Skeleton-based deformations
Why?
Why?
Where?

Halo 3

Crowd simulation

Paths & physics

Bolt

modelling

Kinect

Leap motion
What is a skeleton?
How to define what is its transformation?
How to transfer its deformation to the mesh?
How to manipulate it easily?
...
Skeleton structure
Linear blend skinning
Linear blend skinning

We need to define the influence of the bones onto the mesh vertices
Linear blend skinning
Linear blend skinning

$W_{i,2}$
Linear blend skinning

\[ f: v_i \rightarrow \sum_j w_{ij}(R_j \cdot v_i + T_j) \]

- **Vertex** \( v_i \)
- **Weight of bone** \( j \) over vertex \( i \)
- **Deformation function** \( f \)
- **Bones influencing vertex** \( i \)
- **Rotation of bone** \( j \)
- **Translation of bone** \( j \)
Skinning weights properties

\[ f : v_i \rightarrow \sum_j w_{ij}(R_j \cdot v_i + T_j) \]

- Positivity \( w_{ij} \geq 0 \)
- Affinity \( \sum_j w_{ij} = 1 \)

Why?
Skinning weights properties

\[ f : v_i \rightarrow \sum_j w_{ij}(R_j \cdot v_i + T_j) \]

- Sparsity: only a few \( w_{ij} > 0 \)

Why?
Skinning in modelling tools

- Blender
- Maya
- 3DSMax

DEMO
LBS alternatives

LBS: \[ f: v_i \rightarrow \sum_j w_{ij} (R_j \cdot v_i + T_j) \]

Alternatives:
- Dual quaternion skinning (DQS)
- Spline skinning
- Differential blending

180 degrees → « candy wrapper » effect
Blending transformations

\[ f : v_i \rightarrow \sum_j w_{ij}(R_j \cdot v_i + T_j) \]

Blend « the transformed vertices »
Blending transformations

\[ f: v_i \rightarrow (\sum_j w_{ij} R_j).v_i + (\sum_j w_{ij} T_j) \]

Blend « the transformations »

\[
\begin{align*}
\frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}
\end{align*}
\]

Rotation by 0  \quad Rotation by \pi  \quad Not a rotation
Blending transformations

\[ f : v_i \rightarrow (\sum_j w_{ij} R_j) \cdot v_i + (\sum_j w_{ij} T_j) \]

Blend « the transformations »

\[
\frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \leftrightarrow \sqrt{2}/2 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}
\]

Rotation by 0
Rotation by \( \pi \)
Rotation by \( \pi/2 \)
Dual quaternion Skinning

[Kavan et al.] : Skinning with Dual Quaternions
Dual quaternion Skinning

LBS

DQS

« candy-wrapper »

« bulge »
Disney’s CoRs
Key ideas

- LBS and DQS have « orthogonal » problems
- DQS is good at blending the rotations
- The bulge effect is due to a non-optimized translation (or a non-optimized center of rotation)
What is a good CoR?

- Vertices with similar weights will have a similar rotation.
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
What is a good CoR?

- Vertices with similar weights will have a similar rotation.
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
- Requires a similarity function between weights

\[ s(w_1, w_2) = 1 \quad s(w_1, w_3) = 0.01 \quad s(w_1, w_4) = 0 \]
What is a good CoR?

- Vertices with similar weights will have a similar rotation.
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
- Requires a similarity function between weights.

\[
s(w_p, w_v) = \sum_{\forall j \neq k} w_p^j w_p^k w_v^j w_v^k e^{-\frac{(w_p^j w_v^k w_p^k w_v^j)^2}{\sigma^2}}
\]
Idea

• Consider the LBS transformation of the mesh
• Use the DQS rotation for each vertex
• Optimize per-vertex translation to fit the LBS deformation while enforcing rigid sections.
Idea

\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|_2^2 \, dv \]

where: \[ \tilde{v} = \sum_{j=1}^{m} w_{pj} (R_{pj} v + t_{pj}) \]
Idea

\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|_2^2 \, dv \]

where: \[ \tilde{v} = \sum_{j=1}^{m} w_{pj} (R_j v + t_j) \]
Idea

\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|^2 dv \]

where: \[ \tilde{v} = \sum_{j=1}^{m} w_{pj} (R_{j} v + t_{j}) \]
Computation

\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|_2^2 \, dv \]

where: \[ \tilde{v} = \sum_{j=1}^{m} w_{pj} (R_j v + t_j) \]
Computation

\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|_2^2 \, dv \]

where: \[ \tilde{v} = \sum_{j=1}^{m} w_{p_j} (R_j v + t_j) \]

\[ t_p = \frac{\int_{v \in \Omega} s(w_p, w_v) (\tilde{v} - R_p v) \, dv}{\int_{v \in \Omega} s(w_p, w_v) \, dv} \]
\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|_2^2 \, dv \]

where: \[ \tilde{v} = \sum_{j=1}^{m} w_{pj} (R_j v + t_j) \]

\[ t_p = \frac{\int_{v \in \Omega} s(w_p, w_v) (\tilde{v} - R_p v) \, dv}{\int_{v \in \Omega} s(w_p, w_v) \, dv} \]

\[ = \sum_{j=1}^{m} w_{pj} \left( R_j \frac{\int_{v \in \Omega} s(w_p, w_v) v \, dv}{\int_{v \in \Omega} s(w_p, w_v) \, dv} + t_j \right) - R_p \frac{\int_{v \in \Omega} s(w_p, w_v) v \, dv}{\int_{v \in \Omega} s(w_p, w_v) \, dv} \]
Computation

\[ t_p = \arg \min_t \int_{v \in \Omega} s(w_p, w_v) \| R_p v + t - \tilde{v} \|^2 dv \]

where: \( \tilde{v} = \sum_{j=1}^{m} w_{pj} (R_j v + t_j) \)

\[ t_p = \frac{\int_{v \in \Omega} s(w_p, w_v) (\tilde{v} - R_p v) dv}{\int_{v \in \Omega} s(w_p, w_v) dv} \]

\[ = \sum_{j=1}^{m} w_{pj} \left( R_j \frac{\int_{v \in \Omega} s(w_p, w_v) v dv}{\int_{v \in \Omega} s(w_p, w_v) dv} + t_j \right) - R_p \frac{\int_{v \in \Omega} s(w_p, w_v) v dv}{\int_{v \in \Omega} s(w_p, w_v) dv} \]

\[ = \sum_{j=1}^{m} w_{pj} (R_j p^* + t_j) - R_p p^* \]

where: \( p^* = \frac{\int_{v \in \Omega} s(w_p, w_v) v dv}{\int_{v \in \Omega} s(w_p, w_v) dv} \)
Algorithm 1 Skeletal Skinning with Optimized Centers of Rotation

Input: $n$ vertices, vertex $i$ includes:
- Rest pose position $\mathbf{v}_i \in \mathbb{R}^3$
- Skinning weights $\mathbf{w}_i \in \mathbb{R}^m$
- CoR $\mathbf{p}_i^* \in \mathbb{R}^3$ computed by Eq. (1) and Eq. (4)
- $m$ bones, bone $j$ transformation is $[\mathbf{R}_j \ \mathbf{t}_j] \in \mathbb{R}^{3 \times 4}$

Output: Deformed position $\mathbf{v}_i' \in \mathbb{R}^3$ for all vertices $i = 1..n$

1: for each bone $j$ do
2:   Convert rotation matrix $\mathbf{R}_j$ to unit quaternion $\mathbf{q}_j$
3:   end for
4: for each vertex $i$ do
5:   $\mathbf{q} \leftarrow w_{i1} \mathbf{q}_1 \oplus w_{i2} \mathbf{q}_2 \oplus \cdots \oplus w_{im} \mathbf{q}_m$

   where: $\mathbf{q}_a \oplus \mathbf{q}_b = \begin{cases} 
\mathbf{q}_a + \mathbf{q}_b & \text{if } \mathbf{q}_a \cdot \mathbf{q}_b \geq 0 \\
\mathbf{q}_a - \mathbf{q}_b & \text{if } \mathbf{q}_a \cdot \mathbf{q}_b < 0
\end{cases}$

   ($\mathbf{q}_a \cdot \mathbf{q}_b$ denotes the vector dot product)
6:   Normalize and convert $\mathbf{q}$ to rotation matrix $\mathbf{R}$
7:   LBS: $[\mathbf{\tilde{R}} \ \mathbf{\tilde{t}}] \leftarrow \sum_{j=1}^{m} w_{ij} [\mathbf{R}_j \ \mathbf{t}_j]$
8:   Compute translation: $\mathbf{t} \leftarrow \mathbf{\tilde{R}} \mathbf{p}_i^* + \mathbf{\tilde{t}} - \mathbf{R}_i \mathbf{p}_i^*$ (Eq. (3b))
9:   $\mathbf{v}_i' \leftarrow \mathbf{R} \mathbf{v}_i + \mathbf{t}$
10: end for

$$p^* = \int_{v \in \Omega} \frac{s(\mathbf{w}_p, \mathbf{w}_v) \mathbf{v} \ dv}{\int_{v \in \Omega} s(\mathbf{w}_p, \mathbf{w}_v) \ dv}$$
CoRs

\[ p^* = \frac{\int_{v \in \Omega} s(w_p, w_v) v \, dv}{\int_{v \in \Omega} s(w_p, w_v) \, dv} \]
Results

LMS
(Log-matrix Skinning)
[Alexa 2002, M-Thalmann et al. 2004]

SBS
(Spherical Blend Skinning)
[Kavan and Zara 2005]

LBS
(Linear Blend Skinning)
[M-Thalmann et al. 1988]

DQS
(Dual Quaternion Skinning)
[Kavan et al. 2008]

Our Method
Results
Results
Results

[Images showing different methods (LMS, SBS, LBS, DQS) and a comparison with an optimized method]
Finally: LBS with Complex bones

Before: \[ f : v_i \mapsto \sum_{j \in B(i)} w_{ij} (R_j \cdot v_i + T_j) \]

Now: \[ f : v_i \mapsto \sum_{j \in B(i)} w_{ij} (R_j(v_i) \cdot v_i + T_j(v_i)) \]
Automatic weights computation methods

• Input:
  - Mesh
  - Skeleton

• Output:
  - Skinning weights for each mesh vertex
HeatBones

- Rather simple
- Very fast
- Lightweight implementation

[Baran & Popovic] : Automatic rigging and animation of 3d characters
HeatBones : principle

\[- \Delta w_j + H w_j = H \chi_j\]

Laplacian matrix \hspace{1cm} Stiffness diagonal matrix \hspace{1cm} Voronoi indicative function

\[H_{jj} = \frac{c}{d(j)^2}\]

Solve a linear equation, for each bone j.

Positivity and affinity naturally fulfilled.

Intersections with kd-tree.
HeatBones: principle

What the algorithm does is simple in spirit: it takes the Voronoi indicative functions, and it blurs them.
BoneGlow : variant of HeatBones

\[
- \Delta w_j + H w_j = H \chi_j
\]

BoneGlow: variant of HeatBones

Bone Heat

Bone Glow
BoneGlow: variant of HeatBones

\[-\Delta w_j + H w_j = H \chi_j\]

Replace the binary Voronoi indicative function by a softer bone visibility test.

(a) Bone Heat

(b) Bone Glow
Automatic weights (2)
Bounded Biharmonic Weights (BBW)

- Rather simple
- Rather slow
- Difficult to implement if positivity constraints are enforced

[Jacobson et al. 2011]: Bounded Biharmonic Weights for Real-Time Deformation
BBW : principle

\[
\arg \min_{w_j, \ j=1, \ldots, m} \sum_{j=1}^{m} \frac{1}{2} \int_{\Omega} \| \Delta w_j \|^2 dV
\]

subject to: 
\[
w_j \big|_{H_k} = \delta_{jk}
\]

\[
\sum_{j=1}^{m} w_j(p) = 1
\]

\[
0 \leq w_j(p) \leq 1
\]

Minimize the bi-Laplacian on a tetrahedral mesh, with linear inequalities. → slow
Automatic methods
Inverse kinematics
**Cyclic-Coordinate Descent**

- Starting with the root of our effector, R, to our current endpoint, E.
- Next, we draw a vector from R to our desired endpoint, D.
- The inverse cosine of the dot product gives us the angle between the vectors: \( \cos(a) = \vec{RD} \cdot \vec{RE} \)

*Fig3: Cyclic-Coordinate Descent (CCD)*
One type of IK Solutions

Cyclic-Coordinate Descent
Rotate our link so that RE falls on RD

Fig3: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

Cyclic-Coordinate Descent
Move one link up the chain, and repeat the process

Fig3: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

**Cyclic-Coordinate Descent**

The process is basically repeated until the root joint is reached. Then the process begins all over again starting with the end effector, and will continue until we are close enough to D for an acceptable solution.

---

**Fig 5: Cyclic-Coordinate Descent (CCD)**
One type of IK Solutions

Cyclic-Coordinate Descent

Fig3: Cyclic-Coordinate Descent (CCD)
Cyclic-Coordinate Descent

We’ve reached the root. Repeat the process

Fig 3: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

Cyclic-Coordinate Descent

Fig3: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

Cyclic-Coordinate Descent

Fig 3: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

Cyclic-Coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

Cyclic-Coordinate Descent
One type of IK Solutions

Cyclic-Coordinate Descent

Fig5: Cyclic-Coordinate Descent (CCD)
One type of IK Solutions

**Cyclic-Coordinate Descent**

We’ve reached the root again. Repeat the process until solution reached.

*Fig3: Cyclic-Coordinate Descent (CCD)*
Using IK in Game Development

Examples of IK in action:

- Character Animation Demo (Softimage XSI 5.0, Blender, Maya, everywhere)

- Real-Time calculations: E3 2003 Demo Footage of Half-Life 2
Conclusions

- Skeletons are adapted to character animation
- You find them in games, shape recognition, movies, ...
- The influence of the bones needs to be defined by weights
- Plenty of possibilities for the weights
- Plenty of possibilities for the blending model (LBS, DQS, …)
- Simple structure → advanced deformation mechanisms (IK)
- Plenty of open problems