HAL
open science

A Muscle Model for Enhanced Character Skinning

Juan Ramos, Caroline Larboulette

To cite this version:

Juan Ramos, Caroline Larboulette. A Muscle Model for Enhanced Character Skinning. Journal of WSCG, 2013, 21 (2), pp.107–116. hal-00911527

HAL Id: hal-00911527

https://hal.science/hal-00911527v1

Submitted on 2 Dec 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Muscle Model for Enhanced Character Skinning

Juan Ramos
Universidad Rey Juan Carlos
Modeling and Virtual Reality Group
C/ Tulipàn, s/n
28933, Mòstoles, Spain
Tomillo1235@gmail.com

Caroline Larboulette
University of South Brittany
IRISA-UBS Lab
Campus de Tohannic
56000, Vannes, France
Caroline.Larboulette@gmail.com

ABSTRACT

This paper presents a novel method for deforming the skin of 3*D* characters in real-time using an underlying set of muscles. We use a geometric model based on parametric curves to generate the various shapes of the muscles. Our new model includes tension under isometric and isotonic contractions, a volume preservation approximation as well as a visually accurate sliding movement of the skin over the muscles. The deformation of the skin is done in two steps: first, a skeleton subspace deformation is computed due to the bones movement; then, vertices displacements are added due to the deformation of the underlying muscles.
We have tested our algorithm with a GPU implementation. The basis of the parametric primitives that serve for the muscle shape definition is stored in a cache. For a given frame, the shape of each muscle as well as its associated skin displacement are defined by only the splines control points and the muscle's new length. The data structure to be sent to the GPU is thus small, avoiding the data transfer bottleneck between the CPU and the GPU. Our technique is suitable for applications where accurate skin deformation is desired as well as video games or virtual environments where fast computation is necessary.

Keywords

Muscle, Skinning, Character Animation, Real-Time, GPU

1 INTRODUCTION

While the rendering of static scenes can now be so realistic that it is very often difficult to distinguish virtual images from real photographs, the same isn't true for animation, especially for characters. Despite the tremendous progress that has been made, anybody can tell the difference between a video of a real person in movement and a 3*D* character animation. This is because a character is a complex object composed of many different parts in interaction and a character is familiar to human beings which leaves little room for inaccuracy.

It is common practice to decompose the animation of a character into two problems: the animation of the skeleton and the animation of what's around it. In this paper, we address the second one. By describing the individual layers composing an object rather than just its surface, we believe that it is possible to generate more accurate deformations of the outside skin layer.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

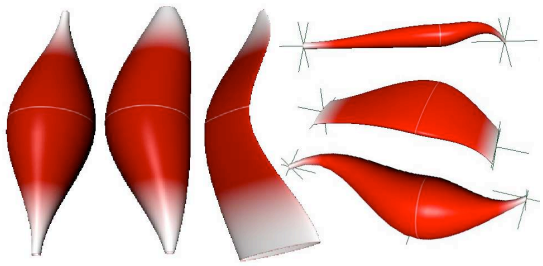

Figure 1: Various muscle shapes generated with our Maya plugin. From left to right: original template shape; non-linear sweeping spline; flat muscle; arbitrary muscles. The white line shows the position of the muscle belly center **C**.

Anatomically based models have first been introduced to Computer Graphics in 1992 by Chen and Zeltzer [Che92a] who modeled an accurate frog's muscle using finite elements. Subsequent simpler models have then been used with the sole goal of enhancing 3*D* characters' skin deformation through the animation of individual muscles. In contrary to what happens in real life, they deform as a response to bone deformation and their only purpose is to obtain visually more appealing deformations. Although that approach has been used for movie production (Shrek, Lord of the Rings [Gra]) and is included in recent versions of

Autodesk Maya Software [May13a] it has remained computationally too expensive and too difficult to set up for a wider use including video games or virtual environments. We propose a new muscle model that addresses both problems.

We do not intend to model dynamic effects or wrinkles that can be added as additional layers [Cap02a, Lar04a]. We do however provide a *tension* parameter that may be used to control a soft tissues dynamics simulation such as [Jam02a, Lar05a]. These additional layers are outside the scope of this paper that aims at presenting the muscle model and it's GPU implementation.

Our muscle model is defined by a few parameters that allow artists to express complicated shapes (see figure 1). Although it is best suited for fusiform and multipennate (broad origins and insertions) muscle shapes, multi-belly muscles such as the *pectoralis mayor* or the *trapezius* can be generated by grouping several fusiform shapes like in [Sch97a].

Animating muscle deformation as a response to bone movement has the advantage that it resembles the way artists work, but this implies that skin deformation due to the tension of the underlying muscles is not taken into account. To overcome this problem, we introduce a muscle *tension* parameter allowing the muscle to change the shape of its cross-section while preserving its overall volume. In addition, we achieve a sliding behavior of the skin over the muscles by restricting the displacement of each skin vertex to directions perpendicular to the muscles' fibers which visually improves the results obtained by previous geometric models where the skin vertices were rigidly attached to the muscles primitives.

To address the challenges of accurately animating anatomically based character models in real-time, we introduce (1) a new parametric fusiform muscle model capable of both isometric and isotonic contraction that preserves its overall volume (section 3), (2) a new geometric algorithm for skin deformation that achieves skin sliding behavior without the high cost of a physically based simulation (section 4), and (3) a GPU implementation of our model including normal correction that is suitable for use in video games (section 5). We achieve frame rates of 25 Hz on a dual core 3.0 GHz Pentium 4 PC equipped with a GeForce 8800 GTX graphics board for 25 instances of a mesh composed of 82000 triangles and 56 muscles, which represents a computational overhead of $20 - 25\%$ compared to a smooth skinning only.

2 RELATED WORK

The modeling and animation of individual muscles has been an extensive topic of research for a few decades. For a comprehensive state-of-the-art, the reader may refer to [Lee12a].

Previous work range from anatomically accurate models to models only vaguely faithful to the real anatomy. On the one hand, we find the muscle models that serve as a mean to move the bones of the skeleton. Because they are difficult to control in their actual state, those models are still marginal although they continue to receive some attention [Lee06a]. A complete biomechanical model of the upper body can be found in [Lee09a]. On the other hand, we find techniques that try to mimic some muscle deformation of the skin although either no individual muscle is defined [Wan02a, Moh03a, Cap07a] or the muscles do not correspond to real ones [Pra05a]. The skinning based techniques give nice results provided the user designs accurate input shapes [Wan02a]. However, with only anisotropic scaling [Cha89a, Wan02a, Moh03a] along X, Y and Z-axis, it is very difficult to obtain realistic muscle deformations. [Wan07a, Web07a, San08a] present data driven techniques for synthesizing skin deformations from skeletal motion. While the results produced are definitely more accurate, those techniques require a set of examples and thus cannot be applied to characters that need exaggerated deformations or for movements the actor did not perform or the animator did not create.

The alternative is to define individual muscles that serve as a mean to deform the outer skin layer but do not control the bones and the skeleton movement like in reality. Our work fits into this paradigm. In 1997, Wilhelms and Scheepers et al. [Wil97a, Sch97a] both proposed a first generic muscle model based on ellipsoids. While the results were visually appealing at that time, there were several limitations to the model. Firstly, the shape of the muscle belly and of the tendons was constrained to elliptical ones which made it hard to cover all of the body muscles, and even the *biceps* and *triceps* of the arm could not be correctly approximated. Subsequent parametric models [Wil97b, Lee07a] tried to solve the setup problem. Parametric muscles that have been designed for one character are easy to re-use onto another character as only a few scalings need to be done. Our muscle model is also based on the idea that only a small set of parameters is sufficient to create a wide variety of shapes. Moreover, unlike [Lee07a], our parameterization is suitable for a GPU implementation.

More recently, physically based models of muscles have been investigated. Whether they are based on mass-spring systems [Ned00a, Aub00a], finite elements methods [Zhu98a] or finite volume methods [Ter03a, Ter05a] they are difficult to set up and several orders of magnitude slower than the model we propose. In addition, the muscle shape is defined by either a polygonal mesh or a volumetric decomposition in tetrahedra which is totally unsuitable for the modern GPU pipelines.

To deform the skin and obtain a sliding effect of the skin over the muscles, [Wil97b] uses masses and springs to model the skin as well as to anchor the skin to the underlying muscles and bones. While this gives pleasing results, this step is very heavy in terms of setup (stiffness and damping coefficients) and computation. Turner et al. [Tur93a] use a physically based simulation of elasticity to deform an elastic skin. We avoid the heavy cost of a physically based simulation by proposing a new geometric algorithm to displace the vertices of the skin as a response to the muscles deformation, which allows sliding effects. This is an improvement compared to other geometric models where the skin is rigidly anchored to underlying components [Wil97b, Lee07a].

3 OUR MUSCLE MODEL

Similar to previous models [Wil97b, Ned00a, Aub00a, Lee07a], we make the assumption that the force exerted by the muscle follows a fictive *action line* $A(t)$ that lies on the main axis of the muscle shape. This action line is anchored to the bones at the *origin* and *insertion* points, thus elongating and shortening as the skeleton moves. The muscle belly is attached through deformable tendons at both ends.

Overview The general shape of our muscle model is a modified generalized cylinder defined by the sweep of a *thickness curve* $C_T(t)$ along a *sweeping curve* $C_S(t)$. $C_S(t)$ is a three-dimensional spline representing the profile of the muscle. It is defined by two $3D$ points: the origin $\mathbf{O}$ and insertion $\mathbf{I}$. $C_T(t)$ is a one-dimensional function representing the muscle's thickness as a function of t. The muscle's cross-section at t is thus defined as an ellipse placed along $C_S(t)$ whose thickness is given by $C_T(t)$ and a scaling factor $\mathbf{s_f}$. $\mathbf{s_f}$ is applied in a user-chosen direction $\overrightarrow{\mathbf{S_d}}$ perpendicular to the action line. This scaling is only applied to the sections, not to the spline curves. See figure 2 for a sketch.

To avoid intersections between sections and to simplify the volume conservation computation detailed in section 3.2.1, the sections do not follow the curvature of $C_S(t)$ but are always perpendicular to the action line $A(t)$.

Modeling We have implemented a Maya plugin that allows the user to modify the curves control points, the scaling factor $\mathbf{s_f}$ as well as the scaling direction $\overrightarrow{\mathbf{S_d}}$. As we will see in section 3.1, the control points are constrained and cannot be freely moved, which reduces the number of variables to control. In addition, the muscle belly center $\mathbf{C}$ can be displaced along the segment $\mathbf{OI}$. It is thus possible to obtain a wide variety of asymmetrical shapes. See figure 1 for examples.

3.1 Shape Computation

$C_S(t)$ and $C_T(t)$ Bézier curves defining the muscle shape are each composed of two cubic spline segments

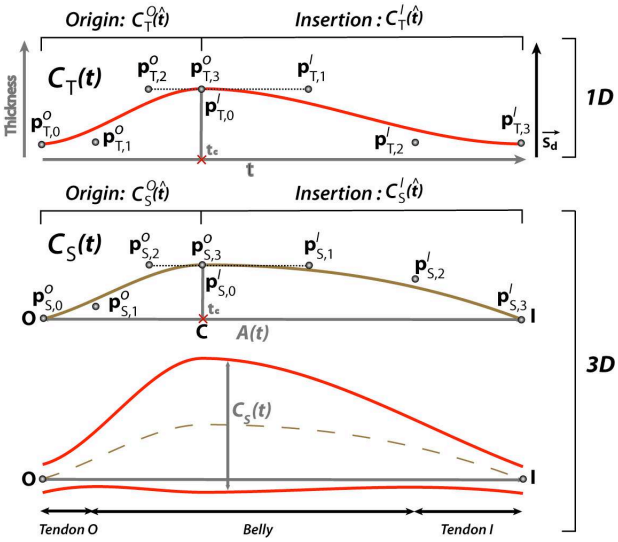

Figure 2: Top: *thickness curve* $C_T(t)$; center: *sweeping curve* $C_S(t)$; bottom: muscle's longitudinal section resulting from the sweeping of $C_T(t)$ along $C_S(t)$ with $C_S(t)$ represented in dashed line.

$C_S^O(\hat{t}), C_S^I(\hat{t})$ and $C_T^O(\hat{t}), C_T^I(\hat{t})$ respectively called *origin* and *insertion segments*, joined with $\mathscr{G}^1$ continuity. Mapping between t and $\hat{t}$ is explained later. $C_T(t)$ is a piecewise one-dimensional function which represents the varying thickness of the muscle at a parametric point $t \in [0,1]$ along $C_S(t)$.

Each Bézier cubic spline segment is defined by four control points $\mathbf{p}_j$, $j \in [0,3]$ and the cubic Bernstein polynomials $B_j^3(\hat{t})$ by:

$$\begin{aligned} C(\hat{t}) &= \textstyle\sum_{j=0}^{3} \mathbf{p}_j B_j^3(\hat{t}) \\ &= (1-\hat{t})^3\mathbf{p}_0 + 3\hat{t}(1-\hat{t})^2\mathbf{p}_1 + \\ &\quad 3\hat{t}^2(1-\hat{t})\mathbf{p}_2 + \hat{t}^3\mathbf{p}_3 \end{aligned} \tag{1}$$

Henceforward we will write $\mathbf{p}_{0-3}^O$ and $\mathbf{p}_{0-3}^I$ the four control points of the *origin* and *insertion* spline segments respectively, regardless of their dimension.

$\mathscr{G}^1$ continuity is obtained by merging $\mathbf{p}_3^O$ and $\mathbf{p}_0^I$ and aligning $\mathbf{p}_2^O$, $\mathbf{p}_3^O$ and $\mathbf{p}_1^I$ in the cubic spline segments for both the *thickness* and the *sweeping* curves (see figure 2). $\mathbf{p}_0^O$, $\mathbf{p}_1^O$, $\mathbf{p}_2^I$ and $\mathbf{p}_3^I$ can take any arbitrary values chosen by the user.

The action line $A(t)$ defined by the origin $\mathbf{O}$ and the insertion $\mathbf{I}$ of the muscle can be expressed by the following parametric equation:

$$A(t) = \mathbf{O} + t(\mathbf{I} - \mathbf{O}) \quad t \in [0,1] \tag{2}$$

Let $\overrightarrow{\mathbf{S}}$ be the unit vector perpendicular to both $\overrightarrow{\mathbf{S_d}}$ and $A(t)$:

$$\overrightarrow{\mathbf{S}} = \frac{\overrightarrow{\mathbf{OI}} \times \overrightarrow{\mathbf{S_d}}}{\|\overrightarrow{\mathbf{OI}} \times \overrightarrow{\mathbf{S_d}}\|} \tag{3}$$

The orthogonal basis $\mathscr{B} = \{\overrightarrow{\mathbf{S}}; \overrightarrow{\mathbf{S_d}}; \overrightarrow{\mathbf{OI}}\}$ defines the *section space*. Each of $C_S(t)$ control points, $\mathbf{p}^O_{S,0-3}$ and $\mathbf{p}^I_{S,0-3}$, can be defined in *section space* by projecting them onto $\mathscr{B}$ orthogonal vectors.

Efficient skin vertices displacement as a response to muscles deformation such as described in section 4.2 relies on having a direct mapping of the parametric space of the action line onto the parametric space of the spline segments. This is simply obtained by making local Bézier parameter $\hat{t}$ linearly dependent on t. Thanks to the linear precision property of the Bézier spline, uniformly distributed control points on a straight line produce that straight line:

$$\sum_{j=0}^{n} \frac{j}{n} B^n_j(t) = t \tag{4}$$

In our case $n = 3$. The control points $\mathbf{p}^O_{S,0-3}$ on $C^O_S(t)$ are spaced by $\|\overrightarrow{\mathbf{OC}}\|/3$ along its t coordinate and the control points $\mathbf{p}^I_{S,0-3}$ on $C^I_S(t)$ are spaced by $\|\overrightarrow{\mathbf{CI}}\|/3$. In *section space*, $\overrightarrow{\mathbf{OI}}$ coordinate is t, so $\mathbf{p}^O_{S,0-3}$ and $\mathbf{p}^I_{S,0-3}$ positions can be expressed with only two coordinates plus t.

Let $\mathbf{C} = A(t_c)$ be the muscle belly center such as $t_c = \|\overrightarrow{\mathbf{OC}}\|/\|\overrightarrow{\mathbf{OI}}\|$. The local $\hat{t}$ is obtained by:

$$\hat{t} = \begin{cases} t/t_c & \text{if } t <= t_c, \\ (t-t_c)/(1-t_c) & \text{if } t > t_c. \end{cases} \tag{5}$$

and used in $C^O_S(\hat{t})$, $C^O_T(\hat{t})$ and $C^I_S(\hat{t})$, $C^I_T(\hat{t})$ respectively to define the shape of the muscle as a function of t along the action line.

3.2 Muscle Deformation

Muscle is an incompressible, anisotropic, hyper-elastic material that undergoes two types of contractions: isotonic and isometric. In both cases, the overall volume of the muscle remains roughly constant [Wil97b].

In the case of an *isotonic* contraction, the bones move but the tension of the muscle remains constant. The change in shape is thus only due to the shortening/elongation of the muscle. If it shortens, the belly of the muscle bulges. It decreases in the other case. The action line, anchored to the bones at **O** and **I**, changes length as the skeleton moves. In order to preserve the volume of the muscle, we recompute the new belly thickness while keeping tendons thicknesses constant. This is detailed in section 3.2.1.

In the case of an *isometric* contraction, the bones do not move and the muscle length doesn't change. However, the shape of the muscle is modified. When the tension increases, the muscle bulges in one direction while narrowing in the perpendicular dimension. Tendons also bulge in the same direction and become more visible. The muscle and the tendons all relax to the rest state when the tension decreases. Again, the overall volume of the muscle remains constant. This is achieved through a tension parameter **tension** that allows the muscle to scale in the direction $\overrightarrow{\mathbf{S_d}}$ chosen by the artist. The inverse of that scale is applied in the perpendicular direction $\overrightarrow{\mathbf{S}}$. This is detailed in section 3.2.2.

In real life, both types of contraction happen at the same time. First, the tension of the muscle increases. When it reaches a certain threshold, the bones start moving while the tension decreases until the bones reach the new position. By moving the bones and varying the tension parameter of our model, it is possible to achieve this natural behavior.

3.2.1 Volume Preservation

As muscle sections are parallel to each other along the sweeping curve, two consecutive sections form the elliptic bases of an oblique truncated cone of height the distance between the two sections. The volume of the muscle is thus equal to the sum of an infinite number of such cones of infinitely small height. The volume of an oblique truncated cone is the same as the one of a right cone, therefore we can compute the volume as if $C_S(t)$ were straight and equal to $A(t)$.

Cone volume computation is costly and we have found that a very coarse discretization of the muscle's volume in only two truncated elliptic cones was a good and very stable approximation (see figure 3). Indeed, it is more important to keep the differential volume error low between the different muscle states (elongated, rest, compressed) rather than having an exact volume. For the muscles we used in our demos, we have measured an error of 10% in average with respect to the exact volume computation but only a 1% error deviation between extreme muscle states (150% elongated, 50% shortened). This means that the volume remains constant with an error of only 1%, which is negligible.

Let (a_X, b_X) be the axes of the three ellipses defining the two cones at the origin **O**, center **C**, and insertion **I** points. The a_X axes are aligned with the scaling direction $\overrightarrow{\mathbf{S_d}}$ and their length is $\mathbf{a}_X$. b_X axes are perpendicular to the a_X ones and their lengths $\mathbf{b}_X$ are defined as $\mathbf{b}_O = \mathbf{p}^O_{T,0}$, $\mathbf{b}_C = \mathbf{p}^O_{T,3} = \mathbf{p}^I_{T,O}$ and $\mathbf{b}_I = \mathbf{p}^I_{T,3}$. The ratios $\mathbf{a}_X/\mathbf{b}_X = \mathbf{s_f}$.

The total volume of the muscle is computed by equation (6) which, divided by $\mathbf{s_f}$, results in equation (7).

$$\begin{aligned} V &= \|\overrightarrow{\mathbf{OC}}\|\pi\left(\frac{2\mathbf{a}_C\mathbf{b}_C+\mathbf{a}_C\mathbf{b}_O+\mathbf{a}_O\mathbf{b}_C+2\mathbf{a}_O\mathbf{b}_O}{6}\right) \\ &+ \|\overrightarrow{\mathbf{CI}}\|\pi\left(\frac{2\mathbf{a}_C\mathbf{b}_C+\mathbf{a}_C\mathbf{b}_I+\mathbf{a}_I\mathbf{b}_C+2\mathbf{a}_I\mathbf{b}_I}{6}\right) \end{aligned} \tag{6}$$

$$\begin{aligned} V &= \frac{\pi\mathbf{s_f}}{3}(\|\overrightarrow{\mathbf{OC}}\|(\mathbf{b}_C{}^2+\mathbf{b}_O{}^2+\mathbf{b}_C\mathbf{b}_O) \\ &+ \|\overrightarrow{\mathbf{CI}}\|(\mathbf{b_C}{}^2+\mathbf{b}_I{}^2+\mathbf{b}_C\mathbf{b}_I)) \end{aligned} \tag{7}$$

The initial volume is computed at rest state. To keep a constant volume as well as a constant thickness for the tendons (desired feature for muscles deforming under isotonic contraction), $\mathbf{b}_C$ needs to be recomputed at each timestep. From (7), we obtain a quadratic equation $A\mathbf{b}_C{}^2 + B\mathbf{b}_C + C = 0$ with:

$$\begin{aligned} A &= \|\overrightarrow{\mathbf{OI}}\| \\ B &= \|\overrightarrow{\mathbf{OC}}\|\mathbf{b}_O + \|\overrightarrow{\mathbf{CI}}\|\mathbf{b}_I \\ C &= \|\overrightarrow{\mathbf{OC}}\|\mathbf{b}_O{}^2 + \|\overrightarrow{\mathbf{CI}}\|\mathbf{b}_I{}^2 - \frac{3V_{initial}}{\pi \mathbf{s_f}} \end{aligned} \tag{8}$$

It can be solved by taking its positive solution $\mathbf{b}_C{}' = \frac{-B+\sqrt{\delta}}{2A}$ with $\delta = B^2 - 4AC$.

As mentioned in section 3.1, to keep the belly shape continuous, $\mathbf{p}^O_{T,2}$, $\mathbf{p}^O_{T,3}$ and $\mathbf{p}^I_{T,1}$ must be kept collinear as $\mathbf{b}_C$ changes and the slope of segment $\mathbf{p}^O_{T,2}\mathbf{p}^I_{T,1}$ must also be kept constant.

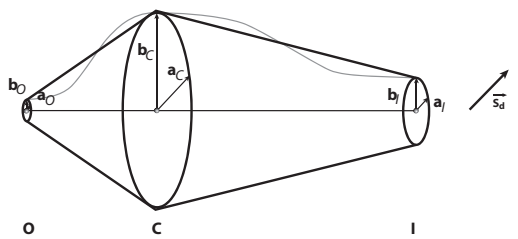

Figure 3: The volume of the muscle is approximated by the volume of two elliptic cones defined by $(\mathbf{O}, \mathbf{a}_O, \mathbf{b}_O)$, $(\mathbf{C}, \mathbf{a}_C, \mathbf{b}_C)$ and $(\mathbf{C}, \mathbf{a}_C, \mathbf{b}_C)$, $(\mathbf{I}, \mathbf{a}_I, \mathbf{b}_I)$.

3.2.2 Tension Parameter

From equation (7) it is straightforward that to maintain a constant volume, the scaling factor $\mathbf{s_f}$ should be kept constant. As $\mathbf{s_f} = \mathbf{a}_X/\mathbf{b}_X$, this can be achieved by multiplying axes lengths $\mathbf{a}_X$ by **tension** while dividing axes lengths $\mathbf{b}_X$ by the same amount at the origin, center and insertion points.

The new axes lengths $\tilde{\mathbf{a}}_X$ and $\tilde{\mathbf{b}}_X$ when the muscle is under isometric contraction can thus be expressed as follows:

$$\begin{aligned} \tilde{\mathbf{a}}_X &= \mathbf{a}_X * \mathbf{tension} \\ \tilde{\mathbf{b}}_X &= \mathbf{b}_X / \mathbf{tension} \end{aligned} \tag{9}$$

Figure 4 shows a muscle in isometric contraction for different values of the tension parameter. At rest state (left), **tension** $= 1$. The muscle is under tension for **tension** > 1, and, for increasing values of **tension**, the muscle bulge is more noticeable (middle and right).

At any time during animation, both types of contraction are involved. The volume of the muscle is preserved by first applying equation (7) that accounts for changes in length of the action line, then equation (9) that takes the strain into account.

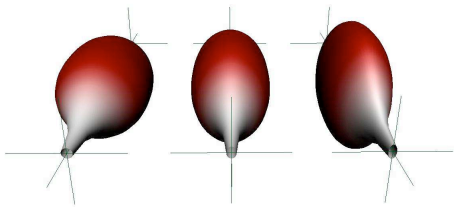

Figure 4: Varying tension parameter. From left to right: **tension** $= 1$, **tension** $= 1.15$ and **tension** $= 1.3$.

4 SKIN DEFORMATION & RENDERING

The skin/clothes layer is the only visible layer on a character. The muscles serve as an underlying tool to help deform the skin in a more accurate way. We use a two-steps approach. First, the skin layer is modified using a linear blend skinning as a response to the skeleton's movement. Note that other skinning techniques [Kav05a, Kav07a, Kav08a] can be used. The skin layer is subsequently deformed as a response to the muscles deformation by applying a displacement to each influenced vertex. We perform this step on the GPU.

The way a muscle deforms the surrounding skin is crucial to obtain good results. As the relationship between the skin and the muscle can be very different depending on the part of the body affected, we introduce a weighted deformation scheme as well as a fat layer that allow the animator to adjust the skin deformation. The fat layer is in charge of keeping a distance between the skin and the underlying muscle and more or less attenuates the deformation due to the muscle bulging depending on the chosen weighting system (section 4.1).

To achieve a skin sliding behavior over the muscles, we propose a vector oriented vertices displacement (section 4.2).

4.1 Muscle Influence

A skin vertex is influenced by a muscle if it lies in its neighborhood. This influence together with an associated weight are evaluated at the bind pose. In existing commercial programs or muscle-dedicated plugins the user paints the vertices influences by hand for each muscle. We provide an automatic way to assign normalized weights varying from 0 to 1 to skin vertices, a weight of 0 meaning no influence at all, and 1 meaning full muscle influence.

The influence computation is achieved by projecting perpendicularly each vertex of the concerned limb $\mathbf{V_i}$ onto the action line $A(t)$ of each muscle. If the projection point lies outside **OI**, then this particular vertex is not influenced by the muscle. In the other case the muscle's influence depends on a neighborhood template assigned by the user.

We provide three neighborhood templates:

Radial template: all of the vertices that lie within a given distance from the action line are influenced.

Half-space template: all of the vertices that lie on the upper side of the plane defined by the action line and having $\overrightarrow{\mathbf{S_d}}$ as a normal are influenced.

Fitted template: similar to the half-space template but with the additional restriction that the ray $(\mathbf{V_i}, -\overrightarrow{\mathbf{S_d}})$ must intersect the muscle shape for the vertex $\mathbf{V_i}$ to be influenced.

The user chooses a template depending on how he wishes the skin to be influenced by the muscle. Examples of the three templates are shown on figure 5, top row, for a given muscle.

If the vertex $\mathbf{V_i}$ is influenced by the muscle, its weight $\mathbf{w_i}$ is obtained depending on an automatic weighting system chosen by the user. We have implemented two of them:

Binary weighting system: the weight $\mathbf{w_i}$ of $\mathbf{V_i}$ is set to 1 if the vertex $\mathbf{V_i}$ is influenced by the muscle and 0 in the other case.

Spline-based weighting system: the weight is obtained by dividing the muscle thickness $C_T(t)$ at $\mathbf{V_i}$ by $max(\mathbf{p}^O_{T,2}, \mathbf{p}^O_{T,3}, \mathbf{p}^I_{T,1})$. We obtain weights $\mathbf{w_i} \in [0,1]$ that depend on the local thickness of the muscle.

These weighting systems must be used depending on the expected deformation of the muscle on the skin (see figure 5, bottom row). The first one is indicated for very fitted zones with almost no fat, so the tendons can be seen as well as almost the whole shape of the muscle. Note that the skin deformation is continuous and smooth because the muscle shape is $\mathscr{G}^1$ continuous and the skin follows that shape. The second one attenuates the deformation of the skin with respect to the previous one, and is indicated for fatty zones were only a small bulge due to the muscle is expected. The computed weights are also smooth, so the deformation remains continuous.

Figure 5: From left to right: *radial* influence template, *half-space* template, and *fitted* template. Top row shows the corresponding *binary* weights while bottom row shows the *spline-based* weights.

4.2 Skin Vertices Displacement

Vector oriented displacement makes the skin slide over the muscles by preserving a distance between them without rigidly anchoring the skin vertices to the muscles' primitives. To explain the deformation technique we first suppose that there is no fat layer and that all vertices have a weight of 1. It results in the skin being displaced to fit the muscles surfaces. How the fat layer and weights affect the vertices displacement is explained in section 4.3.

Let $\mathbf{V_i}$ be the current skin vertex we want to displace onto the muscle's surface $\mathbf{V'_i}$. This vertex is displaced in $\overrightarrow{\mathbf{S_d}}$ direction defined by the user as the main direction of bulging.

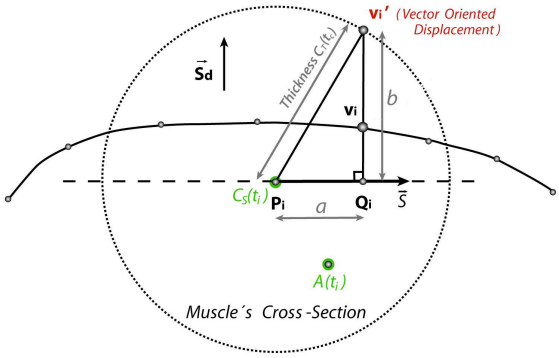

Figure 6: Skin vertices displacement algorithm: $\mathbf{V_i}$ is displaced in order to lie on the muscle's surface at $\mathbf{V'_i}$.

The algorithm works as follows (see figure 6): we first compute $\mathbf{P_i} = C_S(t_i)$ as the central point of a muscle section on $C_S(t)$ curve. The muscle's section on which we solve the intersection now contains $\mathbf{P_i}$ and $\mathbf{V_i}$. Let $C_T(t_i)$ be the thickness of the muscle at the point $\mathbf{P_i}$ and $\overrightarrow{\mathbf{S}}$ be the unit vector perpendicular to both the action line $A(t)$ and the displacement vector $\overrightarrow{\mathbf{S_d}}$ (see equation (3)).

The point $\mathbf{Q_i}$ is defined as the projection of $\mathbf{V_i}$ onto $\overrightarrow{\mathbf{S}}$. The distance from $\mathbf{P_i}$ to $\mathbf{Q_i}$ can simply be expressed as $\mathbf{a} = \overrightarrow{\mathbf{P_i V_i}} \cdot \overrightarrow{\mathbf{S}}$. Because $\mathbf{a}$ and $\mathbf{b}$ (see figure 6) form a right triangle, the new position of vertex $\mathbf{V_i}$ is given by the two following equations:

$$\mathbf{b} = \sqrt{C_T(t_c)^2 - \mathbf{a}^2} \tag{10}$$

$$\mathbf{V'_i} = \mathbf{Q_i} + \overrightarrow{\mathbf{S_d}}\,\mathbf{b} \tag{11}$$

At this point we have found $\mathbf{V'_i}$ for a circular muscle section. To make the algorithm work with elliptical sections we only have to multiply $\mathbf{b}$ by $\mathbf{s_f}$ and **tension**.

$$\mathbf{V'_i} = \mathbf{Q_i} + \overrightarrow{\mathbf{S_d}} \cdot \mathbf{b} \cdot \mathbf{s_f} \cdot \mathbf{tension} \tag{12}$$

4.3 Fat Layer and Weighted Deformation

We have introduced vertex weights $\mathbf{w_i} \in [0,1]$ in section 4.1 to attenuate the effect of muscles on the deformation of the skin. In the case where no fat layer

exists, those weights directly influence the vertex displacement $||\overrightarrow{\mathbf{V_i V_i'}}||$. The new vertex position $\mathbf{V_i''}$ is thus given by:

$$\mathbf{V_i''} = \mathbf{V_i} + \overrightarrow{\mathbf{V_i V_i'}}\, \mathbf{w_i} \tag{13}$$

However, muscles can deform the skin through a fat layer. Our implementation of the fat layer consists in calculating a distance $\mathbf{L_{fat}}$ between the muscle and the skin at bind pose and keeping it constant during animation. To keep distance consistency $\mathbf{L_{fat}}$ must measure vector oriented distance to the muscle.

In that case, the weights $\mathbf{w_i}$ are applied to the fat layer offset $\mathbf{L_{fat}}$ and the new vertex position $\mathbf{V_i''}$ is given by:

$$\mathbf{V_i''} = \mathbf{V_i} + \frac{\overrightarrow{\mathbf{V_i V_i'}}}{||\overrightarrow{\mathbf{V_i V_i'}}||}\,(||\overrightarrow{\mathbf{V_i V_i'}}|| + \mathbf{L_{fat}}\, \mathbf{w_i}) \tag{14}$$

4.4 Normal Correction

Visual appearance of 3D characters is dramatically improved by using several textures for diffuse lighting, bump mapping, and specular mapping. To apply those algorithms, it is necessary to have the normal and binormal vectors for each mesh vertex. When a muscle displaces a vertex $\mathbf{V_i}$ into a new position, the normal $\overrightarrow{\mathbf{N_i}}$ obtained from the previously applied bone skinning technique must be recomputed (see figure 7).

Figure 7: Left: displaced wireframe mesh; center: same mesh with texture applied but no normal correction; right: same mesh with texture and normal correction.

As the normals of the muscle shape can't be directly mapped to the skin due to the fat layer and influence weighting, the ideal solution is to recompute the skin normals from vertex adjacency data. Current GPUs only have edge adjacency available at the input of the pipeline, so we have used a pre-calculated buffer that contains the adjacency information for each vertex.

The character deformation is divided into two passes: the first pass computes the mesh deformation due to the skinning and the muscles; and the second pass renormalizes the normals by looking up in the adjacency buffer and computing regular per vertex normals. We need to recalculate the normals for the entire mesh as skeleton subspace transformed normals are different (see figure 8).

Figure 8: Arm example without muscles. Left: skeleton subspace normals calculation for the whole mesh; center: a discontinuity can be seen when using different normal calculation algorithms; right: normal calculation mask used in the central picture: in black, GPU skeleton subspace normals calculation; in grey, renormalization by computing normals per vertex.

5 GPU IMPLEMENTATION AND RESULTS

We have implemented the vertices displacements due to muscle deformation on a vertex program on the GPU. Two key points make our implementation run in real-time: the use of a parametric muscle model that needs to send little data to the GPU per frame, and the discretization of the Bézier spline basis to avoid high computational cost.

5.1 CPU-GPU data transfer

Muscle volume preserving and movement due to the skeleton animation is calculated on the CPU while the vertex bone skinning, muscle displacement and normal correction are computed on the GPU. Table 1 lists the data sent per animation timestep to the GPU for each muscle. While some of this data is redundant for reducing the amount of computation on the GPU, the total size is of only 10 vectors per muscle per frame.

Type	Information	Symbol	Size (bytes)
float3	origin	$\mathbf{O}$	12
float4	unit side vector center point	$\overrightarrow{\mathbf{S}}$ $\mathbf{C}$	16
float4	action line and its size	$\overrightarrow{\mathbf{OI}}$, $\|\overrightarrow{\mathbf{OI}}\|$	16
float4	unit scale direction, scale amount	$\overrightarrow{\mathbf{S_d}}$, $\mathbf{s_f}$	16
float4	C_T^O control points	$\mathbf{p}_{T,0-3}^O$	16
float4	C_T^I control points	$\mathbf{p}_{T,0-3}^I$	16
float4	C_S^O control points X	$\mathbf{p}_{S,0-3}^O$	16
float4	C_S^O control points Y	$\mathbf{p}_{S,0-3}^O$	16
float4	C_S^I control points X	$\mathbf{p}_{S,0-3}^I$	16
float4	C_S^I control points Y	$\mathbf{p}_{S,0-3}^I$	16
Total Size			**152**

Table 1: Size, in bytes, of the data sent to the GPU per muscle, per frame.

$C_T(t)$ control points are one-dimensional and can be stored in 2 vectors. $C_S(t)$ three-dimensional control

points need 6 vectors. By using the linear precision property (see equation (4)) we can save memory space by encoding $\mathbf{p}^O_{S,0-3}$ and $\mathbf{p}^I_{S,0-3}$ points in the $2D$ basis $\mathscr{B} = \{\overrightarrow{\mathbf{S}};\overrightarrow{\mathbf{S_d}}\}$, the missing dimension is implicitly defined by keeping a constant distance between $\mathbf{p}^O_{S,0-3}$ and $\mathbf{p}^I_{S,0-3}$ as described in section 3.1. The $\mathscr{B}$ basis is very efficient as the majority of GPU calculations are done in *section space*.

Data containing the index of the muscle influencing each vertex and the associated fat offsets $\mathbf{L_{fat}}$ is sent to the GPU only once with the rest of per-vertex data.

5.2 Bézier Spline Basis Discretization

To speed-up the computation of $C(\hat{t})$ (see equation (1)), we store the basis functions of the spline on the GPU in an array of size n. We discretize the basis for n values t_i of $\hat{t}$ and we store the results in $[1 \times 4]$ vectors of the form:

$$[\mathbf{B}^3_0(\mathbf{t_i}), \mathbf{B}^3_1(\mathbf{t_i}), \mathbf{B}^3_2(\mathbf{t_i}), \mathbf{B}^3_3(\mathbf{t_i})] \tag{15}$$

In practice, we have found $n = 100$ and $t_i = 1/n$ to produce smooth deformations. Increasing the discretization only increases the buffer's size but has no influence on the computation time and doesn't visually improve the results.

During animation, the projection of each vertex $\mathbf{V_i}$ onto the action line $A(t)$ gives the value of t for this vertex. We obtain $\hat{t}$ from equation (5). The thickness at $\hat{t}$ is the result of the dot product of the closest t_i spline basis vector we have previously stored with vector $\mathbf{p}^O_{T,0-3}$ if $t <= t_c$ and $\mathbf{p}^I_{T,0-3}$ otherwise. $C_S(t)$ in $\mathscr{B}$ basis is computed in a similar way for each of the two dimensions.

Having $C_T(t)$ and $C_S(t)$ we compute $\mathbf{V'_i}$ by applying equation (12). The final skin deformation is calculated with equation (13) or (14). All these computations are performed on the GPU.

5.3 Results

All of the timings and animations presented in this section have been computed on a dual core 3.0 GHz Pentium 4 PC equipped with a GeForce 8800 GTX graphics board under Windows Vista using the DirectX 10 API.

5.3.1 *Male Gymnast*

We have tested our algorithm on a male gymnast walking and jumping. The model is composed of 82000 triangles. We have attached 56 muscles to the arms, legs, chest and neck of the character (see figure 9). It took about a day for a non-skilled animator to shape the muscles, attach them to the bones and choose appropriate skin influence templates. Note that the number of triangles greatly exceeds the one of standard meshes used in regular video games. This is due to the fact that higher resolution meshes are needed to correctly appreciate the muscles' influences. We obtain real-time for as many as 25 characters, which corresponds to a total of 1400 muscles animated using both isometric and isotonic contractions. We observe a loss of $20-25\%$ in the frame rate due to the use of muscles (17 fps) compared to the use of a linear blend skinning only (26 fps).

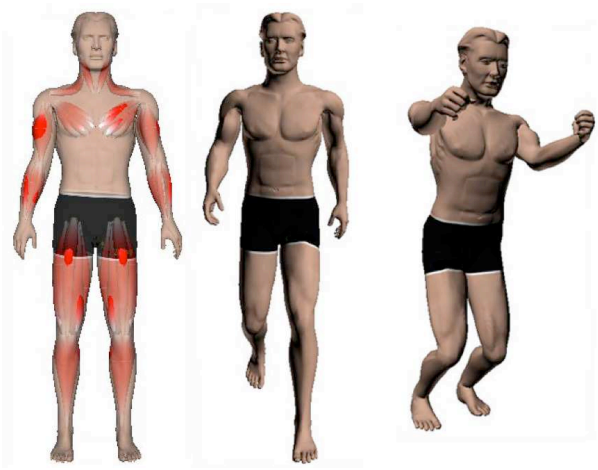

Figure 9: From left to right: *male gymnast* model composed of $82K$ polygons and 56 muscles (setup time about 8 hours); 2 frames from the GPU real-time animation.

5.3.2 *Dinosaur Leg*

We have also tested our algorithm on a simple dinosaur leg. The model is composed of 7800 triangles to which we have added 4 muscles (see figure 10).

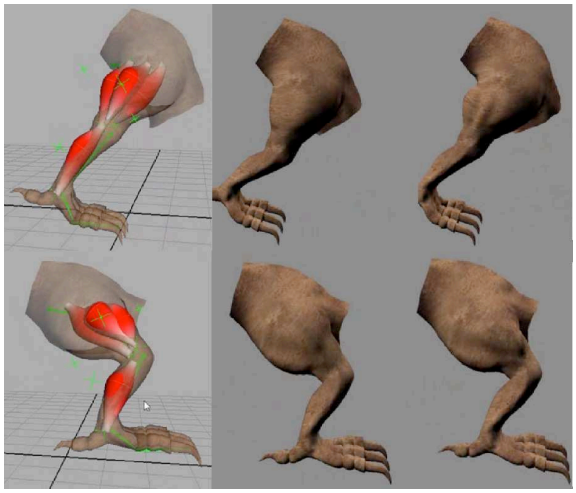

Figure 10: Two frames of the *dinosaur leg* animation. From left to right: design of 4 muscles under Maya (setup time about 20 min); Animation on GPU without muscles; Animation on GPU with muscles.

5.3.3 Crowd Animation

Finally, we have tested our GPU algorithm on wider crowds (see figure 11). Up to 200 instances for a total of 11200 muscles could be animated in interactive time. Note that no GPU instancing technique was used.

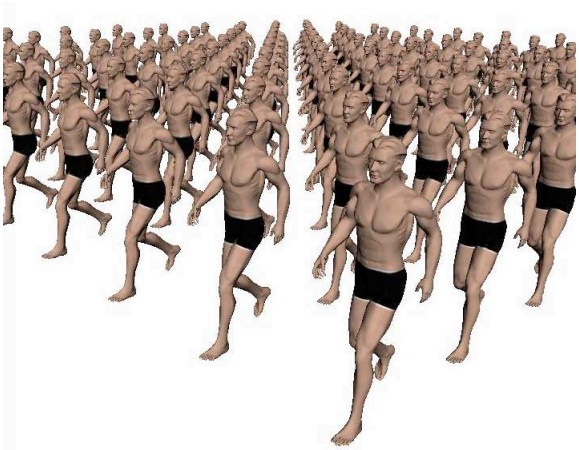

Figure 11: Crowd animation of 100 instances of the male character which corresponds to 5600 muscles total. Frame rate is 4.3 fps mainly limited by the high number of polygons: 8.2 millions.

Table 2 shows the frame rates obtained for different numbers of instances used for the man and dinosaur leg models.

Mesh	# of Instances	Total # of Muscles	Fps
Man (82*K* tri.)	20	1120	21
Man (82*K* tri.)	25	1400	17
Man (82*K* tri.)	50	2800	9
Man (82*K* tri.)	100	5600	4.3
Man (82*K* tri.)	200	14000	2.1
Leg (8*K* tri.)	100	300	32
Leg (8*K* tri.)	200	600	16
Leg (8*K* tri.)	400	1200	3.5

Table 2: Frames per second per animation for both models and a number of instances varying from 20 to 400.

6 CONCLUSION AND FUTURE WORK

We have presented a new parametric muscle model suitable for real-time character animation as well as a new skin vertices displacement algorithm as a fast alternative to physically based simulation of elasticity. Our model is suitable for both procedural animation and GPU animation. The user is provided with 2 spline curves that allow them to design a wide range of possible muscles shapes. We have thus reached our goals of accuracy, efficiency and user's usability. Compared to previous muscles models, our model is complete in the sense that all kinds of skeletal muscles can be generated (fusiform and flat); the skin layer is not rigidly attached to the muscles but slides over them without requiring the heavy cost of a physically based simulation; both isometric and isotonic contractions can be simulated thanks to our tension parameter; when deforming, the muscle preserves its volume; we offer a normal correction; our algorithm is suitable for Graphics Hardware.

Currently, there are two limitations to our work. The first one is that there is no muscle-muscle or muscle-bone interaction. While it may seem incorrect from an anatomical point of view, it may easily be overcome by the fact that the anchor points of our muscles do not need to be onto the bone. It is thus possible to keep the anchor points, hence the action line, from a certain distance to a bone or a muscle, which acts as if there were interactions. The important muscles to be modeled are the ones on the surface because they influence the outside appearance of the skin.

The second shortcoming is the modeling time. While adding muscles following an anatomy book is achievable by most users, it still takes some time to shape the muscles. One of our future work is to use medical data [Ter05a] to compute the initial shape of our muscles and provide retargetting, especially for characters of the same species.

In addition, we plan on combining our model with a model for dynamics of soft tissues whose behavior will be controlled though our *tension* parameter (when muscles are tense, the surrounding tissues jiggle less and vice-versa). We believe it should not be directly part of the muscle model because dynamics is not only due to the muscles, but also to fatty tissues. Last but not least, we would like to address the problem of automatically computing the tension of all of the muscles of a character given its animation through inverse dynamics.

7 REFERENCES

[Aub00a] A. Aubel and D. Thalmann. Realistic deformation of human body shapes. In *Proceedings of Computer Animation and Simulation 2000*, pages 125–135.

[Cap02a] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. In *Proceedings of SIGGRAPH'02, ACM ToG*, 21(3), pages 586–593.

[Cap07a] S. Capell, M. Burkhart, B. Curless, T. Duchamp, and Z. Popović. Physically based rigging for deformable characters. *Graph. Models*, 69(1):71–87, 2007.

[Cha89a] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable

animated characters. In *Proceedings of SIGGRAPH'89, ACM Computer Graphics*, 23(3), pages 243–252.

[Che92a] D. T. Chen and D. Zeltzer. Pump it up : Computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics*, 26(2), 1992.

[Gra] Z. Gray and M. Hutchinson. Procedural muscle and skin simulation. http://www.fourthdoor.com/muscle/.

[Jam02a] D. L. James and D. K. Pai. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of SIGGRAPH'02, ACM ToG*, 21(3), pages 582–585.

[Kav05a] L. Kavan and J. Zara. Spherical blend skinning: A real-time deformation of articulated models. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2005*, pages 9–16.

[Kav07a] L. Kavan, S. Collins, J. Zara, and C. O'Sullivan. Skinning with dual quaternions. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2007*, pages 39–46.

[Kav08a] L. Kavan, S. Collins, J. Zara, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4), 2008.

[Lar04a] C. Larboulette, M.-P. Cani. Real-Time Dynamic Wrinkles. In *Proceedings of Computer Graphics International 2004*, pages 522–525.

[Lar05a] C. Larboulette, M.-P. Cani, and B. Arnaldi. Dynamic skinning: Adding real-time dynamic effects to an existing character animation. In *Proceedings of Spring Conference on Computer Graphics 2005*, pages 87–93.

[Lee06a] S.-H. Lee and D. Terzopoulos. Heads up! biomechanical modeling and neuromuscular control of the neck. In *Proceedings of SIGGRAPH'06, ACM ToG*, 25(3), pages 1188–1198.

[Lee07a] K. S. Lee and G. Ashraf. Simplified muscle dynamics for appealing real-time skin deformation. In *Proceedings of International Conference on Computer Graphics and Virtual Reality*, 2007.

[Lee09a] S.-H. Lee, E. Sifakis, and D. Terzopoulos. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph.*, 28(4):99:1–99:17, 2009.

[Lee12a] D. Lee, M. Glueck, A. Khan, E. Fiume, and K. Jackson. Modeling and simulation of skeletal muscle for computer graphics: A survey. *Found. Trends. Comput. Graph. Vis.*, 7(4):229–276, 2012.

[May13a] Maya. Autodesk, 2013.

[Moh03a] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. In *Proceedings of SIGGRAPH'03, ACM ToG*, 22(3), pages 562–568.

[Ned00a] L. P. Nedel and D. Thalmann. Anatomic modeling of deformable human bodies. *The Visual Computer*, pages 306–321, 2000.

[Pra05a] M. Pratscher, P. Coleman, J. Laszlo, and K. Singh. Outside-in anatomy based character rigging. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2005.

[San08a] S. I. Park and J. K. Hodgins. Data-driven modeling of skin and muscle deformation. *ACM Trans. Graph.*, 27(3):1–6, 2008.

[Sch97a] F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy-based modeling of the human musculature. In *Proceedings of SIGGRAPH'97, ACM Computer Graphics*, pages 163–172.

[Ter03a] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2003.

[Ter05a] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, Cynthia Lau, and Ronald Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Trans. on Visualization and Comp. Graph.*, 11(3):317–328, 2005.

[Tur93a] R. Turner and D. Thalmann. The elastic surface layer model for animated character construction. *Proceedings of Computer Graphics International'93*, pages 399–412.

[Wan02a] X. C. Wang and C. Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of SCA'02*, pages 129–138.

[Wan07a] R. Y. Wang, K. Pulli, and J. Popović. Real-time enveloping with rotational regression. *ACM Trans. Graph.*, 26(3):73, 2007.

[Web07a] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum*, 26(3), 2007.

[Wil97a] J. Wilhelms. Animals with anatomy. *IEEE Computer Graphics and Applications*, 17(3):22–30, 1997.

[Wil97b] J. Wilhelms and A. Van Gelder. Anatomically based modeling. In *Proceedings of SIGGRAPH'97, ACM Computer graphics*, pages 173–180.

[Zhu98a] Q. H. Zhu, Y. Chen, and A. Kaufman. Real-time biomechanically-based muscle volume deformation using fem. *Computer Graphics Forum*, 17(3):275–284, 1998.