Dynamic implicit muscles for character skinning

Valentin Roussellet, Nadine Abu Rumman, Florian Canezin, Nicolas Mellado, Ladislav Kavan, Loic Barthe

Dynamic implicit muscles for character skinning

Valentin Roussellet[1], Nadine Abu Rumman[1], Florian Canezin[1]
Nicolas Mellado[1], Ladislav Kavan[2], Loïc Barthe[1]

[1] Université de Toulouse, IRIT, CNRS
[2] University of Utah, School of Computing

Abstract: Most current methods for character skinning can be categorized into 1) geometric techniques, which are fast and easy to use but often lack physical realism, 2) data-driven approaches, which require a large set of examples that are tedious to edit, and 3) physics-based methods, which are highly realistic but slow and difficult to use. Recently introduced geometric Implicit Skinning methods can solve contact interactions and skin elasticity with results comparable to physics-based simulation in real-time. In this paper we introduce an animation method that adds anatomical plausibility while benefiting from the advantages of Implicit Skinning. We propose an efficient way to model muscle primitives with implicit surfaces. Volumetric extrusions of individual muscles are attached to muscle center lines simulated with a fast, low-dimensional physics-based approach (Position Based Dynamics of one-dimensional line segments). This combination of physics-based simulation with implicit modeling allows us to elegantly resolve muscle-muscle and muscle-bone collisions and add dynamic effects such as flesh jiggling while guaranteeing volume preservation (which is a property of real biological muscles) and producing visually plausible skin-skin contact behavior. Our method runs at interactive frame-rates and features intuitive modeling parameters which allow animators to quickly explore a variety of designs and physics-based effects.

1. Introduction

Character animation is a central component of animated digital media such as films, computer games, and virtual reality. However, the production of compelling and appealing deformations of a virtual character to imbue it with life remains very challenging due to the complexity of the human (or humanoid) body, which is composed of bones, muscles and soft tissues. Specific challenges include run-time performance and parameter tuning. Ideally, the resulting deformation models should run interactively on commodity machines and rely only on intuitive parameters with predictable outcomes on the result. While providing basic skeletal articulation is mandatory to produce animated characters, taking into account the motion of the underlying anatomy significantly increases the realism of the final animation. Existing methods simulating the behavior of all anatomic tissues exist, but they require extensive computational resources, limiting their use in interactive applications.

In this paper, we explore the idea of adding muscle deformations to skeleton-based geometric skinning approaches. We focus specifically on modelling the muscle's shape deformation when they contract, expand and activate under effort. Unlike fat tissues, muscles tend to stay very tense and stiff. We model their elasticity with a dynamic simulation.

Three main families of techniques have been considered to follow this direction by extending Linear Blend Skinning (LBS) [1] or Dual Quaternion Skinning (DQS) [2]. The first are *pose-based* approaches, based on modeling and animation tools, allowing users to edit the character skin in user-specified key poses [3]. Despite being very general and enabling users to model arbitrary skin deformations, manual sculpting is very time-consuming and tedious, especially if similar edits need to be repeated multiple times. A second family of methods uses *muscle primitives* which are positioned inside the character body and act as deformers generating additional skin deformations, usually designed to preserve volume. Geometric modeling of muscles provides interactive parameter tuning [4]. However, deformation parameters are often tedious to set and the resulting skinning solution is subject to the well-known Linear Blend Skinning (LBS) and Dual Quaternion Skinning (DQS) limitations, i.e., no skin-contact deformation and volume loss or gain near joints. The third family is represented by *data-driven* methods, which learn a set of mesh deformers from captured data [5, 6]. However, the produced deformations are limited by the extent of the training set and the difficulty to artistically control the results.

Our contribution is an approach that combines the advantages of *muscle primitive* deformers with an interactive geometric skinning technique producing high-quality skin deformation including contact handling (Figure 1). We rely on Implicit Skinning [7, 8] to resolve self-intersections and represent skin elasticity. However, unlike Implicit Skinning, in this paper we also consider several important phenomena contributing to muscle and skin shape, such as muscle-muscle and muscle-bone collisions and dynamic effects induced by the skeletal motion.

Our technical contributions are two-fold: first, we introduce new *muscle primitives* mimicking the shape and deformation of real muscles (including bulging, deflation, and activation), ranging from relatively simple ones such as the biceps to more complicated pectoral muscles. During animation, our muscles maintain nearly constant volume. We define our muscles as curve-sweep surfaces, with the dynamics of the curves guided by Position Based Dynamics [9]. The use of 3D scalar fields for our muscle primitives enables the integration of these muscles in the Implicit Skinning framework, which provides fast skin-skin contact modeling. The implicit representation of muscles is also perfectly suited for collision response, allowing us to efficiently resolve muscle-bone and muscle-muscle collisions.

Secondly, we show how to adequately integrate our muscle primitives in the Implicit Skinning framework so that the final skinning solution naturally benefits from its skin contact resolution.

In our implementation, muscle shapes and dynamic behavior are controlled by a small set of intuitive parameters, avoiding the need for tedious sculpting of corrective shapes (as in pose-based methods), as well as expensive computational cost and complicated parameter tuning of physical simulation techniques. During the rigging phase, setting the skinning parameters can be done interactively and the total computation time of our method, given a character including more than fifty animated muscles, is

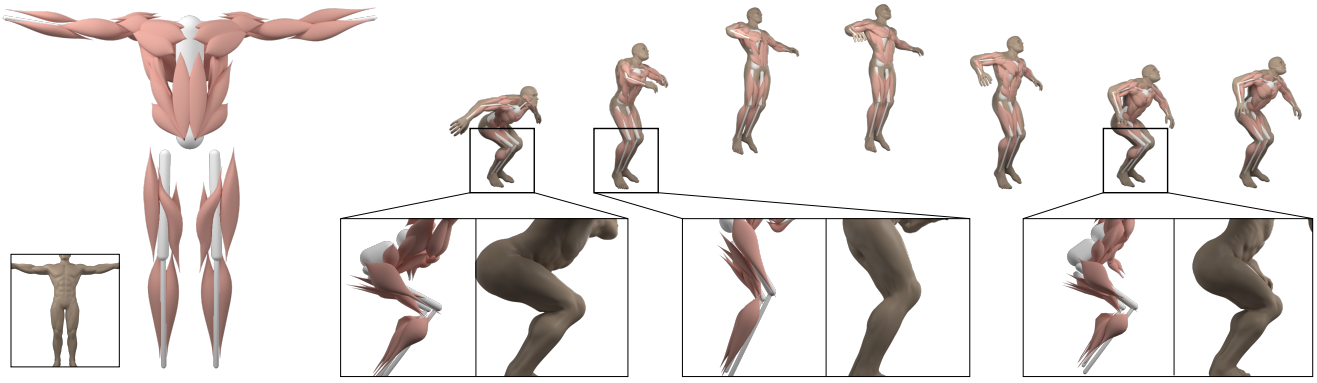


Figure 1. On the left, the muscular and bone structure of the character are shown in the bottom inset. These muscles are represented by new implicit primitives deforming at constant volume with elasticity and collisions computed with Position Based Dynamics. On the right, the different poses of the character jumping illustrate the change in muscle shape due to activation/relaxation, stretch/elongation and the result of the muscle-bone collision resolution. Our muscles adequately seamlessly fit into the Implicit Skinning framework, allowing us to produce character animation with skin contact and elasticity.

less than a second per frame on a standard CPU.

2. Related work

The production of plausible skin-muscle deformations from skeletal motion is a long standing problem in computer animation and several types of approaches emerged in previous work.

Capture-based methods. Data-driven methods [10] can be divided into 1) methods where artists directly craft the data such as target skin or muscle shapes and 2) methods based on capturing data of real subjects, e.g., using 3D scanning or motion capture. Seminal work in the latter category introduced statistical shape models explaining both body and pose-based variations [11, 12, 13]. Subsequent work focused on capturing and modeling flesh and muscle motion using traditional motion capture systems [14, 15] or multi-camera setups with dots painted on the skin [16]. Loper et al. [17] showed how realistic pose and body shapes can be extracted from standard motion capture markers by using a data-driven human body model, and Tsoli et al. [18] focused on capture-based modeling of breathing. Fast models for real-time applications rely on combining linear blend shapes [19], rotation regression [6] or helper-bone rigs that have proven useful for delivering effects such as muscle bulging [5, 20]. Another exciting strand of current research is learning dynamics from data, either using statistical methods [21] or by combining statistical methods with physics-based simulation [22].The most recent work [22] is starting to show promise of being able to generalize to imaginary artist-designed characters. Nonetheless, capture-based methods are generally limited by the availability of real-world human subjects.

Physics-based methods. The advantages of biomechanical modeling of the human body have been recognized early on in computer graphics [23, 24, 25, 26, 27]. More recently, realistic Finite Element-based simulation of flesh has been explored [28, 29] and extended into a comprehensive biomechanical upper-body model [30]. Biomechanical models are usually more focused on accurate computation of the force exerted by muscles and less on their mass or shape [31, 32]. These models have been used to represent the underlying tendons and muscles of an anatomy-based model to create realistic skin deformation of the hand [33, 34]. Advanced numerical simulation methods have been developed for production environments [35, 36]. Fan et al. [37] explored an Eulerian-on-Lagrangian approach to simulate musculoskeletal systems, extended to tendons by Sachdeva et al. [38].

An important advantage of physics-based simulation is the automatic handling of skin contact, e.g., when bending the elbow, which is tedious to model accurately with data-driven methods. The main concern of physics-based methods is speed, which motivated the development of fast physics-based methods capable of handling collisions [39]. More recent works explore the use of physics-based simulation in modeling [40] and the combination of physics-based and data-driven methods [41, 22].

Geometric methods. The need for producing animations of both realistic and stylized characters led to the development of various skinning methods providing interactive and intuitive user control. These methods usually start with geometric techniques to produce pose-based articulation [1, 42, 43, 44, 45, 46, 47, 48, 49], which can be subsequently enriched with artist-provided "correctives" in a set of poses to produce flesh-like effects such as muscle bulging [3, 50, 51]. Finite Element Methods with model reduction in the rig-space [52] or in the pose-space domain [53] may be used to help the artist. However, creating corrective shapes to achieve the desired anatomical effects remains very tedious [54]. Although these methods are included in several commercial simulation packages such as *Maya Muscle* [55] and Weta's *Tissue System* [56], setting up musculoskeletal models with adequate physical properties is time-consuming even for experienced artists.

To avoid the manual modeling of muscle deformations in key poses, muscles may be represented by geometric primitives using parametric ellipses or more elaborated sweeping objects [4,

57]. Muscles are deformed at constant volume [58] and muscle dynamics can be added by representing the muscle sweeping axis with a mass-spring system [59]. Muscle transformations are then weighted over the mesh representing the skin. In a similar spirit, Hyun et al. [60] represented the character body with radial sections that are deformed by the muscle primitives. A contact plane is also introduced to resolve self-collisions at simple joints, i.e., elbows and knees. Finally, Leclercq et al. [61] proposed to represent the muscles as elliptic scalar fields. Even though fast to compute, these approaches do not resolve muscle-bone collisions that are required for realistic muscle deformation at complicated joints (e.g. the shoulder). In general, they also do not handle skin self-collisions and skin elasticity.

3. Technical background

Let us briefly summarize the basic concepts of Implicit Skinning [8] and introduce the corresponding notation. The input is an animated skeleton with the mesh representing the animated character equipped with skinning weights and segmented according to skeletal bones. This segmentation associates a part defined by a set of mesh vertices to each bone, each of them representing an articulated body part(finger phalanx, upper and lower arm, thigh, leg, torso, etc.). In Implicit Skinning, each part is approximated with an isosurface in a 3D scalar field with compact support $f_j : \mathbb{R}^3 \to \mathbb{R}$. Vaillant et al. [7] use Hermite-Radial-Basis-Functions (HRBF) [62] to define these scalar fields f_j for their natural ability to approximate positions and normals with smooth scalar fields. Following the standard convention used in compact support implicit surface modeling [63], the approximating isosurface of f_j is the 0.5-isosurface.

By composing this set of by-part scalar fields f_j, a single scalar field f representing the entire character skin is then defined [64, 65]. Specific compositions achieve desirable deformations of the 0.5-isosurface of f at joints, and include a contact surface where the skin self-collides during its animation. At this step, each mesh vertex stores its initial field value in f.

At run-time, the mesh is incrementally animated (i.e., each animation frame provides the initial pose for the computation of the following one) via geometric skinning and, simultaneously, the field function f by applying rigid transformations to the underlying field functions f_j. The mesh vertices then march following the gradient of f with interleaved tangential relaxations minimizing a modified As-Rigid-As-Possible (ARAP) energy [66] until they reach their initial field value.

4. Overview

In this paper, we present a new type of muscle primitives and a way to embed them into the Implicit Skinning framework [8]. The key idea of Implicit Skinning is to produce the skin deformation on the 0.5-isosurface of f (representing the character) and let the mesh track these deformations while maintaining its elasticity. In order to insert muscle dynamics into this framework, we propose to modify the definition of the scalar fields f_j attached to the bones in order to include muscle deformations. Doing so, the field f (and thus the mesh after tracking)

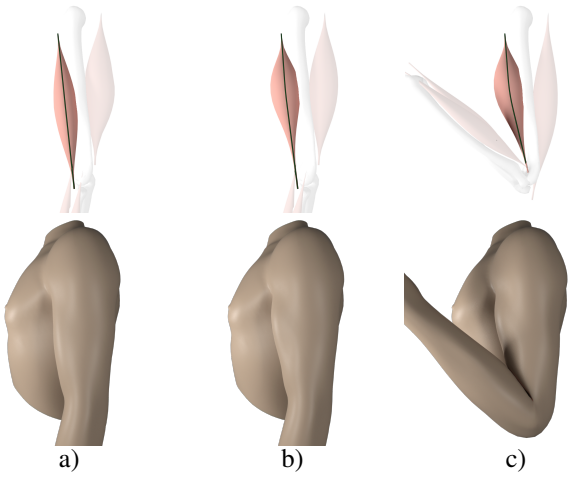


Figure 2. Influence of our muscle deformations on Implicit Skinning. Starting from an input pose (a), the muscle is activated (b) and the arm flexed (c). All along, the primitives preserve their volumes and are deformed to avoid collisions. The shape of the muscle central line (in green) is deformed using Position Based Dynamics (PBD) [9].

also includes these deformations and self-contact surfaces by composition of scalar fields f_j.

We thus define muscle shapes and dynamics using new field functions f_M (Sections 5 and 6), and insert them as additional primitives in the definition of fields f_j (Section 7). Muscle primitives are parameterized as follows (see Figure 3):

- two extremity points: the origin $\mathbf{m_0}$ and the insertion $\mathbf{m_1}$, attached relatively to the bones of the animation skeleton and moving kinematically (Section 5.1),
- a set of user parameters to control the muscle geometry, e.g. longitudinal/radial profiles and volume (Section 5.2),
- a rest shape, an activated shape, and a shape interpolation scheme at constant volume to smooth the transition between these two muscle states (Figure 2 and Section 5.2),
- a set of particles to simulate elastic deformation of the muscle, resolve muscle-muscle and muscle-bone collisions and add dynamics effects using Position Based Dynamics (Section 6).

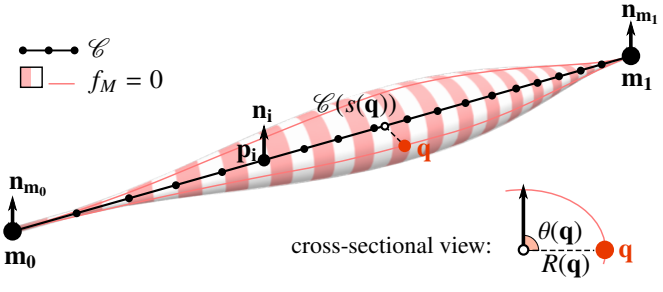


Figure 3. Schematic view of our muscle primitive with notations.

5. Muscle Primitive

The range of shapes that our muscle model produces relies on its adequation to the muscle shapes provided in an anatomic human model [67], the set of muscle shapes and deformations currently used in the computer graphics literature [68, 37], and the capacity of the representation to satisfy our technical (collisions, insertion in the Implicit Skinning) and performance requirements.

We propose a formulation producing fusiform muscles defined by swept primitives with a minimal number of user parameters: two parameters for the longitudinal and one for the radial profile, with a scaling parameter. Our model is able to change from rest to activation shape at constant volume. More complex muscles are represented by combining multiple primitives, as explained in Section 5.3. We present the details of our muscle model in this section and discuss the user parameters in Sections 8.3 and 8.4.

5.1. Model

We initially define a muscle M as a scalar field $f_M : \mathbb{R}^3 \to \mathbb{R}$ constructed by sweeping a profile function R along a central polyline $\mathscr{C}$ (see Figure 3 for notations). We design f_M as a smooth distance field with 0-isovalues describing the muscle boundaries.

The evaluation of f_M at a point $\mathbf{q}$ consists of three steps detailed below: construction of the central polyline $\mathscr{C}$, projection of $\mathbf{q}$ on $\mathscr{C}$ to compute the distance $d(\mathbf{q}, \mathscr{C})$, and evaluation of the sweeping profile function $R(\mathbf{q})$. The value of the scalar field at any point $\mathbf{q}$ is then given by

$$f_M(\mathbf{q}) = d(\mathbf{q}, \mathscr{C}) - R(\mathbf{q}). \tag{1}$$

Central polyline construction.. The two endpoints $\mathbf{m_0}$ and $\mathbf{m_1}$ of the muscle primitive are each attached to an animation bone, so they move kinematically during the animation. The line segment $[\mathbf{m_0}, \mathbf{m_1}]$ is divided into N parts with intermediate control points $\mathbf{p_i}$. The resulting polyline $\mathscr{C}$ is parameterized by $s \in [0, 1]$, and we denote as $s(\mathbf{q})$ the curvilinear parameter of the projection of $\mathbf{q}$ on $\mathscr{C}$. In order to model muscles with elliptic profiles, the polyline is oriented at each endpoint by given normal vectors $\mathbf{n_{m_0}}$ and $\mathbf{n_{m_1}}$. We associate to each control point of the polyline $\mathbf{p_i}$ a normal vector $\mathbf{n_i}$, defined as follows:

1. we compute an interpolated normal vector by spherical linear interpolation of the normal vectors of the two endpoints,
2. we project this interpolated vector on the plane spanning $\mathbf{p_{i-1}}$, $\mathbf{p_i}$ and $\mathbf{p_{i+1}}$ to account for the local twist of the polyline. If these three points are aligned, we interpolate the normals of the two closest polyline control points for which they are defined.

The polyline normal at $s(\mathbf{q})$ is then computed by spherical linear interpolation of the control point normals $\mathbf{n_i}$, $\mathbf{n_{i+1}}$ of the polyline segment it belongs to.

Projection operator and distance to polyline.. The projection of a query point $\mathbf{q}$ on $\mathscr{C}$ is a critical step of our approach, as it shapes the properties (e.g. continuity) of the distance function d, and thus the primitive scalar field (see Eq. 1). The distance function d is defined as follows:

$$d(\mathbf{q}, \mathscr{C}) = \left\| \mathbf{q} - \text{proj}(\mathbf{q}, \mathscr{C}) \right\|_2 .$$

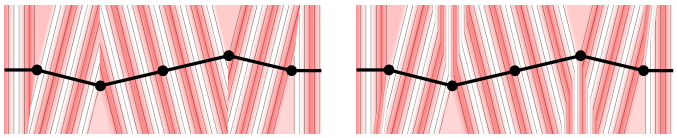
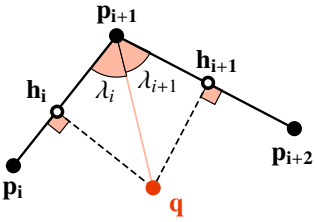


Figure 4. Comparison between standard point-to-segment orthogonal projection, and ours. Colors are computed w.r.t. the curvilinear axis $s(\text{proj}(\mathbf{q}))$. Note the discontinuities appearing when using standard orthogonal projection.

As shown in Figure 4 (left), projecting the query point on the closest segment yields discontinuities in the interior regions of the dihedral angles formed by the segments of the polyline. Similar problems arise also with interpolation on triangle meshes [69, 70].

We fix this problem by reparameterizing the projection on the areas where a point $\mathbf{q}$ projects on the interior of two consecutive segments using the cotangent of the angles λ_i and λ_{i+1} formed between the point and the two segments, as illustrated in the right inset. The projection algorithm is detailed in algorithm 1. As shown in Figure 4 (right), our operator does not exhibit discontinuities and allows projection onto the polyline vertices even in the interior regions of the dihedral angles formed by the segments of the polyline.



Algorithm 1 Reparameterization of the polyline projection

for all segments $[\mathbf{p_i}, \mathbf{p_{i+1}}]$ of $\mathscr{C}$ **do**
  compute $\mathbf{h_i}$, the nearest point from $\mathbf{q}$ to the segment and s_i its parameter
end for
Let $\mathbf{h}$ be the nearest point from $\mathbf{q}$ among all $\mathbf{h_i}$, and s_h its parameter.
if $\exists i$ such as (($\mathbf{h} = \mathbf{h}_i$ or $\mathbf{h} = \mathbf{h}_{i+1}$) **and** (none of $\mathbf{h}_i$, $\mathbf{h}_{i+1}$ belongs to $\{\mathbf{p_0}..\mathbf{p_N}\}$)) **then**
  $s(\mathbf{q}) = \frac{s_i \cot \lambda_i + s_{i+1} \cot \lambda_{i+1}}{\cot \lambda_i + \cot \lambda_{i+1}}$
else
  $s(\mathbf{q}) = s_h$
end if
return $\mathscr{C}(s(\mathbf{q}))$

Sweep surface. The sweep surface is defined by "sweeping" a profile function $R(\mathbf{q})$ along $\mathscr{C}$. In order to describe the overall shape of fusiform muscles, we parametrize $R(\mathbf{q})$ w.r.t.

- $s(\mathbf{q})$, the curvilinear parameter,
- $\theta(\mathbf{q})$, the angle between the polyline normal at $s(\mathbf{q})$ and the vector $\mathbf{q} - \text{proj}(\mathbf{q}, \mathscr{C})$ (Figure 3).

We specify R to be separable to enable easy evaluation of the muscle's volume, i.e.,

$$R(\mathbf{q}) = w\ \Phi\left(s\left(\mathbf{q}\right)\right)\ r\left(\theta\left(\mathbf{q}\right)\right), \tag{2}$$

where $\Phi(s)$ represents the distribution of mass along the axis, r represents the polar profile of the muscle and w is a width scale factor. A sufficient condition for the muscle volume to remain constant is ensuring that $\int_0^1 (\Phi(s))^2\ \mathrm{d}s$ and $\int_0^{2\pi} (r(\theta))^2\ \mathrm{d}\theta$ are constants, as will be shown in the next section. We define our functions Φ and r so that these integrals both equal 1.

5.2. High level shape parameters

The shape of our muscle primitive is defined by the muscle central polyline $\mathscr{C}$, the scale factor w, and the two sweeping functions r and Φ. In this section we discuss how these parameters offer control over 1) the muscle volume, 2) the longitudinal and radial shape profiles, 3) rest/activation muscle shapes. Parameters controlling the muscle shape are detailed in Section 8.4.

Volume. When $\mathscr{C}$ is a straight line we can evaluate the volume of the muscle in cylindrical coordinates as

$$V = \int_{s=0}^{1}\int_{\theta=0}^{2\pi}\int_{\rho=0}^{w\Phi(s)r(\theta)} \rho \mathrm{d}\rho\ \mathrm{d}\theta\ l \mathrm{d}s = \pi w^2 l, \tag{3}$$

where l is the length of the polyline $\mathscr{C}$ (see the derivation details in Appendix A). When the muscle length l changes during the animation, we simply recompute the scale $w = \sqrt{V/\pi l}$ in Equation 2. This makes the muscle inflate when it contracts and shrink when it is stretched, with constant volume (see Figure 2, top row). Our volume computation is global and performed on a straight polyline $\mathscr{C}$, it thus becomes approximate when $\mathscr{C}$ is curved. In practice the volume loss due to the polyline curvature is indiscernible (around 1-3% in our experiments, see Section 8.2). Note that local computations could be considered for a more accurate volume conservation, but this would involve significant computations for a very subtle effect on our muscle parameters adjustment.

Shape longitudinal profile. We define Φ, the distribution of mass along the muscle axis, as

$$\Phi(s) = \varphi(\alpha, \beta; s),$$

where α, β are scalar parameters controlling the shape of the profile. Our definition of φ is inspired by the Euler *beta function*, as it provides geometric profiles similar to muscle shapes. According to Equation 3, volume conservation is guaranteed when the integral of the square of our base function is constant, regardless of α and β. We thus define

$$\varphi(\alpha, \beta; s) = \frac{\varphi_0(\alpha, \beta; s)}{\|\varphi_0(\alpha, \beta; s)\|_2}$$

where φ_0 is defined for $s \in [0, 1]$ as

$$\varphi_0(\alpha, \beta; s) = s^{\alpha-1}(1-s)^{\beta-1}.$$

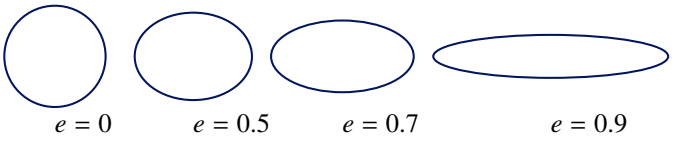


Figure 5. Elliptic profiles at constant area for different values of the ellipse eccentricity e.

As such, $\varphi(\alpha, \beta; s)$ can be explicitly written as

$$\varphi(\alpha, \beta; s) = \frac{s^{\alpha-1}(1-s)^{\beta-1}}{\sqrt{\int_0^1 y^{2(\alpha-1)}(1-y)^{2(\beta-1)}\mathrm{d}y}}. \tag{4}$$

In principle, α and β can be any positive numbers. For muscle profiles we consider only integer values for efficiency of evaluation. We additionally impose $\alpha > 1$ and $\beta > 1$ to yield a function where $\varphi(0) = \varphi(1) = 0$, and $\alpha \leq 9$ and $\beta \leq 9$, larger values leading to very sharp profiles that do not correspond to realistic muscle shapes. Note that the denominator of Equation 4 is independent of s and can be pre-computed for all allowed values of α and β using the Euler *beta function*.

The ratio α/β controls the asymmetry of the shape (with the distribution being symmetric for $\alpha = \beta$) while the individual values of α and β control the sharpness of the function's rise on each side, as illustrated by Figure A.16 in the Appendix.

Shape radial profile. In our muscle representation, the radial profile is elliptic and controllable by a single parameter, the ellipse eccentricity e as illustrated in Figure 5. Formally, we set r to be an ellipse of semi-axis lengths u and v:

$$r(\theta) = \frac{uv}{\sqrt{u^2\cos^2\theta + v^2\sin^2\theta}}. \tag{5}$$

The ellipse parameters u and v are directly computed from the eccentricity e so that the product $uv = 1$ (i.e., the area of the ellipse is always equal to that of a circle of radius 1):

$$u = \sqrt[4]{(1-e^2)} \quad v = 1/u.$$

Rest and activation shapes. When modeling a muscle, there are two shapes to control the muscle state during the animation: the rest shape corresponding to a relaxed state of the muscle fibers and the activation shape corresponding to an increase of tension in the muscle fibers. As illustrated in Figures 2(a) and 2(b) changing of state does not require a change in muscle length, it is an abrupt modification of muscle shape (i.e longitudinal and radial profiles) that can be smoothed over a small sequence of frames by an interpolation at constant volume.

We propose to model this change of muscle state by interpolating between the two sets of shape parameters (α_0, β_0, e_0) and (α_1, β_1, e_1). We use $a \in [0, 1]$ to denote the interpolation parameter. By convention, the rest shape of the muscle corresponds to $a = 0$, and full activation to $a = 1$. While parameters e_0 and e_1 are linearly interpolated, we need to re-normalize the interpolation of the pairs α_0, β_0 and α_1, β_1 to preserve volume. This is done as follows:

$$\Phi(s) = \frac{(1-a)\varphi(\alpha_0, \beta_0, s) + a\varphi(\alpha_1, \beta_1, s)}{\sqrt{F(a)}},$$

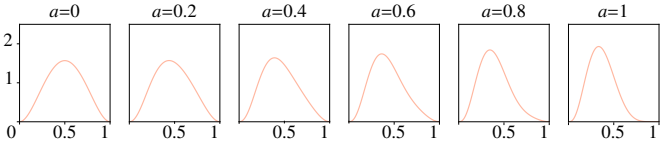


Figure 6. Profiles of the Φ function for $\alpha_0 = \beta_0 = 3$ and $(\alpha_1, \beta_1) = (4, 7)$.

where

$$F(a) = \int_0^1 ((1-a)\varphi(\alpha_0, \beta_0; s) + a\varphi(\alpha_1, \beta_1; s))^2 \, \mathrm{d}s.$$

While the value of F might appear to be expensive to compute for each a, it can be formulated as

$$F(a) = (1-a)^2 + a^2 + 2a(1-a)\ K(\alpha_0, \alpha_1, \beta_0, \beta_1),$$

where K is a constant term which can be expressed in terms of the Euler *beta function B*

$$K(\alpha_0, \alpha_1, \beta_0, \beta_1) = \frac{B(\alpha_0 + \alpha_1 - 1, \beta_0 + \beta_1 - 1)}{\sqrt{B(2\alpha_0 - 1, 2\beta_0 - 1) B(2\alpha_1 - 1, 2\beta_1 - 1)}}$$

(see derivation in Appendix B). In practice, K can be precomputed and tabulated in preprocessing, resulting in fast runtime evaluations. Figure 6 shows an example family of functions Φ at various levels of interpolation a .

5.3. Extension to non-fusiform muscles

This model produces fusiform muscles such as *biceps brachii* of the upper limb or *quadriceps* from the lower limb. To model more complex muscles such as the *pectoralis major*, we follow a method similar to the one proposed by Scheepers et al. [57] and Murai et al. [71]. We represent these muscles by instancing several fusiform shapes, each integrating a set of muscle fibers. Figure 7 shows an example of pectoral and shoulder muscles. Depending on the desired deformation effects, muscle collisions can be ignored (Figure 7 between green muscles) or enabled as explained in Section 6.3 (Figure 7 between blue muscles).

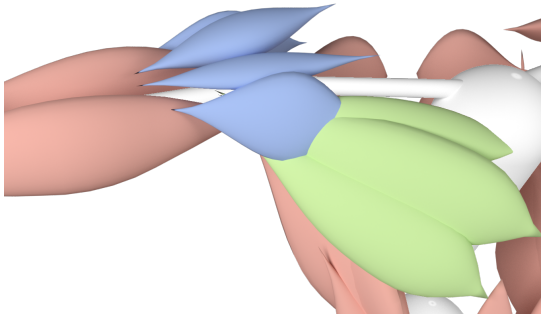


Figure 7. Pectoral (green) and shoulder (blue) muscles represented by sets of fusiform fibers. In this case, shoulder muscles collide against each other, while the pectorals are allowed to overlap to better approximate the desired shape.

6. Dynamic Muscle Deformations

In Section 5 we have introduced a purely kinematic model for muscles, which deforms at constant volume and whose shape is controlled by high level user parameters. In this section, we show how we extend this model (Section 6.1) to deform the muscles according to their reaction to the animation motion, e.g. muscle jiggling (Section 6.2), and their surroundings: other muscles, bones, or skin (Section 6.3).

6.1. Deformation modeling

To imbue muscles with dynamic behavior, we deform their shape by letting the control points $\mathbf{p_i}$ of the central polyline $\mathscr{C}$ be driven by physics-based simulation. While the two extremities $\mathbf{m_0}$ and $\mathbf{m_1}$ of $\mathscr{C}$ still follow the kinematics of their skeleton, each internal point $\mathbf{p_i}$ is represented by a particle animated by Position Based Dynamics (PBD) [9]. Thus the muscle elasticity and collisions are expressed as constraints solved with the standard PBD Gauss-Seidel solver. This solver iterates on each constraint sequentially yielding a correction applied to the particle's position (see Figure 8). A strong benefit of our approach is that muscles are defined by 3D scalar fields, which yields very efficient distance queries and allows to interactively resolve collisions (Section 6.3), even for a large number of constraints and particles.

6.2. Muscle elasticity and animation-induced motion

The mass of the whole muscle is set by computing its initial volume multiplied by the average density of muscle tissues $\rho_M = 1.06$ g.cm^{-3} [72]. Each moving particle of the muscle's axis is assigned a fraction of this mass proportionally to the width of the muscle at their initial position on the axis.

The axis particles are tied together by elastic distance constraints which act as springs in the PBD framework. Each of these constraints is parameterized by its rest length l_0 and stiffness k. The rest length is set as l_0 =2% of the initial distance between the central particles, to provide tension to the muscle axis. A rest length of l_0 =100% is assigned to the first 10 % of the length of the muscle at each extremity to represent the more rigid tendons linking the muscle to the bones. The stiffness values k controls the strength of the muscle tone. As discussed in Section 8, in our setup a value of $k \approx 0.1$ generates a muscle following the motion of its attachment bones with lots of inertial effects, while a value of $k \approx 0.7$ produces a very tense muscle (almost no inertial effects).

6.3. Collision resolution

In our body, the shape of muscles is constrained by the presence of rigid bones, other muscles, soft tissues, and skin. To generate more plausible muscle deformations and to better capture the effect of volume preservation, it helps to resolve collisions among the individual organs. The use of scalar fields in conjunction with PBD provides an efficient framework for collision resolution.

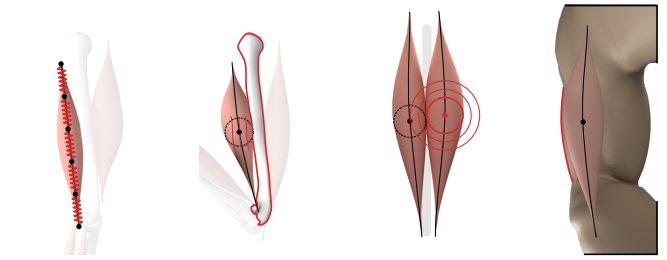


Figure 8. Constraints used to model muscle properties and interactions with surrounding elements.

Muscle-bone collision. We resolve muscle-bone collisions by constraining the particles to stay above a certain radius from the bones. To do so, each particle behaves as a sphere which collides against the bone surface. The collision radius of the particles $\mathbf{p_i}$ is set as $w\Phi(\mathbf{p_i})$, that approximates the average radius of the muscle's surface. Since the muscle's shape evolves during the animation, this radius has to be updated every frame before the PBD solver is executed. The distance between particles and bones is computed efficiently using precomputed distance fields associated with bone shapes (as illustrated in Figure 2) or analytically computing the distance to cylindrical bone proxies (Figure 1).

Muscle-muscle collision. We optionally model collision between muscles in a similar fashion, by constraining the particles of one muscle against the distance field f_M of another muscle. This feature is useful for modeling complex muscles consisting of multiple parts that would otherwise overlap.

Muscle-skin interaction. To represent the effect of the skin enclosing the muscles, we add a third field-based constraint to keep the particles themselves inside the 0.5-isosurface of the nearest HRBF field (see Section 7 for the integration of our muscle primitives with the initial HRBF f_j in the Implicit Skinning framework). This is especially useful for muscles whose shape runs astride a complex joint such as the pectoral across the shoulder, since the tendency of the axis is to form a straight line outside the body. It also prevents inertial effects from dragging the particles outside of the skin in fast motions such as jumping or running.

Regularization. Additionally, to model the visco-elastic properties of biological soft tissues, we introduce a global damping coefficient μ of 0.9 on the velocities at each integration step of the PBD solver. This damping models resistance due to interactions of the muscles with other soft tissues (e.g. fat) and prevents long-term muscle vibrations by dissipating their velocity. Also, similarly to rigid-body PBD [73], we add a tangential friction term for each particle which collides against a bone or a muscle. This friction force is opposed to the current velocity of the particle, simulating the loss of energy when the two anatomical objects interact.

7. Integration with Implicit Skinning

7.1. Composition operators

As explained in Section 4, the dynamic deformations of our muscles have to be included in the scalar fields f_j (defined by HRBF) before they are composed to produce the final field f whose 0.5-isosurface represents the skin.

The field representation f_M introduced in Section 5 is a distance field with a global support. Implicit Skinning relies on compactly supported field functions and an homogeneous formulation is required to adequately integrate our functions [63]. We thus convert our muscles to compact support by composing f_M with a fall-off transfer function as done for the HRBF fields described by Vaillant et al. [7]. We then assemble together all muscular fields associated with a given animation bone with a union operator before blending them with the HRBF to produce the new fields f_j.

The mesh tracking step of Implicit Skinning (Section 3) relies on a gradient descent in the field f. Gradient directions should thus be smooth and singular points (where $\nabla f = 0$) in tracking areas must be avoided. The standard max union operator on field functions [75, 76] is known to produce gradient discontinuities and we rather compose muscles with clean union operators [77, 78] that generate a smoothed gradient. The blending of the assembled muscles with the HRBF introduces singular points along the muscle axis in the fields f_j. It also produces gradient directions pointing to the bone rather than the skin in the bone-facing part of muscles as illustrated in Figure 9-center. This problem in implicit modeling has been raised by Canezin et al. [74] when small details are added to a large object with a blending operator. The solution is to use the detail blending operator of Canezin et al. [74] that blends only the outside part of the muscle without including the central singular points and the bone-facing parts in the resulting field, as illustrated in Figure 9-right.

During the animation, we want the skin to capture the underlying muscle's shape when it bulges out but also when it contracts. In our approach, muscles are added to the initial HRBF skin approximation using a blending operator (union with a smoothed transition between the combined objects). This means that if the muscle deforms completely inside the HRBF, it will not modify its shape, leaving the skin represented by f_j unchanged. In order to deform the skin shape f_j with all visible muscle deformations, we reduce the HRBF surface along muscles so that it represents a muscular rest surface. The blending of the muscle primitives then locally adds the muscle shapes in the limb representation, and the skin shape in this area follows the deformations of the muscles.

7.2. Integration in the Implicit Skinning pipeline

Figure 10 illustrates the framework to follow for the production of an animation frame. At each new frame, the animation skeleton is transformed to its new position. Data which is kinematically bound to the animation bones, such as the individual HRBF fields, the muscles endpoints and the scalar fields representing the rigid bones are updated. Keyframed per-muscle shape parameters (eccentricity, activation and width exaggeration) are also updated during this stage.

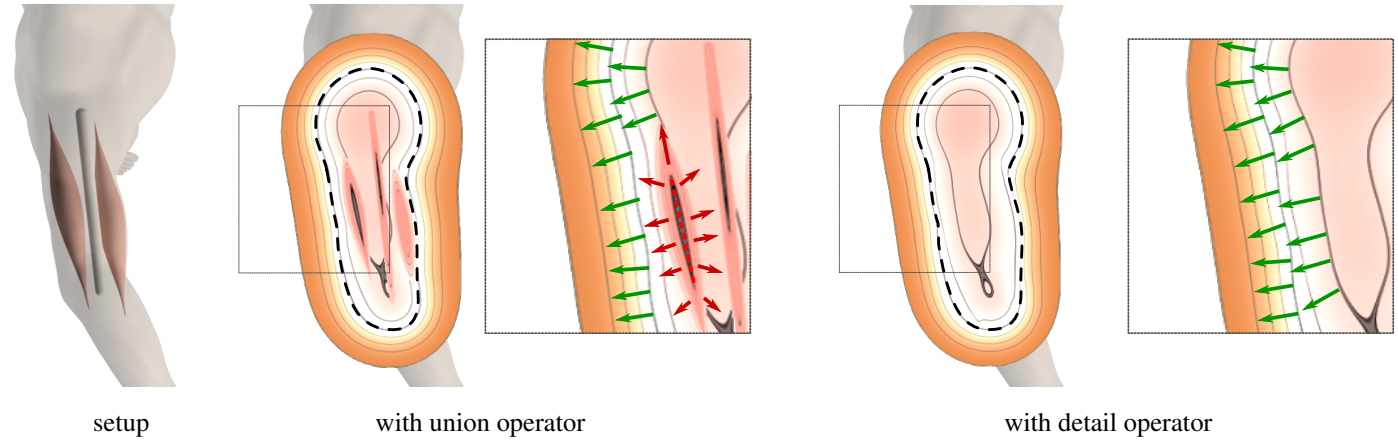


Figure 9. Left: a typical skinning setup with muscles and bone primitives blended with the HRBF field. Center: using a union operator yields singular points (red dotted line) and wrong gradient directions (red arrows) near the 0.5-isosurface. Right : the use of Canezin et al. [74] detail blending operator (right) avoids these problems, eliminating the singular points and yielding a smooth gradient.

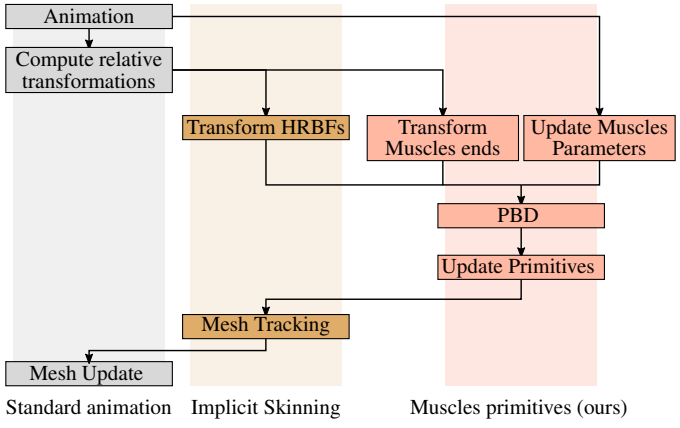


Figure 10. Breakdown of the Implicit Skinning pipeline with muscles. Steps pictured in grey are the standard geometric skinning pipeline, steps in brown are the Implicit Skinning correction algorithm and steps in salmon are our new anatomic-related workflow. Each arrow represents a direct dependency between steps.

Table 1. Summary of the scenes used for our results. The number of PBD constraints is given for 30 particles per muscle.

Scene	# vertices	# bones	# muscles	# constraints
Arm shake	2172	3	4	456
Biceps curl	2172	3	4	456
Jump	19296	12	50	4250
Run	19296	12	50	4250

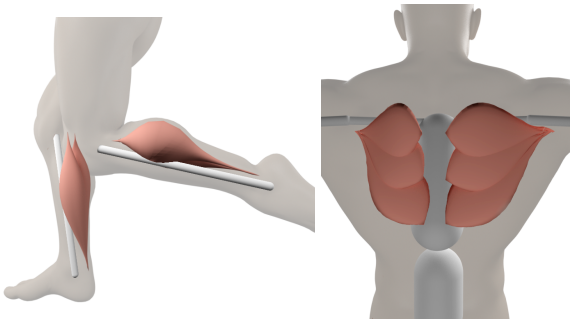


Figure 11. Calf muscles in their least activated state (left; $\alpha_0 = \beta_0 = 3, a = 0$) and their most activated state (right; $\alpha_1 = 4$, $\beta_1 = 8$, $a = 1$). Dorsal muscles are shown in their rest state ($\alpha_0 = 2, \beta_0 = 3$).

Next, the PBD solver computes the new position of the particles. The PBD timestep can be set to a fraction of the animation framerate, which yields an overall stiffer behavior of the physics engine. After the new particle positions are computed, each muscle updates its sampled normals $\mathbf{n_i}$ as described in Section 5.1 and scales its width according to the new length of its axis, as explained in Section 5.2. All underlying field functions are now set and the final skin field f is ready to be evaluated by the Implicit Skinning tracking to skin the character mesh vertices.

8. Results and discussions

We set up a variety of scenes ranging from simple motions, such as an arm shake or a biceps curl (involving only a few muscles) to challenging motions, such as jumping and running with a fully rigged model. Please see Table 1 for details.

Results were generated on a 3.6 GHz Intel Xeon E5-1650 CPU with 64 GB of memory. Our method does not add significant memory overhead to Implicit Skinning due to the compact representation of muscles in memory. The blending operator of Canezin et al. [74] and the optional cached distance fields for anatomic bones are the only memory overhead required for adding our muscle deformations. This represents less than 200 KB for the operator and 8 MB per optional anatomic bone.

The default shape setting for our muscles is $\alpha_0 = \beta_0 = 3$ and $\alpha_1 = 4, \beta_1 = 7$, yielding a smooth symmetric shape in its inactivated state and a significant bulge in its fully activated state, as depicted in Figure 11, left. Other shapes are also used in large flat muscles such as dorsals ($\alpha_0 = 1, \beta_0 = 3$) (Figure 11, right) or pectorals ($\alpha_0 = 3, \beta_0 = 9$) (Figure 7).

The stiffness of the distance constraints between the particles affects mostly the inertial motion of the muscle: stiff muscles ($k = 0.6$) tend to closely follow the skeleton's motion while

Table 2. Average times in seconds per frame for our different scenes and 30 particles per muscle. Times are given for Implicit Skinning without muscles (IS only), for Implicit Skinning with muscles but without the PBD simulation (IS), and for the PBD simulation (PBD).

Scene	Standard IS	IS + Muscles		
		Tracking	PBD	Total
Arm shake	0.008	0.013	0.055	0.068
Biceps curl	0.005	0.015	0.042	0.057
Jump	0.050	0.462	0.096	0.558
Run	0.051	0.473	0.096	0.569

softer muscles ($k = 0.1$) tend to drag and jiggle much more.

8.1. Timings

Table 2 shows the average time per frame for each of our scenes. For reference, we included the time per frame of standard Implicit Skinning without muscles. While Implicit Skinning runs at real-time frame rates even in more demanding scenes, adding more muscles or increasing the number of particles per muscle increases both the physics solver time and the run-time of the Implicit Skinning algorithm itself. The former happens because more constraints have to be solved, while the latter occurs because the Implicit Skinning algorithm has to evaluate the muscle fields several times for the neighboring vertices. Nevertheless, even with our most complex scene with 50 muscles each including 30 particles, we maintain a total frame time below one second.

When editing the muscle parameters it is possible to disable Implicit Skinning to have interactive response times (up to 5-6 fps for the jumping model with the physics of all muscles activated). The parameters of each individual muscle can be tuned in real time, taking advantage of the fact that only one muscle is edited. Also, muscles subject to small deformations can be represented by fewer particles, which reduces the number of constraints and increases the simulation speed while decreasing the evaluation cost of the muscle scalar fields (see Figure 12). Note that even though subtle in Figure 12-left, the muscle scalar field computation time increases with the number of particles. This increase is due to the larger number of segments to consider for the evaluation point projection on the muscle polyline (see Section 5.1).

8.2. Volume conservation

We measured experimentally the volume variation of our muscle models due to the bending of the central axis (see Section 5). Figure 13 shows the variation of a numerical approximation of the volume of different muscles during the biceps and the jump animations (using the volume of the representative mesh as a proxy). The highest variation is approximately 3%.

Our model also allows a user to transgress this volume constraint to amplify the muscle growth for expressive effect by tuning the scale factor w of the muscles as shown in Figure 14.

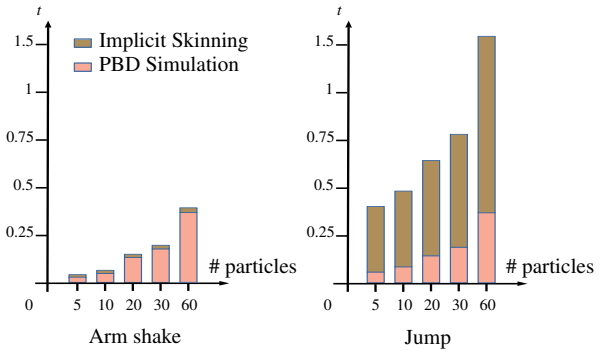


Figure 12. Average times in seconds for the computation of an animation frame, for different numbers of particles per muscle. The brown represents the computation of Implicit Skinning excluding the PBD simulation, and the pink represents the PBD simulation.

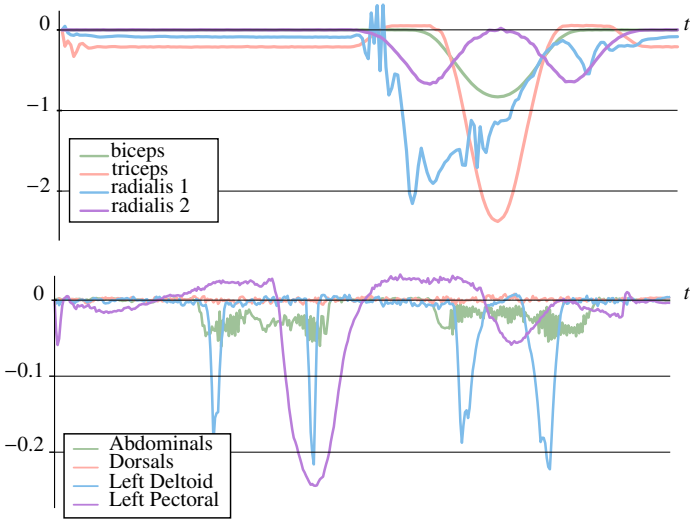


Figure 13. Relative variation over time (in %) of muscles volume during, top - the biceps curl and bottom - the jump animation.

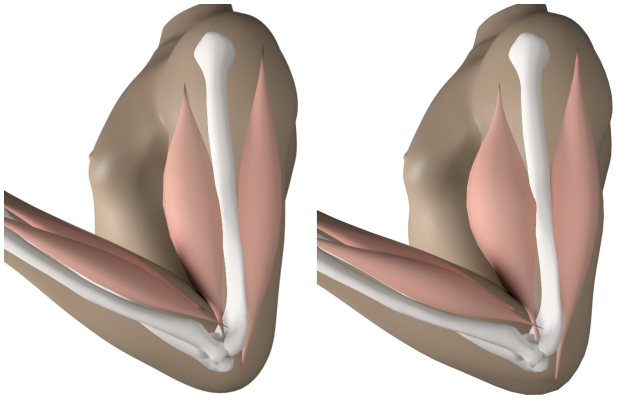


Figure 14. Left, normal biceps curl. Right, muscle width increased by 30%.

8.3. User parameters

Our model exposes two sets of muscle parameters, i.e. the geometric parameters and the dynamic parameters, and a muscle state, i.e relaxed and activated.

8.3.1. Geometric parameters

Geometric parameters are introduced in Section 5.2 with our volume equations. They are α and β for the muscle longitudinal shape, the eccentricity e for its radial cross-section, and the scale factor w.

As shown in Figure A.16, the value of α and β, from 2 to 9, control the position of the bulge along the axis and the sharpness of its rise on each end of the axis. The shape is symmetrical when $\alpha = \beta$, with higher values corresponding to a narrower bulge. The more the values differ, the further the maximum point is from the axis center.

The eccentricity parameter produces circular radial shapes when $e = 0$ and smoothly interpolates towards more elongated elliptic cross-sections when it approaches 1, as illustrated in Figure 5. Note that this parameter has a non linear impact on the ellipse dimensions. This can easily be compensated by adjusting the input value provided by the user, for instance when using a slider moving from circular to flat.

The last geometric parameter, w controls the uniform scale over the muscle width to adjust its global size.

8.3.2. Muscle activation

Two different shapes of a muscle can be defined using the geometric parameters α, β, e and optionally w: the muscle rest shape, and the activated state. A muscle activation is in general almost instantaneous and the muscle may change its shape from rest to activation in a few frames. Our model can vary continuously from one shape to the other using the muscle shape interpolation at constant volume introduced in Section 5.2. Once these two shapes are set, the muscle state and its corresponding shape are defined by the interpolation parameter a with $a = 0$ for the rest state and $a = 1$ for the fully activated shape.

8.3.3. Dynamic parameters

To be deformed in response to the skeleton motion as explained in Section 6, each muscle is parametrized by its mass, its rest length and its stiffness k. The PBD solver is also controlled by global parameters: the timestep δt and the damping coefficient μ. The mass, rest length and damping parameters are preset as explained in Sections 6.2 and 6.3. Changing the global engine parameters is a possibility to access a wider variety of motions, but this requires adjusting the stiffness of every muscle in the scene to keep the muscles' behavior consistent. This is a limitation of the PBD approach that is discussed further in Section 8.6 and we thus suggest to avoid the modification of these paramters.

In our experiments, we thus only provide a single dynamic user parameter, the stiffness k, to adjust per-muscle to obtain a stiffer or looser behavior of a given muscle.

8.4. Parameter setting

8.4.1. Keyframing parameters

Keyframing is a standard approach for controlling parameters over an animation. All the geometric parameters and the stiffness can be keyframed in order to allow the user to have a maximal control of the deformations. We linearly interpolate the eccentricity e, interpolation a, scaling w and the stiffness k while α and β are interpolated with our interpolation at constant volume (see Section 5.2). If required, any higher degree interpolation may be used for e, w and k.

8.4.2. User parameter setting

The parameters can be set in an interactive muscle modelling session, using standard UI controllers such as sliders for the continuously varying values e, w and k, and a set of icons for automatically selecting the values of α and β. Common muscles of a body may also be preset, together with their activated shape and directly proposed to the user that just have to adjust the endpoint positions and the scaling w, optionally tweaking the other parameters.

Once both rest and activation shapes are set, the interpolation parameter a can be keyframed by the user or automatically set if forces information is provided by the animation system.

In our experiments, the manual edition of a muscle geometric parameters including rest and activation shapes in a prototype script interface takes around 10 minutes and the keyframing of the muscle state interpolation a and its stiffness k takes a few minutes for about a minute of animation. The use of a dedicated UI as the one provided in a professional software would greatly lower these timings by enabling the setting of individual muscles in an interactive visual session.

8.4.3. Automatic parameter setting

One may consider automatic adjustment methods to set the muscle parameters, for example by adapting existing anatomical models with Anatomy Transfer [79]. Another direction of investigation is to use a data-based approach to set the muscle shape parameters from sparse motion capture marker inputs, by optimizing the muscle activation parameters as proposed by Sifakis et al. [80]. Similarly, muscular activity models such as the locomotion model of Lee et al. [81] can control the evolution of muscle activation parameters.

Table 3. Average times in seconds per frame for minimizing PBD constraints, without and with collision constraints (muscle/bones, muscle/muscle, interactions with skin), in our different scenes.

Scene	PBD - only elasticity	PBD - with collision
Arm shake	0.003	0.055
Biceps curl	0.003	0.042
Jump	0.050	0.096
Run	0.048	0.096

8.5. Effects of collisions

As shown by Table 3, the addition of collision constraints in the PBD solver doubles its convergence time on a full model. However, the use of collisions is mandatory as without them, two main phenomena occur. The first is that a significant part of the muscle grows inside the bone and neighbor muscles when it inflates. This cancels most of the inflation effect naturally produced on the skin by the muscle volume preservation and thus significantly reduces the deformation plausibility, as illustrated in Figure 15 and in the accompanying video. The second is the unnatural behavior of muscles that are allowed to move through where the bones should be, instead of moving around them (for instance on a shoulder).

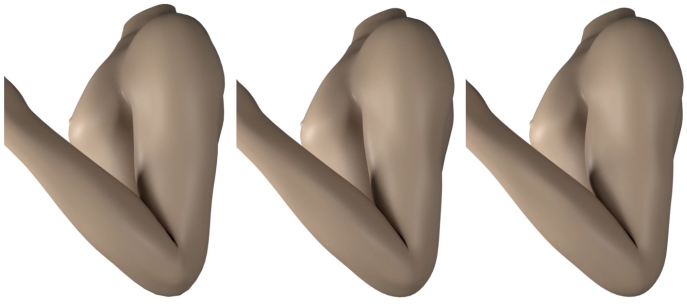


Figure 15. The arm bent from left to right with Implicit Skinning only, with our muscle but without resolving muscle/bones collisions and with our muscles and muscle/bones collisions resolved. Without collision, the muscle deformation in the middle is very similar to the solution without muscle shown on the left. On the right, the collision makes the muscle deformation clearly visible.

8.6. Muscle dynamics

Even though PBD is appreciated for its robustness, fast execution, and ease of implementation, there are also certain trade-offs. One issue is the dependence between the time integration step and the stiffness of the constraints: larger time steps yield softer constraints, thus a more rubbery material. This issue could possibly be avoided using the method proposed by Macklin et al. [82], as it could improve predictability of the results with different time steps.

Projective Dynamics [83] have been considered in place of PBD. It averages all constraint projections in one global step which limits oscillations. However, the pre-factorization of the global step matrix in Projective Dynamics complicates collision resolution, especially in close proximity scenarios such as muscle-bone or muscle-muscle scenarios. Therefore, we have for now decided not to pursue this approach.

Unless muscles are set with a too low stiffness, which is unrealistic as muscles are stiff organs compared to soft tissues, especially when activated, the muscle dynamic is stable or most muscles (as pectorals, biceps, triceps muscles) even with muscle/bones and muscle/muscle collisions enabled. Some difficulties may arise when muscles undergo large deformations due to collisions. In this case collisions are strongly contradicting the muscle elasticity and undesired oscillations may arise. In such situations, our solution is to pinch the muscle between two other muscles (as a small decomposition in fibers) and increase the stiffness. This is a limitation of our model under its current form that would be interesting to solve in further research.

Another limitation when using PBD is that the jiggling of muscles may be hard to generate. While simple oscillations are easy to obtain (such as the one shown in the video on the biceps when shaking the arm), it is far more challenging to generate plausible longitudinal jiggling as those expected on muscle legs during a run, even acting on all PBD parameters. This is partially due to the use of a dynamic system based on constraints rather than forces.

8.7. Integration in other skinning frameworks

On the one hand, while our muscle primitives have been specifically designed to enrich the Implicit Skinning approach [8], they may benefit weight-based geometric skinning techniques [58, 59, 61]. In these systems, a set of weights is defined over the mesh for each muscle. Mesh vertices with non-zero weights store the initial field value of the corresponding muscle primitives in rest pose. During the animation, muscle deformation and skinning are performed. Then, for each vertex with non-zero weight, a displacement corresponding to its projection on its initial field value in the deformed muscle field is computed (using for instance a gradient descent). Vertex positions are then updated by applying the weighted displacements.

On the other hand, real-time physically or dynamic based skinning methods, such as position-based skinning [84], usually rely on a volumetrical mesh deformed by minimizing a function defined by constraints. In that case, the definition and insertion of constraints transferring the muscle field function deformations on the volumetrical mesh remains an open challenge. In addition, the interaction of these new constraints with the others will also have to be specifically studied, especially because some of them may violate each other. A way to avoid these issues is to apply the muscle deformations as a post-process, but all physical properties and eventual collision-handling would be lost.

8.8. Limitations

As can be seen in Figure 12, computations required by Implicit Skinning are the main bottleneck for a full model. This is due to the very large number of field functions that are structured in a composition tree [85] and evaluated several times per mesh vertex for the isosurface tracking. This phenomenon is accentuated by the increased number of deformed vertices due to the muscle dynamics. Even though we use a bounding box hierarchy to avoid unnecessary field evaluations, there is room for specifically improving the field function filtering and

composition tree evaluations. We currently rely on a CPU implementation which is multi-threaded, but not heavily optimized. Both Implicit Skinning and PBD may be efficiently implemented on a GPU, and it would be interesting to study a dedicated GPU implementation including our muscle primitives.

The variety of muscle shapes produced with our model is well-suited for the production of plausible body dynamics of a human character. The main limitation is the application to non-fusiform muscles as the pectoral, for which we allow the introduction of visible deformations, but we provide a coarse approximation of the muscle shape and the deformations could be enhanced with a more accurate model. Different types of muscle primitives may be studied to adapt to a larger variety of virtual characters, including animals and imaginary creatures.

9. Conclusion

In this paper, we have presented a method to animate a character's muscular system which achieves expressive, artistically controllable results while maintaining interactive frame-rates.

To achieve this goal, we designed a family of shapes defined as isosurfaces of a scalar field. These shapes mimic the appearance of human muscles and are able to reproduce the contraction, extension, volume preservation and activation of the real muscles. Furthermore, we let the central axis of our muscles be driven by physics-based simulation, thus enabling highly dynamic effects such as jiggling. Using 3D scalar field representation allows us to resolve collisions efficiently between muscles, bones and skin. Another advantage of this implicit representation is the ability to use the muscles as skin deformers while reaping the benefits of Implicit Skinning.

As future work, a sketching approach may be investigated, enabling users to sketch the muscle deformation in different poses. We also believe that combining physical simulation with implicit modeling for animation is a promising line of research. Indeed, using a similar representation for other anatomic elements with highly dynamic behaviour and effect on the skin such as fat tissues or cartilages (nose, ears) could improve the ability of animators to efficiently produce realistic visual effects.

10. Acknowledgements

This research has been partially supported by the FOLD-Dyn project (ANR-16-CE33-0015-01) and the CIMI Labex (ANR-11-LABX-0040). This material is also based upon work supported by the National Science Foundation under Grant Numbers IIS-1617172, IIS-1622360 and IIS-1764071. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also gratefully acknowledge the support of Activision, Adobe, and hardware donation from NVIDIA Corporation. The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. Finally, we wish to thank the Anatoscope company and more specifically Pr François Faure for the gracious provision of their anatomical model.

References

[1] Magnenat-Thalmann, N, Laperrire, R, Thalmann, D. Joint-Dependent Local Deformations for Hand Animation and Object Grasping. In: Proceedings of Graphics Interface. GI 1988; Edmonton, Canada; 1988, p. 26–33.

[2] Kavan, L, Collins, S, Žára, J, O'Sullivan, C. Skinning with Dual Quaternions. In: Proceedings of the Symposium on Interactive 3D Graphics and Games. I3D '07; Seattle, USA: ACM. ISBN 978-1-59593-628-8; 2007, p. 39–46. URL: http://doi.acm.org/10.1145/1230100.1230107. doi:10.1145/1230100.1230107.

[3] Lewis, JP, Cordner, M, Fong, N. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '00; New Orleans, USA: ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5; 2000, p. 165–172. URL: http://dx.doi.org/10.1145/344779.344862. doi:10.1145/344779.344862.

[4] Wilhelms, J, Van Gelder, A. Anatomically-based modeling. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '97; New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7; 1997, p. 173–180. URL: http://dx.doi.org/10.1145/258734.258833. doi:10.1145/258734.258833.

[5] Mohr, A, Gleicher, M. Building efficient, accurate character skins from examples. ACM Transactions on Graphics 2003;22(3):562–568. URL: http://doi.acm.org/10.1145/882262.882308. doi:10.1145/882262.882308.

[6] Wang, RY, Pulli, K, Popović, J. Real-time enveloping with rotational regression. ACM Transactions on Graphics 2007;26(3). URL: http://doi.acm.org/10.1145/1276377.1276468. doi:10.1145/1276377.1276468.

[7] Vaillant, R, Barthe, L, Guennebaud, G, Cani, MP, Rohmer, D, Wyvill, B, et al. Implicit Skinning: Real-time Skin Deformation with Contact Modeling. ACM Transactions on Graphics 2013;32(4):125:1–125:12. doi:10.1145/2461912.2461960.

[8] Vaillant, R, Guennebaud, G, Barthe, L, Wyvill, B, Cani, MP. Robust Iso-surface Tracking for Interactive Character Skinning. ACM Transactions on Graphics 2014;33(6):189:1–189:11. URL: http://doi.acm.org/10.1145/2661229.2661264. doi:10.1145/2661229.2661264.

[9] Müller, M, Heidelberger, B, Hennix, M, Ratcliff, J. Position based dynamics. Journal of Visual Communication and Image Representation 2007;18(2):109 – 118. URL: http://www.sciencedirect.com/science/article/pii/S1047320307000065. doi:http://dx.doi.org/10.1016/j.jvcir.2007.01.005.

[10] Mukai, T. Example-Based Skinning Animation. Springer International Publishing; 2016, p. 1–21.

[11] Allen, B, Curless, B, Popović, Z. Articulated body deformation from range scan data. ACM Transactions on Graphics 2002;21(3):612–619. URL: http://doi.acm.org/10.1145/566654.566626. doi:10.1145/566654.566626.

[12] Anguelov, D, Srinivasan, P, Koller, D, Thrun, S, Rodgers, J, Davis, J. Scape: Shape completion and animation of people. ACM Transactions on Graphics 2005;24(3):408–416. URL: http://doi.acm.org/10.1145/1073204.1073207. doi:10.1145/1073204.1073207.

[13] Allen, B, Curless, B, Popović, Z, Hertzmann, A. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association; 2006, p. 147–156.

[14] Park, SI, Hodgins, JK. Capturing and animating skin deformation in human motion. In: ACM SIGGRAPH 2006 Papers. SIGGRAPH '06; New York, NY, USA: ACM. ISBN 1-59593-364-6; 2006, p. 881–889. URL: http://doi.acm.org/10.1145/1179352.1141970. doi:10.1145/1179352.1141970.

[15] Park, SI, Hodgins, JK. Data-driven modeling of skin and muscle deformation. ACM Transactions on Graphics 2008;27(3):96:1–96:6. URL: http://doi.acm.org/10.1145/1360612.1360695. doi:10.1145/1360612.1360695.

[16] Neumann, T, Varanasi, K, Hasler, N, Wacker, M, Magnor, M, Theobalt, C. Capture and statistical modeling of arm-muscle deformations. Computer Graphics Forum 2013;32(2):285–294.

[17] Loper, M, Mahmood, N, Black, MJ. Mosh: Motion and shape capture from sparse markers. ACM Transactions on Graphics 2014;33(6):220.

[18] Tsoli, A, Mahmood, N, Black, MJ. Breathing life into shape: capturing, modeling and animating 3d human breathing. ACM Transactions on Graphics 2014;33(4):52.

[19] Loper, M, Mahmood, N, Romero, J, Pons-Moll, G, Black, MJ. SMPL: A skinned multi-person linear model. ACM Transactions on Graphics 2015;34(6):248:1–248:16. URL: http://doi.acm.org/10.1145/2816795.2818013. doi:10.1145/2816795.2818013; proceedings of SIGGRAPH Asia.

[20] Mukai, T, Kuriyama, S. Efficient dynamic skinning with low-rank helper bone controllers. ACM Transactions on Graphics 2016;35(4):36:1–36:11. URL: http://doi.acm.org/10.1145/2897824.2925905. doi:10.1145/2897824.2925905.

[21] Pons-Moll, G, Romero, J, Mahmood, N, Black, MJ. Dyna: A model of dynamic human shape in motion. ACM Transactions on Graphics 2015;34(4):120:1–120:14. URL: http://doi.acm.org/10.1145/2766993. doi:10.1145/2766993.

[22] Kim, M, Pons-Moll, G, Pujades, S, Bang, S, Kim, J, Black, MJ, et al. Data-driven physics for human soft tissue animation. ACM Transactions on Graphics 2017;36(4):54:1–54:12. URL: http://doi.acm.org/10.1145/3072959.3073685. doi:10.1145/3072959.3073685.

[23] Schneider, DC, Davidson, TM, Nahum, AM. In vitro biaxial stress-strain response of human skin. Archives of Otolaryngology 1984;110(5):329–333. URL: +http://dx.doi.org/10.1001/archotol.1984.00800310053012. doi:10.1001/archotol.1984.00800310053012.

[24] Girard, M, Maciejewski, AA. Computational modeling for the computer animation of legged figures. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '85; New York, NY, USA: ACM. ISBN 0-89791-166-0; 1985, p. 263–270. URL: http://doi.acm.org/10.1145/325334.325244. doi:10.1145/325334.325244.

[25] Chadwick, JE, Haumann, DR, Parent, RE. Layered construction for deformable animated characters. SIGGRAPH Computer Graphics 1989;23(3):243–252. URL: http://doi.acm.org/10.1145/74334.74358. doi:10.1145/74334.74358.

[26] Zajac, FE. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. Critical reviews in biomedical engineering 1989;17 4:359–411.

[27] Chen, DT, Zeltzer, D. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. SIGGRAPH Computer Graphics 1992;26(2):89–98. URL: http://doi.acm.org/10.1145/142920.134016. doi:10.1145/142920.134016.

[28] Teran, J, Blemker, S, Ng-Thow-Hing, V, Fedkiw, R. Finite volume methods for the simulation of skeletal muscle. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '03; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 1-58113-659-5; 2003, p. 68–74. URL: http://dl.acm.org/citation.cfm?id=846276.846285.

[29] Teran, J, Sifakis, E, Blemker, S, Ng-Thow-Hing, V, Lau, C, Fedkiw, R. Creating and simulating skeletal muscle from the visible human data set. IEEE Transactions on Visualization and Computer Graphics 2005;11(3):317–328. doi:10.1109/TVCG.2005.42.

[30] Lee, SH, Sifakis, E, Terzopoulos, D. Comprehensive biomechanical modeling and simulation of the upper body. ACM Transactions on Graphics 2009;28(4):99:1–99:17. URL: http://doi.acm.org/10.1145/1559755.1559756. doi:10.1145/1559755.1559756.

[31] Pai, DK, Sueda, S, Wei, Q. Fast physically based musculoskeletal simulation. In: ACM SIGGRAPH 2005 Sketches. SIGGRAPH '05; New York, NY, USA: ACM; 2005,URL: http://doi.acm.org/10.1145/1187112.1187141. doi:10.1145/1187112.1187141.

[32] Pai, DK. Muscle mass in musculoskeletal models. Journal of Biomechanics 2010;43(11):2093 – 2098. URL: http://www.sciencedirect.com/science/article/pii/S0021929010002149. doi:http://dx.doi.org/10.1016/j.jbiomech.2010.04.004.

[33] Sueda, S, Kaufman, A, Pai, DK. Musculotendon simulation for hand animation. ACM Transactions on Graphics 2008;27(3):83:1–83:8. URL: http://doi.acm.org/10.1145/1360612.1360682. doi:10.1145/1360612.1360682.

[34] Sueda, S, Jones, GL, Levin, DIW, Pai, DK. Large-scale dynamic simulation of highly constrained strands. ACM Transactions on Graphics 2011;30(4):39:1–39:10. URL: http://doi.acm.org/10.1145/2010324.1964934. doi:10.1145/2010324.1964934.

[35] McAdams, A, Zhu, Y, Selle, A, Empey, M, Tamstorf, R, Teran, J, et al. Efficient elasticity for character skinning with contact and collisions. ACM Transactions on Graphics 2011;30(4):37:1–37:12. URL: http://doi.acm.org/10.1145/2010324.1964932. doi:10.1145/2010324.1964932.

[36] Patterson, T, Mitchell, N, Sifakis, E. Simulation of complex nonlinear elastic bodies using lattice deformers. ACM Transactions on Graphics 2012;31(6):197:1–197:10. URL: http://doi.acm.org/10.1145/2366145.2366216. doi:10.1145/2366145.2366216.

[37] Fan, Y, Litven, J, Pai, DK. Active volumetric musculoskeletal systems. ACM Transactions on Graphics 2014;33(4):152:1–152:9. URL: http://doi.acm.org/10.1145/2601097.2601215. doi:10.1145/2601097.2601215.

[38] Sachdeva, P, Sueda, S, Bradley, S, Fain, M, Pai, DK. Biomechanical simulation and control of hands and tendinous systems. ACM Transactions on Graphics 2015;34(4):42:1–42:10. URL: http://doi.acm.org/10.1145/2766987. doi:10.1145/2766987.

[39] Gao, M, Mitchell, N, Sifakis, E. Steklov-poincaré skinning. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '14; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association; 2014, p. 139–148. URL: http://dl.acm.org/citation.cfm?id=2849517.2849541.

[40] Saito, S, Zhou, ZY, Kavan, L. Computational bodybuilding: Anatomically-based modeling of human bodies. ACM Transactions on Graphics 2015;34(4):41:1–41:12. URL: http://doi.acm.org/10.1145/2766957. doi:10.1145/2766957; proceedings of SIGGRAPH 2015.

[41] Kadleček, P, Ichim, AE, Liu, T, Křivánek, J, Kavan, L. Reconstructing personalized anatomical models for physics-based body animation. ACM Transactions on Graphics 2016;35(6):213:1–213:13. URL: http://doi.acm.org/10.1145/2980179.2982438. doi:10.1145/2980179.2982438.

[42] Mohr, A, Tokheim, L, Gleicher, M. Direct manipulation of interactive character skins. In: Proceedings of the 2003 Symposium on Interactive 3D Graphics. I3D '03; New York, NY, USA: ACM. ISBN 1-58113-645-5; 2003, p. 27–30. URL: http://doi.acm.org/10.1145/641480.641488. doi:10.1145/641480.641488.

[43] Merry, B, Marais, P, Gain, J. Animation space: A truly linear framework for character animation. ACM Transactions on Graphics 2006;25(4):1400–1423. URL: http://doi.acm.org/10.1145/1183287.1183294. doi:10.1145/1183287.1183294.

[44] Forstmann, S, Ohya, J, Krohn-Grimberghe, A, McDougall, R. Deformation styles for spline-based skeletal animation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association; 2007, p. 141–150.

[45] Kavan, L, Collins, S, Žára, J, O'Sullivan, C. Geometric skinning with approximate dual quaternion blending. ACM Transactions on Graphics 2008;27(4):105. URL: http://doi.acm.org/10.1145/1409625.1409627. doi:10.1145/1409625.1409627.

[46] Jacobson, A, Baran, I, Popović, J, Sorkine, O. Bounded biharmonic weights for real-time deformation. ACM Transactions on Graphics 2011;30(4):78:1–78:8. URL: http://doi.acm.org/10.1145/2010324.1964973. doi:10.1145/2010324.1964973.

[47] Jacobson, A, Sorkine, O. Stretchable and twistable bones for skeletal shape deformation. ACM Transactions on Graphics 2011;30(6):165.

[48] Kavan, L, Sorkine, O. Elasticity-inspired deformers for character articulation. ACM Transactions on Graphics 2012;31(6):196:1–196:8. URL: http://doi.acm.org/10.1145/2366145.2366215. doi:10.1145/2366145.2366215.

[49] Le, BH, Hodgins, JK. Real-time skeletal skinning with optimized centers of rotation. ACM Transactions on Graphics 2016;35(4):37:1–37:10. URL: http://doi.acm.org/10.1145/2897824.2925959. doi:10.1145/2897824.2925959.

[50] Sloan, PPJ, Rose III, CF, Cohen, MF. Shape by example. In: Proceedings of the Symposium on Interactive 3D Graphics. I3D '01; Research Triangle Park, USA: ACM. ISBN 1-58113-292-1; 2001, p. 135–143. URL: http://doi.acm.org/10.1145/364338.364382. doi:10.1145/364338.364382.

[51] Kry, PG, James, DL, Pai, DK. Eigenskin: Real time large deformation character skinning in hardware. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.

SCA '02; San Antonio, USA: ACM. ISBN 1-58113-573-4; 2002, p. 153–159. URL: http://doi.acm.org/10.1145/545261.545286. doi:10.1145/545261.545286.

[52] Hahn, F, Martin, S, Thomaszewski, B, Sumner, R, Coros, S, Gross, M. Rig-space physics. ACM Transactions on Graphics 2012;31(4):72:1–72:8. URL: http://doi.acm.org/10.1145/2185520.2185568. doi:10.1145/2185520.2185568.

[53] Xu, H, Barbič, J. Pose-space subspace dynamics. ACM Transactions on Graphics 2016;35(4):35:1–35:14. URL: http://doi.acm.org/10.1145/2897824.2925916. doi:10.1145/2897824.2925916.

[54] Lee, GS, Hanner, F. Practical experiences with pose space deformation. In: Sketches of ACM SIGGRAPH Asia. SIGGRAPH Asia '09; Yokohama, Japan: ACM; 2009, p. 43:1–43:1. URL: http://doi.acm.org/10.1145/1667146.1667201. doi:10.1145/1667146.1667201.

[55] Palamar, T. Mastering Autodesk Maya 2016. 1st ed.; Alameda, CA, USA: SYBEX Inc.; 2015. ISBN 1119059828, 9781119059820.

[56] Weta Digital, . FEM horse simulation. Web Article; 2015. URL: https://www.wetafx.co.nz/research-and-tech/technology/tissue/.

[57] Scheepers, F, Parent, RE, Carlson, WE, May, SF. Anatomy-based modeling of the human musculature. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '97; New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7; 1997, p. 163–172. URL: http://dx.doi.org/10.1145/258734.258827. doi:10.1145/258734.258827.

[58] Ramos, J, Larboulette, C. A muscle model for enhanced character skinning. Journal of WSCG 2013;21(2):107–116.

[59] Lee, KS, Ashraf, G. Simplified muscle dynamics for appealing real-time skin deformation. In: Proceedings of the International Conference on Computer Graphics and Virtual Reality. CGVR'07; Las Vegas, Nevada: CSREA Press. ISBN 1-60132-028-0; 2007, p. 160–168.

[60] Hyun, DE, Yoon, SH, Chang, JW, Seong, JK, Kim, MS, Jüttler, B. Sweep-based human deformation. The Visual Computer 2005;21(8):542–550. URL: https://doi.org/10.1007/s00371-005-0343-x. doi:10.1007/s00371-005-0343-x.

[61] Leclercq, A, Akkouche, S, Galin, . Mixing triangle meshes and implicit surfaces in character animation. In: Proceedings of Eurographics Workshop on Computer Animation and Simulation. 2001, p. 37–47.

[62] Macêdo, I, Gois, JP, Velho, L. Hermite radial basis functions implicits. Computer Graphics Forum 2011;30(1):27–42. URL: http://dx.doi.org/10.1111/j.1467-8659.2010.01785.x. doi:10.1111/j.1467-8659.2010.01785.x.

[63] Barthe, L, Wyvill, B, de Groot, E. Controllable binary CSG operator for "soft objects". International Journal of Shape Modeling 2004;10(2):135–154. URL: http://www.worldscientific.com/doi/abs/10.1142/S021865430400064X. doi:10.1142/S021865430400064X.

[64] Gourmel, O, Barthe, L, Cani, MP, Wyvill, B, Bernhardt, A, Paulin, M, et al. A Gradient-based Implicit Blend. ACM Transactions on Graphics 2013;32(2):12:1–12:12. URL: http://doi.acm.org/10.1145/2451236.2451238. doi:10.1145/2451236.2451238.

[65] Angles, B, Tarini, M, Wyvill, B, Barthe, L, Tagliasacchi, A. Sketch-based implicit blending. ACM Transactions on Graphics 2017;36(6):181:1–181:13. URL: http://doi.acm.org/10.1145/3130800.3130825. doi:10.1145/3130800.3130825.

[66] Sorkine, O, Alexa, M. As-rigid-as-possible surface modeling. In: Proceedings of the Symposium on Geometry Processing; vol. 4 of *SGP 07*. Barcelona, Spain; 2007, p. 109–116.

[67] Anatomical digital model. Anatoscope Inc 2015;URL: https://www.anatoscope.com/.

[68] Lee, D, Glueck, M, Khan, A, Fiume, E, Jackson, K. Modeling and simulation of skeletal muscle for computer graphics: A survey. Foundations and Trends in Computer Graphics and Vision 2012;7(4):229–276. doi:10.1561/0600000036.

[69] Kobbelt, L, Vorsatz, J, Seidel, HP. Multiresolution hierarchies on unstructured triangle meshes. Computational Geometry 1999;14(1-3):5–24.

[70] Panozzo, D, Baran, I, Diamanti, O, Sorkine-Hornung, O. Weighted averages on surfaces. ACM Transactions on Graphics 2013;32(4):60.

[71] Murai, A, Hong, QY, Yamane, K, Hodgins, JK. Dynamic Skin Deformation Simulation Using Musculoskeletal Model and Soft Tissue Dynamics. In: Grinspun, E, Bickel, B, Dobashi, Y, editors. Pacific Graphics Short Papers. The Eurographics Association. ISBN 978-3-03868-024-6; 2016,doi:10.2312/pg.20161335.

[72] Urbanchek, MG, Picken, EB, Kalliainen, LK, Kuzon Jr., WM. Specific force deficit in skeletal muscles of old rats is partially explained by the existence of denervated muscle fibers. The Journals of Gerontology: Series A 2001;56(5):B191–B197. URL: http://dx.doi.org/10.1093/gerona/56.5.B191. doi:10.1093/gerona/56.5.B191.

[73] Deul, C, Charrier, P, Bender, J. Position-based rigid body dynamics. Computer Animation and Virtual Worlds 2014;27(2):103–112. URL: http://dx.doi.org/10.1002/cav.1614. doi:10.1002/cav.1614.

[74] Canezin, F, Guennebaud, G, Barthe, L. Adequate inner bound for geometric modeling with compact field functions. Computers & Graphics 2013;37(6):565–573. URL: http://www.sciencedirect.com/science/article/pii/S009784931300099X. doi:10.1016/j.cag.2013.05.024; shape Modeling International (SMI) Conference 2013.

[75] Sabin, M. The use of potential surfaces for numerical geometry. Tech. Rep. VTO/MS/153; British Aerospace Corporation; Weybridge, United Kingdom; 1968.

[76] Ricci, A. A constructive geometry for computer graphics. The Computer Journal 1973;16(2):157–160. doi:10.1093/comjnl/16.2.157.

[77] Pasko, A, Adzhiev, V, Sourin, A, Savchenko, V. Function representation in geometric modeling: concepts, implementation and applications. The Visual Computer 1995;11(8):429–446. doi:10.1007/BF02464333.

[78] Bernhardt, A, Barthe, L, Cani, MP, Wyvill, B. Implicit blending revisited. Computer Graphics Forum 2010;29(2):367–375. URL: http://webhome.cs.uvic.ca/~blob/publications/ibr.pdf.

[79] Ali-Hamadi, D, Liu, T, Gilles, B, Kavan, L, Faure, F, Palombi, O, et al. Anatomy transfer. ACM Transactions on Graphics 2013;32(6):188:1–188:8. URL: http://doi.acm.org/10.1145/2508363.2508415. doi:10.1145/2508363.2508415.

[80] Sifakis, E, Neverov, I, Fedkiw, R. Automatic determination of facial muscle activations from sparse motion capture marker data. In: ACM SIGGRAPH 2005 Papers. SIGGRAPH '05; New York, NY, USA: ACM; 2005, p. 417–425. URL: http://doi.acm.org/10.1145/1186822.1073208. doi:10.1145/1186822.1073208.

[81] Lee, Y, Park, MS, Kwon, T, Lee, J. Locomotion control for many-muscle humanoids. ACM Transactions on Graphics 2014;33(6):218:1–218:11. URL: http://doi.acm.org/10.1145/2661229.2661233. doi:10.1145/2661229.2661233.

[82] Macklin, M, Müller, M, Chentanez, N. XPBD: Position-based simulation of compliant constrained dynamics. In: Proceedings of the International Conference on Motion in Games. MIG '16; San Francisco, USA: ACM. ISBN 978-1-4503-4592-7; 2016, p. 49–54. URL: http://doi.acm.org/10.1145/2994258.2994272. doi:10.1145/2994258.2994272.

[83] Bouaziz, S, Martin, S, Liu, T, Kavan, L, Pauly, M. Projective dynamics: Fusing constraint projections for fast simulation. ACM Trans Graph 2014;33(4):154:1–154:11. URL: http://doi.acm.org/10.1145/2601097.2601116. doi:10.1145/2601097.2601116.

[84] Abu Rumman, N, Fratarcangeli, M. Position-based skinning for soft articulated characters. Computer Graphics Forum 2015;34(2):240–250. URL: http://dx.doi.org/10.1111/cgf.12533. doi:10.1111/cgf.12533; proceedings of Eurographics.

[85] Wyvill, B, Guy, A, Galin, E. Extending the CSG tree: Warping, blending and boolean operations in an implicit surface modeling system. Computer Graphics Forum 1999;18(2):149–158. URL: http://dx.doi.org/10.1111/1467-8659.00365. doi:10.1111/1467-8659.00365.

Appendix A. Muscle volume derivation

The volume enclosed by the muscle's surface is given by Equation 3

$$V = \pi w^2 l$$

Proof.

$$V = \int_{s=0}^{1} \int_{\theta=0}^{2\pi} \int_{\rho=0}^{w\Phi(s)r(\theta)} \rho \mathrm{d}\rho \; \mathrm{d}\theta \; l \mathrm{d}s$$

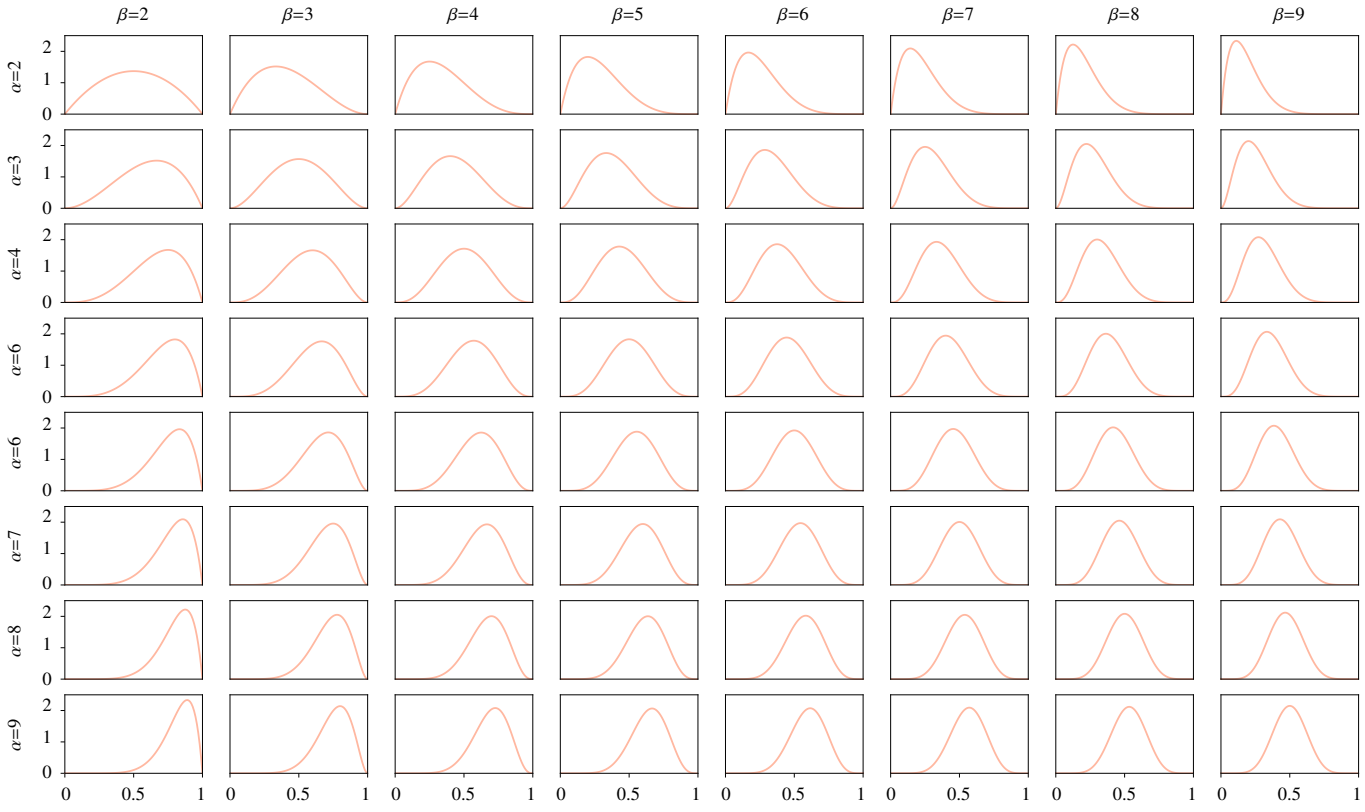


Figure A.16. Profiles of the φ function for values of α and β.

Integrating in ρ yields:

$$V = l \int_{s=0}^{1} \int_{\theta=0}^{2\pi} \frac{(w\Phi(s)r(\theta))^2}{2} \mathrm{d}\theta \mathrm{d}s$$
$$= w^2 l \int_{s=0}^{1} (\Phi(s))^2 \, \mathrm{d}s \int_{\theta=0}^{2\pi} \frac{(r(\theta))^2}{2} \mathrm{d}\theta$$

The integral in θ is the area of an ellipse of semi-axis length u and v (see Equation 5)

$$V = w^2 l \int_{s=0}^{1} (\Phi(s))^2 \, \mathrm{d}s \, \pi u v$$

Since we enforce $uv = 1$, it yields

$$V = \pi w^2 l \int_{s=0}^{1} (\Phi(s))^2 \, \mathrm{d}s$$

We also constrain $\int \Phi^2 = 1$, thus we have

$$V = \pi w^2 l$$

□

Appendix B. Derivation of function K

The Euler *beta function B* is defined as

$$B(\alpha, \beta) = \int_0^1 u^{\alpha-1}(1-u)^{\beta-1} \mathrm{d}u$$

Let us define

$$I(\alpha, \beta) = \int_0^1 u^{2(\alpha-1)}(1-u)^{2(\beta-1)} \mathrm{d}u \; = B(2\alpha - 1, 2\beta - 1)$$

which appears in the denominator of Equation 4.

We define the linear interpolation $\Phi(s)$ between $\varphi(\alpha_0, \beta_0; s)$ and $\varphi(\alpha_1, \beta_1; s)$, governed by the activation parameter a, such that $\int_0^1 (\Phi(s))^2 \, \mathrm{d}s = 1, \forall a \in [0, 1]$ as:

$$\Phi(s) = \frac{(1-a)\varphi(\alpha_0, \beta_0; s) + a\varphi(\alpha_1, \beta_1; s)}{\sqrt{F(a)}}$$

where $\sqrt{F(a)}$ is the $\mathcal{L}^2$-norm of the numerator, i,e

$$F(a) = \int_0^1 ((1-a)\varphi(\alpha_0, \beta_0; s) + a\varphi(\alpha_1, \beta_1; s))^2 \mathrm{d}s$$

We show that $F(a)$ can be expressed as a second-order polynomial in a whose coefficients depend only on the chosen values for $\alpha_1, \alpha_2, \beta_1$ and β_2.

Proof.

$$F(a) = \int_0^1 ((1-a)\varphi(\alpha_0, \beta_0; s) + a\varphi(\alpha_1, \beta_1; s))^2 \, \mathrm{d}s$$
$$= (1-a)^2 \int_0^1 (\varphi(\alpha_0, \beta_0; s))^2 \, \mathrm{d}s$$
$$+ a^2 \int_0^1 (\varphi(\alpha_1, \beta_1; s))^2 \, \mathrm{d}s$$
$$+ 2a(1-a) \int_0^1 \varphi(\alpha_0, \beta_0; s)\varphi(\alpha_1, \beta_1; s) \mathrm{d}s$$
$$= (1-a)^2 + a^2 + 2a(1-a)K(\alpha_0, \alpha_1, \beta_0, \beta_1),$$

where $K(\alpha_0, \alpha_1, \beta_0, \beta_1)$ is the constant term equal to:

$$\frac{\int_0^1 s^{\alpha_0+\alpha_1-2}(1-s)^{\beta_0+\beta_1-1}\mathrm{d}s}{\sqrt{\int_0^1 y^{2(\alpha_0-1)}(1-y)^{2(\beta_0-1)}\mathrm{d}y \int_0^1 y^{2(\alpha_1-1)}(1-y)^{2(\beta_1-1)}\mathrm{d}y}},$$

which can be expressed in terms of the *beta function B* as:

$$K(\alpha_0, \alpha_1, \beta_0, \beta_1) = \frac{B(\alpha_0+\alpha_1-1, \beta_0+\beta_1-1)}{\sqrt{B(2\alpha_0-1, 2\beta_0-1)B(2\alpha_1-1, 2\beta_1-1)}}.$$

□