COMP3360:
Data-Driven Computer Animation

Skinning

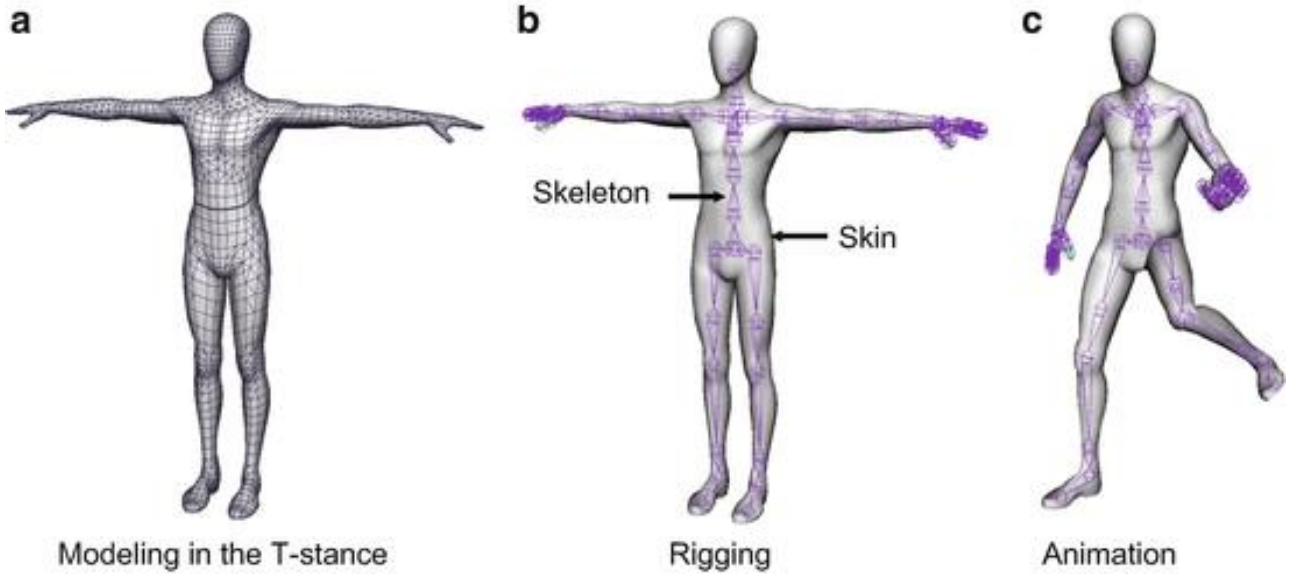
Taku Komura

Overview

- Skinning
  - Background knowledge
  - Linear Blending
  - How to decide weights?
- Dual Quaternion Skinning
- Anatomical models

Skinning

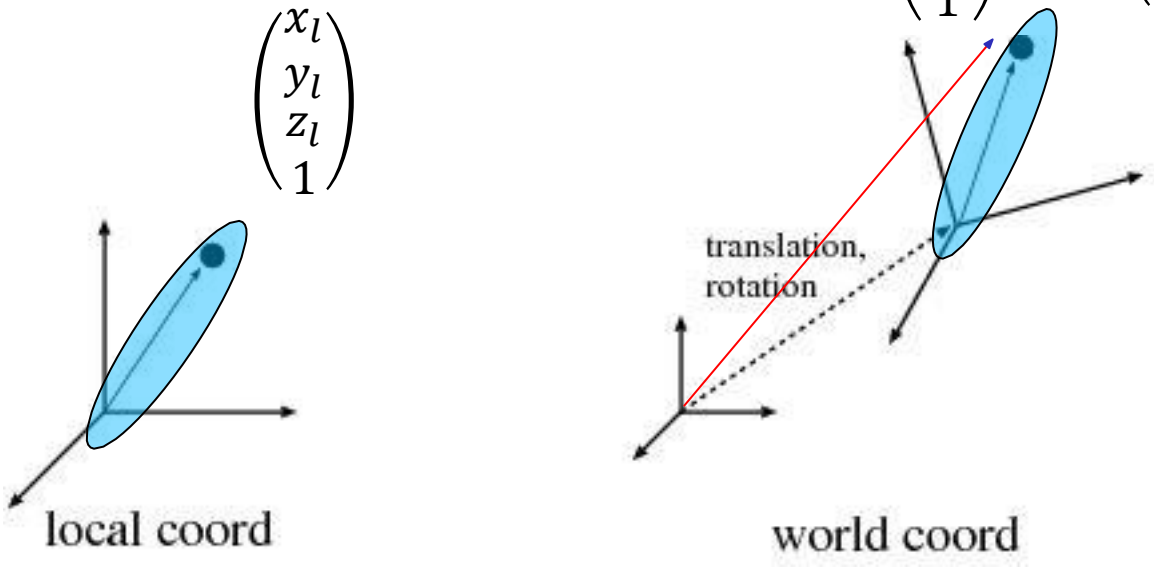
- Assume the movements of the skeleton is determined
- The character’s skin must deform according to the motion of the skeleton
- This process is called skinning



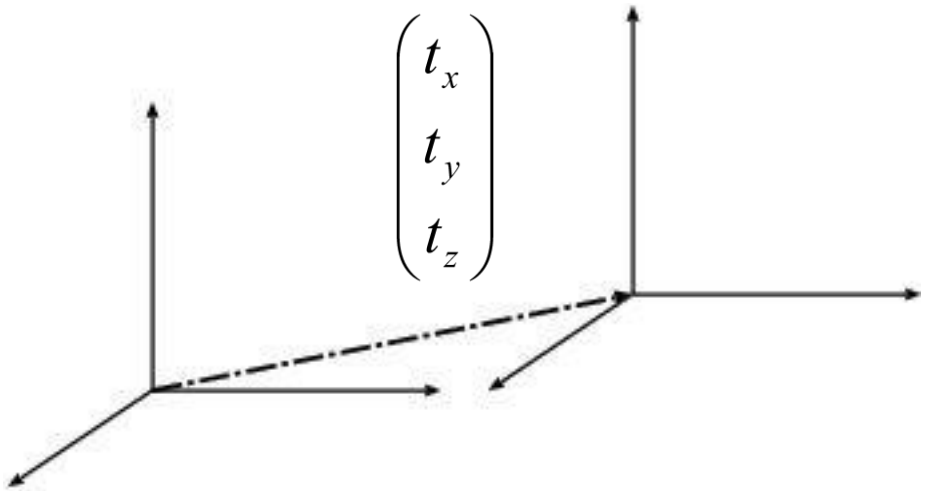
Review: Homogeneous Transformation

- A homogeneous transformation matrix *M* (4x4) is defined per bone
- It represents the pose of the bone
- It converts local coordinates to world coordinates

$$\begin{pmatrix} x_g \\ y_g \\ z_g \\ 1 \end{pmatrix} = \mathbf{M} \begin{pmatrix} x_l \\ y_l \\ z_l \\ 1 \end{pmatrix}$$



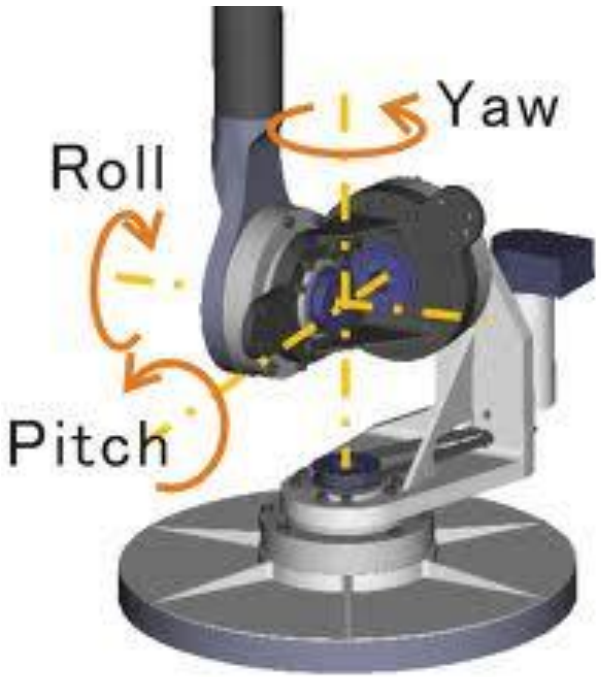
Translation matrix



global coord local coord

$$M = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation matrix



$$Rot\ x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

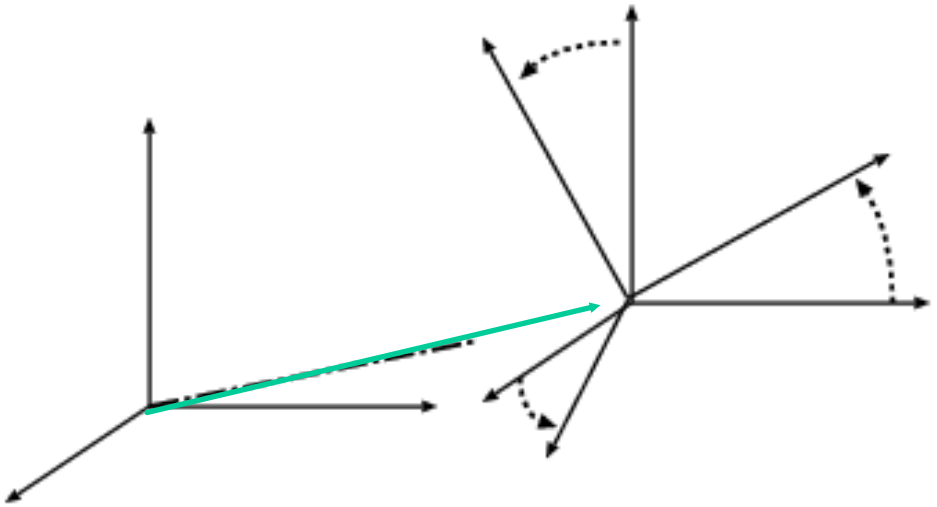
$$Rot\ y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot\ z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- We assume we move together with the local coordinate system while transforming.
- Then, for Euler angles in the order of x, y, z, the matrix can be computed by

$$\begin{bmatrix} \mathbf{R}_{\alpha\beta\gamma} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = Rot\ x(\alpha) \cdot Rot\ y(\beta) \cdot Rot\ z(\gamma)$$

Translation & Rotation matrix



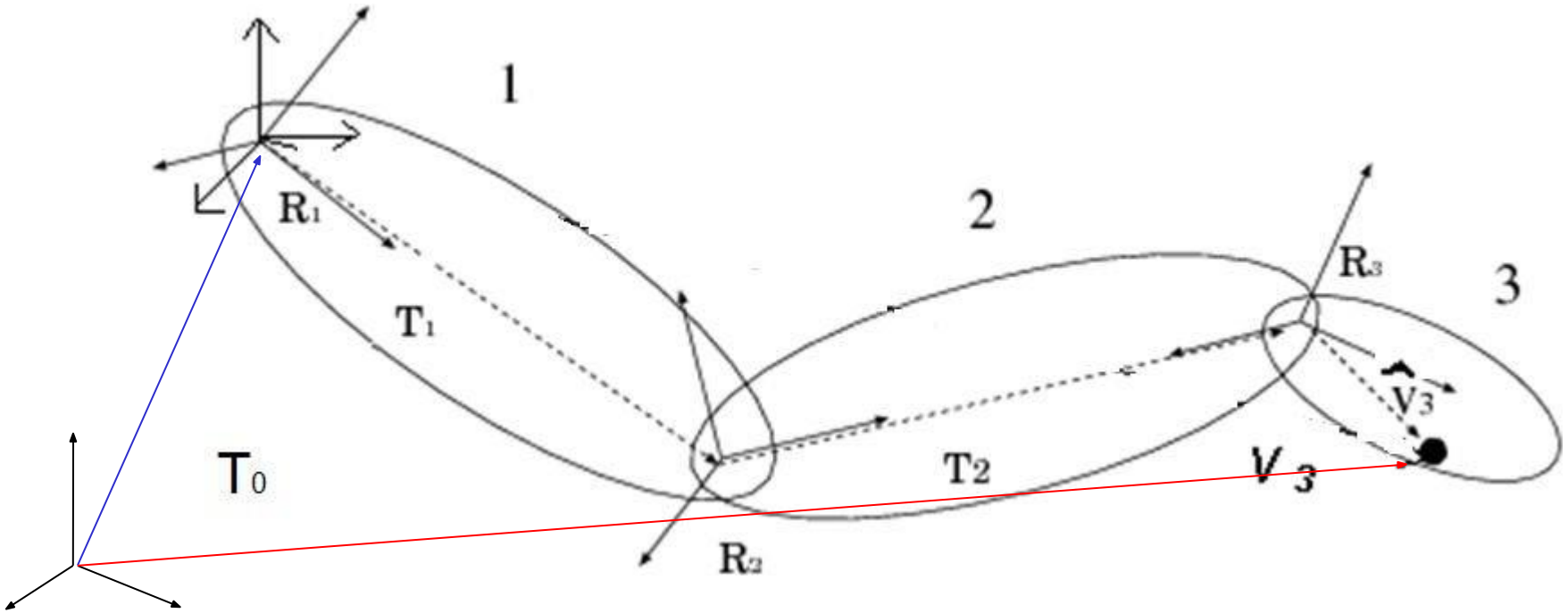
$$M = \begin{bmatrix} & \mathbf{R}_{\boldsymbol{\alpha\beta\gamma}} & \begin{matrix} t_x \\ t_y \\ t_x \end{matrix} \\ 0 & 0 \quad 0 & 1 \end{bmatrix}$$

- First translating the local coordinate to tx, ty, tz
- And then rotating the local coordinate system with the Euler angles in the order of x, y, z,
- The matrix can be computed by $\mathrm{Trans}(tx, ty, tz) \cdot Rot\ x(\alpha) \cdot Rot\ y(\beta) \cdot Rot\ z(\gamma)$

Animating Robots

What is the position of the points composing the segments of the robots?

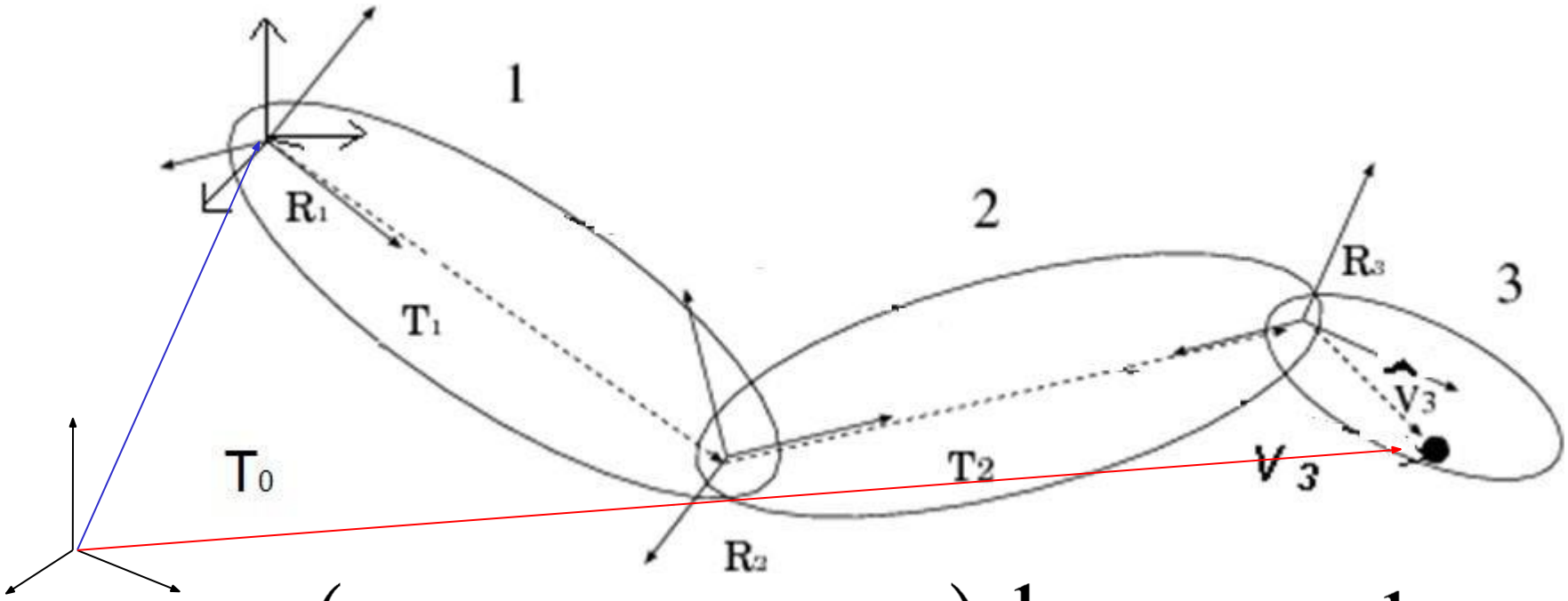
A local-to-global transformation for a hand



$$\mathbf{v}_3 = \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{T}_2\mathbf{R}_3\hat{\mathbf{v}}_3 = \mathbf{M}_3\hat{\mathbf{v}}_3$$

- A body composed of three segments
- A vertex ***v3*** defined in a local coordinate system of segment 3
- Its global position v is computed by the above equation

A global-to-local transformation for a hand



$$\hat{\mathbf{v}}_3 = \left(\mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{T}_2\mathbf{R}_3\right)^{-1}\mathbf{v}_3 = \mathbf{M}_3^{-1}\mathbf{v}_3$$

- Given the global location, the local coordinate of the point can be computed as above.

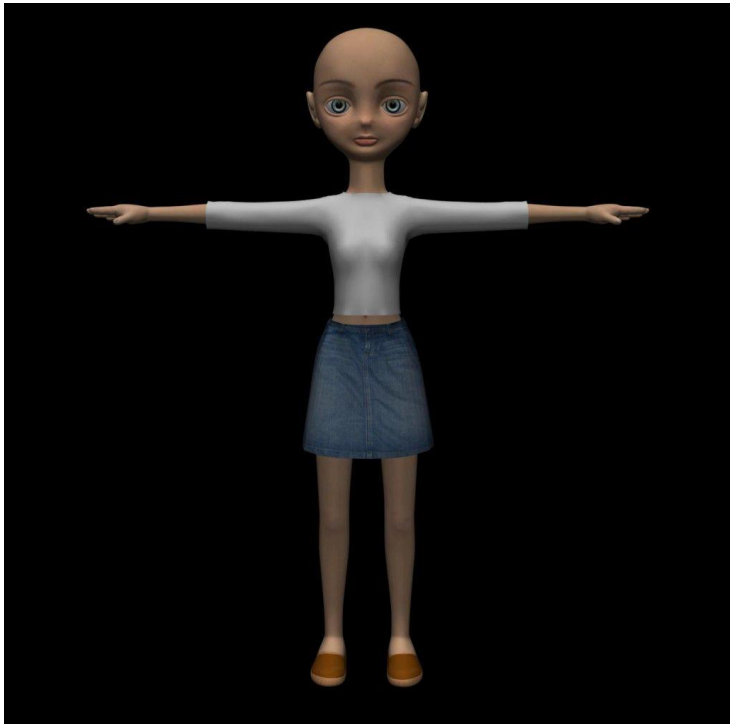
For robots, this is fine

- Compute the local-to-global matrix for all body segments : $\mathbf{M_i}$
- Multiply the local coordinates $\mathbf{v_i}$ of every body segment to this matrix to calculate its global position :

$$\mathbf{v_i} = \mathbf{M_i}\hat{\mathbf{v}}_\mathbf{i}$$

- Then, the global position of all the points can be computed

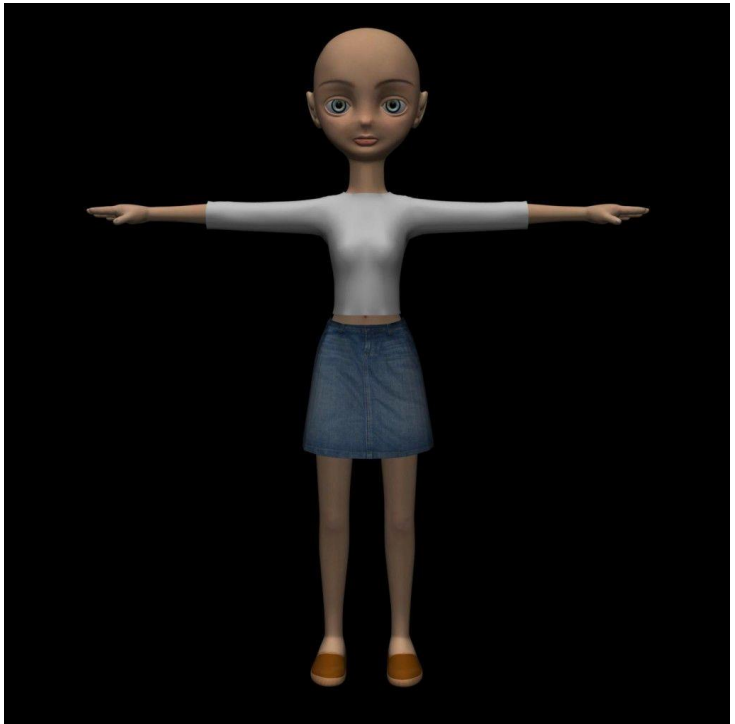
What about polygon characters?



What about polygon characters?

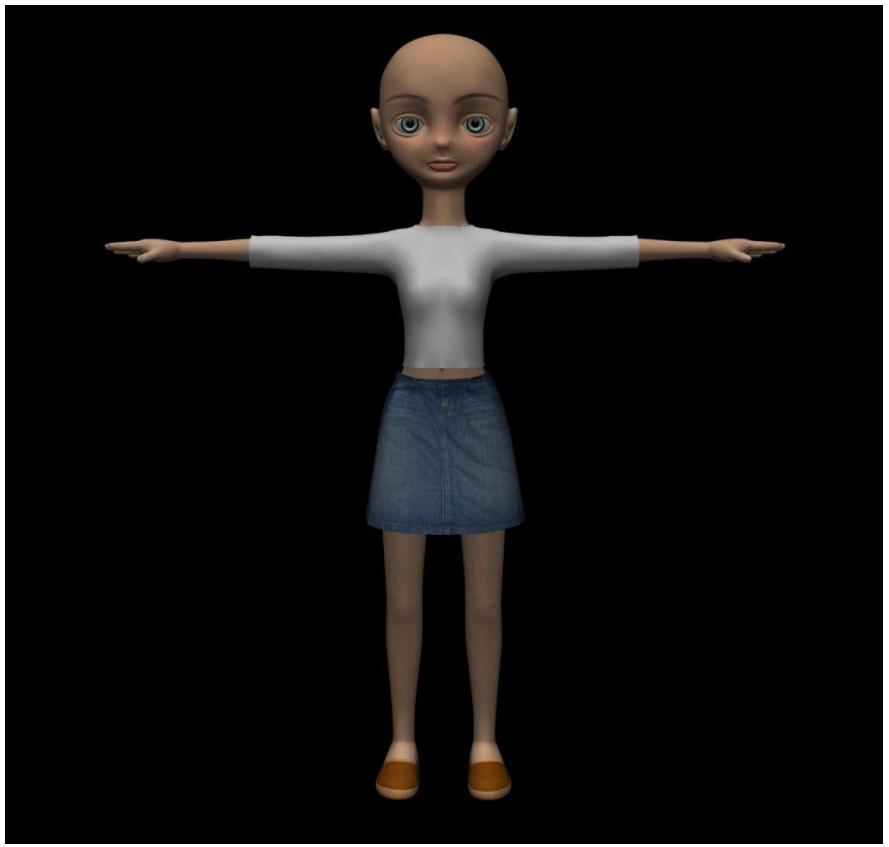
Problem 1. We are only given the polygon – no skeleton structure

Problem 2. Some points on the body do not belong to a single bone but to multiple bones close to it

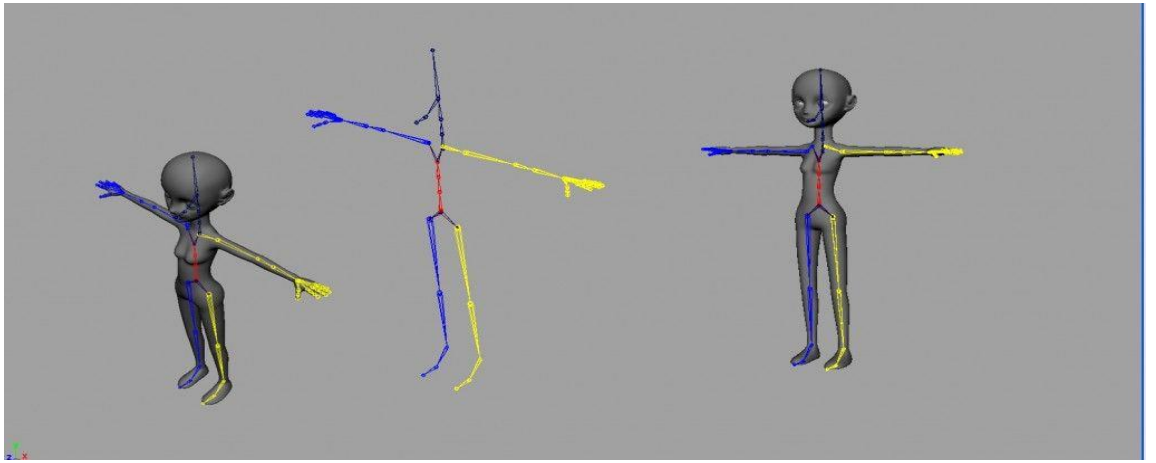


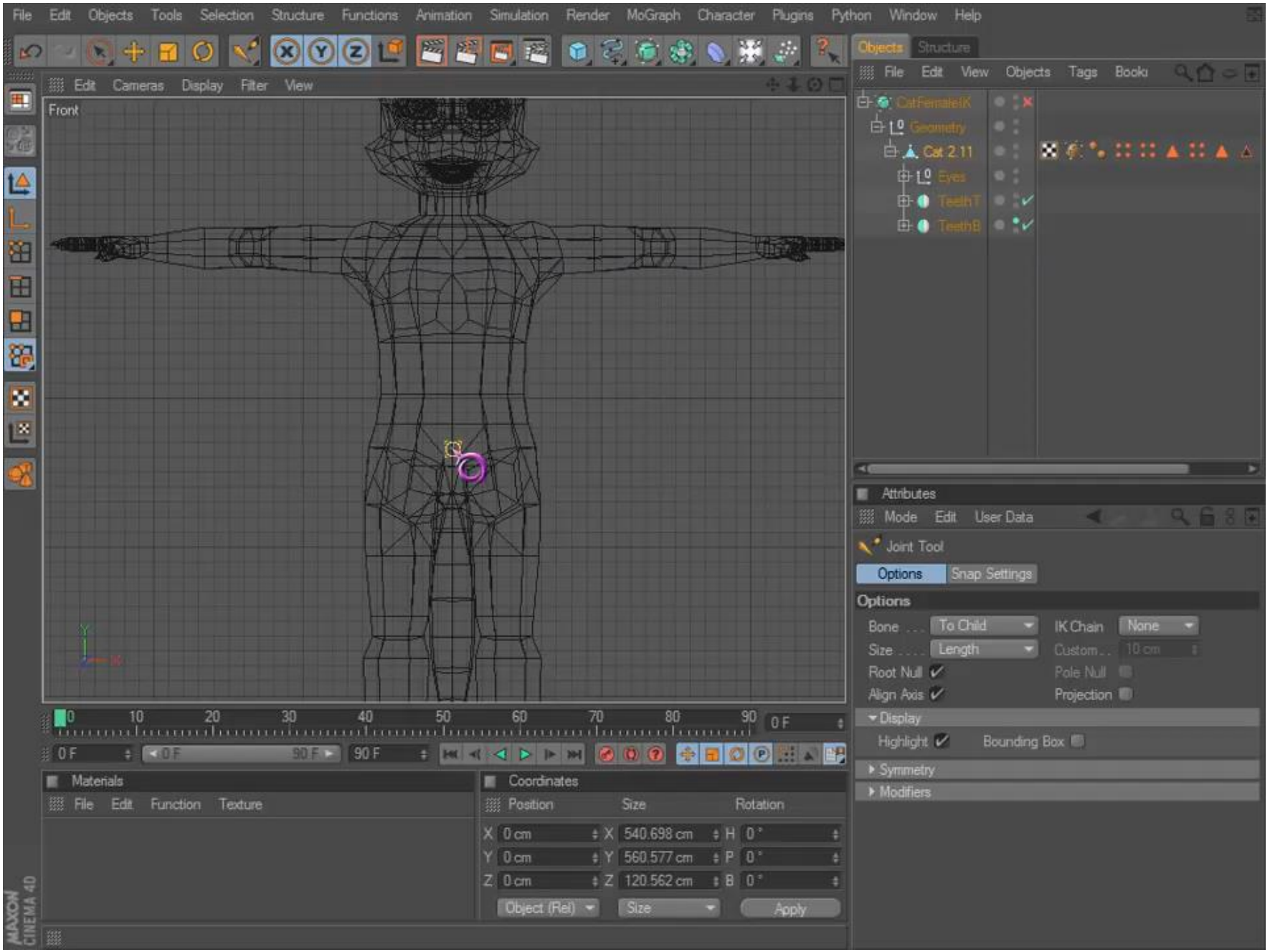
Solution: Skinning

Fitting the skeleton into the polygon model (done manually)



https://www.youtube.com/watch?v=dniWVu55PEc&feature=related

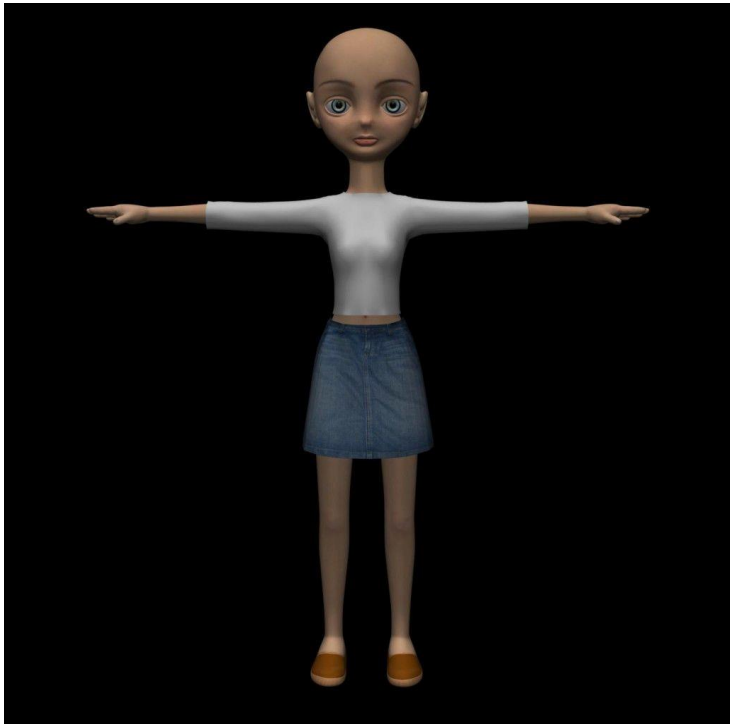



File Edit Objects Tools Selection Structure Functions Animation Simulation Render MoGraph Character Plugins Python Window Help
Edit Cameras Display Filter View
Front
Objects Structure
File Edit View Objects Tags Book
CatFemaleIK
Geometry
Cat 2.11
Eyes
TeethT
TeethB
Attributes
Mode Edit User Data
Joint Tool
Options Snap Settings
Options
Bone To Child
IK Chain None
Size Length
Custom 10 cm
Root Null
Pole Null
Align Axis
Projection
Display
Highlight
Bounding Box
Symmetry
Modifiers
0 10 20 30 40 50 60 70 80 90 0 F
0 F 0 F 90 F 90 F
Materials
File Edit Function Texture
Coordinates
Position Size Rotation
X 0 cm X 540.698 cm H 0 °
Y 0 cm Y 560.577 cm P 0 °
Z 0 cm Z 120.562 cm B 0 °
Object (Rel) Size Apply
MAXON CINEMA 4D

What about polygon characters?

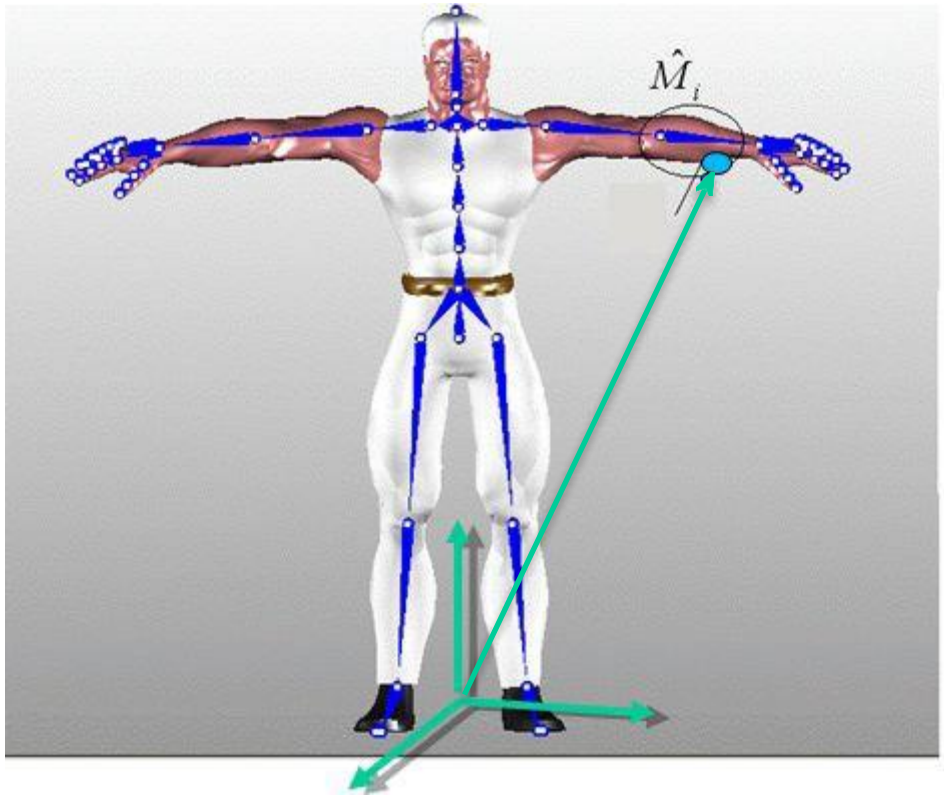
Problem 1. We are only given the polygon – no skeleton structure

Problem 2. Some points on the body do not belong to a single bone but to multiple bones close to it



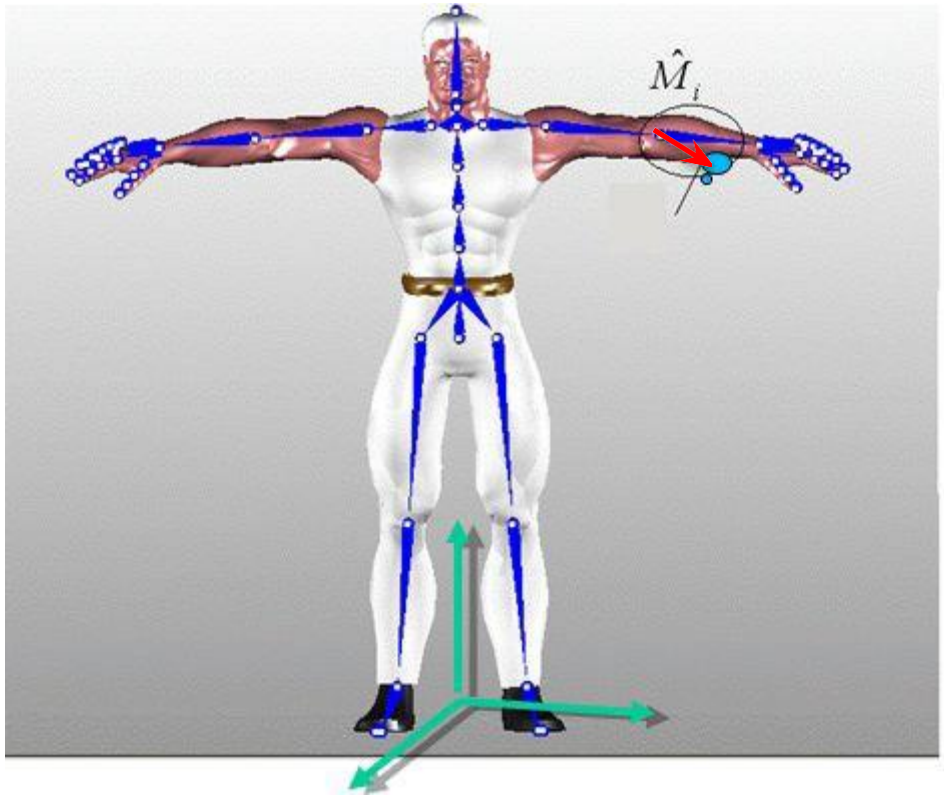
Rest Pose to Bone Coordinate

- the global position of a particular vertex, ***v***, in the rest pose is defined $\mathbf{v}_{\mathbf{g}}$



Rest Pose to Bone Coordinate

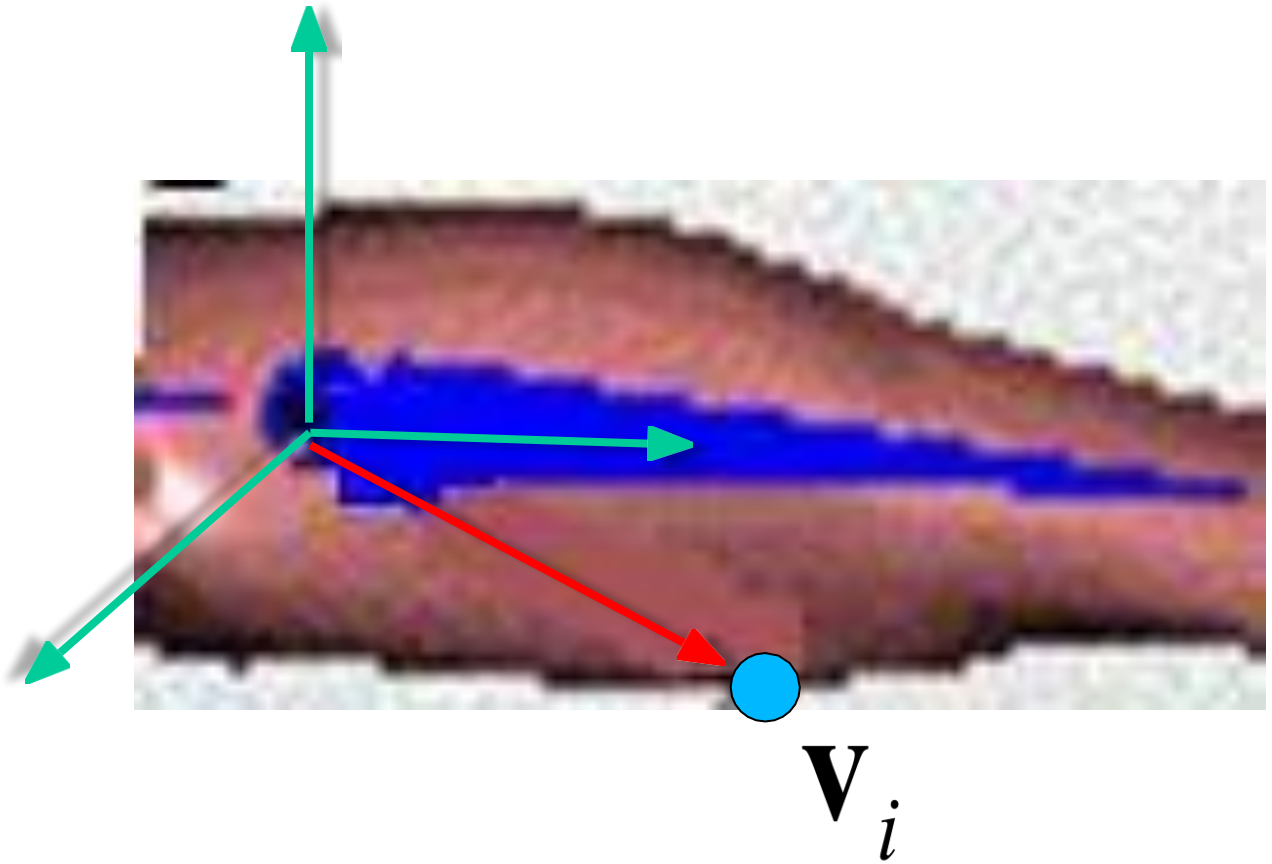
- We want to know where this point $\mathbf{V}_g$ is in the local coordinates



Rest Pose to Bone Coordinate

- for each bone, i, the position of the vertex in the rest pose is first transformed from model coordinates ($\mathbf{v}_g$) to bone coordinates ($\mathbf{v}_i$) by applying the inverse of the rest pose bone transformation:

$$\mathbf{v}_i = \mathbf{M}_i^{-1}\mathbf{v}_g$$

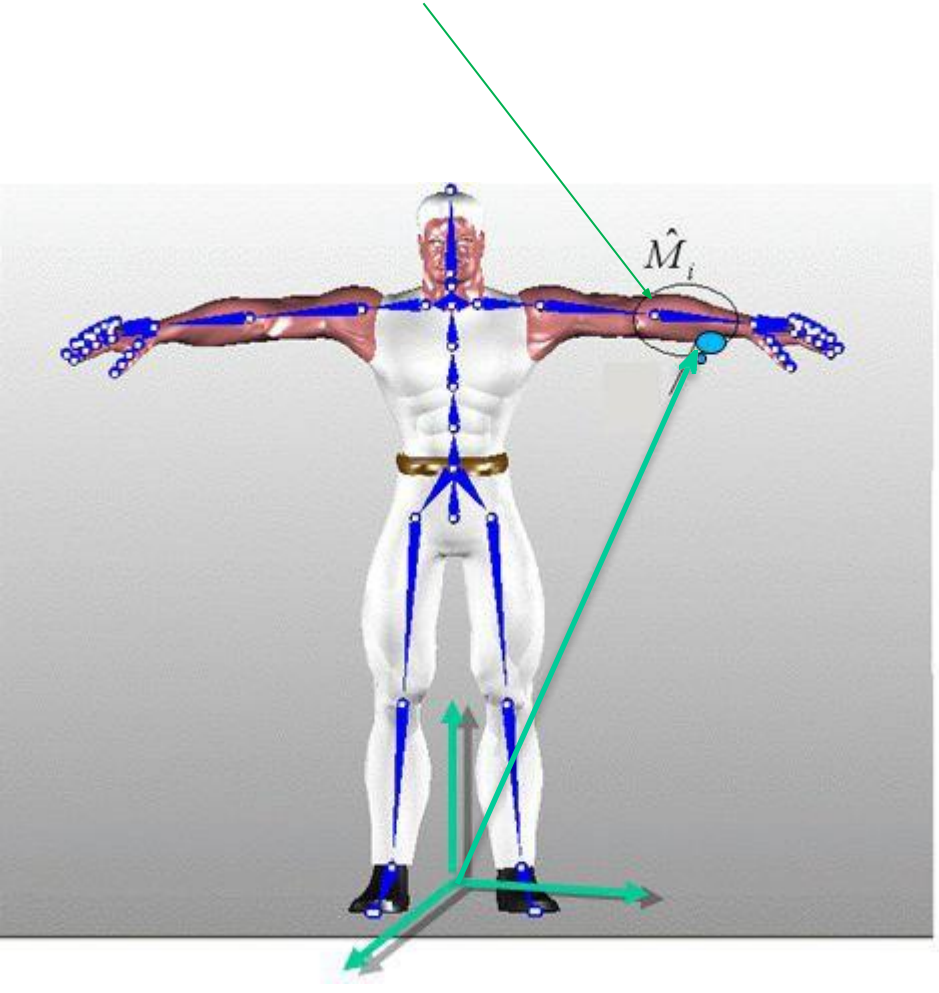


Rest Pose to Bone Coordinate: Example

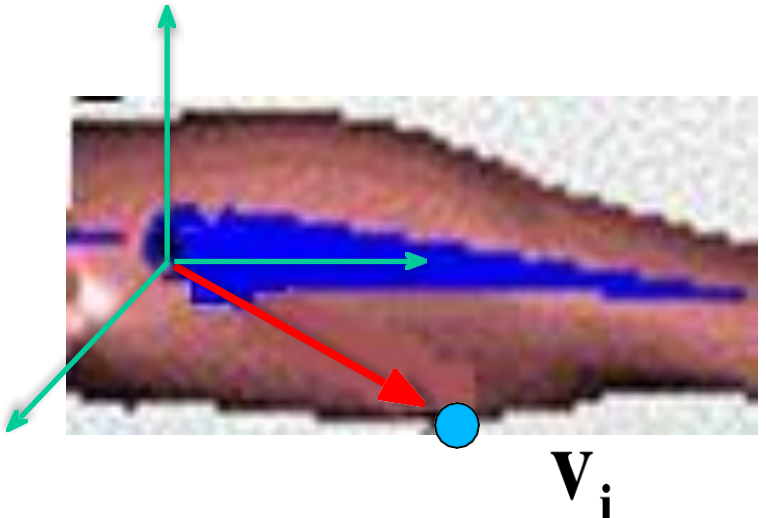
center of elbow joint

(0.7,1.5,0)

$$\begin{bmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 1.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{v}_g = (0.8,1.45,0)$$



Rest Pose to Bone Coordinate

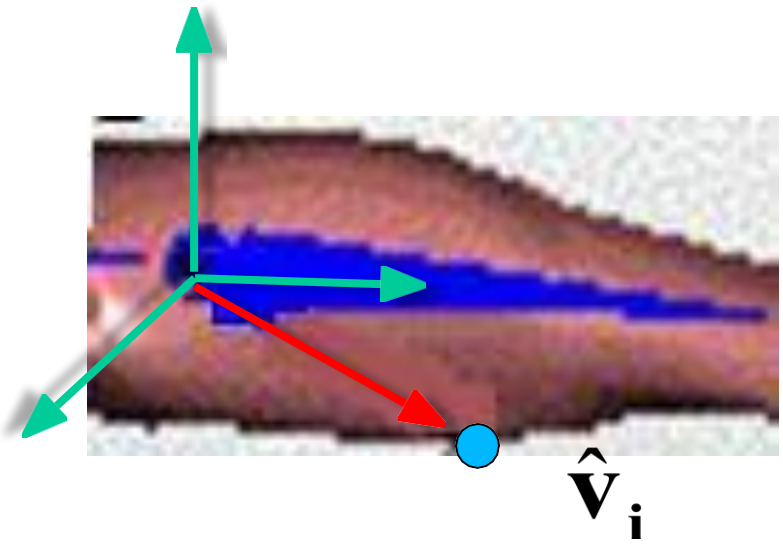
$$\mathbf{v}_i = \mathbf{M}_i^{-1}\mathbf{v}_g$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 1.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0.8 \\ 1.45 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & -0.7 \\ 0 & 1 & 0 & -1.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 \\ 1.45 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.05 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{M}_i = \begin{bmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 1.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v_g = (0.8, 1.45, 0)$$

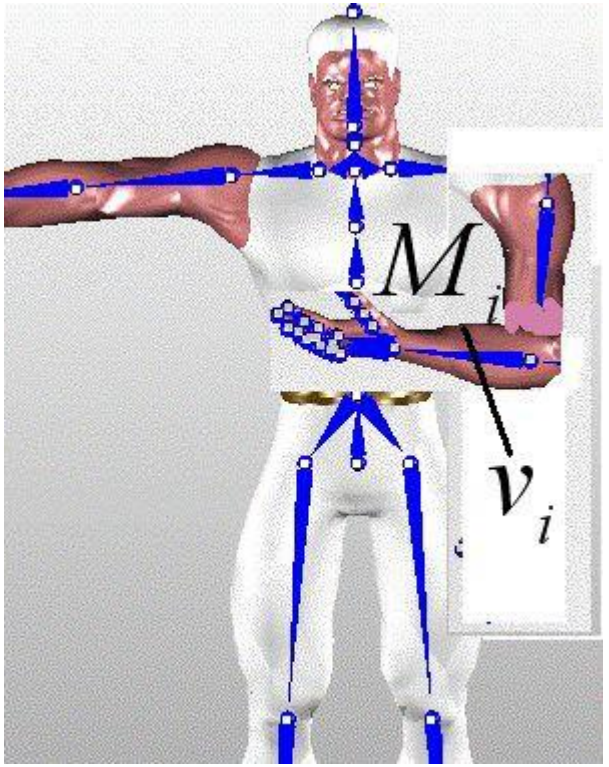


Bone Coordinate to World Coordinate

- The vertex in bone coordinates, $\mathbf{v_i}$ is then transformed back into world coordinates by applying the transformation of the bone in the new pose

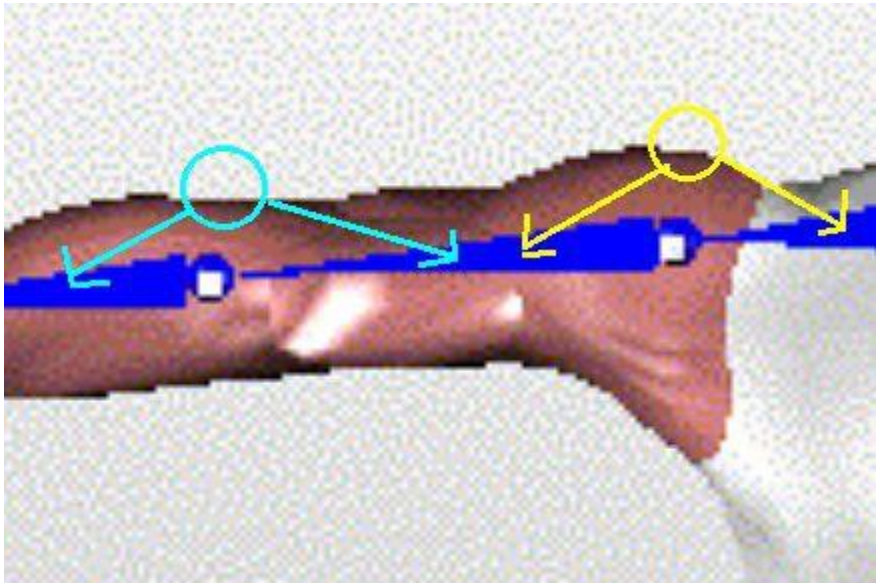
$$\mathbf{v'_g = M'_i M_i^{-1} v_g}$$

$\mathbf{M'_i}$: the local-to-global matrix in the new posture



Back to Problem 2:

- For some points, we don’t know to which body segment it belongs to
- For the points near the elbow joint, it might belong to the upper arm, or the forearm or maybe both
- We want both segments to affect its movements

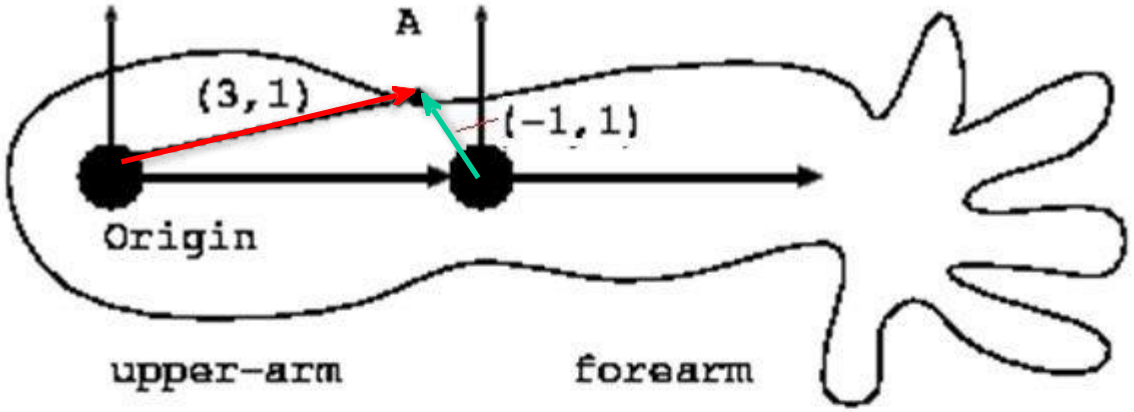


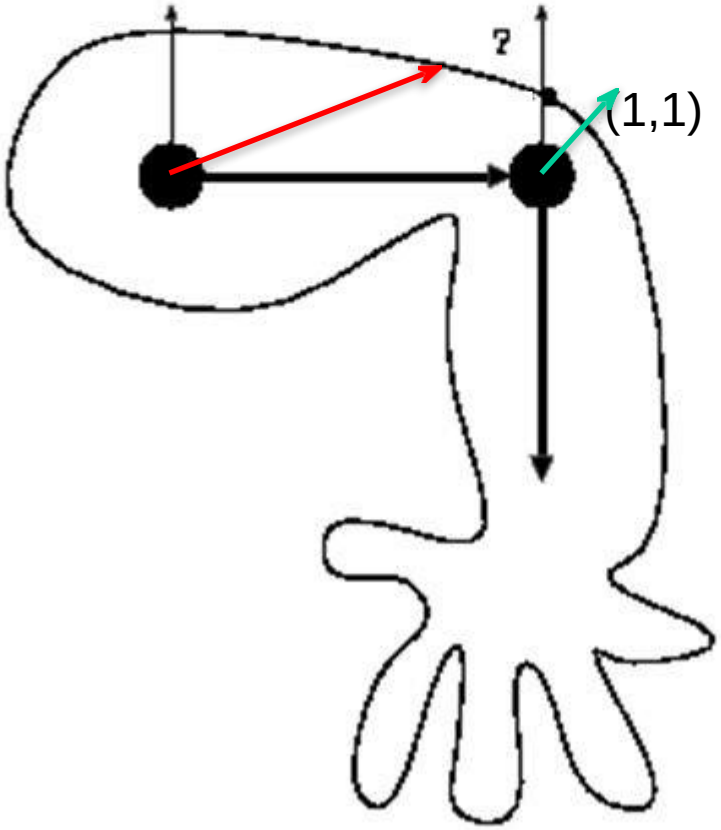
Solution: Linear Blend Skinning

- Linearly blend the results of the vertex transformed rigidly with each bone.
- A scalar weight, w_i, is given to each influencing bone and the weighted sum gives the vertex's position, v, in the new pose, as follows:

$$\mathbf{v} = \sum_{i=1}^{n} w_i \mathbf{M}'_{\mathbf{i}} \mathbf{M}_{\mathbf{i}}^{-1} \mathbf{v}_{\mathbf{g}} = \left(\sum_{i=1}^{n} w_i \mathbf{T}_{\mathbf{i}}\right) \mathbf{v}_{\mathbf{g}} \qquad \sum_{i=1}^{n} w_i = 1$$

n is the number of bones influencing the position of v

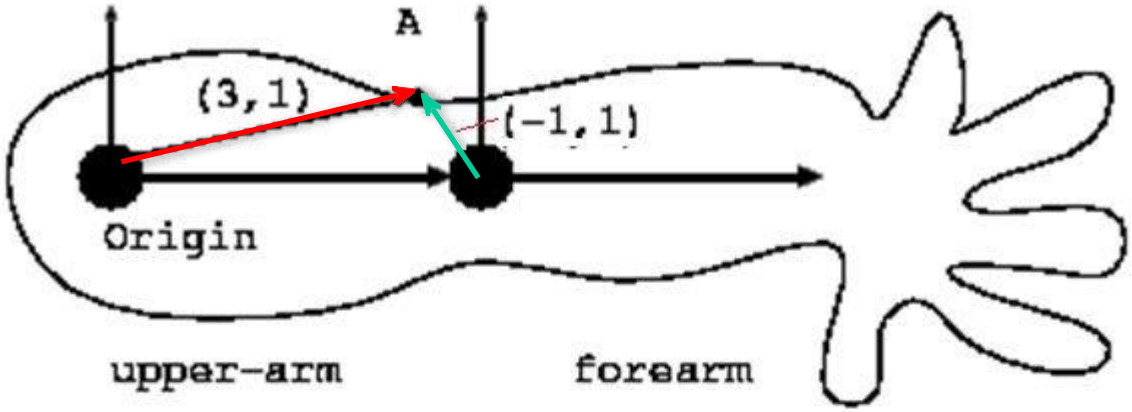




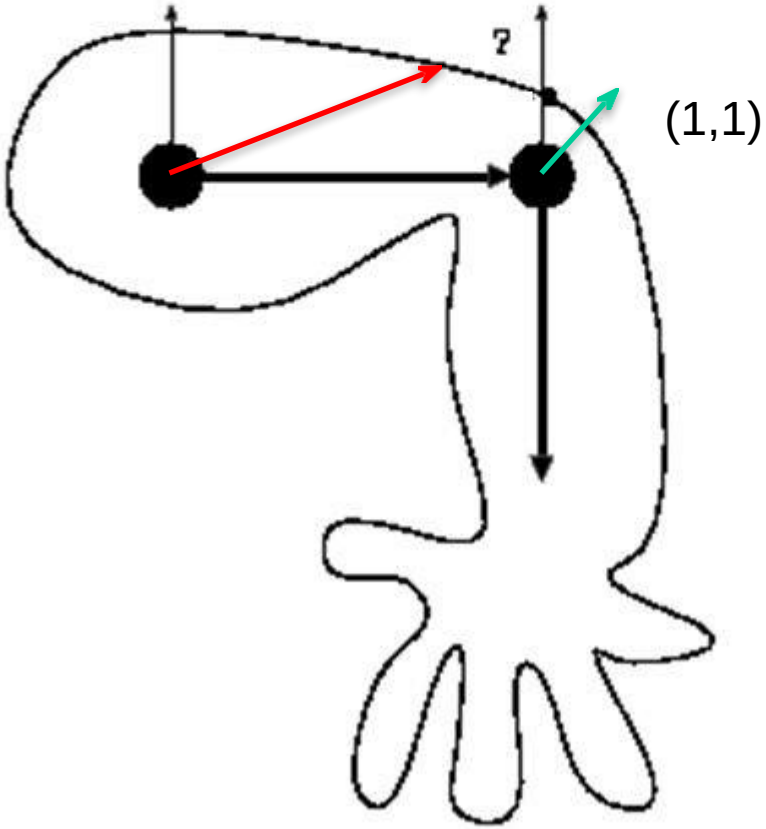
What is the position of point A after the elbow is bent 90 degrees?

- assuming it is a point of the upper arm
- assuming it is a point of the forearm

Assuming the weight is 0.8 for the upper-arm and 0.2 for the forearm



- Assuming it is a point of the upper arm, the position is (3,1)
- assuming it is a point of the forearm, it is (5,1)



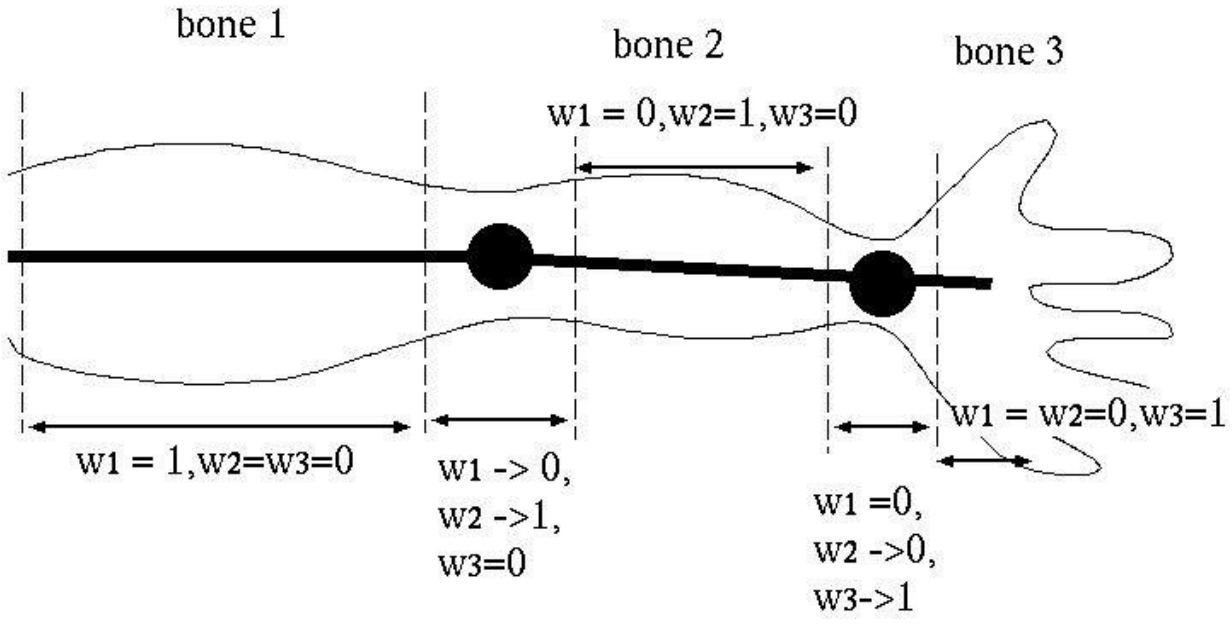
- Assuming the weight is 0.8 for the upper-arm and 0.2 for the forearm, the position is

0.8*(3,1)+0.2*(5,1) =(3.4, 1)

How to decide the weights?

Decide the mapping of the vertex to the bone

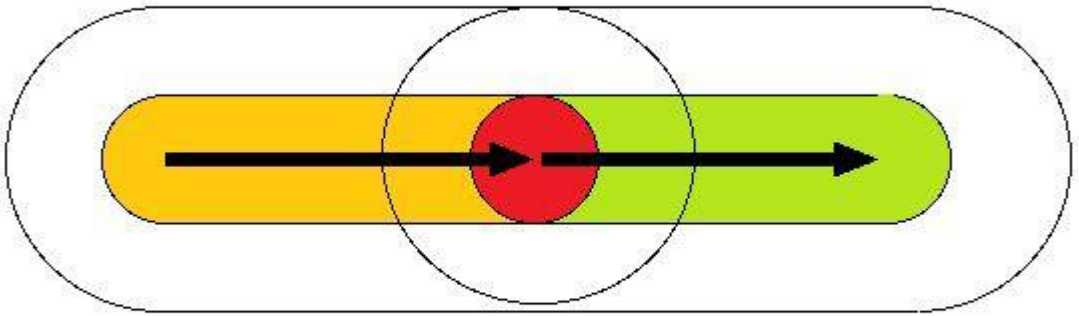
- If vertex v is in the middle of bone i, then $w_i = 1$ and for the rest $w_{j \neq i} = 0$
- If the vertex is near the border of bone i and i+1, *wi* will gradually decrease to 0 and *wi+1* will gradually increase to 1
- If the vertex is affected by more than three bones, the weight can be determined according to its distance to each bone



How to decide the weights?

Example: use the Euclidean distance

- Surround the bones by the inner and outer capsules
- If the vertex is inside only one inner capsule, the weight for the corresponding bone is 1, and the rest are 0
- If inside multiple inner capsules, compute the distance to each bone, and use that to decide the weights (longer distance, lower weight)

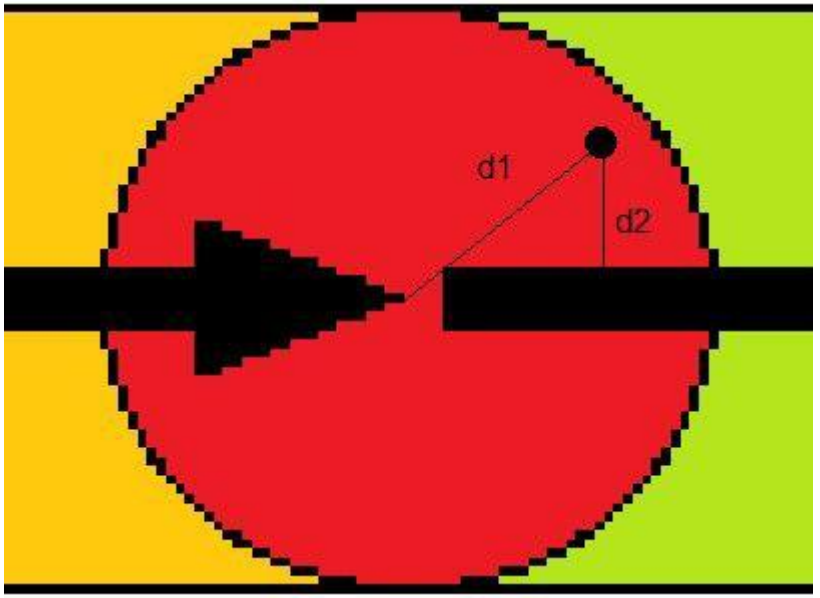


How to decide the weights?

Example: use the Euclidean distance

- Surround the bones by the inner and outer capsules
- If the vertex is inside only one inner capsule, the weight for the corresponding bone is 1, and the rest are 0
- If inside multiple inner capsules, compute the distance to each bone, and set the weights inverse proportional to the distance with normalization

$$w_1 = \frac{\frac{1}{d_1}}{\frac{1}{d_1}+\frac{1}{d_2}}, w_2 = \frac{\frac{1}{d_2}}{\frac{1}{d_1}+\frac{1}{d_2}}$$



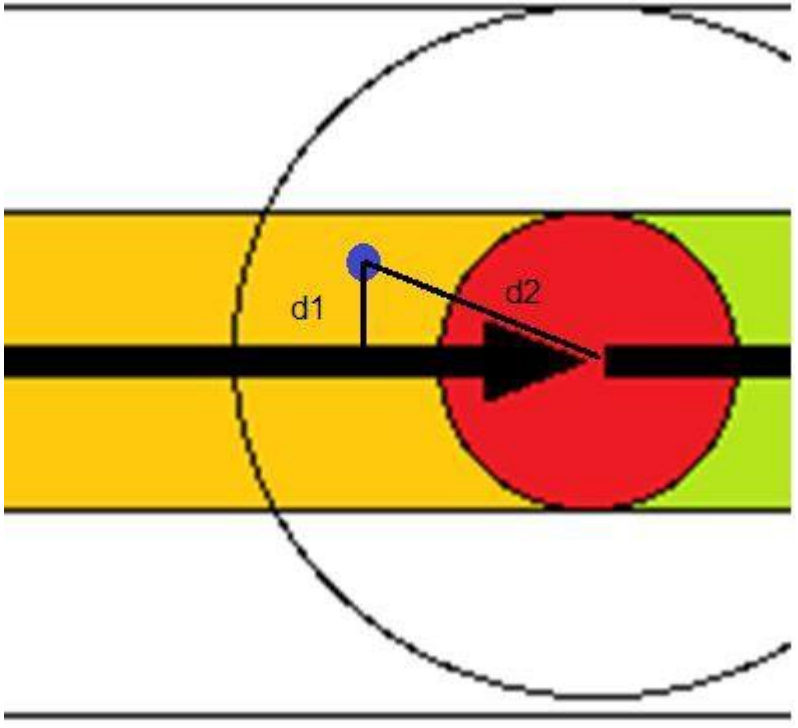
How to decide the weights?

Example: use the Euclidean distance

- If inside the inner capsule of one and in outer capsule of the other, use the distance again but with a fall-off with the other

i.e.

$$w_1 = \frac{\frac{1}{d_1}}{\frac{1}{d_1}+f(d_2)\frac{1}{d_2}}, w_2 = \frac{f(d_2)\frac{1}{d_2}}{\frac{1}{d_1}+f(d_2)\frac{1}{d_2}}$$



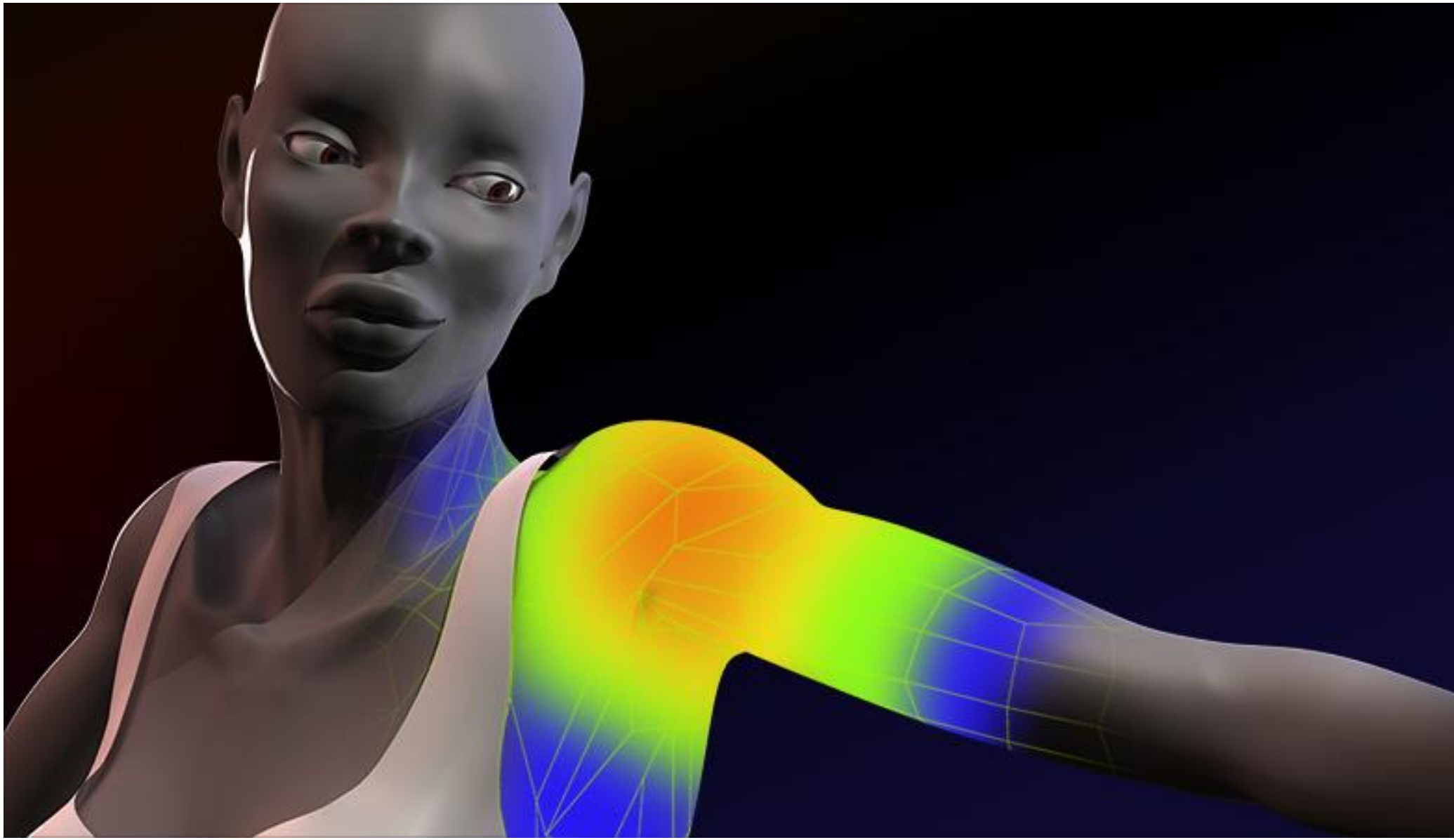
$$\text{f(x)} : \text{fall-off function}$$

$$f(x) = 1 \text{ if } x < r_1$$

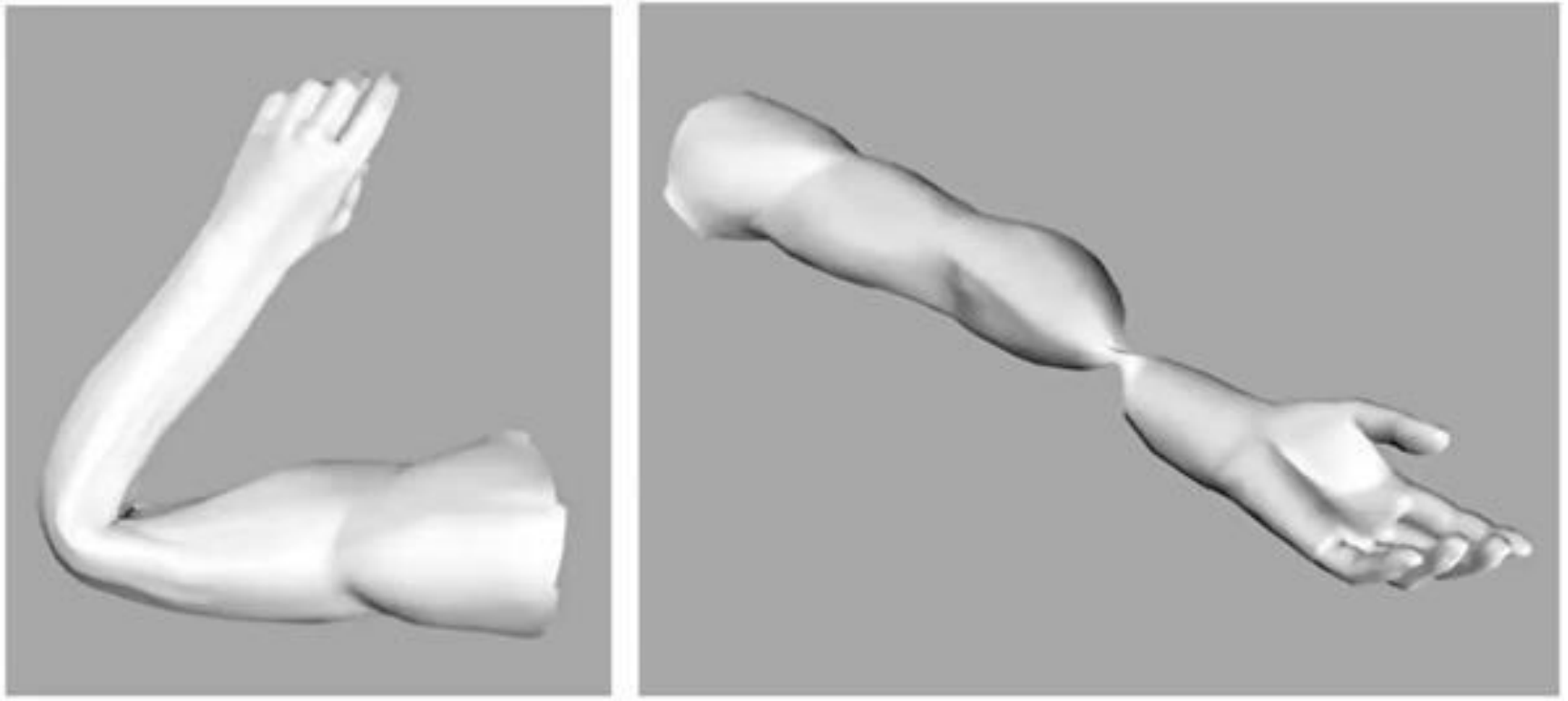
$$f(x) = 0 \text{ if } x > r_2$$

$$f(x) = 1 - \frac{x - r_1}{r_2 - r_1}$$

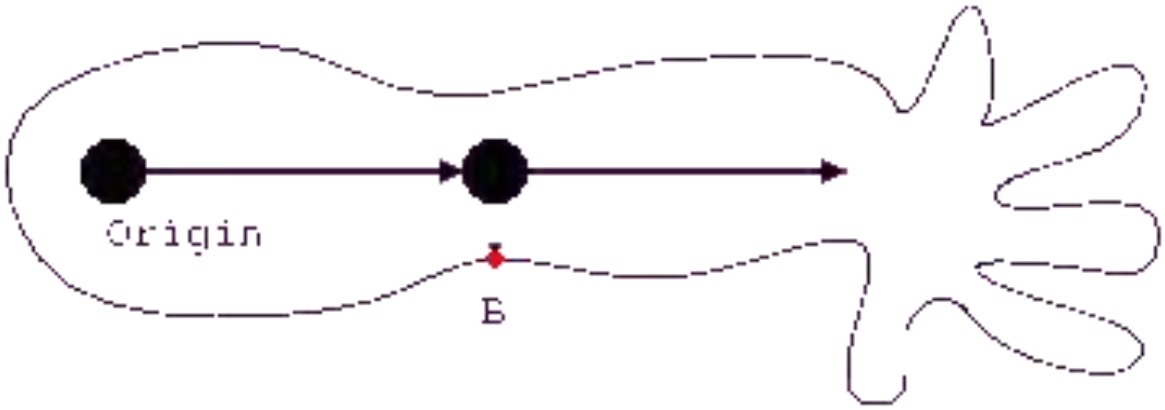
Otherwise, manually edit it using brushing tools

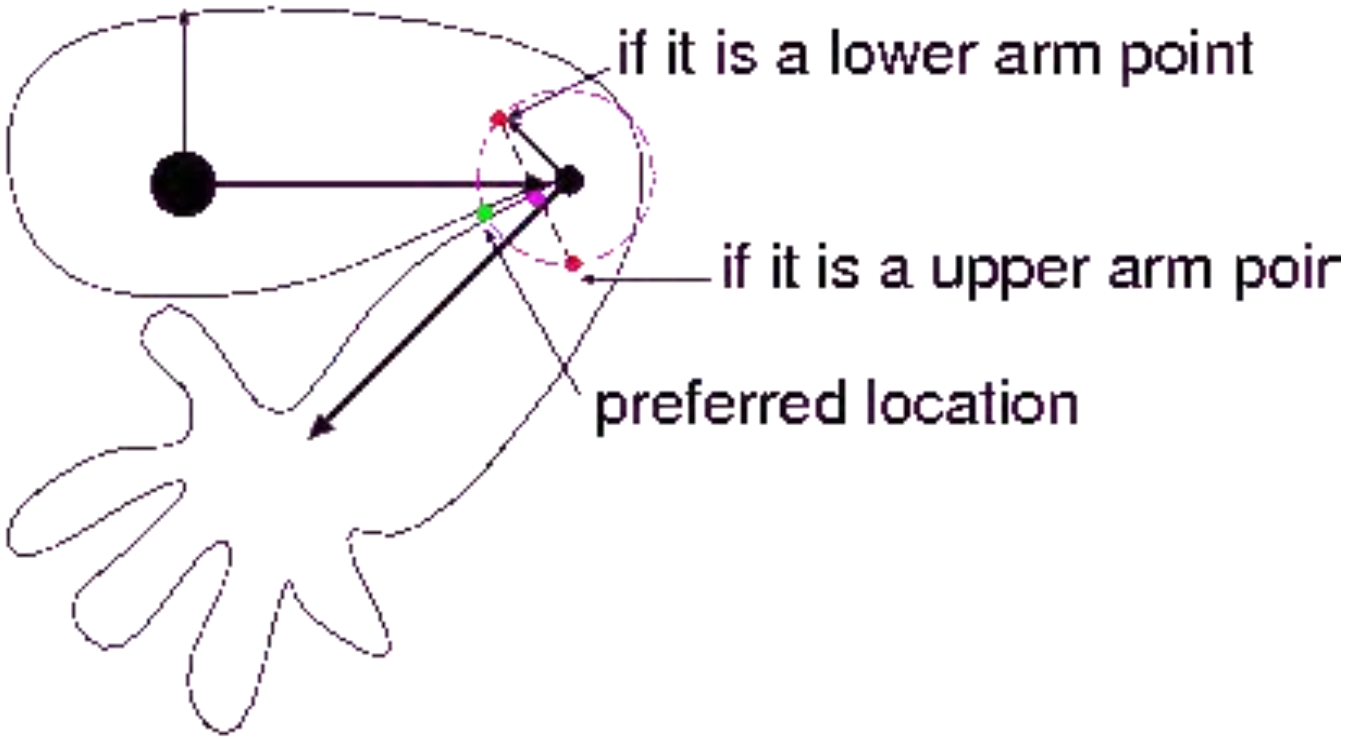


Linear Blend Skinning: Issues Collapse and Candy Wrapper

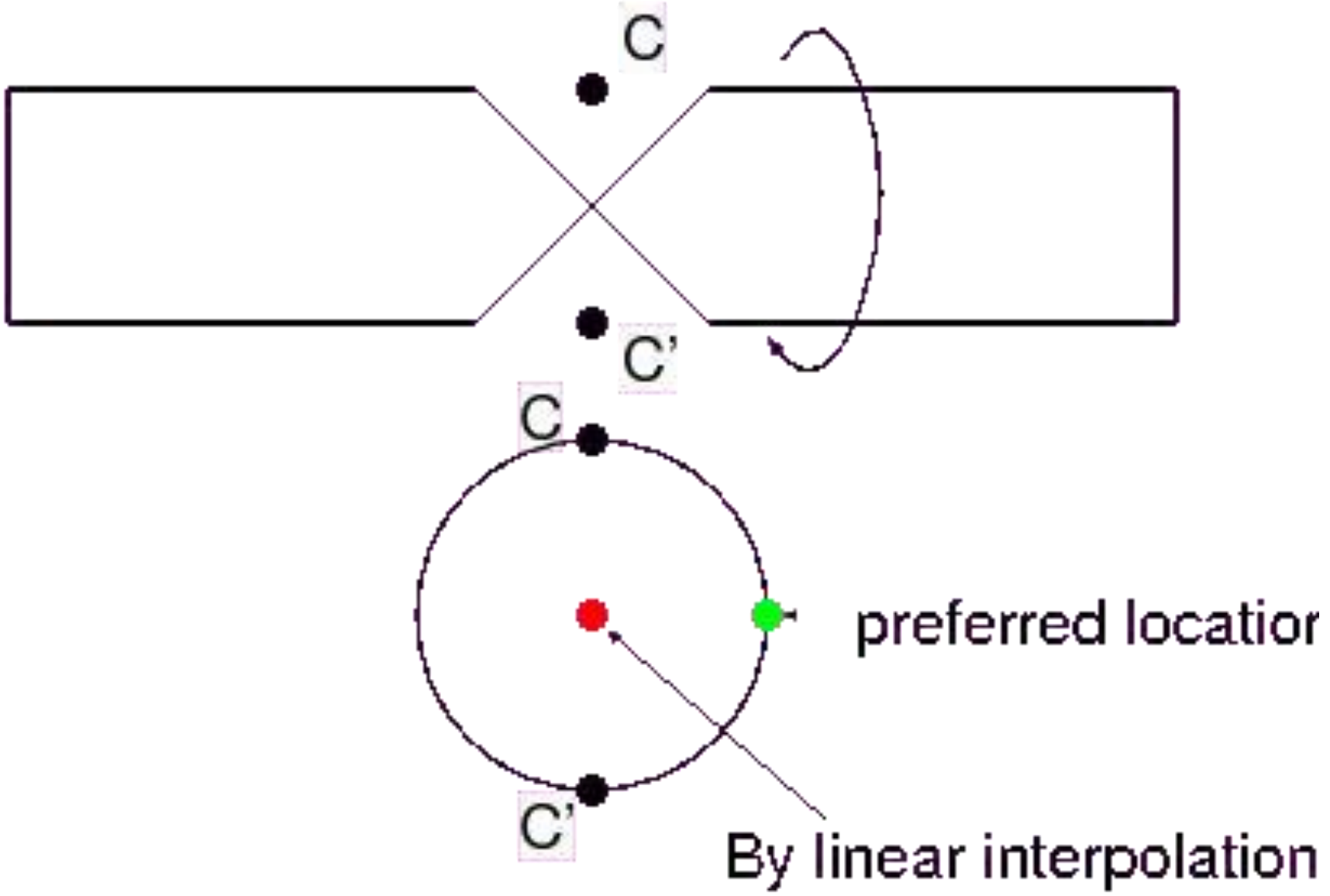


Why does it happen?





Why does it happen?



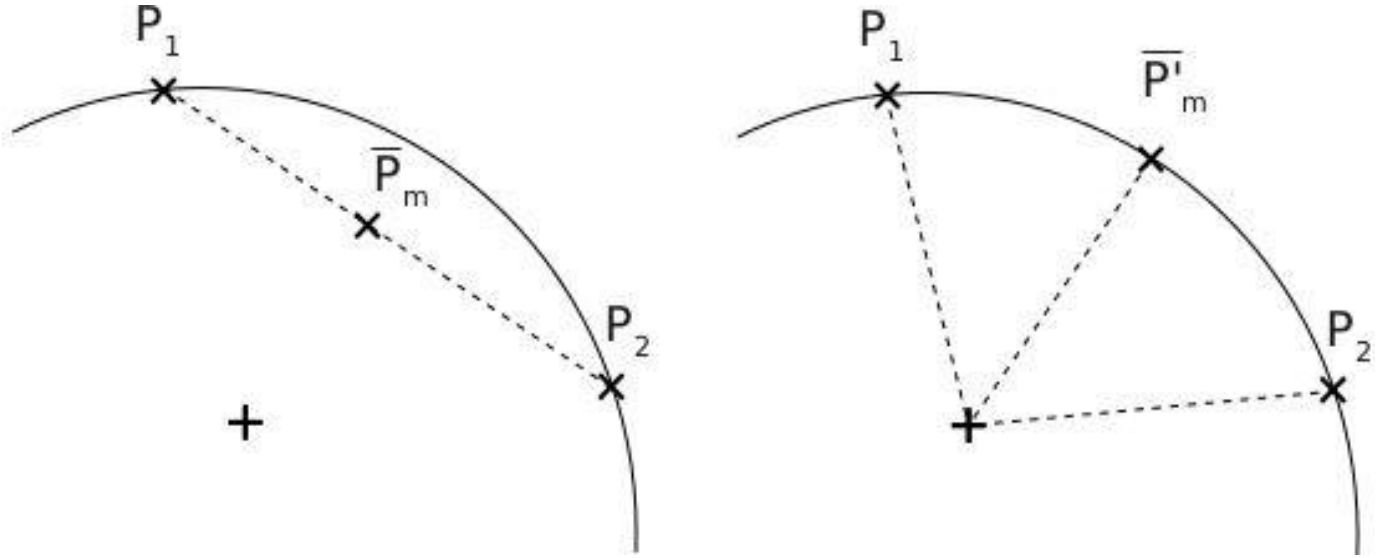
Linear Blend Skinning: Issues

- Issues with the way the interpolation is done
- Remember the equation of Linear Blend Skinning

$$\mathbf{v} = \sum_{i=1}^{n} w_i \mathbf{M'_i}\,\mathbf{M_i^{-1}}\mathbf{v_g} = (\sum_{i=1}^{n} w_i \mathbf{T_i})\mathbf{v_g} \qquad \sum_{i=1}^{n} w_i = 1$$

- Element-wise interpolation of rigid transformation does not result in a rigid body transformation

Dual Quaternion Skinning



Dual Quaternion Skinning

- Instead of using matrices to express the motions of the joints here we use **Dual Quaternions**.
- Unit dual quaternions can represent rigid transformations
- Dual quaternions can be used to linearly blend rigid transformations

LBS

DQBS

Quaternion

- Mathematical object developed by Sir William Rowan Hamilton in 1843 as an extension to the complex number
- Can be written as $\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ where w, x, y, z are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are the quaternion units that satisfies $\mathbf{i^2} = \mathbf{j^2} = \mathbf{k^2} = \mathbf{ijk} = -1$

Quaternion

- They can also be written in a form of a tuple of four real numbers: (w, x, y, z)
- Conjugate: $\mathbf{q}^* = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}$
  where $\mathbf{q}\mathbf{q}^* = w^2 + x^2 + y^2 + z^2$

Quaternion

- A unit quaternion represents a rotation

$$\mathbf{q} = (\cos\frac{\theta}{2}, \mathbf{n_x}\sin\frac{\theta}{2}, \mathbf{n}_y\sin\frac{\theta}{2}, \mathbf{n_z}\sin\frac{\theta}{2})$$

$\mathbf{q}$ represents a rotation of θ around axis $\mathbf{n}$

- Or can be written in a form like:

$$\mathbf{q} = \cos\frac{\theta}{2} + \mathbf{n}\sin\frac{\theta}{2}$$

Rotating a Vector by Quaternion

- Let us think of rotating a vector (v_x, v_y, v_z)
- We define a quaternion

$$\mathbf{v} = 0 + v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$$

$\mathbf{q}$ is a unit

Rotating the vector (v_x, v_y, v_z) can be done by the following operation:

$$\mathbf{qvq}^*$$

Dual Numbers

- Dual numbers are composed of the non-dual part and dual part

$$\hat{a} = a_0 + \varepsilon a_\varepsilon$$

where $\boldsymbol{\varepsilon}$ is a dual unit that satisfies $\boldsymbol{\varepsilon}^2 = 0$

- Dual conjugate

$$\bar{\hat{a}} = a_0 - \varepsilon a_\varepsilon$$

Dual Quaternion

- Dual quaternions are quaternions whose components are dual numbers
- A dual quaternion can be written as

  $$\hat{\mathbf{q}} = \hat{w} + \hat{x}\mathbf{i} + \hat{y}\mathbf{j} + \hat{z}\mathbf{k}$$

  where $(\hat{w}, \hat{x}, \hat{y}, \hat{z})$ are dual numbers

- Dual quaternions can also be represented as 8 real numbers or as the sum of two ordinary quaternions
- $$\hat{\mathbf{q}} = \mathbf{q_0} + \boldsymbol{\varepsilon}\mathbf{q_\varepsilon}$$

Dual Quaternion: Rotation and Translation

- A unit dual quaternion represents transformation with a rotation and a translation.
- The non-dual part represents a rotation
- The dual part represents the translation

$$\hat{\mathbf{q}} = \mathbf{q_0} + \boldsymbol{\varepsilon}\mathbf{q_\varepsilon}$$

Rotation translation

Dual Quaternion: Rotation and Translation

- A unit dual quaternion represents a rotation when the dual part $\mathbf{q}_{\boldsymbol{\varepsilon}}$ is zero

$$\mathbf{q} = \cos\frac{\theta}{2} + \mathbf{n}\sin\frac{\theta}{2}$$

- A dual quaternion representing a translation of (t_0, t_1, t_2) can be written as

$$\hat{\mathbf{t}} = 1 + \frac{\boldsymbol{\varepsilon}}{2}(t_0\mathbf{i} + t_1\mathbf{j} + t_2\mathbf{k})$$

- If we have a 3D point $\mathbf{v} = (v_0, v_1, v_2)$ we define the associated unit dual quaternion as

$$\hat{\mathbf{v}} = 1 + \boldsymbol{\varepsilon}(v_0\mathbf{i} + v_1\mathbf{j} + v_2\mathbf{k})$$

- Applying the quaternion q to a vertex **v**

$$\mathbf{q}\,\hat{\mathbf{v}}\mathbf{q}^{*} = \mathbf{q}(1 + \boldsymbol{\varepsilon}(v_0\mathbf{i} + v_1\mathbf{j} + v_2\mathbf{k}))\,\mathbf{q}^{*}$$
$$= 1 + \boldsymbol{\varepsilon}\mathbf{q}(v_0\mathbf{i} + v_1\mathbf{j} + v_2\mathbf{k})\,\mathbf{q}^{*}$$

  - Rotating a vector by the quaternion **q**

Translation by dual quaternion

$$\hat{\mathbf{t}}\mathbf{v}\overline{\hat{\mathbf{t}}^*}$$

$$\hat{\mathbf{t}} = 1 + \frac{\boldsymbol{\varepsilon}}{2}(t_0\mathbf{i} + t_1\mathbf{j} + t_2\mathbf{k})$$

$$\hat{\mathbf{t}}\mathbf{v}\overline{\hat{\mathbf{t}}^*} = \hat{\mathbf{t}}\big(1 + \boldsymbol{\varepsilon}(v_0\mathbf{i} + v_1\mathbf{j} + v_2\mathbf{k})\big)\left(1 + \frac{\boldsymbol{\varepsilon}}{2}(t_0\mathbf{i} + t_1\mathbf{j} + t_2\mathbf{k})\right)$$

$$= \hat{\mathbf{t}}\left(1 + \boldsymbol{\varepsilon}\left((v_0 + \frac{1}{2}t_0)\mathbf{i} + (v_1 + \frac{1}{2}t_1)\mathbf{j} + (v_2 + \frac{1}{2}t_2)\mathbf{k}\right)\right)$$

$$= \quad 1 + \boldsymbol{\varepsilon}((v_0 + t_0)\mathbf{i} + (v_1 + t_1)\mathbf{j} + (v_2 + t_2)\mathbf{k})$$

Rigid transformation by dual quaternion

$$\hat{\mathbf{t}}(\mathbf{q}\hat{\mathbf{v}}\overline{\mathbf{q}^*})\overline{\hat{\mathbf{t}}^*} = (\hat{\mathbf{t}}\,\mathbf{q})\hat{\mathbf{v}}\left(\overline{\mathbf{q}^*}\,\overline{\hat{\mathbf{t}}^*}\right) = (\hat{\mathbf{t}}\,\mathbf{q})\hat{\mathbf{v}}\overline{(\hat{\mathbf{t}}\,\mathbf{q})^*}$$

Actually, $\hat{\mathbf{t}}\,\mathbf{q}$ is a dual quaternion that performs a rigid transformation

$$\hat{\mathbf{t}}\,\mathbf{q} = \left(1 + \frac{\boldsymbol{\varepsilon}}{2}(t_0\mathbf{i} + t_1\mathbf{j} + t_2\mathbf{k})\right)\mathbf{q}$$

$$= \left(\mathbf{q} + \frac{\boldsymbol{\varepsilon}}{2}(t_0\mathbf{i} + t_1\mathbf{j} + t_2\mathbf{k})\right)\mathbf{q}$$

- Linear blend skinning blends the matrices by the blending weights

$$\mathbf{T} = \sum_{i=1}^{n} w_i \mathbf{T_i}$$

- Dual quaternion skinning blends the dual quaternion of each bone by the blending weights followed by a normalization

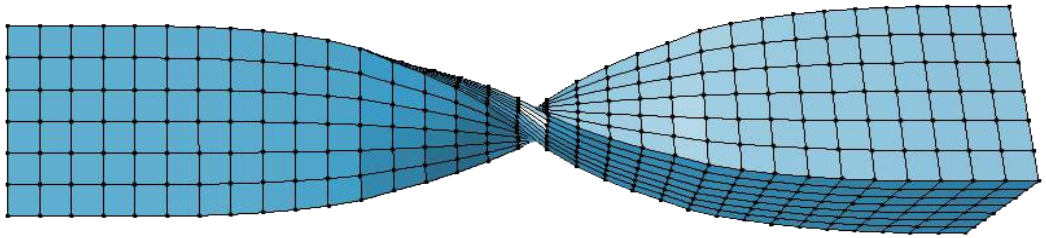
$$\frac{w_1 \hat{\mathbf{q}}_1 + \cdots + w_n \hat{\mathbf{q}}_n}{\| w_1 \hat{\mathbf{q}}_1 + \cdots + w_n \hat{\mathbf{q}}_n \|}$$

Procedure

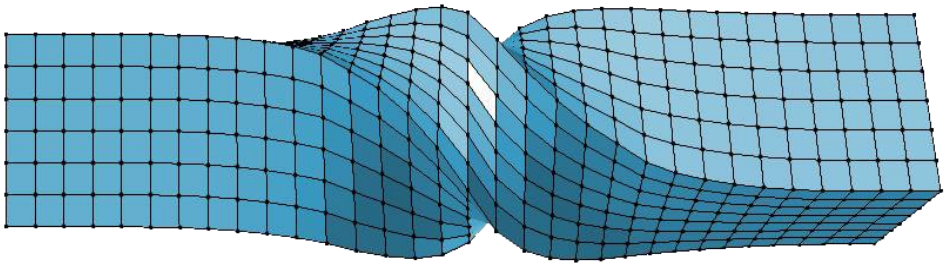
1. Computing the matrices of each bone by FK
2. Convert each matrix to dual quaternion
3. For each vertex compute the weighted dual quaternion by $$\hat{\mathbf{q}} = \frac{w_1\hat{\mathbf{q}}_1 + \cdots + w_n\hat{\mathbf{q}}_n}{\|w_1\hat{\mathbf{q}}_1 + \cdots + w_n\hat{\mathbf{q}}_n\|}$$
4. Compute the global position of the vertex by $$\hat{\mathbf{q}}\hat{\mathbf{v}}\overline{\hat{\mathbf{q}}^*}$$

- Joint collapse and candy wrap can be avoided using dual quaternion skinning.

https://www.youtube.com/watch?v=4e_ToPH-I5o



Linear Blending Skinning



Dual Quaternion Skinning

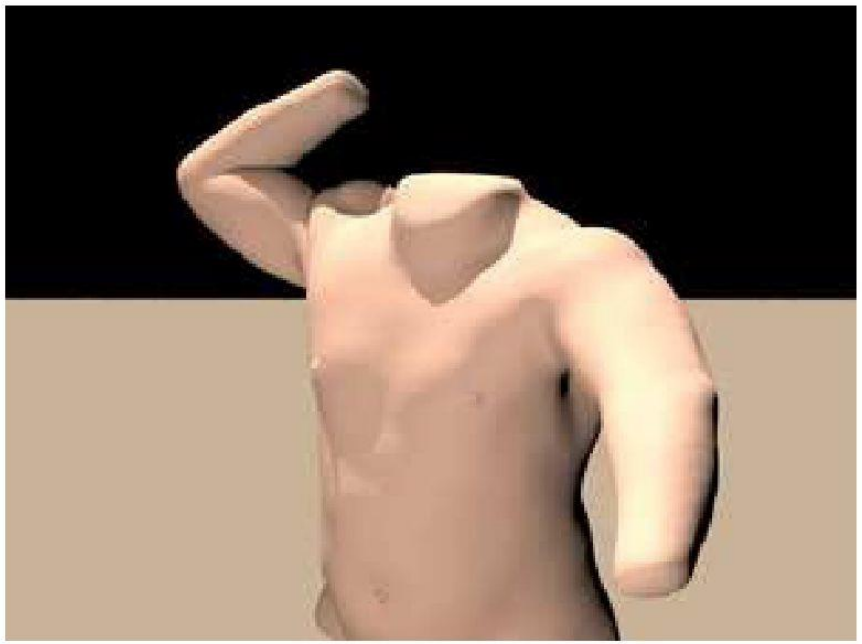
Skinning with Dual Quaternions

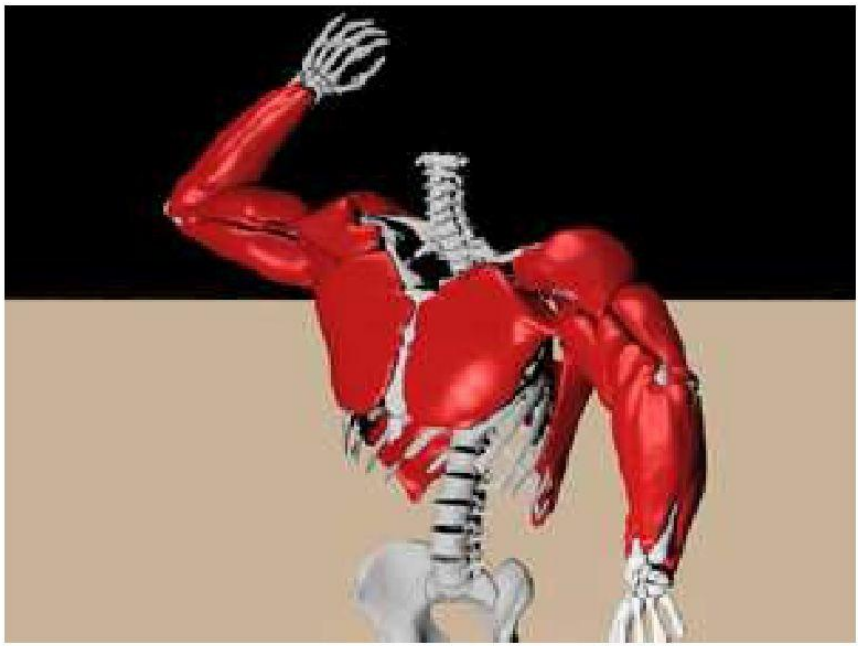
L. Kavan, S. Collins, J. Zara, C. O'Sullivan

Trinity College Dublin
Czech Technical University in Prague

Anatomical models

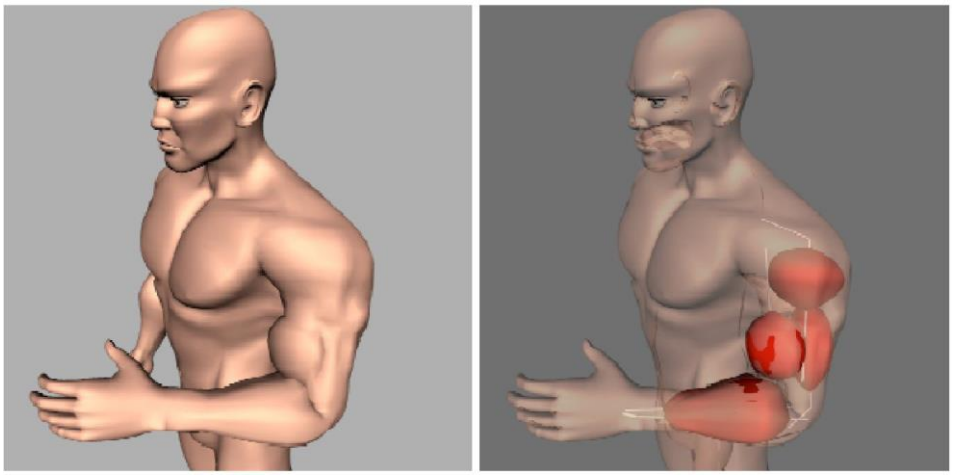
- Model the body by
  - Muscles
  - Fat
  - Skin





Method

1. When the joints are bent, the muscles contract
2. The distance between the origin and insertion point decreases
3. The volume of the muscles are kept the same, so they pump up
4. The skin is deformed to cover the muscles



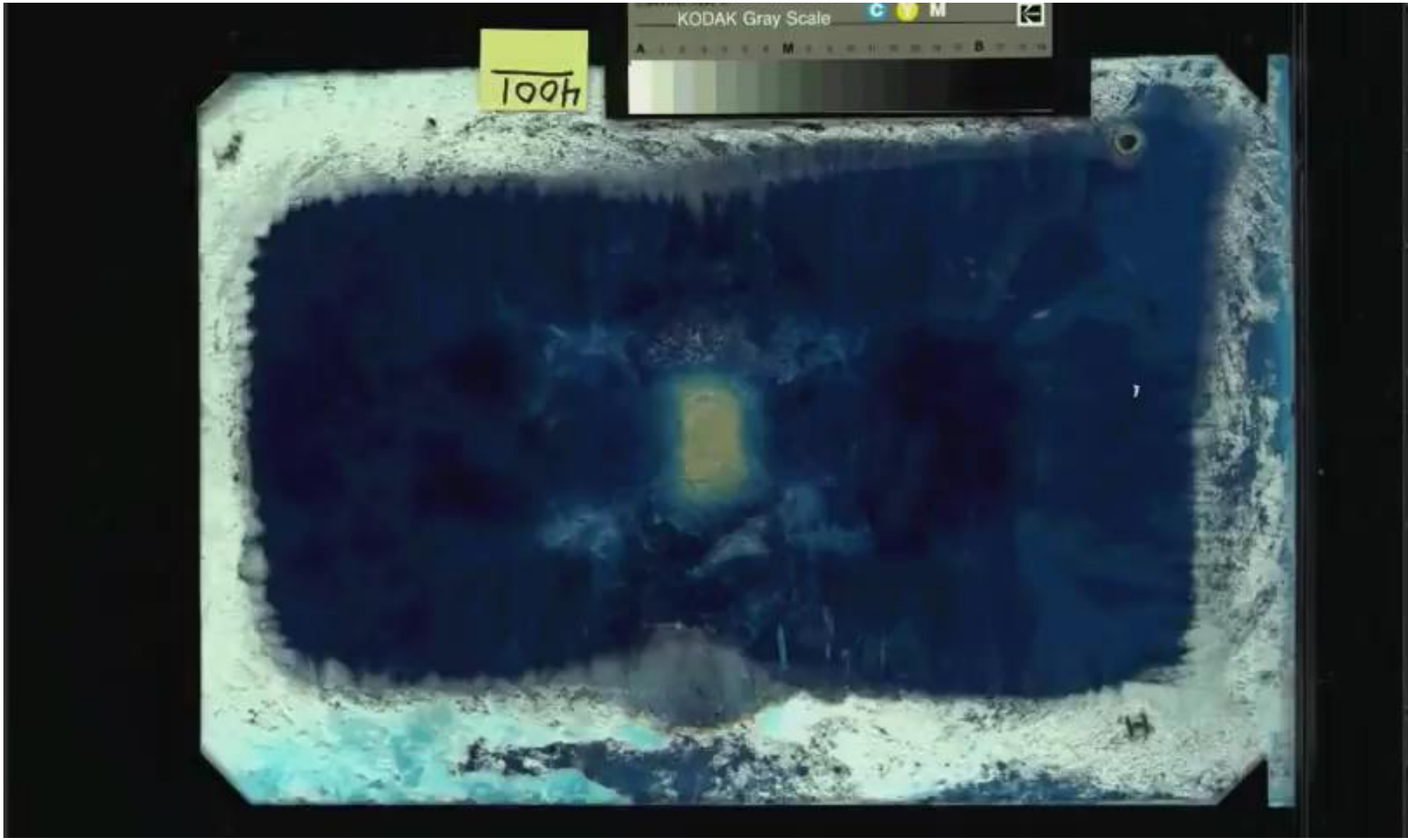
Visible Human Project

Many anatomical models are based the **Visible Human Project** dataset

Two cadavers sliced at 1milimeter intervals from the top to the bottom and photographed by cameras

The CT scans and MRI images were also taken

http://www.youtube.com/watch?v=iWP2HnPSMyo


KODAK Gray Scale
C
Y
M
A
B
100H

Summary

- Skinning
  - Linear Blending
  - Dual Quaternion Skinning
  - Anatomical models

Readings

Skinning

- **A Comparison of Linear Skinning Techniques for Character Animation Afrigraph 2007**
- **Tutorial for Dual Quaternion Skinning** http://rodolphe-vaillant.fr/?e=29
- Lecture notes of COMP3271 Comptuer Graphics
- Geometric Skinning with Approximate Dual Quaternion Blending, SIGGRAPH 2008
- Visible Human project http://www.nlm.nih.gov/research/visible/visible_human.html