Skinning of characters with polygonal mesh

Olga Shevchuk and Aline Normoyle

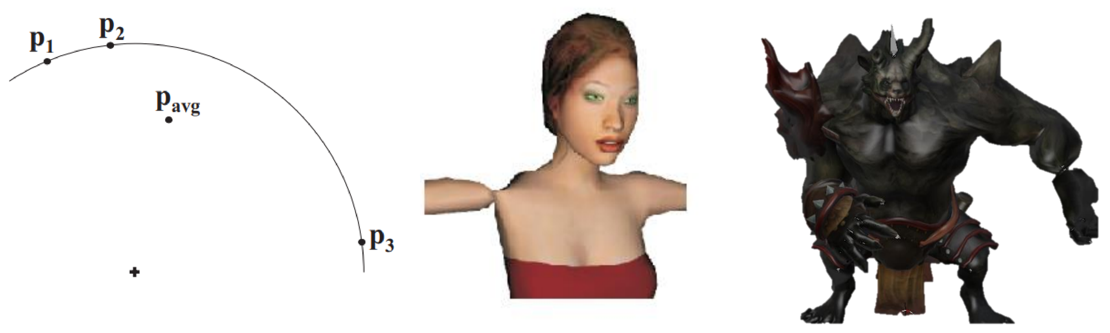
Abstract

Skinning, a process of attaching the mesh of a 3D model to the virtual skeleton, is widely used in a range of applications, such as movie production, computer games and computer aided design. Skinning becomes especially difficult when it comes to certain cases of movement of character joints, and there is a strong tradeoff between the skinning quality, the algorithm speed and implementation simplicity. We will compare two geometric skinning algorithms: the most common one – vertex blending as well as a more recent one – dual quaternion linear blending. We will also review other skinning approaches including some based on physical simulation and data-driven methods providing insight into the advantages and disadvantages of each and their use cases.

Introduction

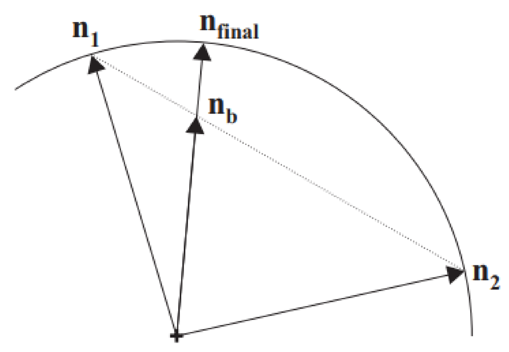
The most common skinning algorithm, called vertex blending or linear blend skinning, is extremely simple to implement, but produces artifacts in certain cases.

Principle: vertices inside a single mesh are transformed by a blend of multiple transforms, represented by matrices.
Problems: Linearly blending the rigid transformation matrices does not in general result in a matrix that represents a rigid body transformation.
Can be compared with the problem of direct averaging point coordinates. Directly multiplying by scalar, summing transformation matrices and multiplying the result by the point we want to transform which is located on the spherical arc might result in a new point which no longer lies on the arc. In extreme cases, the point coincides with the arc's center. This causes "candy-wrapper" artifact, when the skin collapses to a single point.



From left to right. Spherical arc showcasing the problem of averaging point coordinates. "Candy-wrapper" artifact

It is important to find a fast geometric alternative that solves the above artifact. Dual quaternion blending is one of such algorithms, the principle of which is the same as when blending normals (see below). Dual quaternions are blended linearly and the resulting normalized dual quaternion ends up on the arc of the unit sphere. Vertices never deviate to the center of the arc.



Standard normal blending trick: vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ are first blended linearly, giving $\mathbf{n}_b$ and then re-normalized, giving $\mathbf{n}_{final}$.

Besides that, dual quaternions represent the whole rigid transformation, both rotation and translation, the property which solves some of the issues caused by previous skinning approaches. Dual quaternions are graphics hardware friendly, because fewer registers are needed than for rigid transformation matrices.

Linear Blend Skinning

Idea: one vertex (point on the mesh) can be transformed by several different matrices, with the results weighted and blended together. The transform of each bone influences each vertex by a weight defined by the user.
The transformed vertex position $\mathbf{v}'$ is

$$\mathbf{v}' = \sum_{i=1}^{n} w_i T^0_{j_i} \mathbf{v}^{j_i},$$

where the weight w_i is the influence of joint j_i on vertex $\mathbf{v}$, $T^0_{j_i}$ is the transform from the local coordinate system of joint j_i to global coordinates and $\mathbf{v}^{j_i}$ is the position of vertex $\mathbf{v}$ with respect to local coordinate system of j_i

"Candy-wrapper" artifact occurs when we rotate the joint by more than $90°$

Example:

Let j_1 be shoulder joint and j_2 be elbow joint. Two joints have equal influence on the vertex: $w_1 = w_2 = 0.5$. j_1 is at the global origin. j_2 is at (2, 0, 0) with respect to j_1. $\mathbf{v}$ is at (2, 3, 0) with respect to j_1

Goal: animate the arm so that joint j_2 is rotated by 180 degrees around the x-axis.

Joint j_1 is not rotated at all. Thus $T^0_{j_1} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$, where I is the 3×3 identity matrix, which represents no rotation. Then $T^0_{j_1}\mathbf{v}^{j_1} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}$.

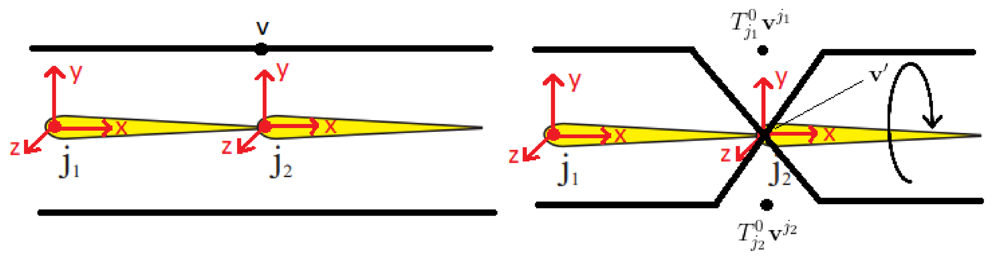
We have $T^0_{j_2} = T^0_{j_1} * T^{j_1}_{j_2} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} R^{j_1}_{j_2} & d^{j_1}_{j_2} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R^{j_1}_{j_2} & d^{j_1}_{j_2} \\ 0 & 1 \end{bmatrix}$, where R and d are are rotation and displacement from j_2 to j_1 respectively.

Then $T^0_{j_2}\mathbf{v}^{j_2} = \begin{bmatrix} R_x(180°) & \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 3 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ -3 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 0 \\ 1 \end{bmatrix}$, where

$$R_x(180°) = \begin{bmatrix} cos(180°) & -sin(180° & 0 \\ sin(180° & cos(180° & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Then if we blend the effects of j_1 and j_2 on $\mathbf{v}$, $\mathbf{v}' = w_1 T^0_{j_1}\mathbf{v}^{j_1} + w_2 T^0_{j_2}\mathbf{v}^{j_2} = \frac{1}{2} * \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} + \frac{1}{2} * \begin{bmatrix} 2 \\ -3 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$

Thus, $\mathbf{v}'$ ends up at the position of j_2 making the skin collapse to a single point (see Figure below).



From left to right. Arm in its rest pose. Arm with an elbow rotated by 180 degrees. $\mathbf{v}'$ ends up at j_2

Dual Quaternion Linear Blending

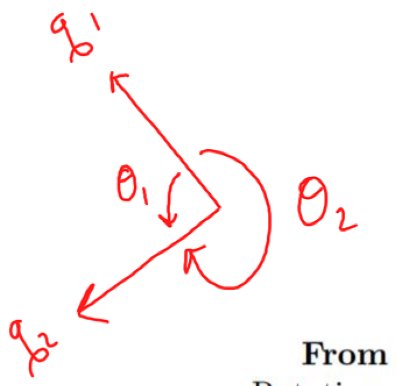
Algorithm

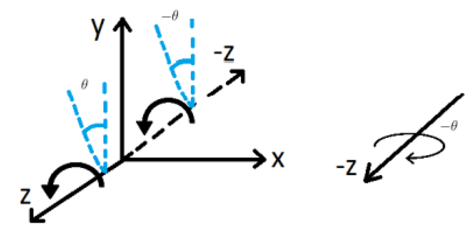
$$DLB(\mathbf{w}; \hat{\mathbf{q}}_1 ... \hat{\mathbf{q}}_n) = \frac{w_1\hat{\mathbf{q}}_1 + ... + w_n\hat{\mathbf{q}}_n}{||w_1\hat{\mathbf{q}}_1 + ... + w_n\hat{\mathbf{q}}_n||}$$, where

$w_1 ... w_n$ are joint weights and $\hat{q}_1 ... \hat{q}_n$ are unit dual quaternions.

Properties:

- Always returns a rigid transformation (DLB computes a unit dual quaternion which can be converted to rigid transformation matrix)
- Avoids the problems with rotation center (the chosen rotation center does not affect the final transformation). This is guaranteed by DLB performing rotation and translation at the same time.
- Interpolates two rigid transformations along the shortest path (to mimic natural skin behavior and avoid excessive stretching). Guaranteed by the antipodality of dual quaternions (a property they inherit from regular quaternions (see figures below)).





From left to right. Rotations by θ and z and $-\theta$ and $-z$ are the same. Rotation around $-z$ looks opposite when viewed from the front.

$\hat{\mathbf{q}}_1 \cdot \hat{\mathbf{q}}_2 \geq 0 \rightarrow \theta_1 < 90°$
$\hat{\mathbf{q}}_1 \cdot \hat{\mathbf{q}}_2 < 0 \rightarrow \theta_2 > 90°$
Since $\hat{\mathbf{q}}_2 = -\hat{\mathbf{q}}_2$, if $\hat{\mathbf{q}}_1 * \hat{\mathbf{q}}_2 < 0$, then $\hat{\mathbf{q}}_1 * (-\hat{\mathbf{q}}_2) \geq 0$

Implementation and speed

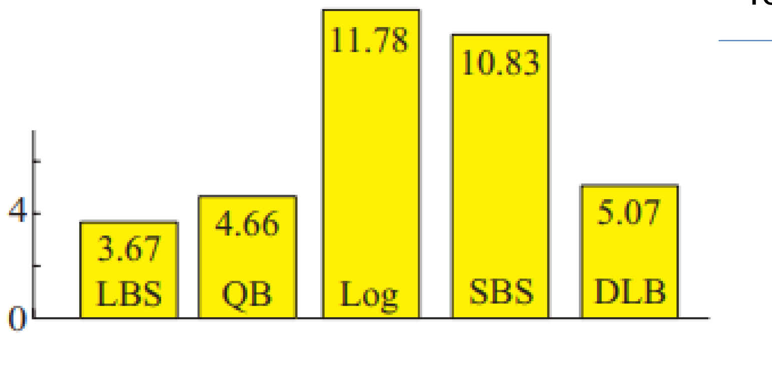
The rotation part of the dual quaternion can be efficiently executed due to the property, according to which regular quaternions cam be expressed efficiently in terms of cross products (shown in definition below). Modern GPUs provide fast cross product operations.

Definition *Let $\boldsymbol{q} = a + \boldsymbol{r}$ be a unit dual quaternion with scalar part a and vector part $\boldsymbol{r}$. Rotation of a vector (v_x, v_y, v_z) represented by the regular quaternion $\mathbf{v} = v_x\boldsymbol{i} + v_y\boldsymbol{j} + v_z\boldsymbol{k}$ can be computed as*

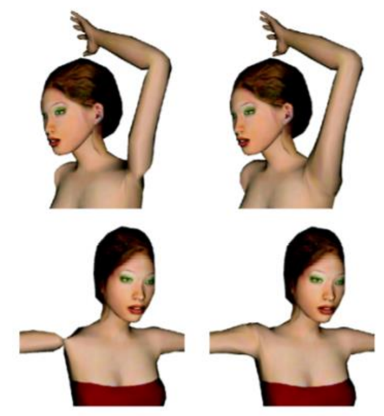
$$\mathbf{v}' = \mathbf{v} + 2\boldsymbol{r} \times (\boldsymbol{r} \times \mathbf{v} + a\mathbf{v}),$$

where $\mathbf{v}'$ is the vector $\mathbf{v}$ rotated by $\boldsymbol{q}$.

Dual quaternion blending is rather efficient as requires only 8 scalars per joint compared to 12 required by linear blend skinning. The average CPU performance of DLB compared to other skinning algorithms is provided in the figure to the right.

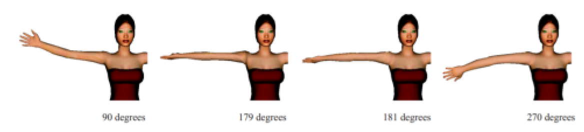


Average CPU performance for skin deformation of the human model in milliseconds (Pentium 4, 3.4GHz): LBS - linear blend skinning, QB - direct quaternion blending, Log - log-matrix blending, SBS - spherical blend skinning, DLB - dual quaternion linear blending. Taken from Kavan et al.



Comparison of linear (left) and dual quaternion blending (right). Dual quaternions preserve rigidity of input transformations and therefore avoid skin collapsing effects. Taken from Kavan et al.

DLB Artifacts



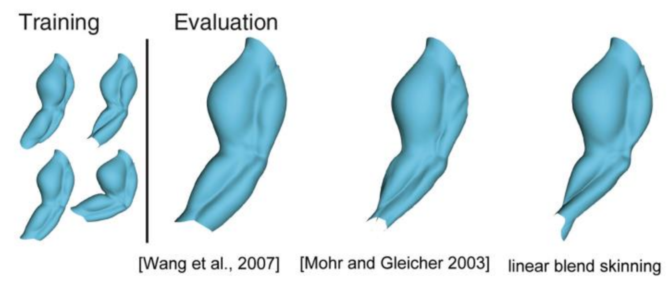
Example of a skin flipping artifact caused by the shortest path property. When rotating from 179 to 181 degrees, skin discontinuously changes its shape, because the other rotation direction becomes shorter. Taken from Kavan et al.

From left to right. Linear blend skinning suffers from volume loss while bending. Dual quaternion skinning successfully eliminates the "candy-wrapper" effect and preserves the volume of the skin, but produces the joint-bulging artefact while bending. Taken from Rumman et al.

Other skinning approaches

Example-based

- **Pose space deformation (PSD):** interpolates vectors among the example poses; example poses are interpolated as a function of a character pose. Pose space is the set of degrees of freedom of character's model. Simple to implement, but require tremendous effort from artists, as they have to create poses by hand for a variety of examples.
- **Single-weight enveloping (SWE) and multi weight enveloping (MWE).** SWE estimates single weight per vertex with rigid character bones, with provisions made for additional bones. MWE is based on a linear framework supporting multiple weights per vertex-bone; provides better approximations than SWE but at the cost of 12 weights per vertex-bone instead of 1 in SWE. Allows a smaller number of poses to be used to generate a larger number of deformations , while introducing more weight parameters.
- **Rotational regression model** which captures common skinning deformation such as muscle bulging and twisting, specifically in challenging regions such as the shoulders. (see figure below)



A set of example poses from an anatomically motivated arm model with both bending and twisting at the elbow. The twisting and muscle bulges are enough to prevent LBS from approximating the examples well. The technique by Mohr does better, but still differ from the given example poses. The model by Wang well-approximate the examples poses.

Physics-based

- Highly enhances the believability and realism of character motions.
- **Mass spring systems** are very simple and efficient. The vertices of the mesh are represented as mass points, governed by Newton's second law of motion, and the edges are elastic massless links (spring). The mesh is deformed when the lengths of the elastic links change.
- These systems suffer from instability and overshooting problems under large timesteps. They might not be accurate since they are strongly topology dependent and are not built based on elasticity theory.
- Too computationally expensive.

Conclusion and Future Directions

We analyzed in detail a most common skinning approach – linear blend skinning and explored the mathematics behind dual quaternion blending. We also investigated other skinning techniques, in particular, example-based and physics-based ones. Future work might involve implementing four skinning approaches: linear blend skinning, dual quaternion blending, spherical blend skinning and log-matrix blending and compare their performance on complex, large meshes. Other future work will explore extensions to skinning that fix artifacts related to self-intersections.