Spherical Blend Skinning: A Real-time Deformation of Articulated Models

Ladislav Kavan* Jiří Žára

Czech Technical University in Prague

Abstract

Skin deformation based on an underlying skeleton is a common method to animate believable organic models. The most widely used skeletal animation algorithm, linear blend skinning, is also known as skeleton subspace deformation, vertex blending, or enveloping. It runs in real-time even on a low-end hardware but it is also notorious for its failures, such as the collapsing-joints artifacts. We present a new algorithm which removes these shortcomings while maintaining almost the same time and memory complexity as the linear blend skinning. Unlike other approaches, our method works with exactly the same input data as the popular linear version. This minimizes the cost of upgrade from linear to spherical blend skinning in many existing applications: the data structures and models need no change at all. The paper discusses also theoretical properties of rotation interpolation, essential to spherical blend skinning.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: skinning, deformation, skeletal animation

1 Introduction

Real-time animation of deformable objects is always a compromise between visual fidelity and computation complexity. Other aspects are quite important as well, for example the amount of artists work necessary to design the model. Therefore, there exist many algorithms for modeling deformable objects in the literature. They differ by the intended area of application and generality of allowed models.

We focus on the real-time animation systems in this paper. Its most popular representative, known generally as the *skeletal animation*, is based on simple but versatile structure. It consists of *joints*, given by their position and orientation. The segments connecting the joints are conveniently interpreted as *bones*. The *skeleton* is, formally speaking, a tree whose nodes are identified with the joints and edges with the bones. The only displayed element is a *skin*, a 3D polygonal mesh, usually equipped with normal and texture data.

Although the terminology is adopted from the virtual humanoid modeling, the skeletal animation is not limited to character animation – it can be applied to a wide range of soft objects, including imaginary (cartoon) creatures, plants, furniture, etc. This is an apparent advantage over complex systems which rely on explicit anatomy.

*e-mail: kavanl1@fel.cvut.cz

The skeleton simplifies the animation task considerably: instead of animating each vertex individually, it is sufficient to manipulate the skeleton, and the skin deforms automatically. The skeletal animation in general does not specify how exactly the skeleton posture should be propagated to the skin. However, there is an established standard used in majority of real-time 3D applications. It comes by many names, all relating to the same algorithm: linear blend skinning (LBS), skeleton subspace deformation, vertex blending, enveloping, or simply skinning. Basically, this algorithm blends between rigidly transformed vertices using vertex weights, which denote the amount of influence of individual joints.

Although LBS is very fast and advantageous to graphics hardware, it suffers from inherent artifacts, known as "collapsing joints", "twisting elbow problem" or a "candy-wrapper artifact". In general, the mesh deformed by LBS loses volume as the joint rotation increases. The cause of this phenomena is explained in section 3, together with the LBS algorithm itself.

The structure of the paper is as follows: in the next section, we summarize the previous work concerning real-time skin deformation and sketch our solution. In section 3, we analyze the problems of the LBS algorithm. Our approach to resolve these problems is presented in section 4. In section 5, we compare the results and discuss possible enhancements.

2 Related Work

An early contribution concerning the animation of deformable objects is [Magnenat-Thalmann et al. 1988], which considers the movement of a human hand. First 3D characters used in numerous computer games were animated by simple, often unpublished algorithms. Later on, the basic principles of LBS were described by the game development community [Lander 1998; Lander 1999]. The artifacts of LBS were discovered soon [Weber 2000]. An improvement based on addition of auxiliary joints has been also proposed in [Weber 2000]. Although this reduces the artifacts, the skin to joints relationship must be re-designed after joint addition. The number and location of the additional joints remains questionable. Another problem is how the movement of the original skeleton should be propagated into the augmented one.

More formal articles consider skin deformation as an interpolation problem, such as [Lewis et al. 2000]. They use radial basis functions to interpolate between example skins with different shapes. Similar method is presented in [Sloan et al. 2001] and [Kry et al. 2002]. The latter de-correlates the deformation displacements using principal component analysis, which reduces the memory requirements considerably. The advantage of example based methods is that they capture the designed shape, including effects like muscle bulging. The drawback is the necessity of acquiring the example skins.

An interesting generalization of LBS is called multi-weight enveloping [Wang and Phillips 2002]. It introduces more parameters and therefore greater flexibility to the deformation algorithm. Instead of one weight per influence (joint) as in LBS, the multi-weight enveloping uses twelve. These numerous parameters are derived from examples using the least squares optimization. The

disadvantage is obvious: while the LBS models can be weighted manually by artists [Steed 2002], this is questionable with multi-weight enveloping. Tools that help animators to design the vertex weights are described in [Mohr et al. 2003]. This article is interesting also from the theoretical point of view, because it describes how to explore the space of all possible LBS deformations.

Another deformation algorithm [Bloomenthal 2002] uses a complex auxiliary structure – a medial. An idea similar to spherical blend skinning (SBS) is bones blending proposed by [Kavan and Žára 2003]. However, bones blending is limited to vertices attached to only two joints. In addition, it requires hand-tuning of special parameters. Another algorithm removes the LBS artifacts by adding additional joints, and computes the vertex weights automatically using examples [Mohr and Gleicher 2003]. A recent skin deformation algorithm presented in [Magnenat-Thalmann et al. 2004] seems to give results competitive to SBS, although it is based on a different mathematical fundament [Alexa 2002]. However, this method is considerably slower than LBS and therefore [Magnenat-Thalmann et al. 2004] recommends to use rather the standard LBS if the joint rotations are small.

To conclude, there are many methods correcting the problems of LBS, but none of them is superior to LBS in all aspects. As a result, the linear blend skinning is still widely used in many applications, in spite of the artifacts.

2.1 Our Contribution

We observed that the artifacts of LBS are caused by the straightforward, linear interpolation of vertex positions. Intuitively, a linear blending is not suitable to capture deformations induced by skeleton, because their nature is rather spherical. Our basic idea is to change the interpolation domain: we interpolate transformations itself instead of transformed vertex positions. Because we consider transformations consisting of a translation and rotation, we suggest to use a quaternion representation.

The transition to non-linear interpolation domain is not elementary. In order to achieve our goal, we cope with two main problems: determination of the center of rotation, and interpolation of multiple quaternions. The first problem follows from the fact that the choice of the center of rotation influences the result of interpolation considerably. We show how to compute a convenient center of rotation in real-time. The second problem is simple in the case of two quaternions [Shoemake 1985], but gets considerably harder for more than two rotations [Buss and Fillmore 2001; Park et al. 2002; Alexa 2002]. Because the previous methods are not efficient enough for our purpose, we use a simple linear quaternion averaging. We justify both theoretically and experimentally that this solution is appropriate for our task (and probably for many others).

Resolving those problems, we obtain a skin animation algorithm that deforms the mesh in much more plausible way then LBS. Because we change only the interpolation domain and not the input data, our program works with exactly the same models as LBS. The proposed algorithm improves a deformed shape even of models that have been designed and carefully tuned for LBS. Considering the high speed and low memory demands of SBS, it provides an attractive alternative to classic LBS.

2.2 Conventions

Let us denote matrices by capital letters, while vectors and quaternions by bold. Vectors are considered column vectors, therefore a multiplication of vector $\mathbf{v}$ by matrix M is written as $M\mathbf{v}$. We do not introduce a different notation for the R^3 vectors and their homogeneous R^4 counterparts with last coordinate equal to 1. The same convention is used for matrices. We denote the dot product of two vectors $\mathbf{v}_1, \mathbf{v}_2$ as $(\mathbf{v}_1, \mathbf{v}_2)$ and the norm $\|\mathbf{v}_1\|$ as a shortcut for $\sqrt{(\mathbf{v}_1, \mathbf{v}_1)}$.

3 Linear Blend Skinning

The input to LBS consists of a polygonal mesh representing the digital skin, a skeleton, and vertex weights for every vertex of the skin. The polygonal mesh and the skeleton are designed in a reference position, e.g. virtual characters are often posed in the da Vinci posture [Steed 2002].

Let us label the joints by integer numbers, assigning zero to the root. Each joint in the reference posture is associated with a homogeneous matrix, describing its position and orientation in the world coordinate system. For j-th joint, we denote this matrix by A_j, like "absolute" (or reference) position. This matrix is computed by multiplying all the transformations of individual joints in the chain from root to joint j. To compute the shape of the deformed skin, we need yet another set of matrices, describing the position and orientation of joints in the actual, animated posture. We call them F_j, standing for the "final" placement of joint j. Matrices F_j are computed in a similar way as the absolute matrices, but including the actual rotation of each joint in the chain (we do not consider translating and scaling joints).

The most simple skin deformation algorithm computes

$$\mathbf{v}' = F_j A_j^{-1} \mathbf{v}$$

where $\mathbf{v}$ is a vertex in the reference skin associated with joint j and $\mathbf{v}'$ is its position in the deformed mesh. The interpretation is following: the first matrix A_j^{-1} transforms $\mathbf{v}$ to the position with joint j's coordinate system aligned to the world coordinate system. The following transformation F_j returns the vertex to its current position induced by the animated skeleton. Because these transformations usually occur together, we define the "complete" matrix $C_j = F_j A_j^{-1}$. Some older computer games animated characters in this way, even though it does not produce nice, smooth deformations.

The linear blend skinning allows assignment of one vertex to multiple bones. Assume that vertex $\mathbf{v}$ is attached to joints $j_1, \ldots, j_n$ with weights $w_1, \ldots, w_n$. The weights are coefficients of a convex combination, i.e. non-negative and $\sum_{i=1}^{n} w_i = 1$. The weight w_i represents the amount of influence of joint j_i. The vertex position in the mesh deformed by LBS is then computed as

$$\mathbf{v}' = \sum_{i=1}^{n} w_i C_{j_i} \mathbf{v} \qquad (1)$$

that is to say, making a convex combination of individual vertex transformations. For example if $n = 2$ then vertex $\mathbf{v}'$ lies on the line segment connecting $C_{j_1}\mathbf{v}$ and $C_{j_2}\mathbf{v}$. The actual position on the segment is given by weight w_1 (or w_2, because $w_1 + w_2 = 1$). As explained in the next section, the SBS works on a circular arc instead of segment, see Figure 1.

If the joint rotations are large, the LBS produces non-natural deformations. In the extremal case of rotation by 180 degrees, the skin can collapse to a single point. It is the notorious "candy-wrapper" artifact, which is demonstrated in Figure 2. The right shoulder of

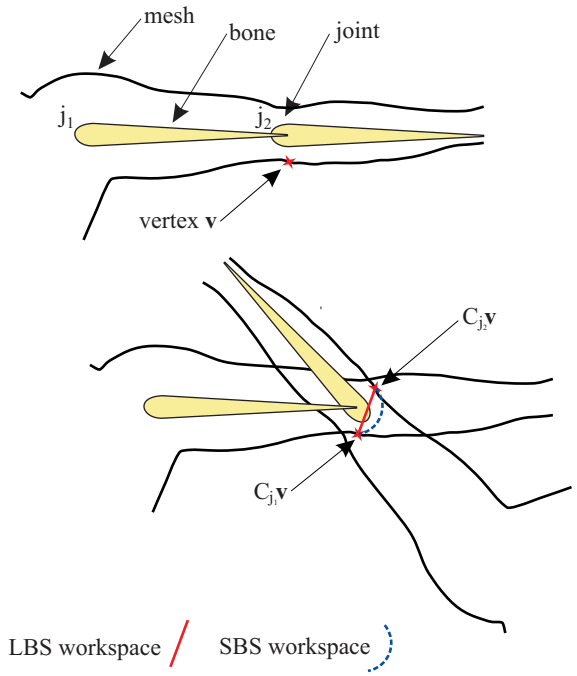

Figure 1: The set of possible results of LBS is a line segment, while SBS gives a circular arc.

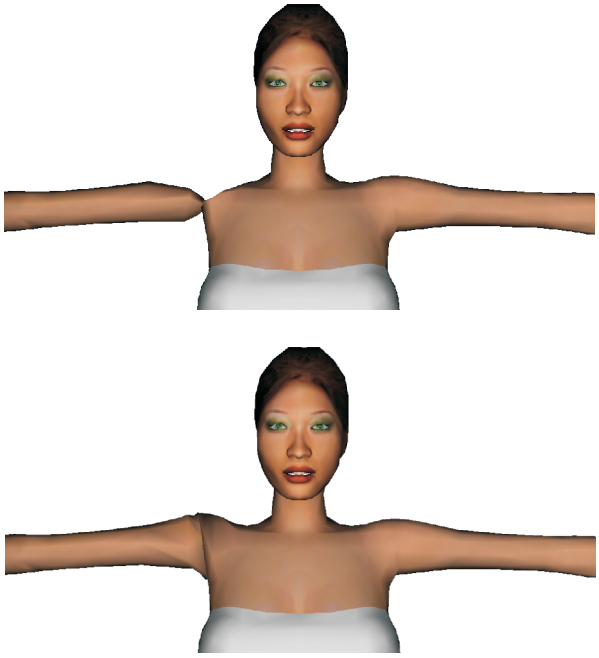

Figure 2: Up – an extreme shoulder twist deformed by LBS, down – the same posture deformed by SBS

the model is twisted by 180 degrees, while the left shoulder is left in the reference pose.

To understand why this undesirable effect occurs, it is sufficient to re-arrange the equation (1)

$$\mathbf{v}' = \left(\sum_{i=1}^{n} w_i C_{j_i} \right) \mathbf{v} \qquad (2)$$

This formula is less efficient, because it blends matrices instead of vectors, but gives us a valuable insight. It is well known that the component-wise interpolation of matrices produces odd results: it does not preserve the orthogonality of the rotational part of the matrix. In some situations, it does not preserve even the rank of the interpolated matrices. This is exactly what happens in the "candy-wrapper" problem: the single point the skin collapses to is a result of transformation by a singular matrix. A similar defect is visible also in the proximity of the singular configuration. Although the matrix is regular, it involves a non-uniform scaling and skewing, which is responsible for the loss of volume of the deformed skin even for small rotations.

4 Spherical Blend Skinning

Instead of trying to correct the bad results of LBS, we propose to change the interpolation method in (2). We focus on the interpolation of rotations – the linear interpolation of the translation part of C_{j_i} matrices is all right. An established interpolation of two rotations is spherical linear interpolation (SLERP) [Shoemake 1985]. Its key of success is the use of quaternions to represent rotations. Unfortunately, it is not possible to simply replace matrices C_{j_i} in (2) with corresponding pairs quaternion-translation. One of the problems is that the linear interpolation of quaternions is not equivalent to SLERP. However, this is not the most serious difficulty, and we address it in section 4.1. The more important problem is to compute a convenient center of the interpolated rotations.

We show that this is really an important problem on an example of human arm. Consider that the arm geometry is influenced by two joints j_1 and j_2, such that j_1 is a parent of j_2, as in Figure 1. The transformation of the whole mesh by C_{j_1} is illustrated in the top row of Figure 3 and the transformation of the same geometry by C_{j_2} in the bottom row (note that the results are identical in both columns of these rows). The rows in the middle show the progress of interpolation between C_{j_1} to C_{j_2}. The only difference between the two columns in Figure 3 is in the choice of the center of rotation. In the left column, the rotation center $\mathbf{r}_c$ is set to the translation part of matrix A_{j_2} (the position of joint j_2 in the reference posture). Note that $C_{j_1}\mathbf{r}_c = C_{j_2}\mathbf{r}_c$, therefore also the transformed rotation center is constant during the interpolation. In the right column of the figure, the rotation center $\bar{\mathbf{r}}_c$ is set to the translation part of A_{j_1}. Because $C_{j_1}\bar{\mathbf{r}}_c \neq C_{j_2}\bar{\mathbf{r}}_c$, the transformed rotation center is linearly interpolated from $C_{j_1}\bar{\mathbf{r}}_c$ to $C_{j_2}\bar{\mathbf{r}}_c$. By comparison with the starting mesh (drawn gray in each frame), it is obvious that the center of rotation choice in the left column is much more advantageous. In this case, the interpolation of every single point is a circular arc (as in Figure 1), whereas a disturbing drift is inherent to any other choice of rotation center (such as $\bar{\mathbf{r}}_c$).

Unfortunately, the condition of zero translation cannot be always satisfied, typically for more than two influencing joints. But even if the vertex is attached to only two joints k and l that are not neighbours of each other, some translation may be inevitable. For example consider that there is no relative rotation between C_k and C_l, but there is a relative translation induced by the joints in the chain between k and l. Clearly no choice of the center of rotation can avoid this translation, because the rotation is identity.

Anyway, it is possible to *define* the rotation center as the point whose transformations by associated matrices are as close as possible. This minimizes the drift and works even if the vertex is assigned to n joints $j_1, \ldots, j_n$. We find the center of rotation $\mathbf{r}_c$ as the

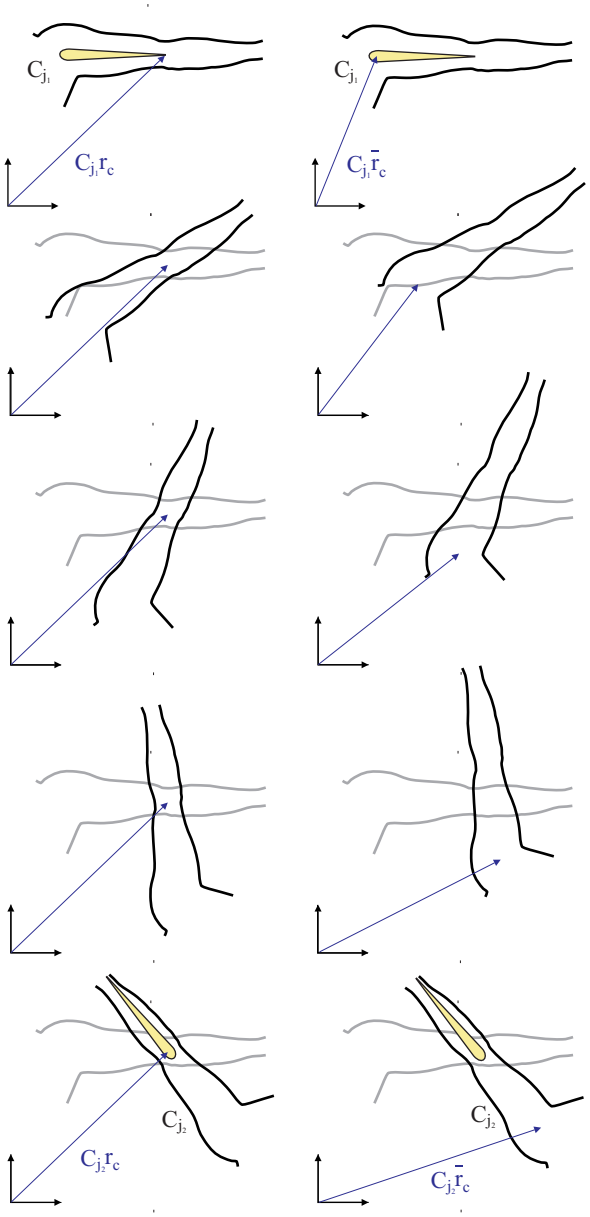

Figure 3: The correct center of rotation is chosen in the left column, while the sub-optimal in the right column. In the middle rows, notice the difference of the elbow position with respect to the original skin.

least-squares solution of the system of $\binom{n}{2}$ linear vector equations

$$C_a\mathbf{r}_c = C_b\mathbf{r}_c,\ a<b,\ \ a,b\in\{j_1,\ldots,j_n\}$$

Each homogeneous matrix C_i has structure

$$C_i = \begin{pmatrix} C_i^{rot} & \mathbf{C}_i^{tr} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

where C_i^{rot} is a 3×3 orthogonal matrix and $\mathbf{C}_i^{tr}$ is a translation vector. This enables us to re-write the linear system to

$$\begin{aligned} C_a^{rot}\mathbf{r}_c + \mathbf{C}_a^{tr} &= C_b^{rot}\mathbf{r}_c + \mathbf{C}_b^{tr} \\ (C_a^{rot} - C_b^{rot})\mathbf{r}_c &= \mathbf{C}_b^{tr} - \mathbf{C}_a^{tr} \end{aligned}$$

If we stack all these equations to one matrix D and the right-hand sides to vector $\mathbf{e}$, we can write the whole system as

$$D\mathbf{r}_c = \mathbf{e}$$

where D is a $3\binom{n}{2}\times 3$ matrix, $\mathbf{r}_c$ is a 3-dimensional unknown vector and $\mathbf{e}$ is $3\binom{n}{2}$-dimensional vector. In general, we cannot make any assumptions about the rank of matrix D, which can vary from 0 to 3 (consider for example $n=2$ and $C_{j_1}=C_{j_2}$). We search the optimal solution $\mathbf{r}_c$ in the least-squares sense. If there are multiple solutions giving the minimal $\|D\mathbf{r}_c-\mathbf{e}\|$, the $\mathbf{r}_c$ with the minimal norm is chosen. This can be done in a robust way using the singular value decomposition (SVD), followed by computation of pseudo-inverse matrix. To perform these computations, we use the LAPACK software [Anderson et al. 1999].

Even though LAPACK routines are efficient, computation of the center of rotation per each vertex would not result in a real-time algorithm. Fortunately, the center of rotation depends only on the transformations of the joints $j_1,\ldots,j_n$ and not the vertex itself. Therefore, if we encounter another vertex assigned to the same set of joints $j_1,\ldots,j_n$, we can re-use the center of rotation computed formerly (cached). Moreover, if there is only one, or two neighboring joints that influence the vertex, we can determine the center of rotation precisely (as indicated in the beginning of this section) and omit the SVD computation at all. It turns out that the number of different non-trivial joint sets, and therefore the number of running the SVD, is surprisingly small for common models – about several tens. This enables the real-time performance.

4.1 Interpolation of Multiple Rotations

As mentioned in the introduction, the interpolation of multiple rotations has already received some attention [Buss and Fillmore 2001; Park et al. 2002] as well as interpolation of multiple general transformations [Alexa 2002]. Unfortunately, all these methods are substantially slower then the simple linear interpolation used in LBS. Since our goal is an algorithm with comparable time complexity as LBS, we propose an approximate but fast linear quaternion blending. For the case of two rotations, we compare our method with the established SLERP.

Recall that a rotation around axis $\mathbf{a}$ (unit length vector) with angle 2α corresponds to quaternion $\mathbf{q}' = \cos\alpha + \mathbf{a}\sin\alpha$. However, this correspondence is not unique, because both quaternions $\mathbf{q}'$ and $-\mathbf{q}'$ represent the same rotation. The SLERP of two unit quaternions $\mathbf{p},\mathbf{q}$ assumes that their dot product $(\mathbf{p},\mathbf{q})\geq 0$. If the dot product $(\mathbf{p},\mathbf{q})<0$, we use $-\mathbf{p}$ instead of $\mathbf{p}$, which is possible because both $\mathbf{p}$ and $-\mathbf{p}$ represent the same rotation. The SLERP of $\mathbf{p},\mathbf{q}$ with interpolation parameter $t\in\langle 0,1\rangle$ is given by the following formula, see for example [Eberly 2001].

$$s(t;\mathbf{p},\mathbf{q}) = \frac{\sin((1-t)\theta)\mathbf{p} + \sin(t\theta)\mathbf{q}}{\sin\theta} \tag{3}$$

where θ is the angle inclined by quaternions $\mathbf{p},\mathbf{q}$, i.e. $\cos\theta = (\mathbf{p},\mathbf{q})$.

The linear interpolation of quaternions (QLERP) is computed as

$$l(t;\mathbf{p},\mathbf{q}) = \frac{(1-t)\mathbf{p}+t\mathbf{q}}{\|(1-t)\mathbf{p}+t\mathbf{q}\|} \tag{4}$$

The difference to SLERP is obvious: QLERP interpolates along the shortest *segment*, and then projects to arc, which does not result in the uniform interpolation of the arc. In spite of this, we claim that QLERP is sufficient for our task. In order to justify this statement, we face an interesting question by itself: how big can be the difference between QLERP and SLERP for the same input rotations?

For $t = 0$, both QLERP and SLERP return of course $\mathbf{p}$. For $t > 0$, we can imagine that both QLERP and SLERP work by concatenating $\mathbf{p}$ with some rotation (multiplying $\mathbf{p}$ with some quaternion). For SLERP, we denote this quaternion as $\mathbf{r}_s(t)$. It can be expressed as $\mathbf{p}^* s(t;\mathbf{p},\mathbf{q})$, because

$$\mathbf{p}\mathbf{r}_s(t) = \mathbf{p}\mathbf{p}^* s(t;\mathbf{p},\mathbf{q}) = s(t;\mathbf{p},\mathbf{q})$$

The rotation $\mathbf{r}_s(t)$ can be written out as

$$\mathbf{r}_s(t) = \mathbf{p}^* s(t;\mathbf{p},\mathbf{q}) = \frac{\sin((1-t)\theta)\mathbf{1} + \sin(t\theta)\mathbf{p}^*\mathbf{q}}{\sin\theta} \qquad (5)$$

The quaternion $\mathbf{1}$ represents the identity (zero angle rotation). From the definition of quaternion multiplication it can be seen that the real part of $\mathbf{p}^*\mathbf{q}$ equals $(\mathbf{p},\mathbf{q}) = \cos\theta$. Since $\mathbf{p}^*\mathbf{q}$ is a unit quaternion, we can express it as

$$\mathbf{p}^*\mathbf{q} = \cos\theta + \mathbf{u}\sin\theta$$

for some axis of rotation $\mathbf{u}$. If we substitute this into equation (5), we obtain

$$\mathbf{r}_s(t) = \frac{\sin((1-t)\theta) + \sin(t\theta)\cos\theta}{\sin\theta} + \mathbf{u}\sin(t\theta)$$

which means that the direction of the axis $\mathbf{u}$ is independent on t.

Let us examine the rotation $\mathbf{r}_l(t)$ following $\mathbf{p}$ in QLERP:

$$\begin{aligned}\mathbf{r}_l(t) &= \mathbf{p}^* l(t;\mathbf{p},\mathbf{q}) = \frac{(1-t)\mathbf{1} + t\mathbf{p}^*\mathbf{q}}{\|(1-t)\mathbf{p} + t\mathbf{q}\|} = \\ &= \frac{(1-t+t\cos\theta)}{\|(1-t)\mathbf{p} + t\mathbf{q}\|} + \mathbf{u}\frac{t\sin\theta}{\|(1-t)\mathbf{p} + t\mathbf{q}\|}\end{aligned}$$

which shows that the axis of rotation has the same direction. We can conclude with an important property: the SLERP can be written as $\mathbf{p}\mathbf{r}_s(t)$ and QLERP as $\mathbf{p}\mathbf{r}_l(t)$, where the rotations $\mathbf{r}_s(t)$ and $\mathbf{r}_l(t)$ have the same axis. Moreover, this axis is constant, i.e. independent on the interpolation parameter t.

It follows that the only difference between QLERP and SLERP is in the angle of rotations $\mathbf{r}_s(t)$ and $\mathbf{r}_l(t)$. Note that both $\mathbf{r}_s(t)$ and $\mathbf{r}_l(t)$ have a form of linear combination of quaternions $\mathbf{1}$ and $\mathbf{p}^*\mathbf{q}$. It means that the results of both $\mathbf{r}_s(t)$ and $\mathbf{r}_l(t)$ always end up in certain 2D subspace of R^4. We can restrict our attention to this subspace (the linear hull of $\mathbf{1}$ and $\mathbf{p}^*\mathbf{q}$).

Since SLERP assumes $\cos\theta = (\mathbf{p},\mathbf{q}) \geq 0$, the angle θ cannot exceed $\pi/2$. To obtain an upper bound of the maximal difference in the angle, we consider the extremal case with $\theta = \pi/2$, depicted in Figure 4.

The angle $\alpha(t)$ on the picture can be computed by *atan*, and $\beta(t)$ by simple linear interpolation of the right angle, which yields the difference function

$$d(t) = \alpha(t) - \beta(t) = atan\left(\frac{t}{1-t}\right) - \frac{\pi}{2}t$$

It remains to find the extremes of $d(t)$ on the interval $\langle 0,1\rangle$. The elementary mathematical analysis discovers the global extremes in points $1/2 \pm \sqrt{(1/\pi - 1/4)}$. The absolute value of $d(t)$ in these points is approximately 0.071 radians (4.07 degrees). As mentioned in the introduction of this section the angle of rotation is twice the angle inclined by quaternions.

To conclude: both SLERP and QLERP interpolate by multiplying the first quaternion with a rotation with the same, fixed axis. The difference between SLERP and QLERP is only in the angle of this rotation, and is strictly less then 0.143 radians (8.15 degrees) for any interpolation parameter $t \in \langle 0,1\rangle$. This is an upper bound; practical results are much smaller and could hardly cause an observable defect in the deformed skin. The big advantage of QLERP is that it can be easily generalized to interpolate multiple rotations – it suffices to make a convex combination and re-normalization of multiple quaternions.

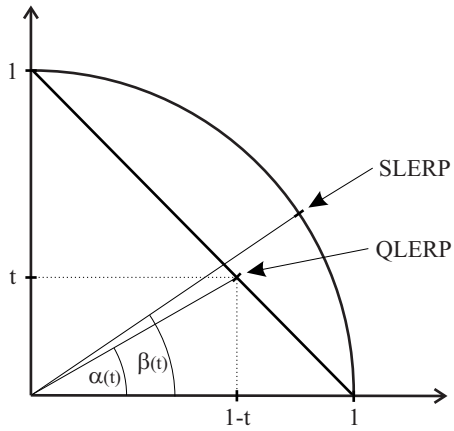

Figure 4: The difference between QLERP angle $\alpha(t)$ and SLERP $\beta(t)$

4.2 Algorithm Overview

Now we have prepared all the ingredients to describe how the SBS algorithm works. The task is to transform a vertex $\mathbf{v}$ influenced by joints $j_1,\ldots,j_n$ with convex weights $W = (w_1,\ldots,w_n)$ to its position $\mathbf{v}'$ in the animated skin. In order to obtain an appealing deformation, it is necessary to respect the computed center of rotation $\mathbf{r}_c$. To achieve this, we extend the QLERP scheme to homogeneous matrices C_{j_i}. We denote the interpolation of matrices C_{j_i} with weights W as

$$q(W;C_{j_1},\ldots,C_{j_n}) = \begin{pmatrix} Q & \mathbf{m} \\ \mathbf{0}^T & 1 \end{pmatrix} \qquad (6)$$

and compute Q and $\mathbf{m}$ as follows. First, the rotation submatrices $C_{j_i}^{rot}$ are converted to quaternions $\mathbf{q}_{j_i}$. One of them, for example $\mathbf{q}_{j_1}$, is chosen as pivot. If $(\mathbf{q}_{j_1},\mathbf{q}_{j_i}) < 0$ for any $i = 2,\ldots,n$, we replace $\mathbf{q}_{j_i}$ with $-\mathbf{q}_{j_i}$ (by analogy to SLERP). Then the QLERP computes $\mathbf{s} = w_1\mathbf{q}_{j_1} + \ldots + w_n\mathbf{q}_{j_n}$, which is subsequently normalized to $\mathbf{s}_n = \mathbf{s}/\|\mathbf{s}\|$. Finally, $\mathbf{s}_n$ is converted to the rotation matrix Q. The translation part is just linearly interpolated, $\mathbf{m} = \sum_{i=1}^n w_i \mathbf{C}_{j_i}^{tr}$.

In order to change the center of rotation from the origin to $\mathbf{r}_c$, we define a homogeneous matrix

$$T = \begin{pmatrix} I & \mathbf{r}_c \\ \mathbf{0}^T & 1 \end{pmatrix} \qquad (7)$$

where I is a 3×3 identity matrix. Then the interpolation of homogeneous matrices with respect to the center of rotation $\mathbf{r}_c$ can be written as

$$Tq(W;T^{-1}C_{j_1}T,\ldots,T^{-1}C_{j_n}T)T^{-1} \qquad (8)$$

Note that the shift of the center of rotation does not influence the interpolated rotation – it manifests only in the translation part. The desired transformation of vertex $\mathbf{v}$ is

$$\mathbf{v}' = Tq(W;T^{-1}C_{j_1}T,\ldots,T^{-1}C_{j_n}T)T^{-1}\mathbf{v}$$

$$= \quad Q(\mathbf{v}-\mathbf{r}_c)+\sum_{i=1}^{n} w_i C_{j_i}\mathbf{r}_c \qquad (9)$$

A detailed derivation of this formula can be found in appendix A. The latter addend represents the translation induced by the new center of rotation.

The equation (9) has to be evaluated once per each vertex, and therefore should be as efficient as possible. The basic optimization is to pre-compute the quaternions $\mathbf{q}_{j_i}$, because they do not depend on the actual vertex – only on the joint's transformation, similarly as the rotation centers $\mathbf{r}_c$. Nonetheless, QLERP has to be executed for each vertex, since weights $w_1,\ldots,w_n$ can vary. In order to challenge the speed of LBS, we apply a following trick.

The vertex $\mathbf{v}$ can be represented by a quaternion with zero real part. In this representation, its rotation by quaternion $\mathbf{q}'$ can be expressed as $\mathbf{q}'\mathbf{v}\mathbf{q}'^{*}$, which is a quaternion with zero real part as well [Eberly 2001]. Although this expression is not efficient for computation (because of slow quaternion multiplication), it enables us to write out the rotation of $\mathbf{v}$ by quaternion $\mathbf{s}_n$ as

$$\mathbf{s}_n\mathbf{v}\mathbf{s}_n{}^{*} = \frac{1}{\|\mathbf{s}\|^2}\mathbf{s}\mathbf{v}\mathbf{s}^{*} = \frac{1}{(\mathbf{s},\mathbf{s})}\mathbf{s}\mathbf{v}\mathbf{s}^{*}$$

This suggests to convert already the quaternion $\mathbf{s}$ to matrix Q' and normalize subsequently by dividing $(\mathbf{s},\mathbf{s})$. Therefore, we can compute the Q matrix from (9) as $Q=\frac{Q'}{(\mathbf{s},\mathbf{s})}$ and save the *sqrt* operation. Some attention must be paid because standard routines for quaternion to matrix conversion assume a unit-length quaternion. The conversion of an arbitrary length $\mathbf{q}'=w+xi+yj+zk$ leads to the following matrix:

$$\begin{pmatrix} x^2+w^2-y^2-z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & y^2+w^2-x^2-z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & z^2+w^2-x^2-y^2 \end{pmatrix}$$

Vertex normal $\mathbf{v}_n$ is transformed in a similar way as vertex position, but ignoring the translation

$$\mathbf{v}'_n = Q\mathbf{v}_n$$

Using the formula (9) we can verify our previous intuitive thinking. First, if we substitute $\mathbf{r}_c$ in place of $\mathbf{v}$, no rotation occurs, which means that $\mathbf{r}_c$ is indeed a center of rotation. Second, if $n=2$ and $C_{j_1}\mathbf{r}_c = C_{j_2}\mathbf{r}_c$ (as in the beginning of section 4), the translation part becomes

$$w_1 C_{j_1}\mathbf{r}_c + w_2 C_{j_2}\mathbf{r}_c = (w_1+w_2)C_{j_1}\mathbf{r}_c = C_{j_1}\mathbf{r}_c$$

which is independent of interpolation parameters (weights), i.e. the translation during interpolation is constant indeed. Third, the equation (9) is nothing but a generalization of LBS to an arbitrary method of rotation interpolation. The choice of QLERP is not important for (9), the matrix Q can be replaced by matrix resulting from any other interpolation scheme, such as [Buss and Fillmore 2001]. If we substitute $Q=\sum w_i C^{rot}_{j_i}$, i.e. a simple linear combination of rotation matrices, we obtain

$$\mathbf{v}' = Q(\mathbf{v}-\mathbf{r}_c)+\sum w_i C_{j_i}\mathbf{r}_c = \sum w_i C^{rot}_{j_i}\mathbf{v} - \sum w_i C^{rot}_{j_i}\mathbf{r}_c + \sum w_i C^{rot}_{j_i}\mathbf{r}_c + \sum w_i \mathbf{C}^{tr}_{j_i} = \sum w_i C^{rot}_{j_i}\mathbf{v} + \sum w_i \mathbf{C}^{tr}_{j_i} = \sum w_i C_{j_i}\mathbf{v}$$

which is exactly the LBS equation (1). This also shows that LBS is a special case, which is independent of the center of rotation.

The whole algorithm can be summarized in the following steps:

- compute matrices C_i for all joints and convert their rotation parts to quaternions $\mathbf{q}_i$
- for each vertex $\mathbf{v}$ influenced by joints $j_1,\ldots,j_n$
  - compute (or re-use a cached) center of rotation $\mathbf{r}_c$ according to section 4
  - blend quaternions $\mathbf{q}_{j_1},\ldots\mathbf{q}_{j_n}$ using QLERP and convert the result to matrix Q
  - compute the position of vertex $\mathbf{v}'$ in the deformed skin using the equation (9)

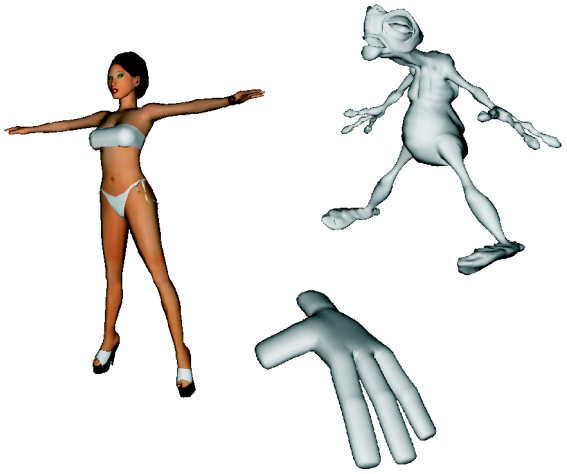

Figure 5: 3D models used for testing

	Hand	**Woman**	**Creature**
vertices	2402	3356	6802
triangles	4800	5205	13590
joints	23	78	56

Table 1: Complexities of example models

5 Results and Comparison

We tested the SBS algorithm on three models, see Figure 5 and Table 1. We compare the shape of the deformed skin on the model of woman, because human eye is most sensitive to the deformations of human body. Figure 6 presents results of LBS and SBS executed on the same posture of the model. Another example has been presented already in Figure 2. For small deformations, both algorithms produce similar results, as in the second row of Figure 6 (although a small loss of volume is noticeable even there). It is remarkable that the results of SBS are better even though the models have been optimised to work with the LBS algorithm.

The performance of both algorithms is compared in Table 2. The measured value is an average time in milliseconds necessary to deform one model on a 2.5GHz Athlon PC (rendering time not included). In the last row of the table the number of different non-trivial joint sets is reported (trivial joint set consists of only one joint

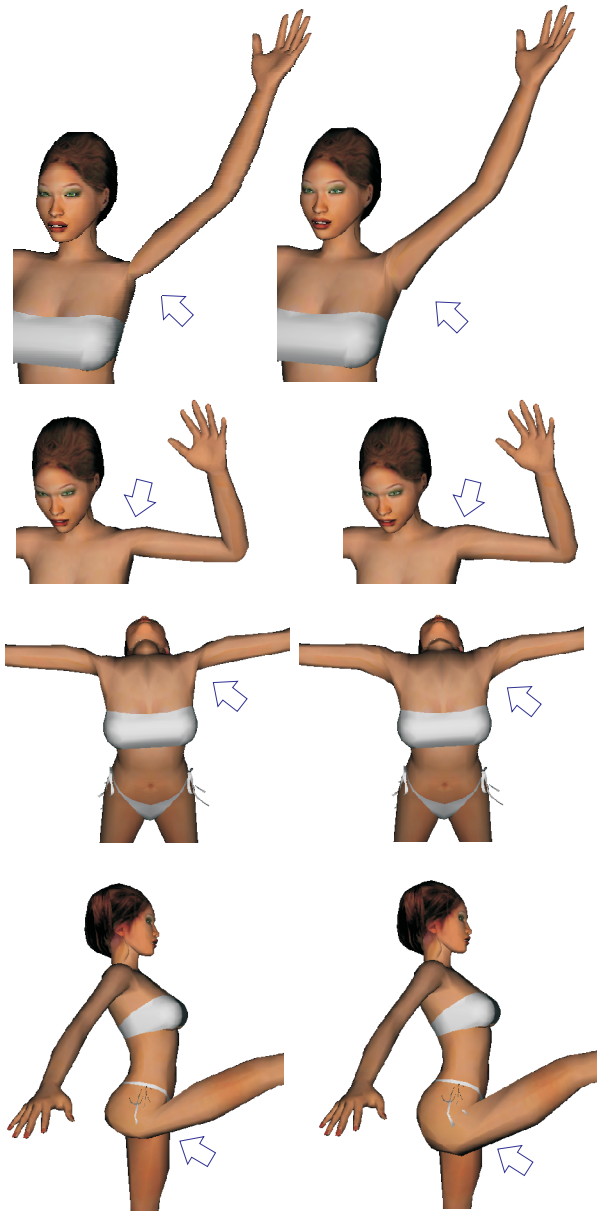

Figure 6: Comparison of deformations by LBS (left) and SBS (right)

or two neighboring joints). Put in another way, it is exactly the number of singular-value decompositions performed by the SBS algorithm. This number participates considerably on the difference between times for LBS and SBS. Theoretically, the number of different non-trivial joint sets could be very high. Fortunately, this number is surprisingly small in practice, because the joint influences tend to be local (e.g. it is unlikely to find vertices influenced by both left and right wrist). The additional memory needed for SBS is dominated by caching the computed centers of rotation. However, this amount of memory is negligible, considering the number of different non-trivial joint sets.

In order to test the accuracy of QLERP, we experimented with spherical weighted averages presented in [Buss and Fillmore 2001]. The algorithm proposed in [Buss and Fillmore 2001] behaves like SLERP for the case of two rotations (in contrast to QLERP, which only approximates SLERP results). On the one hand, the difference in the deformed skin was barely observable, according to the results from section 4.1. On the other hand, the increase in the execution time was quite substantial. For the woman model, the time increased from original 4.54ms to 22.74ms. This only confirmed our choice of QLERP.

	Hand	Woman	Creature
LBS time	3.28	3.59	9.0
SBS time	4.43	4.54	11.37
SVD executions	38	37	56

Table 2: First two rows: run-time of LBS and SBS algorithms in milliseconds; last row: number of SVD executions

5.1 Conclusion and Future Work

The proposed skin deformation system is by no means perfect; it cannot compete with complex, layered models. However, the SBS algorithm offers reasonable price for elimination of the notorious LBS artifacts. The time and memory complexity of both algorithms is comparable. The overhead of replacing an existing LBS implementation by SBS is minimal, because the input data, as well as the internal data structures, are the same. In contrast to other methods, the SBS does not need any additional information, such as the example skins.

The presented algorithm opens many questions and suggests several directions of future work. First of all, we worked only with vertex weights optimised for LBS. These weights are designed to suppress the LBS artifacts, even though they cannot remove them. It would be interesting to find out how much can be the SBS results improved by a set of weights especially designed for SBS. In order to accomplish this, a tool to explore the space of SBS deformations would help considerably. This tool has been presented for LBS in [Mohr et al. 2003], but the situation of SBS is somewhat more complex, because our interpolation method is non-linear. Similarly, it would be possible to estimate the SBS vertex weights from examples, as was done for LBS in [Mohr and Gleicher 2003]. This could also cover additional effects like muscle bulging.

6 Acknowledgments

This work has been partly supported by the Ministry of Education, Youth and Sports of the Czech Republic under research program No.Y04/98:212300014 (Research in the area of information technologies and communications).

We thank to Samuel Buss for providing the algorithm for spherical weighted averages [Buss and Fillmore 2001] and to LAPACK developers for their software. We would also like to thank to Jaroslav Semančík and the anonymous reviewers for valuable comments and to Adam J. Sporka for help with the accompanying video.

A Interpolation of Rotations with an Arbitrary Center

In this appendix we derive the formula (9), which describes the interpolation of rotations with respect to $\mathbf{r}_c$ – a custom center of rotation. Let us denote by K the coordinate system with origin in $\mathbf{r}_c$ and identical basis vectors as the world coordinate system. Then

the matrix T (7) can be interpreted as a transformation from K to the world coordinate system. By analogy, the inverse matrix

$$T^{-1} = \begin{pmatrix} I & -\mathbf{r}_c \\ \mathbf{0}^T & 1 \end{pmatrix}$$

represents the transformation from the world coordinate system to K. It follows that $T^{-1}C_{j_i}T$ is the transformation C_{j_i} expressed with respect to K. By interpolating these matrices with QLERP

$$q(W; T^{-1}C_{j_1}T, \ldots, T^{-1}C_{j_n}T)$$

we obtain a matrix working also on vectors in K coordinates. We can express this matrix with respect to the world coordinate system easily

$$Tq(W; T^{-1}C_{j_1}T, \ldots, T^{-1}C_{j_n}T)T^{-1}$$

which is exactly the formula (8).

Recall that the matrix C_{j_i} has structure

$$C_{j_i} = \begin{pmatrix} C_{j_i}^{rot} & \mathbf{C}_{j_i}^{tr} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

which enables us to write out

$$T^{-1}C_{j_i}T = \begin{pmatrix} C_{j_i}^{rot} & C_{j_i}\mathbf{r}_c - \mathbf{r}_c \\ \mathbf{0}^T & 1 \end{pmatrix}$$

as can be simply verified. Please note that the change of the coordinate system did not influence the rotation part $C_{j_i}^{rot}$ at all. Therefore the result of QLERP will be, according to equation (6)

$$q(W; T^{-1}C_{j_1}T, \ldots, T^{-1}C_{j_n}T) = \begin{pmatrix} Q & -\mathbf{r}_c + \sum_{i=1}^{n} w_i C_{j_i}\mathbf{r}_c \\ \mathbf{0}^T & 1 \end{pmatrix}$$

where Q stands for the interpolation of pure rotations, computed as indicated in section 4.2. Using $T^{-1}\mathbf{v} = \mathbf{v} - \mathbf{r}_c$ and $T\mathbf{x} = \mathbf{x} + \mathbf{r}_c$, we see that

$$\begin{aligned} \mathbf{v}' &= Tq(W; T^{-1}C_{j_1}T, \ldots, T^{-1}C_{j_n}T)T^{-1}\mathbf{v} \\ &= T \begin{pmatrix} Q & -\mathbf{r}_c + \sum_{i=1}^{n} w_i C_{j_i}\mathbf{r}_c \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v} - \mathbf{r}_c \\ 1 \end{pmatrix} \\ &= Q(\mathbf{v} - \mathbf{r}_c) + \sum_{i=1}^{n} w_i C_{j_i}\mathbf{r}_c \end{aligned}$$

is true for any vector $\mathbf{v}$. This is exactly the equation (9).

References

ALEXA, M. 2002. Linear combination of transformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 380–387.

ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. 1999. *LAPACK Users' Guide*, third ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

BLOOMENTHAL, J. 2002. Medial-based vertex deformation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 147–151.

BUSS, S. R., AND FILLMORE, J. P. 2001. Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph. 20*, 2, 95–126.

EBERLY, D. 2001. *3D game engine design: a practical approach to real-time computer graphics*. Morgan Kaufmann Publishers Inc.

KAVAN, L., AND ŽÁRA, J. 2003. Real-time skin deformation with bones blending. In *WSCG Short Papers Proceedings*.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 153–159.

LANDER, J. 1998. Skin them bones: Game programming for the web generation. *Game Developer Magazine* (May), 11–16.

LANDER, J. 1999. Over my dead, polygonal body. *Game Developer Magazine* (October), 17–22.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 165–172.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface '88*, Canadian Information Processing Society, 26–33.

MAGNENAT-THALMANN, N., CORDIER, F., SEO, H., AND PAPAGIANAKIS, G. 2004. Modeling of bodies and clothes for virtual environments. In *CW '04: Proceedings of the 2004 International Conference on Cyberworlds (CW'04)*, IEEE Computer Society, 201–208.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3, 562–568.

MOHR, A., TOKHEIM, L., AND GLEICHER, M. 2003. Direct manipulation of interactive character skins. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, ACM Press, 27–30.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 105–111.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press, 245–254.

SLOAN, P.-P. J., ROSE, III, C. F., AND COHEN, M. F. 2001. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, 135–143.

STEED, P. 2002. *Animating Real-Time Game Characters with CDROM*. Charles River Media, Inc.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 129–138.

WEBER, J. 2000. Run-time skin deformation. In *Proceedings of Game Developers Conference*.