# Multi-Weight Enveloping: Least-Squares Approximation Techniques for Skin Animation

Xiaohuan Corina Wang*
Industrial Light & Magic

Cary Phillips†
Industrial Light & Magic

## Abstract

We present a process called *multi-weight enveloping* for deforming the skin geometry of the body of a digital creature around its skeleton. It is based on a deformation equation whose coefficients we compute using a statistical fit to an input *training exercise.* In this input, the skeleton and the skin move together, by arbitrary external means, through a range of motion representative of what the creature is expected to achieve in practice. The input can also come from existing pieces of handcrafted skin animation. Using a modified least-squares fitting technique, we compute the coefficients, or "weights", of the deformation equation. The result is that the equation generalizes the skin movement so that it applies well to other sequences of animation. The multi-weight deformation equation is computationally efficient to evaluate; once the training process is complete, even creatures with high levels of geometric detail can move at interactive frames rates with a look that approximates that of anatomical, physically-based models. We demonstrate the technique in a feature film production environment, on a human model whose input poses are sculpted by hand and an animal model whose input poses come from the output of an anatomically-based dynamic simulation.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; G.1.2 [Numerical Analysis]: Approximation—Linear approximation; G.1.1 [Numerical Analysis]: Interpolation—Smoothing

**Keywords:** surface deformation, skinning, enveloping

## 1 Introduction

In the animation of digital creatures, the movement of the skin is of utmost importance. This is true of all computer graphic characters, but nowhere is it more true than in the animation of photorealistic animals and humans, because they must convey a true sense of musculature and tissue underneath the skin. The problem of finding computationally efficient and user-friendly techniques for moving the skin in a realistic-looking way is the subject of enormous effort by researchers and practitioners alike.

In practice, the techniques that produce the most visually impressive results are anatomically-driven, physically-based simulations that compute the dynamic effects of muscle, tissue and bones interacting with each other [Cinefex 2000 2001], but these techniques come with a cost. For characters with the level of geometric detail required for high-end feature film production, these techniques generally are not capable of achieving interactive frame rates, which dramatically complicates the animation process because they require a lengthy simulation phase before the animator can see a true representation of the movement of the character's skin.

Many physically-based, anatomically-driven simulation methods operate by adding inertial effects and muscle collisions to an underlying explicit deformation technique. Therefore, such systems require a robust underlying deformation that puts the skin in approximately the proper place where it can then shake and jiggle and collide with underlying muscles and bones.

Often, high-quality character animation involves as much art as science. Sometimes fanciful characters behave in a non-physically achievable manner, and this can be very difficult to describe with a system of muscles and bones. Therefore, there is significant motivation to find fast and explicit alternatives to full-scale dynamic simulation techniques, which still produce similar-looking results.

In a feature film production environment, the desired look of a character's skin must be achieved by whatever means are available. When not using physically-based simulation techniques, this frequently means tedious and difficult touching-up or sculpting by hand, sometimes on a frame-by-frame basis. This type of work often must be repeated over and over again from one piece of animation to another because the by-hand fixes are not general.

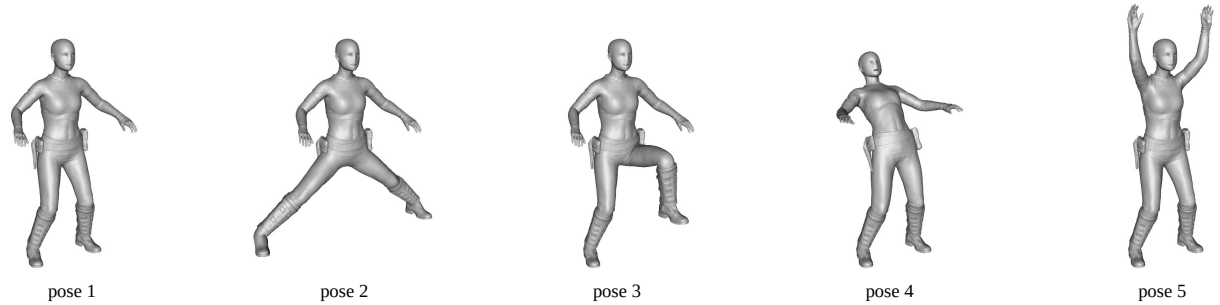
Because of all of this, we desire a skin deformation technique which satisfies these goals:

- It should handle fanciful creatures whose motion cannot readily be described with muscles and bones.
- It should be able to approximate the look of an anatomically-based simulation system, but at interactive display rates.
- It should be able to serve as the underlying deformation on top of which a physically-based simulation system can add inertial effects of jiggling and shaking.
- It should be able to "learn" from existing good examples of how the skin should move.

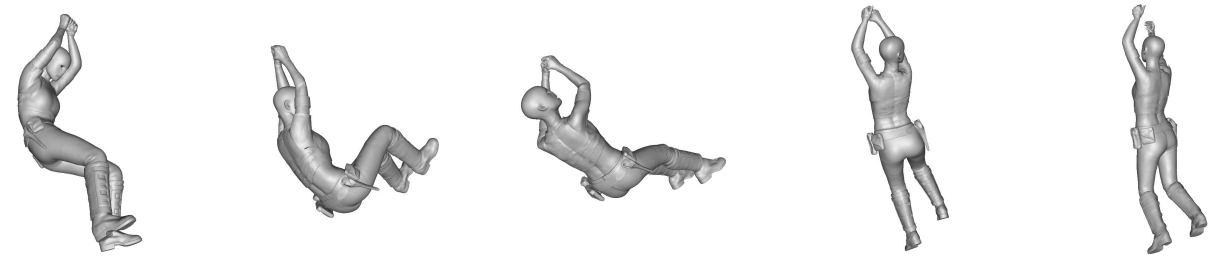
### 1.1 Pose-Based Approach

A *pose-based* approach to skin animation takes as input a series of poses and generates motion that is consistent with the poses. Each pose is a configuration of the skeleton together with the accompanying shape of the skin. Such a system has the very nice quality of being direct, working from the desired results backwards, as opposed to systems that require the indirect construction of muscle abstractions.

*e-mail: cwang@ilm.com
†e-mail:cary@ilm.com




**1-a**: Training Exercise




**1-b**: New Animation Sequence

Figure 1: The multi-weight enveloping skinning process solves for a set of weights that approximate the movement of skin in a training exercise. The result is that the skin movement in the training exercise is generalized so that it applies well to other sequences of animation. **1-a** shows some example poses from a training exercise. **1-b** shows some frames from a new animation sequence.

A pose-based approach need not specify where the poses come from. In practice, there are a variety of sources, from both art and science, including hand sculpting, procedural techniques like dynamic simulation, or even 3D scanning or motion capture.

Building a good sequence of poses by hand can be quite a burden on an artist, who is understandably motivated to produce the minimal number of poses required. But because the pose space of a skeleton is rather difficult concept to visualize, designing a minimal set of poses is likely to be an error-prone process for the artist, and the result is likely to leave portions of the pose space unexplored.

Our particular technique is designed to alleviate this problem by leveraging existing sequences of hand-tweaked animation as a source of poses, using every frame of the animation as a pose. However, this means that the system should be able to handle many, possibly hundreds, of poses, with the expectation that many will be redundant or even conflicting.

## 1.2 A Fitting Solution

One way of computing pose-based motion is to use scattered-data interpolation techniques. Two notable examples are the Lewis et al. *pose-space deformation* [2000], and the Sloan et al. *shape by example* [2001], which we describe in Section 2. These scattered data interpolation techniques hold the input poses as an integral component of the motion representation, which means the complexity of representation grows with the number of poses.

We choose instead to use approximation techniques, together with an explicit deformation equation. Our new deformation equation is a generalization of an old and common deformation technique occasionally referred to as *enveloping* [Softimage 1992], described with its limitations in Section 2.1. This technique defines the position of a point on the skin as a weighted combination of several skeletal coordinate frames. Our *multi-weight enveloping* (MWE) equation, presented in full detail in Section 3, replaces each weighted term in the simpler case with several auxiliary terms, hence the name “multi-weight”.

This new equation is a function of the skeletal coordinate frames. We have found it to be very powerful, and in practice it can represent fairly sophisticated motion of the skin, far beyond that of the original enveloping equation. The multi-weight enveloping equation also has the advantage of being mathematically very concise. Because it is linear in its input variables, we can use a modified linear least-squares solution to derive the weights from the input poses.

We recognize that some fidelity may be lost because we use an approximation technique that does not require the solution to interpolate the input poses. However, this approach is better equipped to handle poses that are close together in pose space because of what might be termed “user error”. In practice, this is quite hard to avoid. Another the advantage of our motion representation is that it does not get more complex with more poses. It can handle hundreds or even thousands of poses, at the expense of a longer solution time, but the amount of memory required to hold the coefficients of the equation is the same, and the subsequent evaluation of deformation equation at runtime remains unaffected by the number of poses.

## 1.3 Applications

With any technique used in a commercial, film production setting, there is a trade-off between time spent up front to build models carefully and ongoing fixes made after the initial setup is complete. Our technique is designed to accommodate both extremes. The ideal initial phase involves carefully building a *training exercise* that carries the skeleton and skin through the complete range of motion it is expected to ever achieve, capturing all the extremes. The alternative approach is to simply produce several sequences of animation using other techniques, without specifically designing them for use with the multi-weight enveloping process, and then provide the

multi-weights as a better-behaving and less finicky alternative that eventually takes over as the deformation technique of choice.

The fitting process can in theory operate on any source of input poses, but we focus on two specific applications. The first, as motivated above, is data generated through simpler deformation techniques augmented by hand-tweaking by an artist. The second is data generated with a full-scale anatomically-driven dynamic simulation. In both of these cases, our technique is able to generalize input and apply it to new sequences of animation. Figure 1 shows some examples of the multi-weight skinning input and output.

Although this training process can be quite involved, once the coefficients of the deformation equation have been computed, the fitting process is left behind, and the resulting representation of motion of the character's skin is quite simple and computationally efficient, making it ideal for any kind of interactive application, such as character animation, as well as games and virtual reality systems.

### 1.4 Overview

In Section 2, we give an overview of skin deformation techniques, some anatomically based and some not. In Section 2.1, we describe the *single-weight enveloping* technique that our deformation equation is generalized from. In Section 3, we describe the *multi-weight enveloping* deformation function. Section 4 presents the fitting process, including the requirements of the input as well as the numerical techniques for computing a solution that generalizes well. Section 5 describes the overall multi-weight skinning process in detail. In Section 6, we discuss some examples of how we have used this technique. Finally, we discuss future work in Section 7.

## 2 Background

Techniques for skin deformations can be characterized by how much and what type of abstraction exists between the skin and the underlying anatomy. Many interesting approaches involve detailed representations of muscle anatomy ([Scheepers et al. 1997] [Wilhelms and Gelder 1997] [Ng-Thow-Hing and Fiume 1997] [Chen and Zeltzer 1992] [Lee et al. 1995] [Nedel and Thalmann 1998]).

Other techniques proceed with less attention to anatomical rigor. Sederberg introduced the idea of a *free-form deformation* (FFD), or rectilinear lattice [Sederberg and Parry 1986]. The control vertices of the lattice act like a 3-dimensional spline volume, and deforming points inside the lattice are given normalized coordinates along each axis. [MacCracken and Joy 1996]. some of the The lattice is thus an abstraction for underlying tissue. Techniques extended from FFD remain an active area of research ([MacCracken and Joy 1996] [Chadwick et al. 1989] [Singh and Kokkevis 2000]).

One of the difficulties of anatomically-based systems is that they are indirect. Achieving a particular look in the skin requires determining what layout of muscles underneath would produce it. Hsu et al. developed a direct manipulation scheme that works with FFDs and allows the user to tug directly on the surface, using numerical pseudo-inverse techniques to determine where the control vertices have to go to yield such a pose [Hsu et al. 1992]. Our approach is similar in flavor, in terms of fitting a desired result to an underlying equation, but our problem domain and underlying formulation are different.

Many popular deformation techniques involve no anatomical abstraction at all and simply represent the skin as a shell that moves as an explicit function of the skeleton. Magnenat-Thalmann et al. introduced the idea of "joint-dependent local deformations" [Magnenat-Thalmann et al. 1988], which are specific local deformation operators based on the nature of the joints.

Shape interpolation is a popular technique for representing object deformations, but it is difficult to apply to an articulated skeleton, although Lewis et al. introduced *pose space deformations* (PSD) [2000] as a hybrid method of shape interpolation and skeleton driven skin animation. It employs scattered-data interpolation with a Gaussian radial basis function, using the difference between the joint configurations as the distance measurement. The falloff parameters are left as animator controls. As the number of pose controls and the number of poses get large, these falloff parameters can be difficult to tune to achieve the desired pose blending. One set of falloff parameters may work well with one set of poses, but when more poses are added, the same parameter mays cause too much overlap among all the poses. This technique works best when the poses are evenly distributed in pose space, but the interpolation becomes more problematic if poses are unevenly spaced, especially when poses are very close together. Lewis et al. discard duplicate poses as user error.

Sloan et al. described another way to use shape interpolation for skin deformation: *shape by example* (SBE) [2001]. As in PSD, SBE takes a set of poses (examples) as input and interpolates them. It combines radial basis functions and linear polynomials to implement shape blending in combination with transform blending (similar to single-weight enveloping) to deform the geometry. All input poses are transformed to a rest configuration for the skeleton where shape blending occurs, and the blended shape is subsequently transformed to the skeleton configuration for the new pose. SBE performs the interpolation in an abstract space defined by adjectives such as gender, age, and bending of a joint. The artist must associate a vector defined in this abstract space to a corresponding input pose. This process of assigning abstract qualities and setting their values for each pose can be a lot of work for the whole body of a complex creature.

In comparing our MWE technique to SBE and PSD, all three techniques take a set of poses or example as input. However, our method differs from SBE and PSD in the following regards:

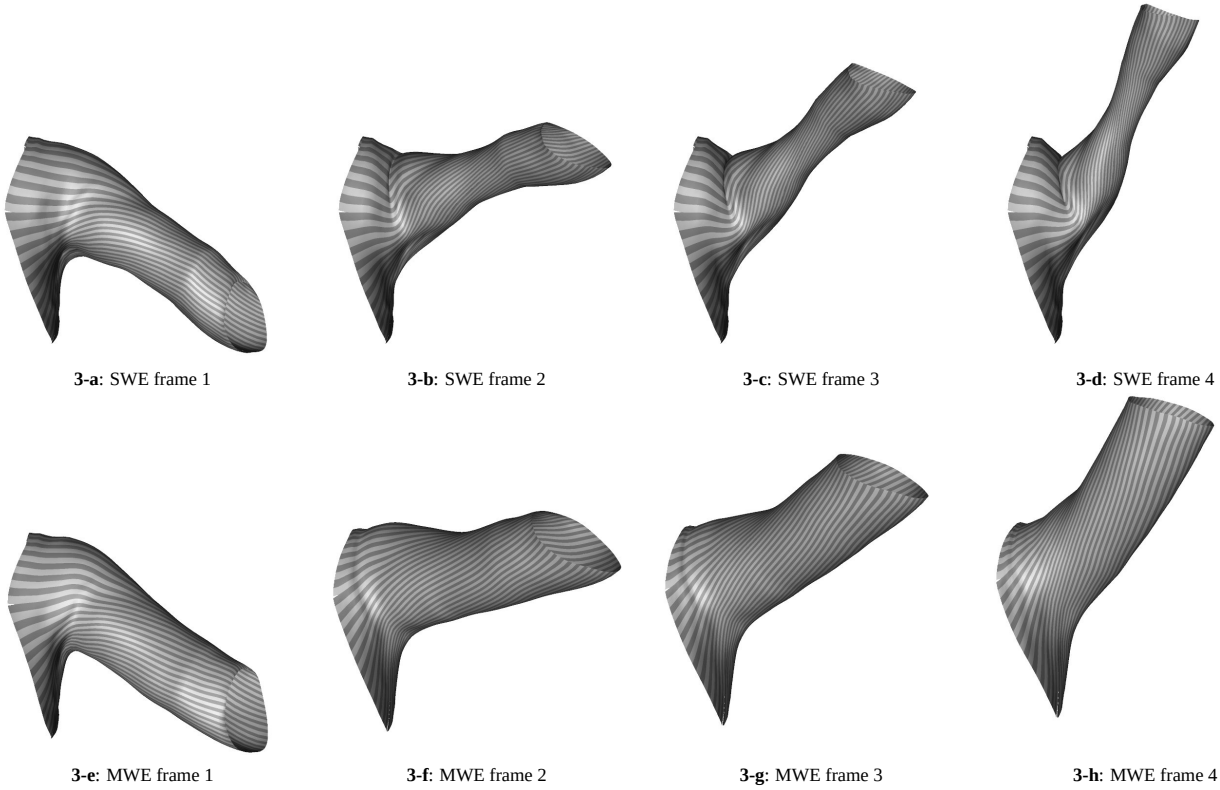
- Both SBE and PSD require all the input poses to be known at runtime. The memory space and evaluation time both increase as the number of poses grow. For MWE, only the sets of weights are needed at runtime. The memory space and evaluation time remain constant as the number of poses change.

- For the whole body animation, the number of pose controls for PSD and the dimension of the abstract space in SBE can be large. As the number of poses increase, adjusting the falloff parameters in PSD and setting the values for all adjectives in the abstract space for each input pose in SBE can become a large amount of work. This make it difficult for both PSD and SBE to make use of substantial portions of an animation sequence for a complex creature, placing a greater burden on the user to properly select representative poses. MWE gracefully handles large numbers of poses, duplicate poses, and poses close together in pose space.

- SBE and PSD are both shape interpolation techniques which interpolate all the input poses and the poses are themselves a part of the motion representation. MWE does not guarantee to reproduce the input poses exactly because of the least-squares approximation technique we employ over all the input poses. But in practice, we have found that our new deformation function is powerful enough to capture the significant details of the skin deformations in the input poses.

The use of training techniques in computer animation is certainly not new, and one technique with a somewhat similar flavor to ours is Grzeszczuk et al's NeuroAnimator [2001], although this application involves training a skeleton instead of the skin. The NeuroAnimator uses a neural net to train a rigid object or skeleton to move by giving it many examples of how such objects move.

Another similar paradigm is Brand's *voice puppetry* [1999]. The voice puppet uses a Hidden Markov Model to learn the mapping



**3-a**: SWE frame 1 **3-b**: SWE frame 2 **3-c**: SWE frame 3 **3-d**: SWE frame 4

**3-e**: MWE frame 1 **3-f**: MWE frame 2 **3-g**: MWE frame 3 **3-h**: MWE frame 4

Figure 3: This example illustrates a tight-fitting shirt with stripes around the shoulder and upper arm area. From left to right the arm is rotated up and twisted toward the front. **3-a** to **3-d** is the SWE deformation based on Equation (1): **3-d** exhibits typical collapsing problem around the shoulder area and the "candy-wrapper" problem around the middle of the upper arm. **3-e** to **3-h** is the MWE deformation computed from only two poses: **3-e** and **3-h** are poses provided by the artist and **3-f** and **3-g** are the MWE results applied on new poses.

from audio signals to facial expression from training video and audio. This mapping can then be applied to a new audio track to generate appropriate facial expression on the speaker.

## 2.1 Single-Weight Enveloping

In practice, one of the most common skin deformation techniques goes by a variety of names, including "enveloping" [Softimage 1992], "skinning" [Alias|Wavefront 1998], and "skeletal subspace deformation" by Lewis et al. [2000]. We will refer to it as *single-weight enveloping* (SWE) to be in contrast to MWE. This algorithm defines the position of a point on a surface as a linear combination of the rest state of the point projected into several moving coordinate frames, or *bones*.

If a point $\mathbf{p}$ is enveloped to a bone whose coordinate frame is $B_k$, let $\mathbf{p}_{B_k}$ be $\mathbf{p}$ in $B_k$'s frame of reference:

$$\mathbf{p}_{B_k} = \mathbf{p}{B_k}^{-1}$$

Then let $\mathbf{p}_k{}'$ be $\mathbf{p}_{B_k}$ carried along with $B_k$ as $B_k$ moves from its rest to its animated position $B_k{}'$:

$$\mathbf{p}_k{}' = \mathbf{p}{B_k}^{-1}B_k{}' = \mathbf{p}M_k$$

where $M_k$ is the *enveloping matrix* for bone $k$. $M_k$ is a transformation matrix that transforms bone $k$ from its rest to its animated frame. The final enveloped position $\mathbf{p}'$ is given by a weighted sum of the $\mathbf{p}_k{}'$ for the bones to which $\mathbf{p}$ is enveloped:

$$\mathbf{p}' = \sum_{k=1}^{n} w_k \mathbf{p} M_k \tag{1}$$

The $w_k$ are the single-weight enveloping weights, $1 \le k \le n$, where $n$ is the number of bones that $\mathbf{p}$ is enveloped to. The values of $w_k$ are left as artist controls. A large value of $w_k$ means $\mathbf{p}'$ follows bone $k$ closely; a small value of $w_k$ means the bone has little influence on the point. Figure 2 (on the color plate page) shows the SWE geometrically with two bones.

This equation has limited ability to describe skin-like behavior. For example, in Figure 2 (on the color plate page), no matter how the artist adjusts $w_1$ and $w_2$, the point $\mathbf{p}'$ is restricted to move along the line formed by $\mathbf{p}_1{}'$ and $\mathbf{p}_2{}'$. In particular, SWE suffers a collapsing problem, in which a bending tube loses volume instead of creasing (Figure 3-d around the shoulder area), which is common around elbows and shoulders. A similar anomaly is the "candy-wrapper" problem (Figure 3-d around the middle of the upper arm), in which a twisting tube collapses completely at rotations near 180 degrees. In general, it is very difficult to make the skin look right at various bone configurations by tweaking the single weight on each bone. One set of weights may work for one bone configuration by not for another.

## 3 Multi-Weight Enveloping

The multi-weight enveloping equation is an extension of Equation (1) in which each entry in the enveloping matrix $M_k$ gets its own weight. If the enveloping matrix from Equation (1) is written out as:

$$M_k = {B_k}^{-1} {B_k}' = \begin{bmatrix} m_{00_k} & m_{01_k} & m_{02_k} & 0 \\ m_{10_k} & m_{11_k} & m_{12_k} & 0 \\ m_{20_k} & m_{21_k} & m_{22_k} & 0 \\ m_{30_k} & m_{31_k} & m_{32_k} & 1 \end{bmatrix} \quad (2)$$

then the multi-weight enveloping equation is:

$$\mathbf{p}' = \sum_{k=1}^{n} \mathbf{p} \begin{bmatrix} w_{00_k} m_{00_k} & w_{01_k} m_{01_k} & w_{02_k} m_{02_k} & 0 \\ w_{10_k} m_{10_k} & w_{11_k} m_{11_k} & w_{12_k} m_{12_k} & 0 \\ w_{20_k} m_{20_k} & w_{21_k} m_{21_k} & w_{22_k} m_{22_k} & 0 \\ w_{30_k} m_{30_k} & w_{31_k} m_{31_k} & w_{32_k} m_{32_k} & 1 \end{bmatrix} \quad (3)$$

where $n$ is the number of bones that $\mathbf{p}$ is enveloped to, and the $w_{ij_k}$ are the multi-weight enveloping weights for bone $k$, $1 \leq k \leq n$, $0 \leq i \leq 3$ and $0 \leq j \leq 2$.

Evaluating this equation for a single point requires 21 multiplications and 9 additions per bone the point is enveloped to. This is in addition to the computation of the $M_k$ matrix, which is a straightforward calculation from the bone transformation matrices, as prescribed by Equation (2).

The enveloping matrix $M_k$ represents the transformation that a point undergoes with respect to bone $k$. By giving a weight to each entry in this matrix, we provide the ability to modify the rotation, translation, scale and shear of $M_k$. This allows the skin point to be expressed with non-rigid as well as rigid transformations. The skin point is no longer restricted to the subspace defined only by rigid transformations of the relevant bones as in the SWE case. Figure 3 shows a simple example of how MWE is capable of capturing the skin-like behavior given only two input poses.

We have designed a new deformation function with a high dimensional input space: 12 dimensions for every relevant bone, to provide the enveloping matrix with the maximum flexibility to transform the skin points to best approximate their desired positions in the training poses. However, given a set of input poses, there may exist many dependencies among these input dimensions. Performing a principal component analysis (PCA) removes correlations among these dimensions and produces a set of independent basis vectors for the training of weights. Details are discussed in Section 4.3.

An advantage of using the matrix components as the variables is that the resulting equation is linear, even though the general movement of the coordinate frames in the skeleton is non-linear, since it is the product of rotations. The trigonometric terms involving the rotation are essentially subsumed into the variables we use in the fitting process. The matrix formulation provides a way of dispensing with messier forms of rotations, like euler angles and quaternions, when represented in a transformation matrix. Even though quaternions behave smoothly, the resulting equation is non-linear which would dramatically complicate the fitting process.

## 4 The Weight-Solving Process

A collection of poses represents samples that pair values of $\mathbf{p}'$ with corresponding values of $m_{ij_k}$. Using these, we solve for the weights $w_{ij_k}$ in Equation (3) using modified least-squares fitting techniques described in Section 4.4. We solve each point independently. For each point, we first determine the subset of bones that affect it, through a process described below in Section 4.1. We can then extract from the input training exercise each pose's data for that bone. If a point has duplicate poses, i.e. poses with very similar $m_{ij_k}$, we average the pose positions $\mathbf{p}'$. This ensures that duplicate poses do not weight the least-squares solution unduly. For the following discussion, let there be $m$ unique poses and $n$ relevant bones for a particular point.

We can solve each dimension independently, since $x$, $y$, and $z$ are unrelated. The multi-weight enveloping equation for the $j$-th dimension,

$${p_j}' = \sum_{k=1}^{n} p_0 w_{0j_k} m_{0j_k} + p_1 w_{1j_k} m_{1j_k} + p_2 w_{2j_k} m_{2j_k} + w_{3j_k} m_{3j_k} \quad (4)$$

is the result of inserting $\mathbf{p} = [p_0\ p_1\ p_2\ 1]$ into Equation (3), with $j$ as 0, 1, or 2 corresponding to the $x$, $y$, or $z$ dimension, respectively.

Each pose yields one equation in the $w_{ij_k}$ variables. If the point $\mathbf{p}$ is enveloped to $n$ bones and there are $m$ poses in the input, then the matrix form of the $m$ equations for the $j$-th dimension is:

$$A\mathbf{w} = \mathbf{b} \quad (5)$$

or

$$\begin{bmatrix} & \vdots & \\ \cdots & A & \cdots \\ & \vdots & \end{bmatrix} \begin{bmatrix} w_{0j_1} \\ \vdots \\ w_{0j_k} \\ w_{1j_k} \\ w_{2j_k} \\ w_{3j_k} \\ \vdots \\ w_{3j_n} \end{bmatrix} = \begin{bmatrix} {p_{j_1}}' \\ \vdots \\ {p_{j_l}}' \\ \vdots \\ {p_{j_m}}' \end{bmatrix}$$

where the matrix of coefficients $A$ is given by:

$$A = \begin{bmatrix} m_{0j_{11}} & \cdots & m_{0j_{k1}} & m_{1j_{k1}} & m_{2j_{k1}} & m_{3j_{k1}} & \cdots & m_{3j_{n1}} \\ \vdots & & & & & & & \vdots \\ m_{0j_{1m}} & \cdots & m_{0j_{km}} & m_{1j_{km}} & m_{2j_{km}} & m_{3j_{km}} & \cdots & m_{3j_{nm}} \end{bmatrix}$$

Matrix $A$ has $m$ rows (one per pose) and $4n$ columns (4 per bone). The right-hand vector $\mathbf{b}$, of dimension $m$, represents the location of the point $\mathbf{p}'$ in each of the poses. $\mathbf{w}$ is the vector of variables being solved, of dimension $4n$.

The standard least-squares process minimizes the error:

$$\varepsilon = \|A\mathbf{w} - \mathbf{b}\|_2^2 \quad (6)$$

$\varepsilon$ is the squared 2-norm of the component-wise difference between the solved location of the point and its position in each of the poses. However, minimizing $\varepsilon$ alone is not practical for several numerical and heuristic reasons. First, matrix $A$ can contain correlated column vectors, which makes it an ill-conditioned matrix that is unstable and difficult to solve. Second, we have to avoid *overfitting*, i.e., computing weights that fit the input poses from the training exercise well (tight fit with small $\varepsilon$), but behave poorly when applied to new poses in the new animation sequences (poor generalization). We discuss these issues further in Section 4.3 and Section 4.4.

### 4.1 Localized Effects from Global Poses

For the input poses, we would like any good pairing between the skin and the skeleton to be a candidate, even if it represents a pose

of the entire creature, not just a portion of it. This way, we can use any piece of animation as a source of poses.

If the training exercise is truly representative of the entire range of motion of the skeleton, the natural correlation between the skin and nearby bones will fall out automatically, but in practice this would require a very large, prohibitively extensive set of training poses. For example, each pose of the fingers would have to be paired with many poses of the toes in order to determine that there is no correlation between the two. In a typical animal skeleton, the skin is fairly localized so that its position depends only on nearby bones.

In order to interpret each pose in the training exercise as a complete pose of the entire skin and skeleton, we require an additional input that defines for each surface point the subset of bones that affect it. This ensures that the dimension of the $A$ matrix in Equation (3) is not the entire number of bones in the skeleton, but is restricted to the subset of the bones that are interesting to a particular point. This map viewed from the point of view of the bone defines what surface points the bone affects.

As described in the next section, there are further benefits to having a scalar field that defines the extent of the influence of each bone, so we require an *influence map* to accompany each bone in the input. The influence of a particular bone includes all surface points where the influence map is non-zero.

Providing these maps is left up to the user. Since the information they convey can readily be visualized as scalar fields on the surface, the interface we rely on is a 3D painting technique, similar to texture map painting. These influence maps are similar to single-weight enveloping weights. Figure 4 (on the color plate page) shows an example of two influence maps painted on an animal. These maps can be painted quite loosely; their purpose is to distinguish the relevant bones from the irrelevant ones. The edges of these maps should be smooth to prevent discontinuities in the skin deformation.

### 4.2 Pose and Bone Influences

In the matrix formulation in Equation (5), we can scale the rows and the columns of $A$ to achieve certain desirable effects. First, the bone influence maps can be used to scale the columns of $A$ to modulate the influence of a particular bone on a point. For each point, the influence from bone $k$, $\mu_k$, is multiplied into the matrix prior to solving the equation so that matrix $A$ becomes:

$$\begin{bmatrix} \mu_1 m_{0j_{11}} & \cdots & \mu_n m_{3j_{n1}} \\ \vdots & & \vdots \\ \mu_1 m_{0j_{1m}} & \cdots & \mu_n m_{3j_{nm}} \end{bmatrix}$$

The $\mu_k$s are then folded into the resulting weights afterwards. A small value of $\mu_k$ gives a bone less effect on the error in Equation (6); a large value gives it greater effect. Smooth bone influence maps provide smooth falloffs of the effects of the bones from one skin point to the next.

It is also possible to increase the influence of the $i$th pose by multiplying the $i$th row of the matrix $A$ and the $i$th entry of $\mathbf{b}$ in Equation (5) by a constant scalar, $1 \leq i \leq m$. The result is that the corresponding pose contributes more to the error that is being minimized. This provides the user with a way of insisting that the solution fit a particular pose more closely than the other poses.

### 4.3 Principal Component Analysis on Input Data

For the input data matrix $A$, note that it uses 12 entries of the bone enveloping matrix, while a bone has fewer than 12 independent degrees of freedom. This means that the column vectors of matrix $A$ may very well be correlated to one another, not to mention that different bones influencing the same point may have their matrix entries correlated with one another. To project the set of correlated variables into a set of more independent variables, we apply PCA. It is a common practice to use PCA to remove the correlation in the input data [Dean 1988].

We apply the PCA method as follows. Compute the eigenvectors of the variance-covariance matrix of $A$: $\mathbf{e_1} \ldots \mathbf{e_{4n}}$, listed in the order of decreasing eigenvalues. Then $\mathbf{e_1}$ is the first principal component of $A$ and represents the direction that has the maximum variance. Specify an eigenvalue threshold which chooses $p$ (with $p < 4n$) eigenvectors to correspond with the $p$ largest eigenvalues. We project the input matrix $A$ onto its principal components by setting $C = AE$, with $E = [\mathbf{e_1} \ldots \mathbf{e_p}]$. Matrix $C$ has $p$ column vectors, which are orthogonal to each other. By choosing only $p$ most significant eigenvectors to transform $A$ into $C$, we get an input matrix $C$ that has fewer dimensions yet still represents a high proportion of the variance of the original input data in $A$.

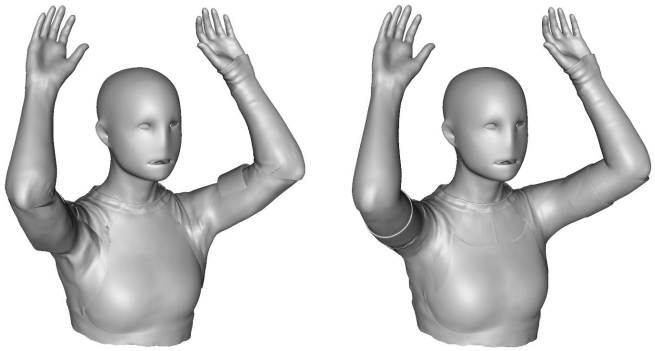
Using PCA, instead of solving Equation (5), we solve $w_c$ in

$$C\mathbf{w_c} = \mathbf{b} \tag{7}$$

The final $\mathbf{w}$ can be computed from $\mathbf{w} = E\mathbf{w_c}$. This is because $C\mathbf{w_c} = \mathbf{b} = A\mathbf{w}$ from Equation (5) and (7), and $C = AE$, which gives us $AE\mathbf{w_c} = A\mathbf{w}$.

### 4.4 Local Ridge Regression

To compute a set of weights from the training poses that generalize well to new animation, we need a way to control the magnitude of the resulting weights. The effect of large weights can be seen by taking the partial derivatives of Equation (4), for example, $\frac{\partial p_j'}{\partial m_{ij_k}} = p_i w_{ij_k}$. If $w_{ij_k}$ is very large, then small changes in the bone transformation, i.e. $m_{ij_k}$, can cause large displacements of the output point, $p_j'$. This will not produce a smooth deformation of the skin when bones move away from their training poses. Instead the skin points will fly away from their desired positions. Figure 5-a demonstrates this kind of overfitting problem as the weights computed from the input poses are applied to a new pose: the skin deforms badly.



**5-a**: Overfitting. Large weights deform skin badly on a new pose.

**5-b**: Fixed. Smoother weights are computed from local ridge regression.

Figure 5: Overfitting Fixed by Local Ridge Regression

In order to achieve a smooth interpolation function, we use a technique called *local ridge regression* [Orr 1995], which offers a trade-off between tight fitting and good generalization. It is a modified least-squares fitting technique and is an extension of the regular ridge regression. The standard least-squares solution of Equation (7) is to minimize the cost function $\|C\mathbf{w_c} - \mathbf{b}\|_2^2$, which is equivalent to solving $C^T C\mathbf{w_c} = C^T\mathbf{b}$, referred to as the standard

normal equation. This can be derived through setting the derivative of the cost function to 0 [Kahaner et al. 1989]. The ridge regression technique was developed by Hoerl and Kennard [1970] to regularize ill-conditioned problems where the inverse of $C^TC$ is unstable (when $\|C^TC\|$ is close to 0). It solves for $(C^TC+\lambda I)\mathbf{w_c} = C^T\mathbf{b}$. The global $\lambda$ is used as a trade-off parameter for tight fitting (small $\lambda$) versus stability (large $\lambda$) in the regular ridge regression.

We find that one global regularization parameter does not suit our needs. We require a regularization parameter that changes according to how extensive the training data is in each dimension. In particular, if one input dimension is not changing at all in the training exercise, it implies we do not have any information how the surface point will move when this input dimension starts to change in the new animation. In this special case, when the variance on one input dimension is 0, we want the weight computed in this dimension to be 0. In general, we want this penalty term to be inversely proportional to the variance on the input coefficient.

The local ridge regression developed by Orr [1995] assigns a penalty term $\lambda_h$ to each $w_{c_h}$: the $h$-th variable in vector $w_c$, with $1 \le h \le p$. It solves for

$$(C^TC+\lambda_h I)\mathbf{w_c} = C^T\mathbf{b} \tag{8}$$

which minimizes the new cost function:

$$\|C\mathbf{w_c}-\mathbf{b}\|_2^2 + \sum_{h=1}^{p} \lambda_h w_{c_h}^2.$$

The larger we choose $\lambda_h$, the smaller the value of $w_{c_h}$ will come out. In the extreme case, we can make $w_{c_h} = 0$ by setting $\lambda_h = \infty$.

There are various heuristics that can be used to compute the regularization parameter $\lambda_h$, some of which are iterative methods such as the one presented by Orr [1995]. We introduce a new way of computing $\lambda_h$ which works well for our technique. It is based on the variance related discussion we have made above:

$$\lambda_h = \begin{cases} \frac{s_f}{\sigma(\mathbf{c_h})} & \text{if } \sigma(\mathbf{c_h}) > 0 \\ \infty & \text{if } \sigma(\mathbf{c_h}) = 0 \end{cases} \tag{9}$$

where $\mathbf{c_h}$ is the $h$th column vector of matrix $C$, $\sigma(\mathbf{c_h})$ is its variance, and $s_f$ is a user provided scale parameter to control overfitting. The idea is that if the coefficient of $w_{c_h}$ is not changing much from the training data, it implies that we do not have enough information to compute the $w_{c_h}$ that would generalize well and we should add a bigger penalty to its magnitude.

Finally, we use singular value decomposition to solve Equation (8), where the user can adjust additional parameters to control the smoothness of the solution by removing small singular values.

### 4.5 Space of Computation

For reasons similar to those discussed above for computing weights with small values, our experience has been that the approximation behaves better if we solve for weights that measure the difference between the pose and the skin rigidly transformed by the skeleton, or even better, deformed by another underlying system, such as SBE. This changes the space of the computation from the local coordinate of the skin point, $\mathbf{p}'$, to a delta between its desired position and the position determined by the underlying system, $\Delta\mathbf{p}'$. This underlying system can be a rigid transformation by the skeleton or single-weight enveloping or any other approach, e.g., lattice-based, that makes the skin roughly follow the bones. The goal is to capture as much as possible of the rigid transformation that the skin undergoes with the underlying system, leaving the MWE solution process to capture the refinement. To make this computation space change, simply replace the $\mathbf{p}'$ with $\Delta\mathbf{p}'$ in Equation (3), (4) and (5).

## 5 The MWE Skinning Process

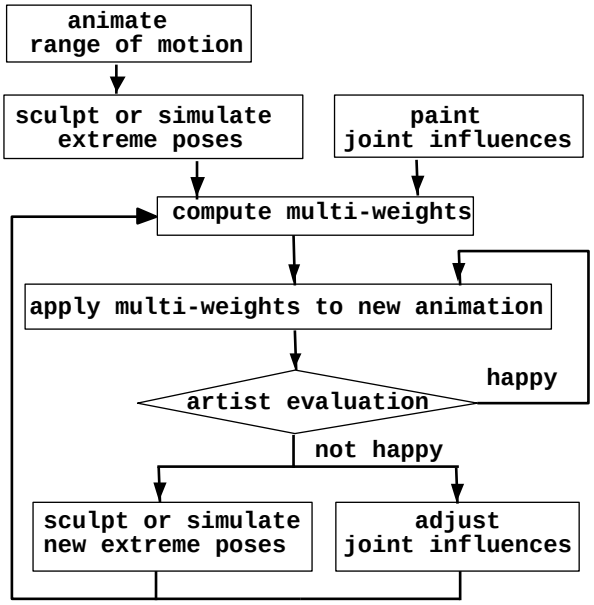



Figure 6: Flow Chart of the MWE Skinning Process

With the new formulation and the statistical approach of automatically computing the weight coefficients, the skinning process is summarized in Figure 6. In Section 6, we present results with two different applications of the technique. In the first example, an artist hand-sculpted isolated frames of a training exercise, using general-purpose geometric modeling software. In the second example, a dynamic simulation process generated the desired poses for the training, which was then augmented by additional sequences of animation.

The skinning process with sculpting by hand is as follows:

1. Animate a training exercise, capturing all extremes that are likely to occur in practice. This animation is a sequence of poses, without having any in-betweens.

2. Paint a bone influence map for each bone in the skeleton, loosely based on anatomy.

3. Hand-sculpt the extreme poses. This may or may not involve making use of an underlying single-weight envelope.

4. Solve for the multi-weights, choosing an appropriate $s_f$ parameter in Equation (9) to control the balance between tight fitting and good generalization.

5. Validate the MWE solution at the input poses. If the fit is not tight enough, adjust the $s_f$ in Equation (9) to be smaller. If some bone movement does not affect particular parts of the skin where it should, extend its influence in the influence map.

6. Validate the MWE solution with test data, i.e. new animation sequences. If skin deforms badly due to poor generalization, e.g., points oscillate about their desired positions, adjust the $s_f$ in Equation (9) to be larger.

7. Where the MWE solution fails to produce aesthetically satisfying interpolation or extrapolation on the new poses, hand-sculpt the failing pose, and add it back into the training set, and repeat steps 4-7.

The skinning process with simulation as input is as follows:

1. Animate a training exercise, capturing all extremes that are likely to occur in practice. This animation must be structured so that the motion is physically achievable by the creature, i.e. attention must be paid to what the simulation process will do between the extremes.

2. Paint a bone influence map for each bone in the skeleton loosely based on anatomy.

3. Run the dynamic simulation on the animation.

4. Solve for the multi-weights using every frame of the simulated training exercise.

It is desirable for the input poses to be taken at static equilibrium so that they provide bulges and creases, but do not include dynamic effects such as jiggling. If the poses do include jiggling, the weight-solving process described in Section 4 averages the different skin point positions in identical poses and approximate their positions in similar poses by minimizing the error in Equation (6).

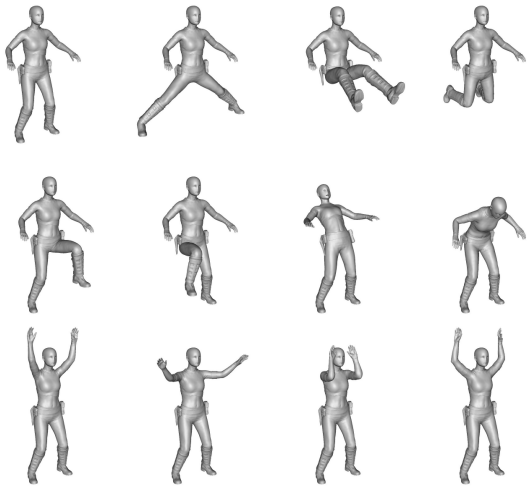
## 6 Results

We have implemented the multi-weight enveloping process and used it in a feature film production environment. Here we present the results on two creatures with different anatomy - a human and an animal creature. The human's poses are sculpted by hand, on top of an SWE, and the animal's poses are done through a dynamic muscle and flesh simulation.

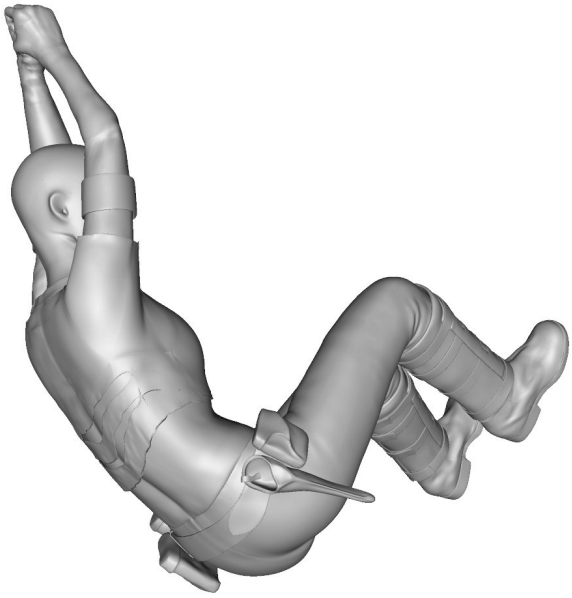
The human figure was built from scanned data and consists of approximately 300,000 spline control vertices and has 59 skeletal bones. Each point is influenced by up to 6 bones. Figure 7-a shows a few extreme poses from the training exercise of the human creature. The training exercise consists of 14 poses, and the process of solving for the multi-weights took 6 minutes on an SGI O2 workstation. Once the multi-weights were solved, we apply them to new sequences of animation where the skin deforms interactively with the skeleton. This allowed artists to visualize realistic skin movement as they adjust the skeleton.

Figure 7-b presents one frame of the MWE result on a new animation sequence, with the multi-weights computed from the original training poses and refined by two additional poses. These two poses were added when the artists were not satisfied by the initial MWE interpolation on two new poses and performed step 7 as described in Section 5. As more poses were added into the training, the multi-weights produced better mappings from bone movement to skin deformation. Figures 7-c and 7-d show a comparison between the SWE and MWE results around the shoulder area. SWE exhibits typical collapsing and crunching artifacts, while MWE shows a smooth shoulder with proper volume. This kind of difference can be seen throughout the entire sequence of the animation.

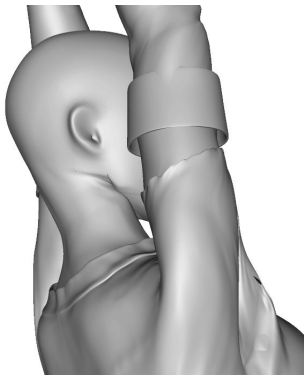
Figure 8 lists a series of training poses of the animal creature. This creature model consists of approximately 200,000 spline control vertices, and it has 55 skeletal bones. Each surface point is influenced by up to 12 bones. Some of these poses are from the initial training exercise, on which we ran a dynamic muscle simulation, and others are from some sequences of animation that had already been simulated, demonstrating that it is possible to make use of any existing example of how the skin is supposed to move. The poses from the initial training exercise are taken at static equilibrium, while the ones from the already simulated sequences contain some jiggling. The total training set consists of 344 poses, and computing the multi-weights took 40 minutes on an SGI Octane. Figure 9-a and 9-b (on the color plate page) present the MWE result



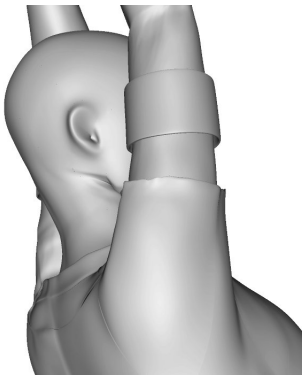
**7-a**: Training Exercise for the Human Creature



**7-b**: Multi-Weight on A New Animation Sequence (frame 9)



**7-c**: Single-Weight. It exhibits typical collapsing and crunching artifacts.



**7-d**: Multi-Weight. It shows a smooth shoulder with proper volume.

Figure 7: Multi-weight Skinning on the Human Creature

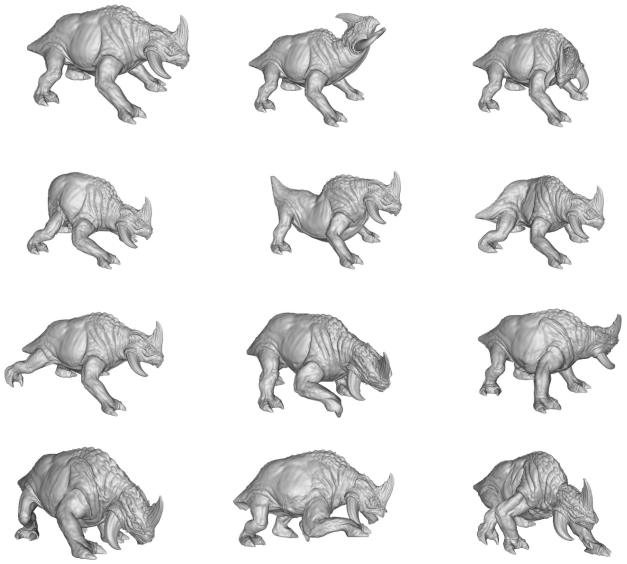


Figure 8: Example of Training Poses for the Animal Creature. Some poses are from the training exercise and some are from additional training animation sequences.

on poses from two new animation sequences. It shows that multi-weights generalize well to new poses and have captured the nice muscle definition under the chest and the bulging volume around the shoulder and front arm.

For both creatures, Figure 10 compares the MWE results with some input poses. Figure 10-a is one of the 16 equally weighted input poses for the human creature and Figure 10-c is one of the 344 equally weighted input poses for the animal creature. Figure 10-b and 10-d show that the MWE approximates the input poses reasonably well, missing only a few small details. For example, the wrinkles on the shirt in Figure 10-b are not as pronounced as in Figure 10-a and the musculature in the upper leg in Figure 10-d is slightly less defined than in Figure 10-c.

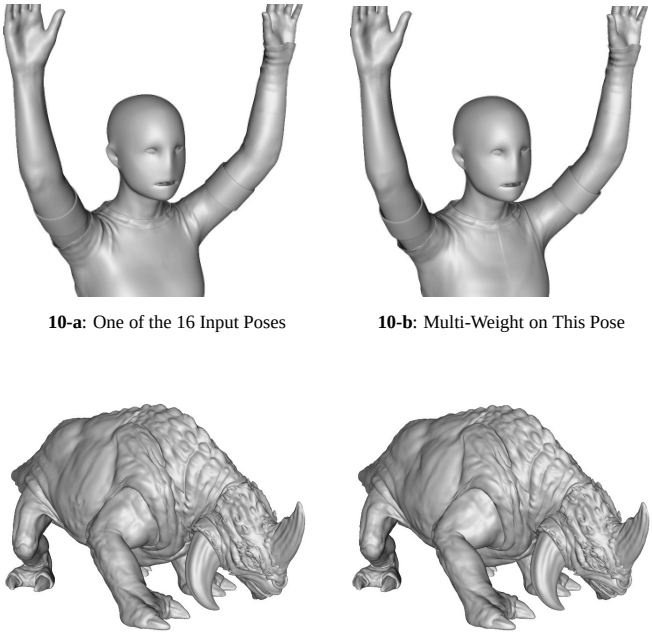
## 7 Conclusion and Future Work

The multi-weight enveloping equation offers a powerful but concise representation for skin movement, and the weight-solving process offers a practical way of harnessing painstaking work performed on one piece of animation for use on another. The multi-weight enveloping process fits well into a film production environment because it makes use of the ability to get one specially constructed animation looking right by any means necessary, including hand tweaking. It is particular useful for a film with many different animation sequences involving the same creature, because the amount of time spent preparing the training exercise is amortized over many shots.

The technique holds up well with large numbers of poses, and the resulting representation of the deformation does not get more complex as the number of poses increases.

The technique also has applications for games and virtual reality, where creature models must deform in real-time but still behave in an anatomically acceptable way. The game development stage would require constructing the training data and solving for the weights, but once this is done, when the creature appears in the game, it will have only its multi-weights describing its motion.

The multi-weight enveloping equation is simple enough that it would lend itself well to hardware acceleration. Game engines could provide this as a primitive operation.



**10-a**: One of the 16 Input Poses

**10-b**: Multi-Weight on This Pose

**10-c**: One of the 344 Input Poses

**10-d**: Multi-Weight on This Pose

Figure 10: Multi-weight Approximates the Input Poses

As we have mentioned, the training data can come from any number of sources, and motion capture is a particularly promising one. If surface data and skeletal information could be captured from real models, the multi-weight process could be used to automatically generate deforming creatures.

## Acknowledgments

We thank John Anderson, Nicolas Popravka, and the entire R&D team at ILM for their valuable insights, discussions, and support. This work would not have been possible without the talented artists at ILM, who have provided us with the incredible models, animation, and extremely useful feedback. We also appreciate the detailed suggestions and comments from all the reviewers.

## References

ALIAS|WAVEFRONT. 1998. *Maya 1.0 Using Maya.* Toronto, Ontario, Canada.

BRAND, M. 1999. Voice puppetry. In *Proceedings of SIGGRAPH 1999,* ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 21–28.

CHADWICK, J., HAUMANN, D., AND PARENT, R. 1989. Layered construction for deformable animated characters. In *Proceedings of SIGGRAPH 1989,* ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 243–252.

CHEN, D. T., AND ZELTZER, D. 1992. Pump it up: Comptuer animation based model of muscle using the finite element method. In *Proceedings of SIGGRAPH 1992,* ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 89–98.

CINEFEX. 2000 2001. *Cinefex*. Riverside, CA, various issues, e.g., vol. 87 (Oct 2001), (Jurassic Park III); vol. 83 (Oct 2000), p. 104 (Hollow Man); vol. 82 (July 2000), p. 68 (Dinosaur).

DEAN, E. B. 1988. Linear least squares for correlated data. In *Proceedings of the Tenth Annual International Conference of the International Society of Parametric Analysts*, http://techreports.larc.nasa.gov/ltrs/PDF/conf-10-ispa.pdf.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 2001. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 9–20.

HOERL, A. E., AND KENNARD, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*, 1, 55–67.

HSU, W., HUGHES, J., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *Proceedings of SIGGRAPH 1992*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 177–184.

KAHANER, D., MOLER, C., AND NASH, S. 1989. *Numerical Methods and Software*. Prentice Hall.

LEE, Y., TERZOPOULOS, D., AND WATERS, K. 1995. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 55–62.

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 SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 165–172.

MACCRACKEN, R., AND JOY, K. 1996. Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH 1996*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 181–188.

MAGNENAT-THALMANN, N., LAPERRIERE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings of Graphics Interface*, 26–33.

NEDEL, L. P., AND THALMANN, D. 1998. Modeling and deformation of the human body using an anatomically-based approach. In *Proceedings of Computer Animation*, 34–40.

NG-THOW-HING, V., AND FIUME, E. 1997. Interactive display and animation of b-spline solids as muscle shape primitives. In *Eurographics Workshop on Animation and Simulation 1997*, 81–97.

ORR, M. J. 1995. Local smoothing of radial basis function networks. In *International Symposium on Artificial Neural Networks*, http://www.anc.ed.ac.uk/ mjo/rbf.html.

SCHEEPERS, F., PARENT, R., CARLSON, W., AND MAY, S. 1997. Anatomy-based modeling of the human musculature. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 163–172.

SEDERBERG, T., AND PARRY, S. 1986. Free form deformations of solid geometric models. In *Proceedings of SIGGRAPH 1986*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 150–161.

SINGH, K., AND KOKKEVIS, E. 2000. Skinning characters using surface oriented free-form deformations. In *Graphics Interface*, 35–42.

SLOAN, P., ROSE, C., AND COHEN, M. 2001. Shape by example. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, ACM Press / ACM SIGGRAPH, ACM.

SOFTIMAGE. 1992. *Softimage V2.51 User's Guide*. Montreal, Ontario, Canada.

WILHELMS, J., AND GELDER, A. V. 1997. Anatomically based modeling. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 173–180.

# Multi-Weight Enveloping: Least-Squares Approximation Techniques for Skin Animation

Xiaohuan Corina Wang
Industrial Light & Magic

Cary Phillips
Industrial Light & Magic

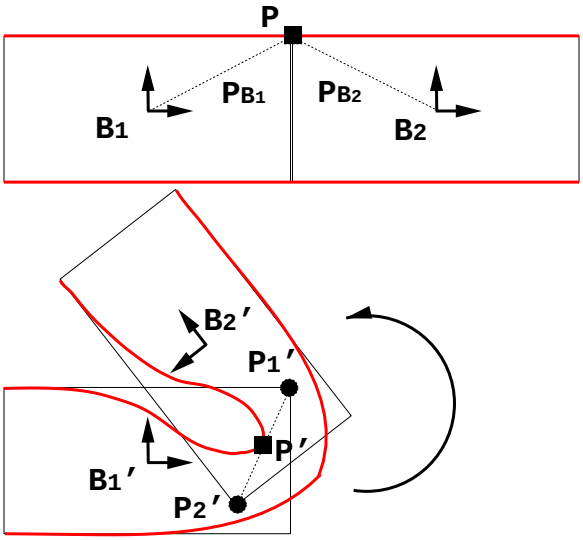



Figure 2: Single-Weight Enveloping. The top figure shows two skin surfaces (red) enveloped to two coordinate frames, $B_1$ and $B_2$, at the rest position. The bottom figure shows how the skin (red) deforms as the angle between $B_1$ and $B_2$ changes, with $w_1 = w_2 = 0.5$.

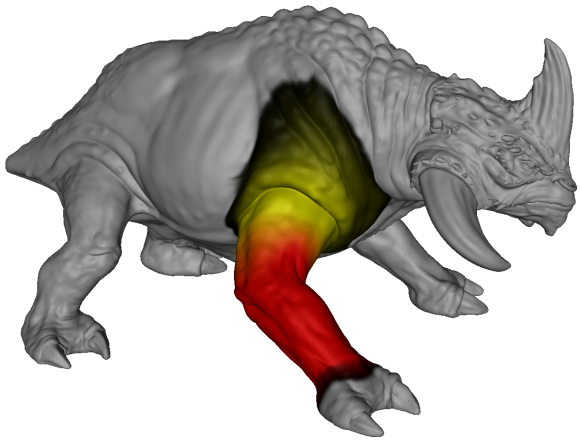


Figure 4: Color coded influences maps from two bones on the animal creature. The yellow area for the upper leg and the red area for the lower leg. The darker the area, the smaller the influence.

**9-a**: Multi-Weight on A New Animation Sequence (frame 42)

**9-b**: Multi-Weight on Another New Animation Sequence (frame 17)

Figure 9: Multi-weight Skinning on the Animal Creature