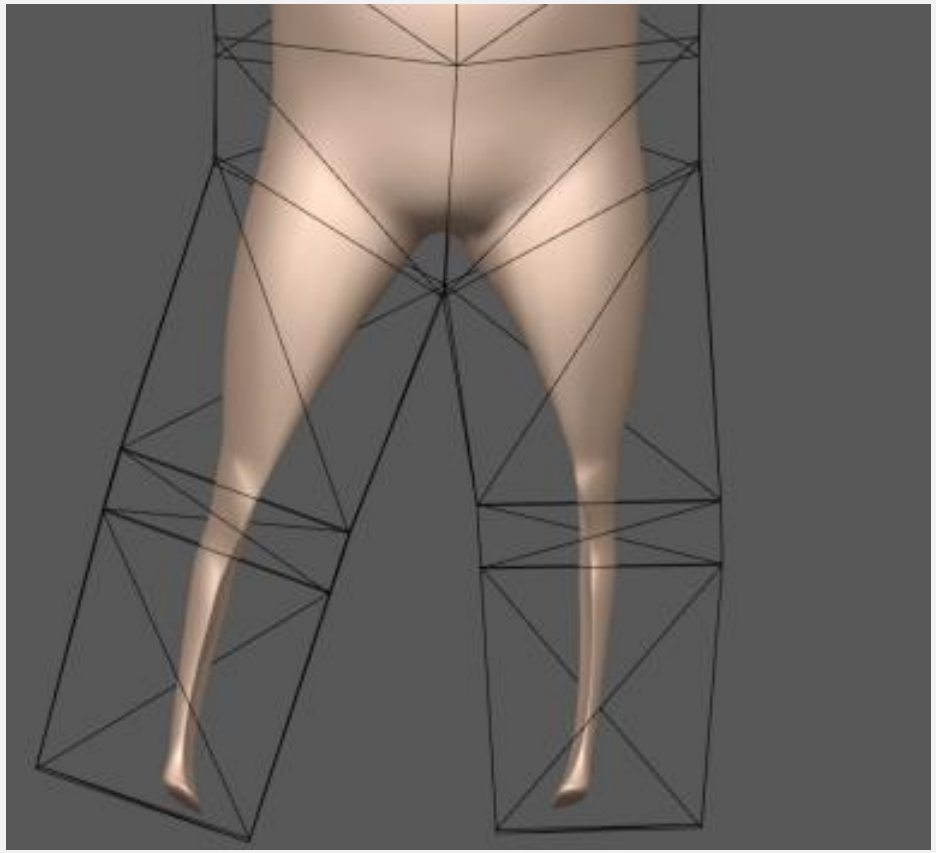
Harmonic Coordinate



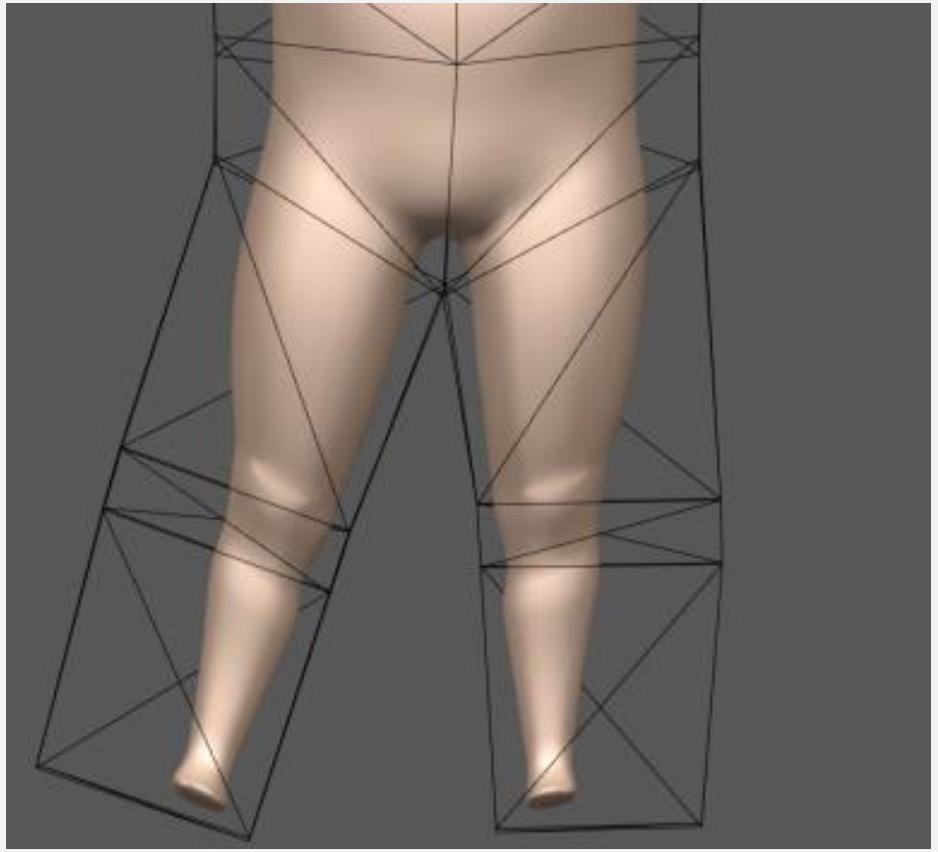
[Joshi et al., SIG'07]

Coordinate Mapping (Cont'd)

Mean Value Coordinate

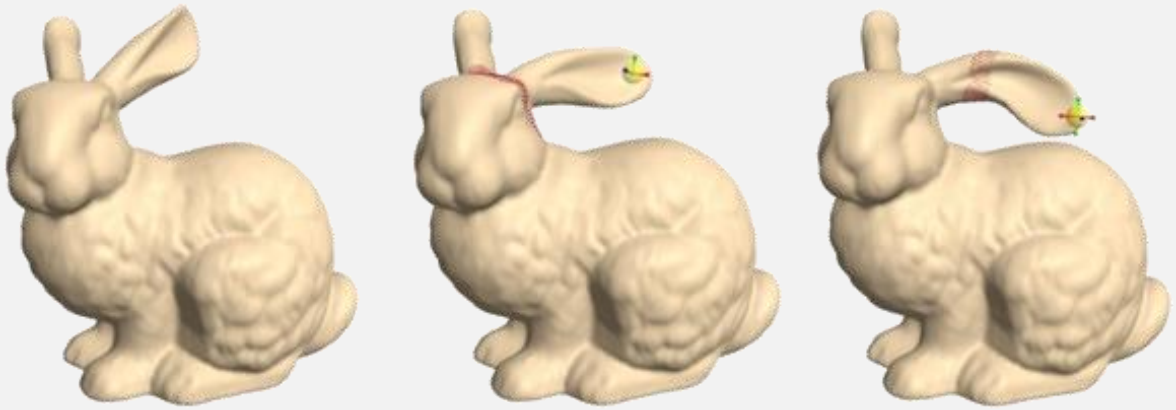


Harmonic Coordinate



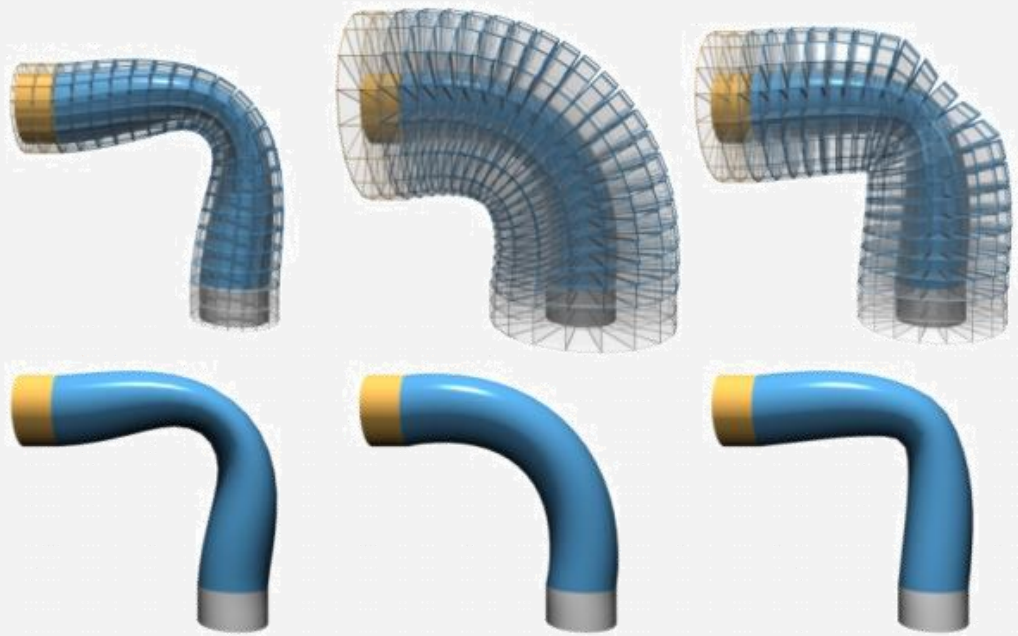
[Joshi et al., SIG'07]

Surface Deformation: $shape = f(shape)$



[Sorkine et al., SGP'04]

[Sorkine and Alexa, SGP'07]



[Botsch et al., SGP'06]

General Framework of Surface Deformation

objective (energy function)

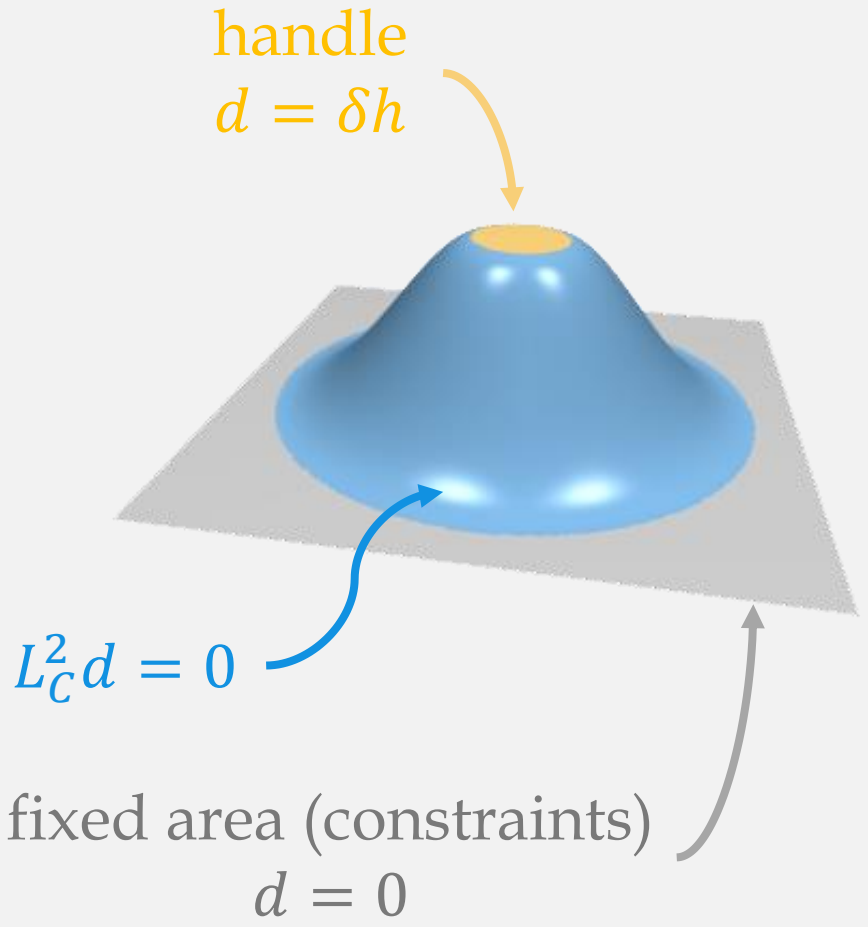
$$x' = \arg\min_{x'} f(x')$$

$$\text{subject to } x_i' = c_i$$

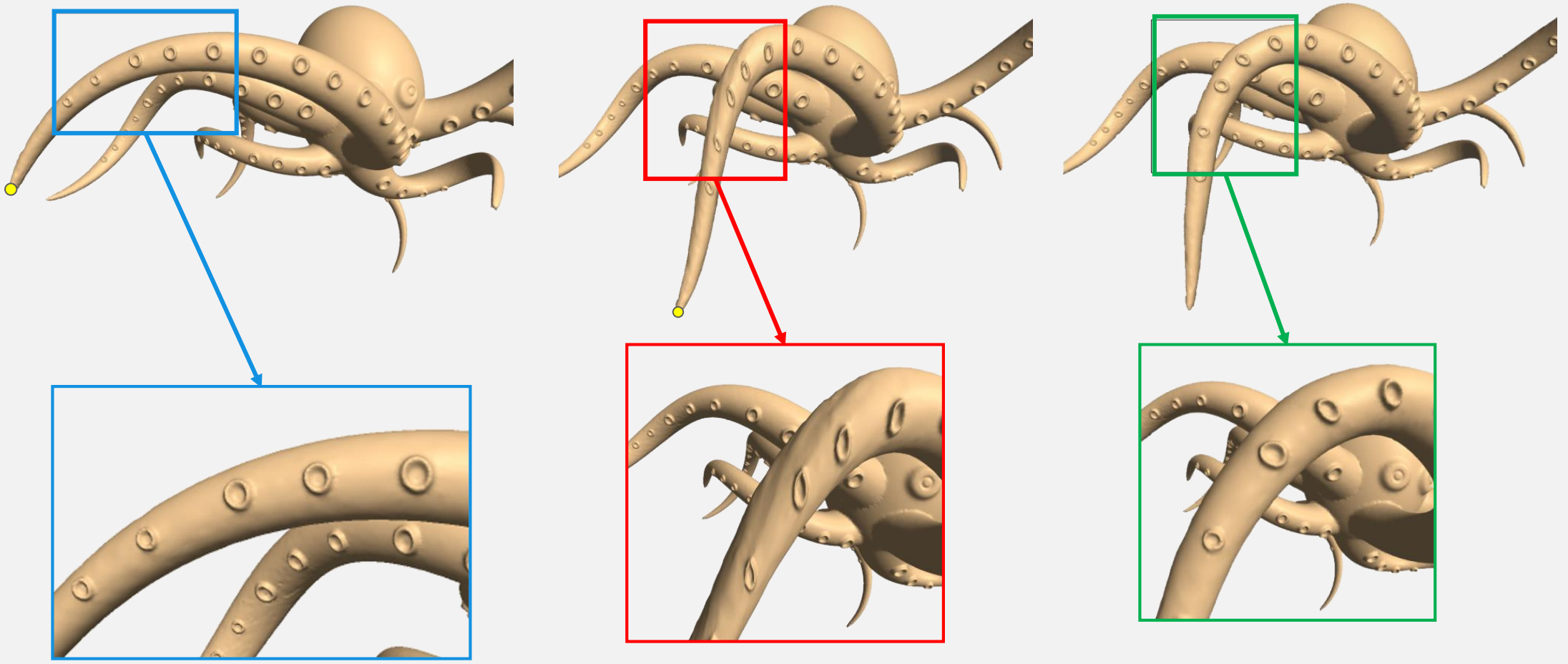
equality constraints

Bi-Harmonic Deformation

$$\begin{bmatrix} & L_c^2 & \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \vdots \\ d_i \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \delta h_i \end{bmatrix}$$



Laplacian Surface Editing



[Sorkine et al., SGP'04]

Laplacian Surface Editing (Cont.)

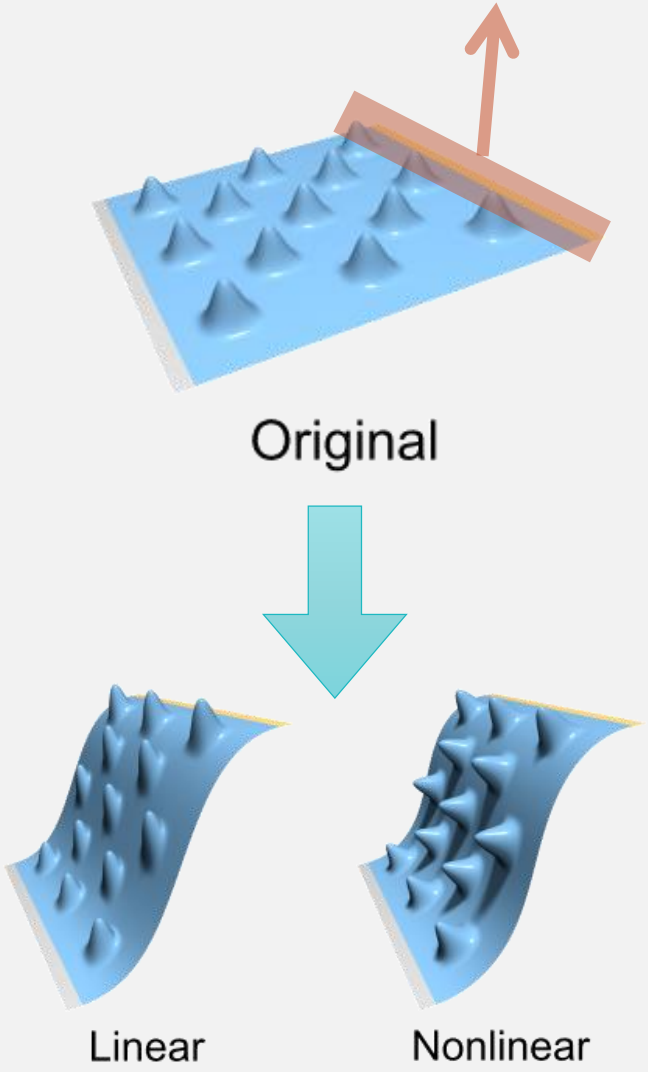
$$v' = arg \min_{v'} \left(\sum_{i=1}^{n} \left\| L_c(v_i') - T_i L_C(v_i) \right\|^2 + \sum_{j \in C} \left\| v_j' - u_j \right\|^2 \right)$$

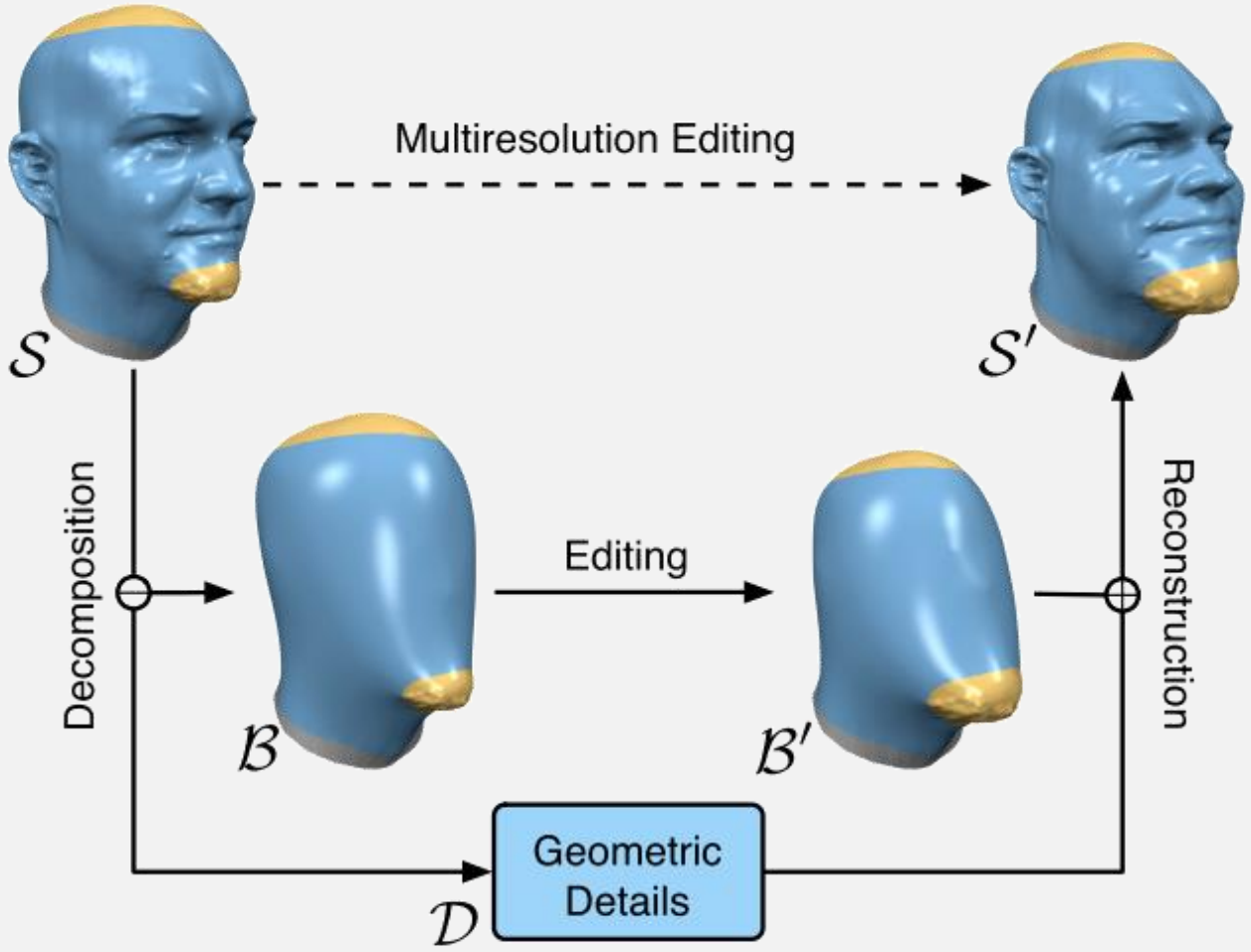
similarity transformation

user constraints

Laplacian coordinate is not rotation invariant, thus we need T_i for alignment (rotation + scale).

Multiresolution Editing



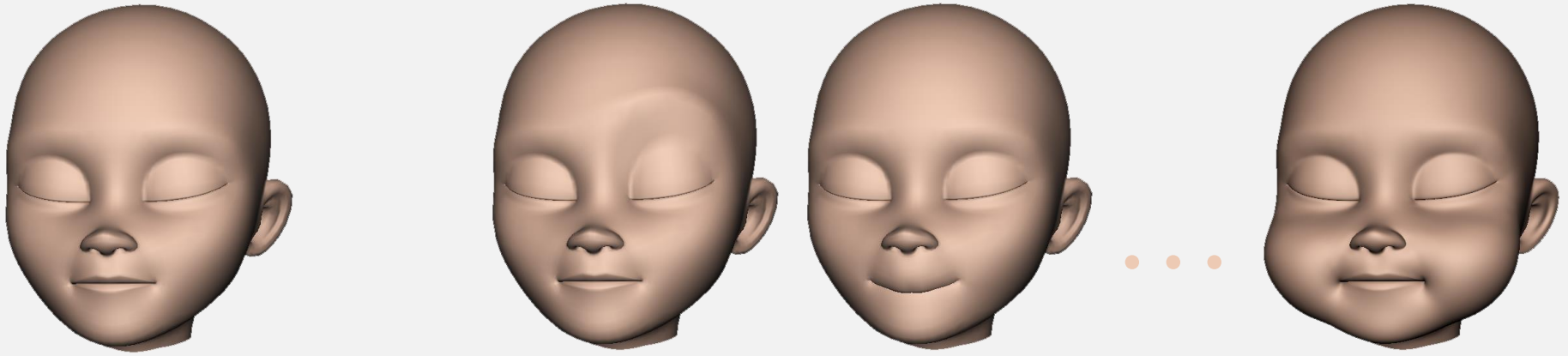


[Botsch and Sorkine. TVCG'08]

Face Animation

- Shape interpolation
  - Blend a set of shapes whose topology is the same
  - aka morphing, morph tgargets, vertex blending, geometry interpolation, etc.
- Final expression could be blended from a neutral expression and several target expression models
- How to tweak the expression via parameters?
  - PCA (Principal Component Analysis)
  - Blend shapes

Blend Shapes



base mesh

morph targets

$$b_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ \vdots \\ x_n \\ y_n \\ z_n \end{bmatrix} \in \mathbb{R}^{3n}$$

$$f = b_0 + \sum_{k=1}^{n} w_k (b_k - b_0)$$

$$\sum_i w_i = 1$$

$$0 \le w_i \le 1$$

$$f = b_0 + Bw$$

Character Rig from Mery Project

Expression Space Decomposition

PCA

- Orthogonal
- Lack the interpretability

Blend Shapes

- Semantic parameterization
- Consistent appearance

x Lack of orthogonality

x Not unique:

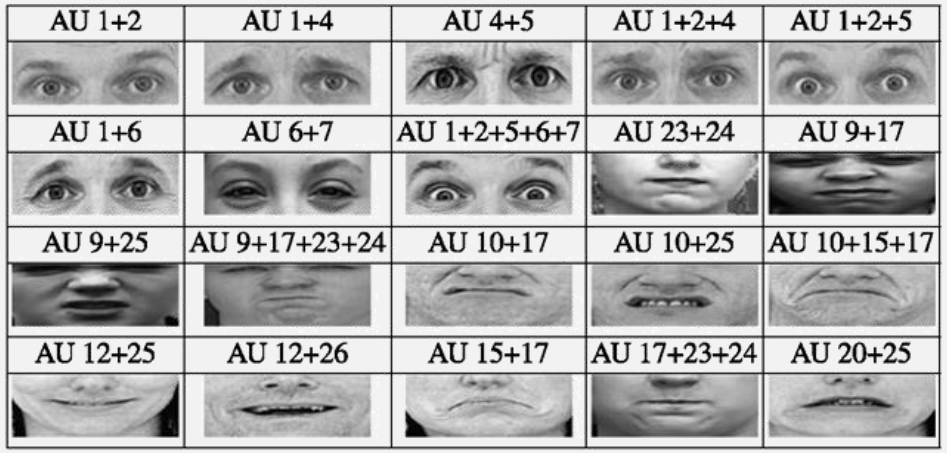
$$f = B(RR^{-1})w$$

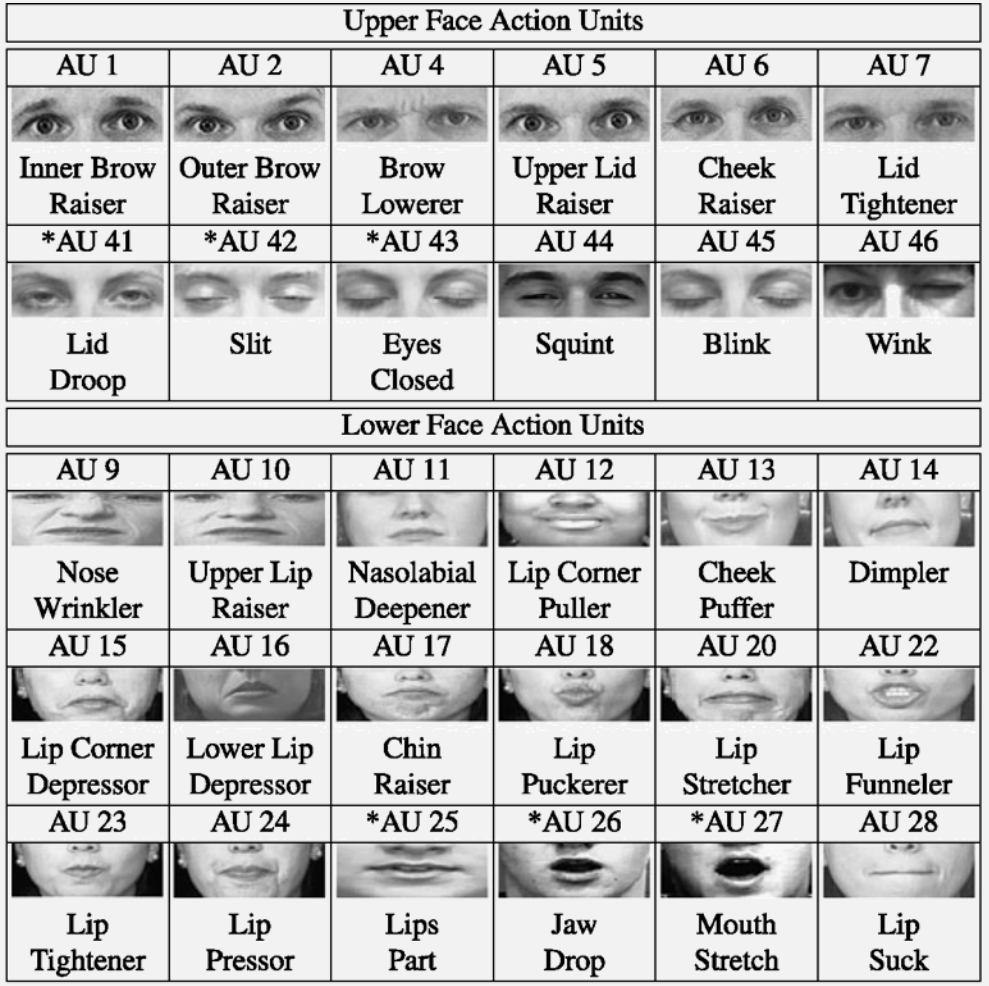
Morph Targets

- The possible facial expression is in the subspace span by all target shapes
- What shapes should we choose as morph targets?

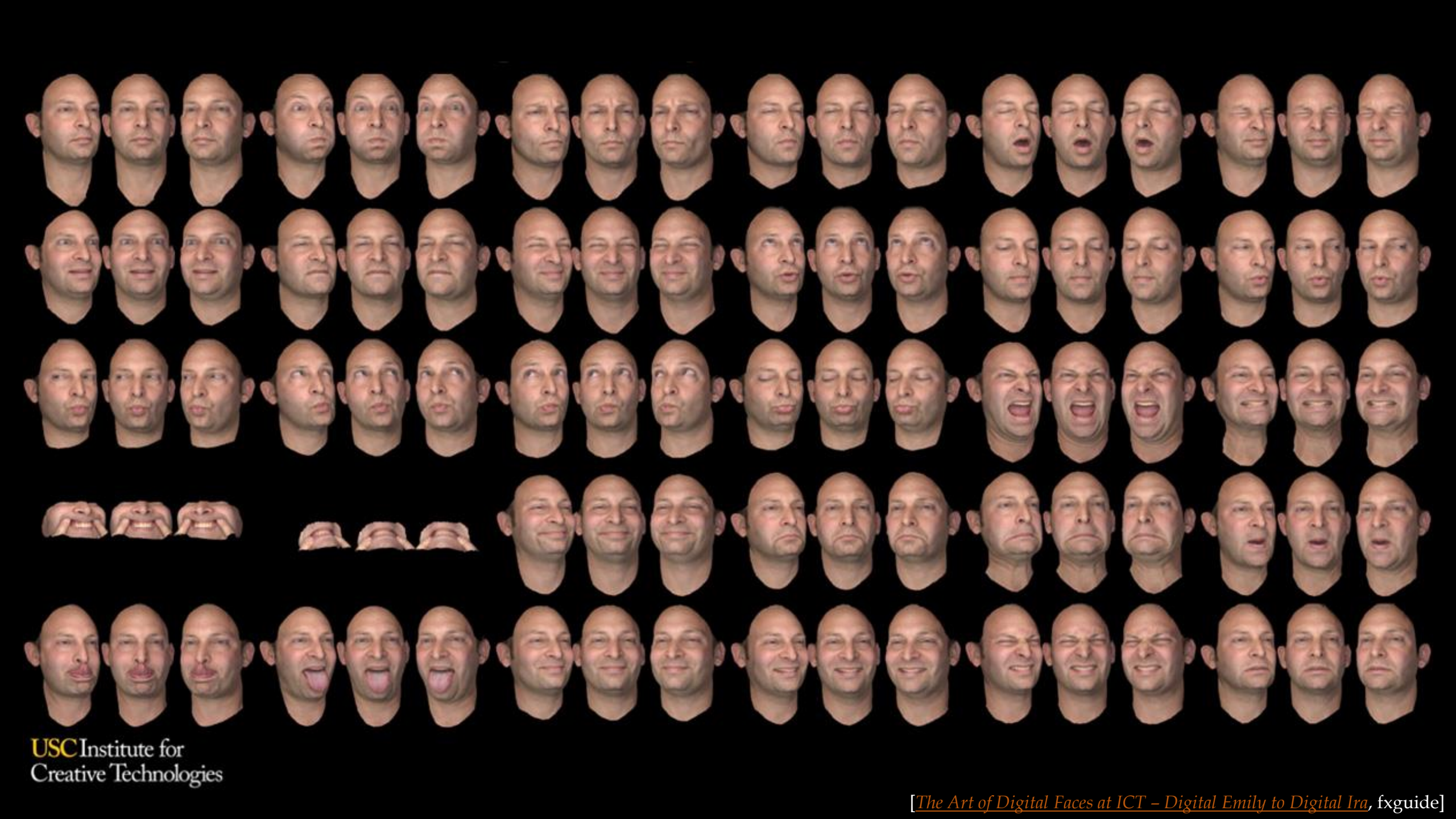
Facial Action Coding System (FACS)

- A system to taxonomize human facial movements by their appearance on the face
- Describes a set of action units that correspond to basic actions
- Expressions are built from basic action units





http://what-when-how.com/face-recognition/facial-expression-recognition-face-recognition-techniques-part-1/



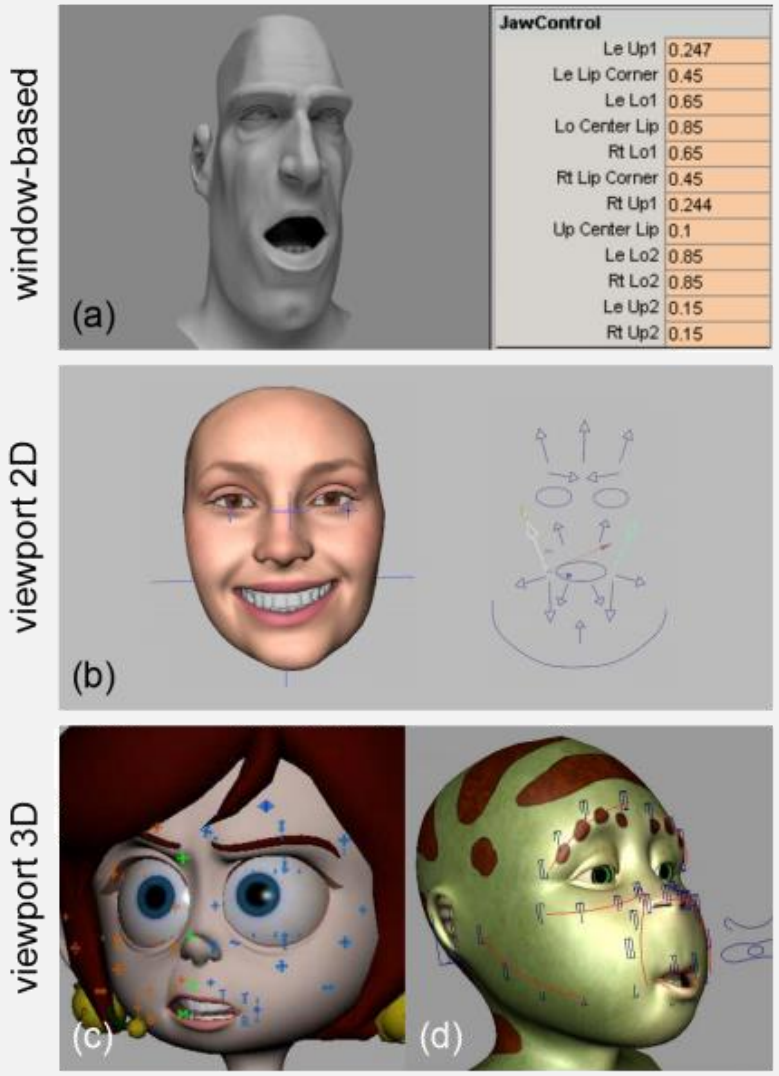
[*The Art of Digital Faces at ICT – Digital Emily to Digital Ira*, fxguide]

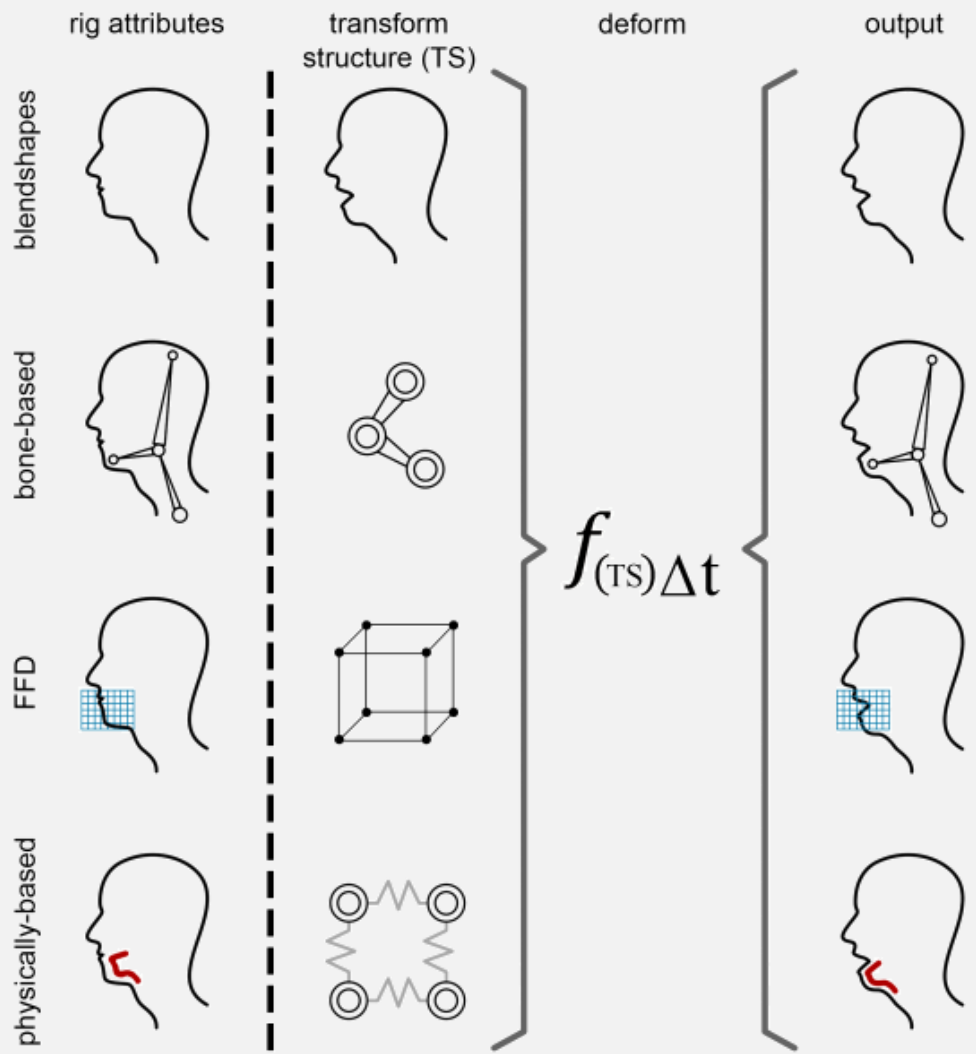
Practical Issues

- How to compress morph targets?
  - Most target expressions would only modify a small range of vertices w.r.t. neutral expression
- Expression transfer between multiple characters
  - Use deformation transfer for morph targets

[Sumner and Popović, SIG'04]

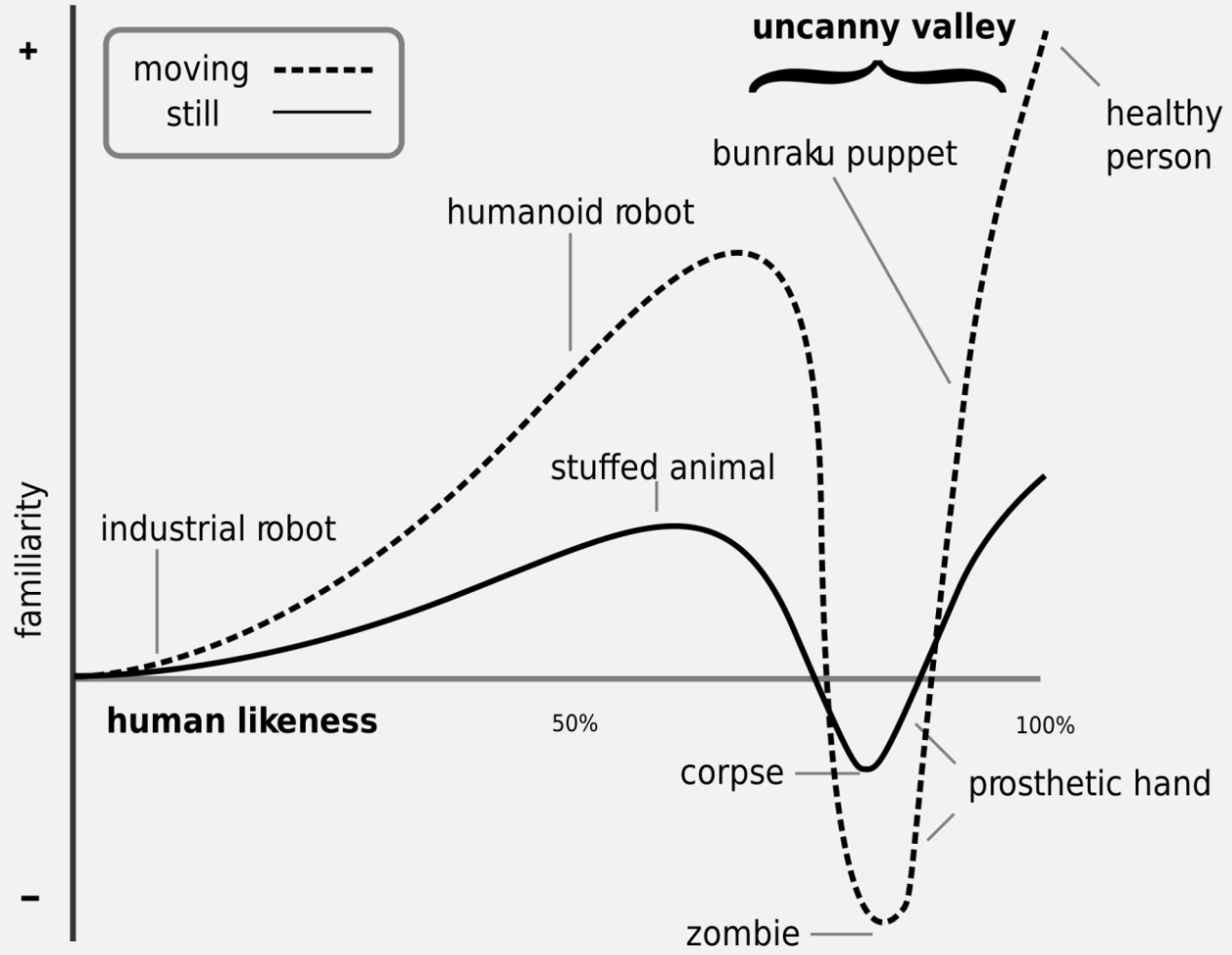
Design of Facial Rigging



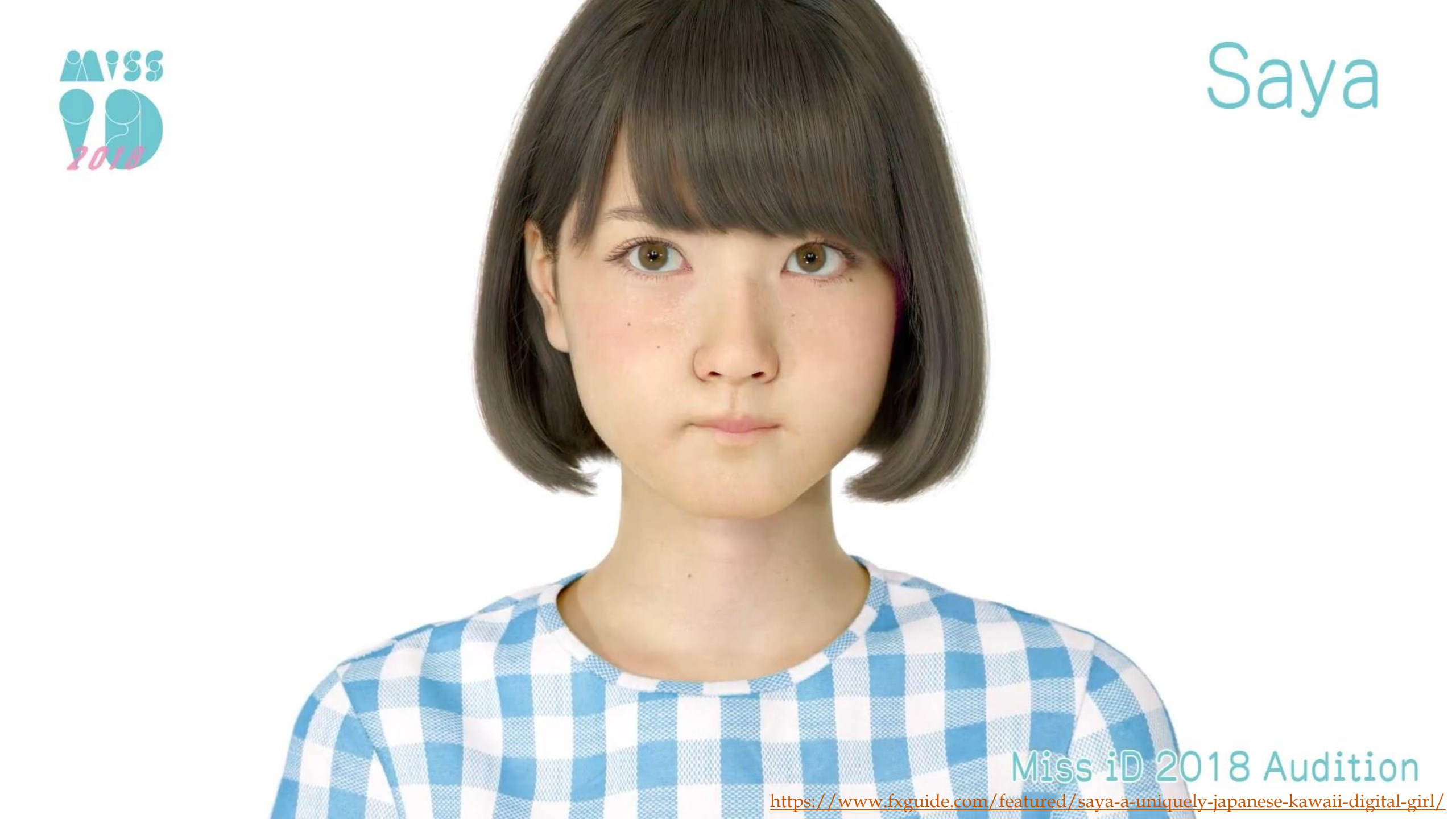


[Orvalho et al., EG'12]

Uncanny Valley



https://en.wikipedia.org/wiki/File:Mori_Uncanny_Valley.svg



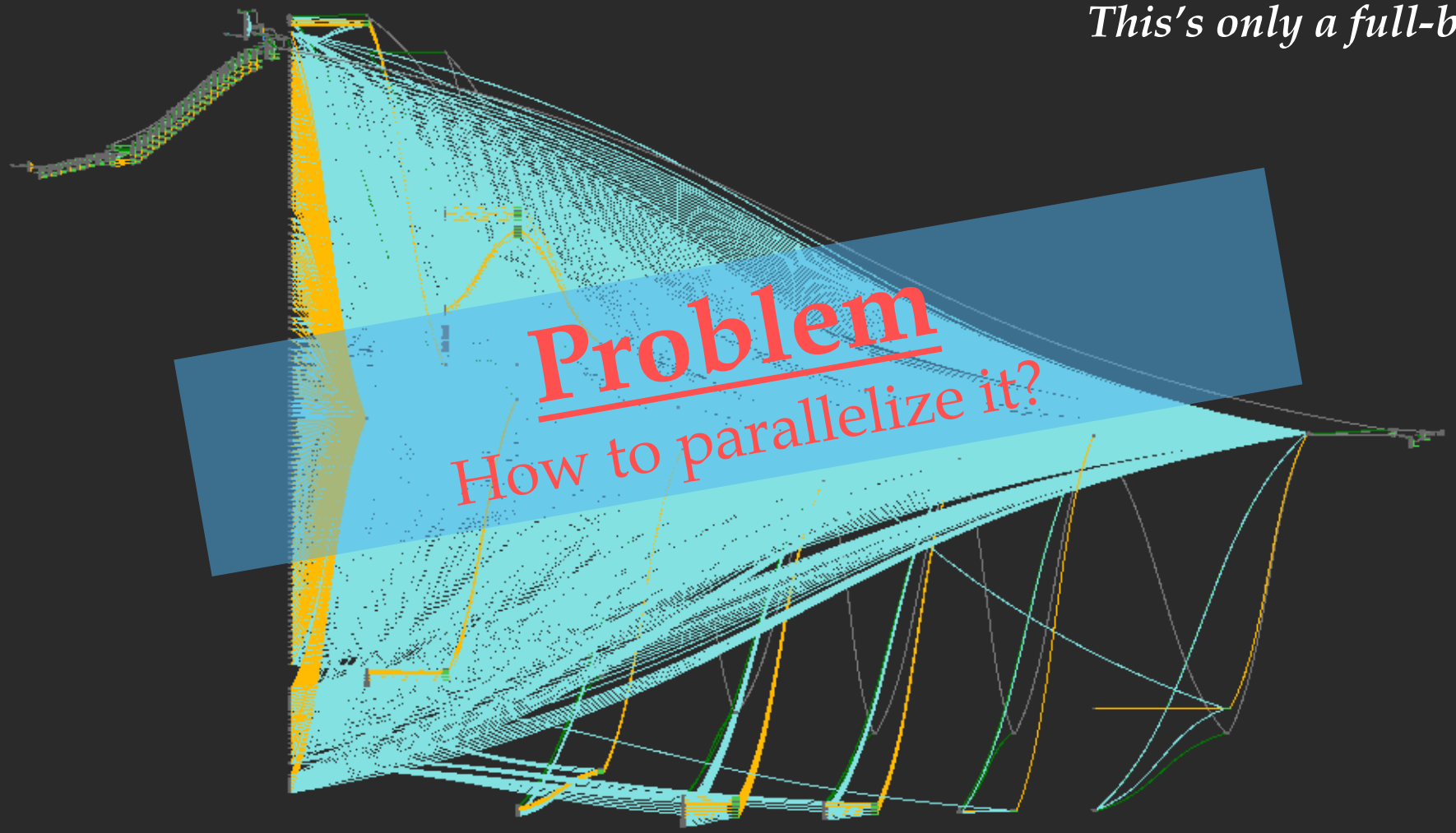
https://www.fxguide.com/featured/saya-a-uniquely-japanese-kawaii-digital-girl/

Practical Issues

- How to provide intuitive controls?
  - Too many => hard to manipulate
  - Not enough => can't get enough animation details
- In node-based framework, computation = graph evaluation
  - How do we separate the evaluation graph for parallelism?

Parallel Graph Evaluation

This's only a full-body character!



Motion Capture

Performance Capture

- Motion capture: the process of recording the movements of objects or people
- Performance capture: the process of recording the movements of an actor and then using the data to create a digital animation
  - Body, finger, facial captures

Andy Serkis Breaks Down His Motion Capture Performances

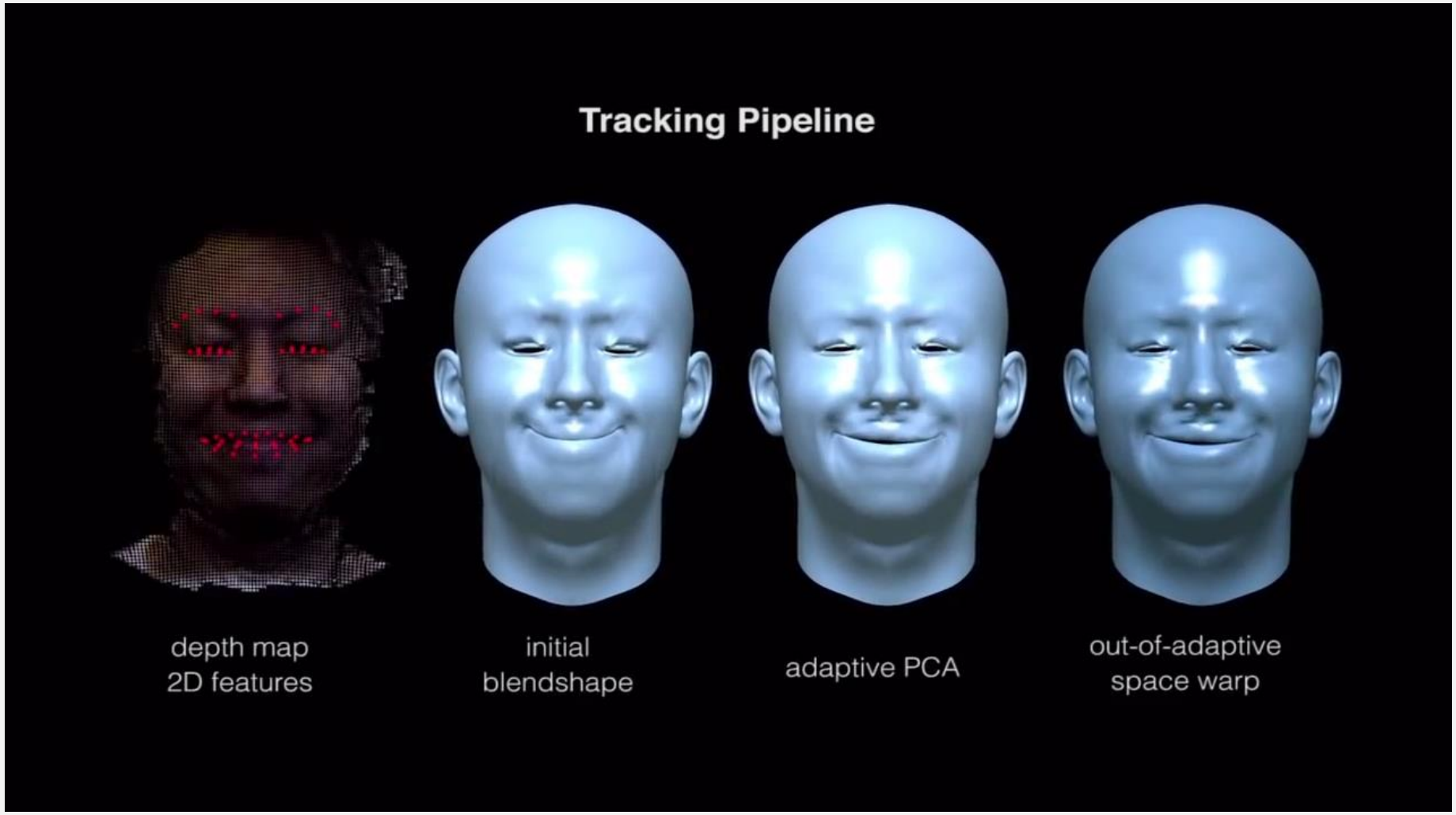
https://www.youtube.com/watch?v=DpRLTfVEhMk

Full Body Motion Capture

Facial Animation with Performance Capture

- Find facial landmarks of the actor
  - With predefined markers
  - Markless becomes mainstream, extract the landmarks from image sequence now
- Given a facial rig with blendshapes, solve the blending weights for each shape with facial landmarks as constraints

Realtime Facial Animation with On-the-fly Correctives




HELLBLADE™
SENUA'S SACRIFICE

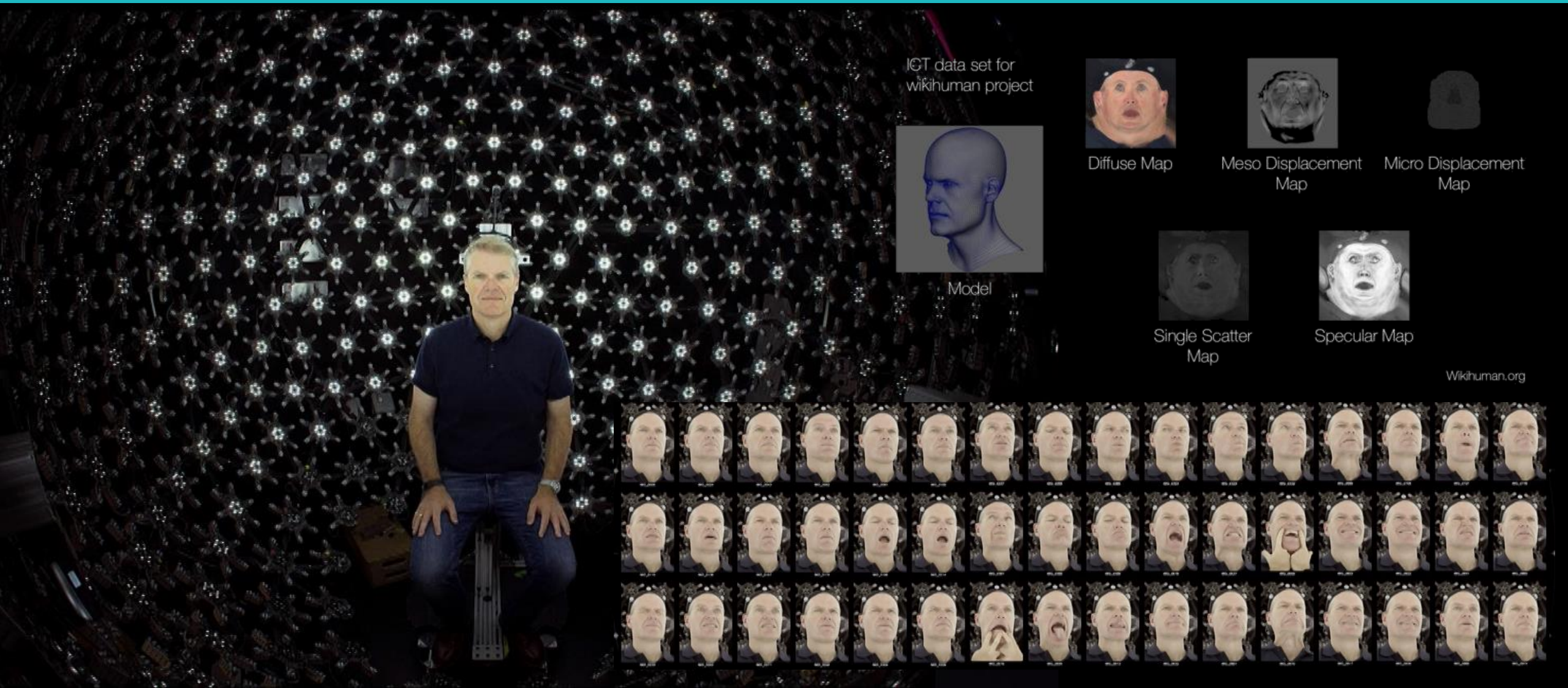
https://www.fxguide.com/featured/real-time-mike/

https://www.youtube.com/watch?v=otmxoK4lCNw&feature=youtu.be&t=16m6s

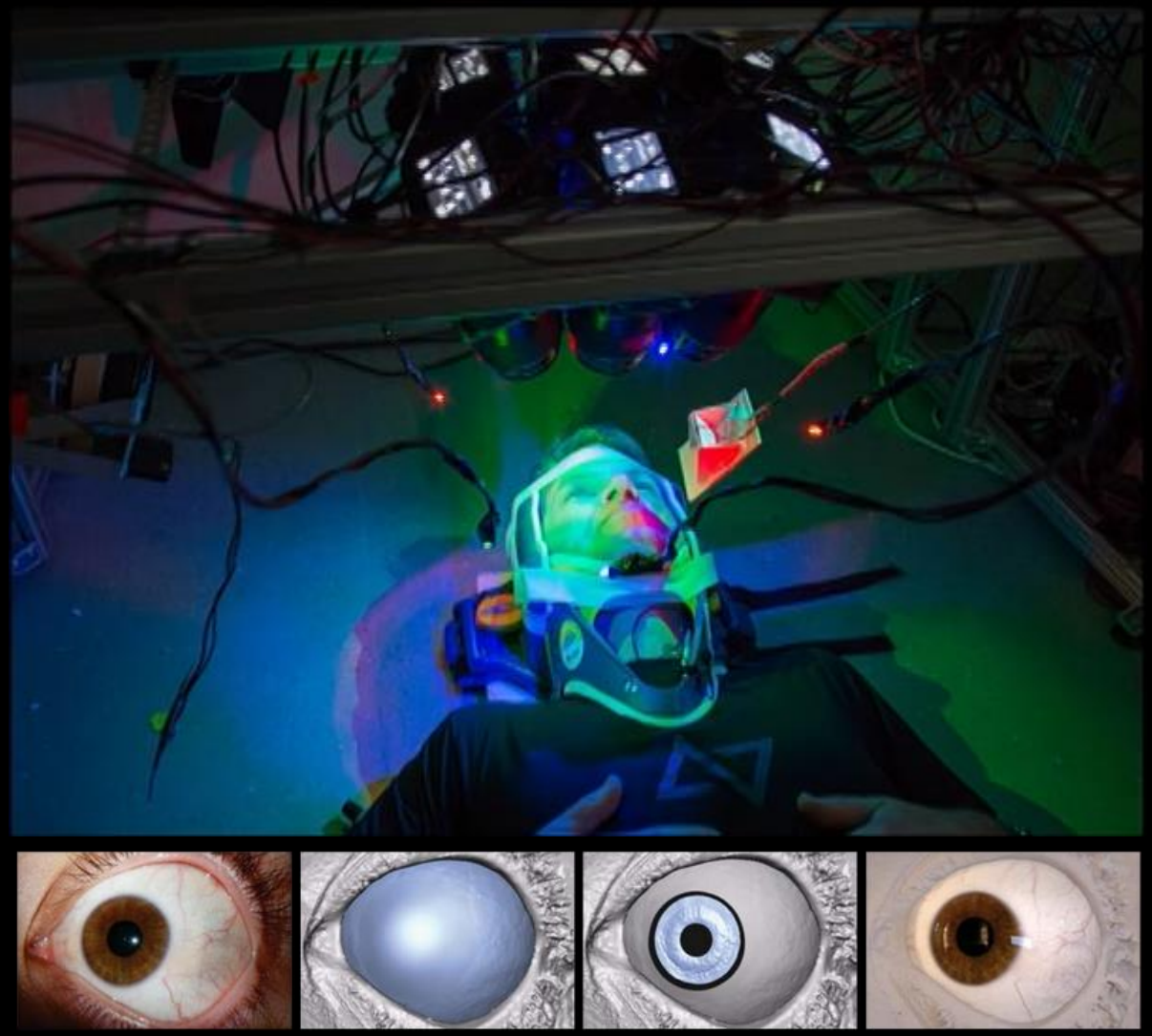
Technical Breakdown

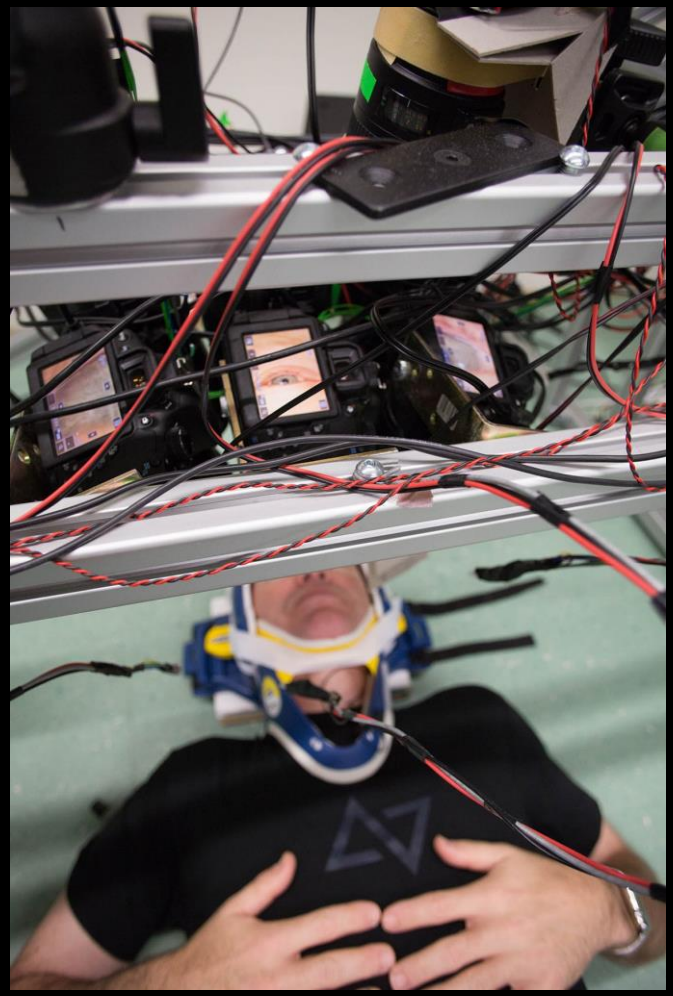
- Asset creation
  - Skin appearance is scaned by using Lightstage at USC ICT
  - Eyes are scaned at Disney Research Zurich
  - Facial rig is created by 3Lateral
- Live system
  - Markless facial landmarks tracking by Cubic Motion
  - Real-time photorealistic shading of skin, hair and eyes are running on UE4

Face Scan with Lightstage



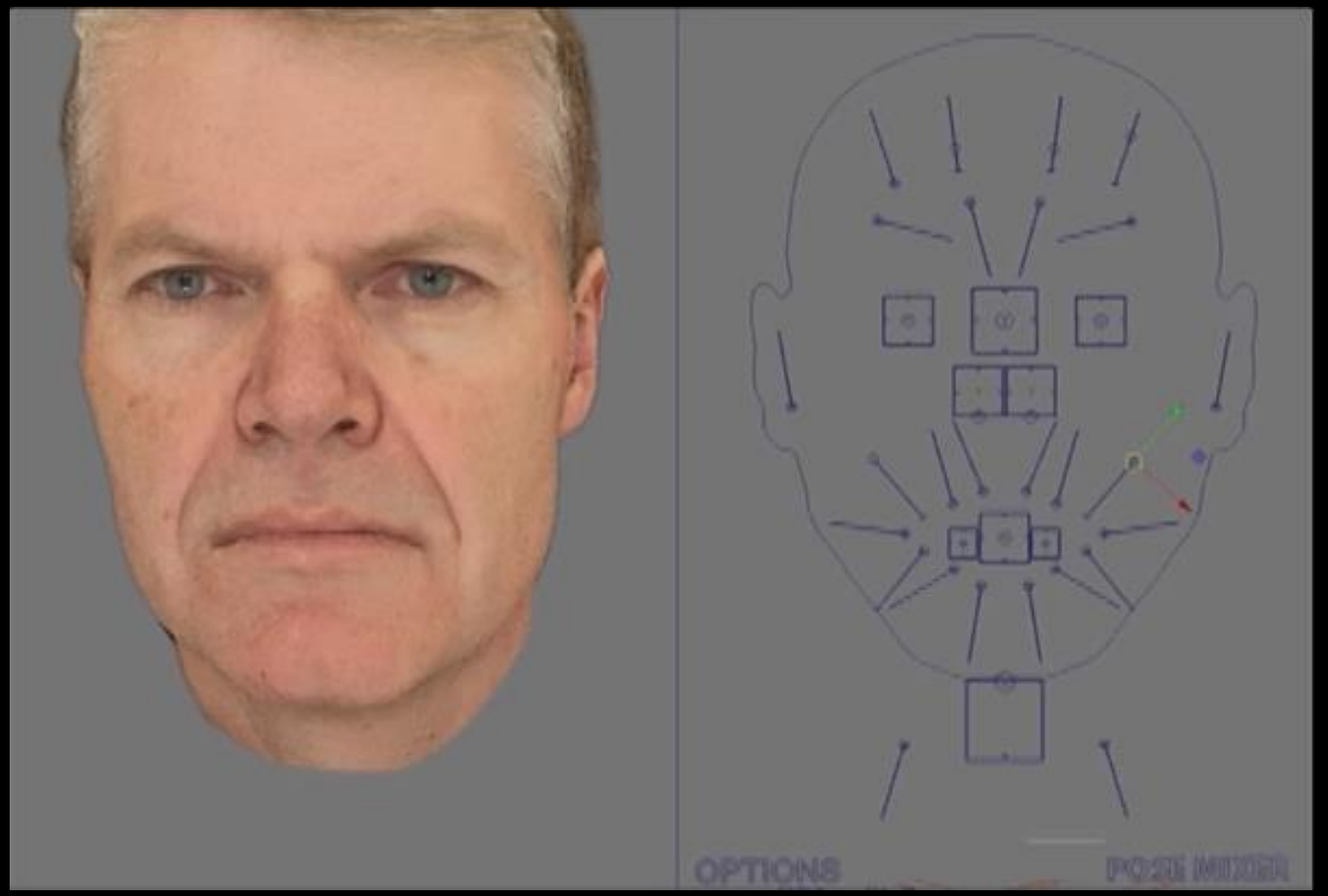
High-Quality Capture of Eyes at Disney Research Zurich



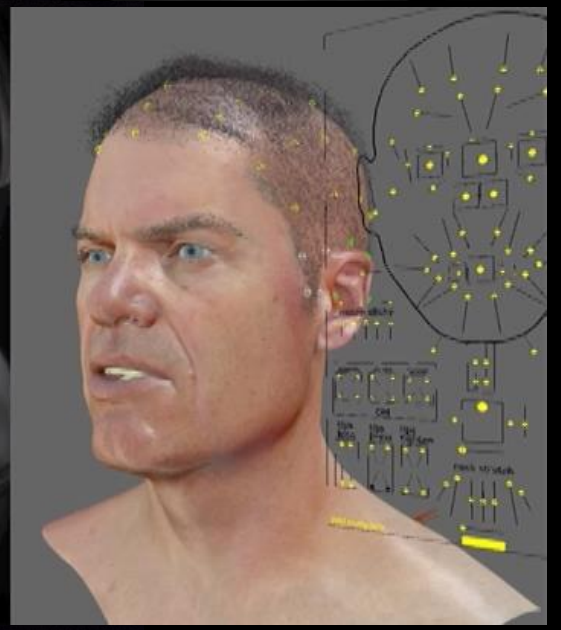


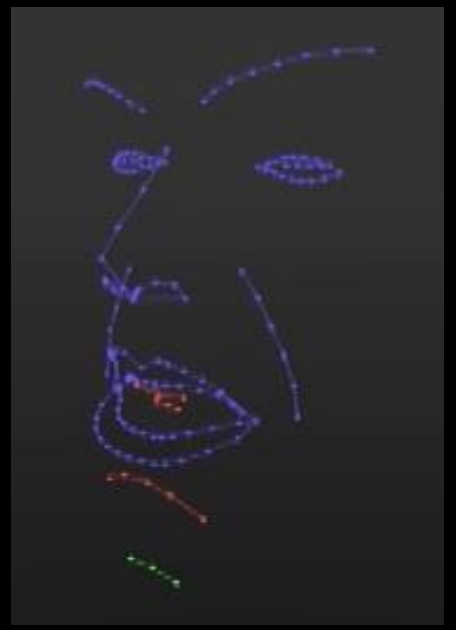
https://www.disneyresearch.com/publication/high-quality-capture-of-eyes/

Facial Rig



Live System





Technical Facts

- About 440,000 triangles being rendered in real time
  - Render 90fps in VR stereo
  - 75% are used for the hair
- About 80 joints in face rig, mostly for the hair and facial hair
- Facial rig: 750 blend shapes and 10 joints for eyes, jaw and tongue
- The system runs on 9 PCs with 32Gig RAM each and 1080ti Nvidia cards

Remarks

- Related subjects
  - Linear algebra
  - Numerical optimization
  - Machine learning (deep learning?)
  - Differential geometry & discrete different geometry
  - A little about fluid dynamics, elasticity, etc.

References

- Quaternions, Ken Shoemake.
- Understanding Rotations, Jim Van Verth.
- On Linear Variational Surface Deformation Methods, Mario Botsch, Olga Sorkine-Hornung.
- Skinning: Real-time Shape Deformation, SIG'14.
- *Laplace-Beltrami: The Swiss Army Knife of Geometry Processing*, SGP'14.