Comprehensive Biomechanical Modeling and Simulation of the Upper Body

Sung-Hee Lee Eftychios Sifakis Demetri Terzopoulos
University of California, Los Angeles

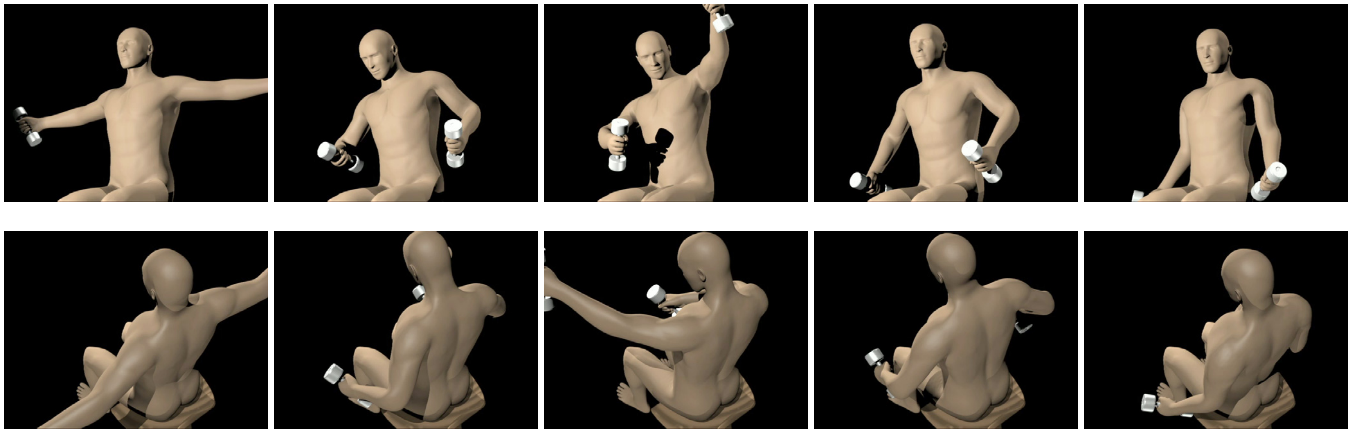


Figure 1: *The biomechanical model in action. A motion controller drives the musculoskeletal system toward a sequence of target poses.*

Abstract

We introduce a comprehensive biomechanical model of the human upper body. Our model confronts the combined challenge of modeling and controlling more or less all of the relevant articular bones and muscles, as well as simulating the physics-based deformations of the soft tissues. Its dynamic skeleton comprises 68 bones with 147 jointed degrees of freedom, including those of each vertebra and most of the ribs. To be properly actuated and controlled, the skeletal submodel requires comparable attention to detail with respect to muscle modeling. We incorporate 814 muscles, each of which is modeled as a piecewise uniaxial Hill-type force actuator. To simulate biomechanically realistic flesh deformations, we also develop a coupled finite element model with the appropriate constitutive behavior, in which are embedded the detailed 3D anatomical geometries of the hard and soft tissues. Finally, we develop an associated physics-based animation controller that computes the muscle activation signals necessary to drive the elaborate musculoskeletal system in accordance with a sequence of target poses specified by an animator.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—[Physically based modeling] I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—[Animation]

Keywords: Human Modeling and Animation, Muscle-Based Animation, Soft Tissue Simulation, Biomechanics, Finite Elements, Rigid/Deformable Dynamics and Control

1 Introduction

Realistic anatomical modeling aimed at achieving high-fidelity human animation remains a major challenge in computer graphics. Since the earliest work [Chadwick et al. 1989], researchers have made significant progress in this domain by focusing their attention on biomechanically modeling various body parts, including the face [Lee et al. 1995; Kähler et al. 2002; Sifakis et al. 2005], the neck [Lee and Terzopoulos 2006], the torso [Zordan et al. 2004; DiLorenzo et al. 2008], the hand [Albrecht et al. 2003; Tsang et al. 2005; van Nierop et al. 2008; Sueda et al. 2008], and the leg [Dong et al. 2002; Komura et al. 2000]. Because of its complexity, however, researchers have thus far shied away from undertaking a detailed biomechanical modeling of the entire human body. In particular, the spine and torso have been drastically simplified in prior work—either they were modeled in a strictly kinematic manner [Monheit and Badler 1991] or the many articular vertebrae and ribs were grouped and treated as compound rigid bodies even in the most detailed dynamic models (e.g., [Nakamura et al. 2005; DiLorenzo et al. 2008]).

In fact, most of the complexity of the human musculoskeletal system is due to the head-neck-trunk complex, in which there are approximately 57 articular bones and many more muscle actuators. Furthermore, the ribs form closed kinematic loops, which introduces additional complexities for biomechanical simulation. Nonetheless, the comprehensive biomechanical modeling and control of the human upper body is the most principled approach to simulating the full range of motions and deformations that it is capable of producing, from pronounced motions such as flexing the arms and spine to more subtle motions such as respiration and laryngeal movements. We develop a detailed biomechanical model of the human upper body, comprising the head, neck, trunk, and arms, for use in physics-based computer animation. Our model features a musculoskeletal system with a full complement of muscle actuators and a coupled finite element simulation of soft tissue deformations. We also develop an associated dynamic animation controller that computes the muscle activation signals necessary to drive the musculoskeletal system in accordance with a sequence of target poses specified by an animator.

We confront the challenge of modeling more or less all of the relevant articular bones, creating a physics-based skeletal model that consists of *68 bones* and *147 DOFs* (degrees of freedom), with each vertebra and most ribs having independent DOFs. To be properly actuated and controlled, our jointed skeletal model requires a comparable level of detail with respect to muscle modeling. We incorporate *814 muscles*, which are modeled as piecewise line segment simplified Hill-type force actuators. Coupled to the musculoskeletal system is a finite element simulation of the soft tissues, which enhances the visual richness of our model consistent with a detailed 3D modeling of the musculature. A total of *354,000 tetrahedral finite elements* (including those in the legs) are simulated to produce realistic, physics-based flesh deformations.

Ours is the most detailed biomechanical model for computer animation developed to date. Its unprecedented complexity requires special attention to making the simulation feasible, through the following features:

- A detailed modeling of the rib bones can introduce closed loops, which would complicate the simulation of the skeleton. In order to exploit an efficient, linear-time dynamics algorithm, we model the skeletal structure topologically as a tree by connecting the ribs to the costal cartilages using soft rather than hard constraints.

- The naive creation of a tetrahedral finite element mesh directly from the 3D surface representations of individual soft tissue components would result in a prohibitive number of finite elements, often with poor aspect ratios. We achieve realistic, robust, and reasonably efficient simulation of soft tissue deformation within the finite element framework by decoupling the visualization geometry from the simulation geometry. We further reduce the complexity of the simulation mesh by allowing T-junctions in the mesh simplification process.

- Our volumetric simulation mesh does not resolve the bones, but it overlaps with them, thus necessitating a special mechanical coupling between the rigid-body and the nonrigid tissue simulations in order to make the soft tissue deform according to the movement of the skeleton. We achieve this using soft constraints that produce coupling forces between sample points on the surface of the bones and appropriate locations in the soft tissue model.

- The existence of bones that are not actuated by muscles (e.g., the sternum) prevents us from applying the recursive Newton-Euler inverse dynamics algorithm. Hence, we perform inverse dynamics by employing a hybrid system dynamics algorithm.

The remainder of the paper is organized as follows: Section 2 discusses additional relevant prior work. Section 3 presents the details of our upper-body model, including the skeleton (Section 3.1), the muscle actuators (Section 3.2), and the soft-tissue simulation (Section 3.3). Section 4 develops the control algorithms that we use to animate our biomechanical model. Section 5 presents our experiments and animation results. Section 6 concludes the paper and discusses promising avenues for future work.

2 Related Work

Our work spans physical and biological modeling and simulation in computer animation, as well as related fields such as anatomy, biomechanics, and control.

Researchers have developed various techniques for anatomical modeling, including muscles and other soft tissues. Scheepers et al. [1997] and Wilhelms and Gelder [1997] modeled muscles geometrically, using primitives such as ellipsoids, and muscle deformations were determined entirely by the configuration of the bones, thus precluding the synthesis of muscle dynamics. Muscle deformation is more accurately simulated biomechanically, by appropriately modeling the stress-strain constitutive characteristics of muscles. Chen and Zeltzer [1992] modeled individual muscles as large finite elements and applied a Hill-type muscle force model to simulate the deformation of muscular soft tissues. Other interesting muscle models include the B-spline solids developed by Ng-Thow-Hing [2001] and the strand model developed by Pai et al. [2005]. Applying the latter, Sueda et al. [2008] created realistic skin deformation of the hand due to the underlying tendons and muscles.

Lee et al. [1995] used volume-preserving, multi-layer mass-spring-damper meshes with embedded muscle actuators to model the soft tissues of the face for biomechanical facial expression synthesis. Irving et al. [2004] enabled the robust simulation of soft tissue deformation by introducing invertible finite elements. Our soft tissue model is based on the work of Sifakis et al. [2005], where the muscles are embedded within flesh modeled using finite elements.

In contrast to facial animation where muscle models have been used for over two decades to generate expressions [Waters 1987], PD servos have traditionally been used to produce articulated skeletal animation [Hodgins et al. 1995; Faloutsos et al. 2001]. Recently, Komura et al. [1997; 2000] incorporated Hill-type muscle models into space-time optimization and motion retargeting to achieve dynamically and physiologically feasible motions. Zordan and colleagues [Zordan et al. 2004; DiLorenzo et al. 2008] developed a detailed torso model and simulated breathing and laughing. They used a Hill-type muscle model to actuate the thorax, but also used PD-servos to control the lumbar vertebrae. Lee and Terzopoulos [2006] biomechanically modeled the human neck-head-face complex and developed a two-layer neuromuscular controller comprising trained neural networks. The head-neck portion of our upper-body model includes the 7 cervical vertebrae present in their model, plus the hyoid bone and the thyroid cartilages modeled as rigid bodies, with a total of 300 muscles, far exceeding the 72 muscles in the earlier model.

The biomechanics community has ongoing efforts to create detailed human musculoskeletal models. Although state of the art models such as the Full-Body SIMM model (www.musculographics.com) and the AnyBody model (www.anybodytech.com) provide accurate muscle parameters, they unfortunately do not provide the data needed to control a very detailed model like ours, notably lacking data for muscles in the trunk. Another problem with these biomechanics models from the point of view of computer animation is that they do not specify the detailed geometries of muscles. In our model, we employ detailed muscle geometry data from the commercially available Ultimate Human model (www.cgcharacter.com). However, this model lacks information regarding the force generating properties of the muscles. We estimate muscle parameters—e.g., attachment points and physiological cross sectional areas—by analyzing the muscle geometry data, as detailed in Section 3.2.

There exists a large biomechanics literature on human motor control mechanisms. Traditionally, biomechanics researchers have attempted to interpret motor control strategy as an optimization process and have devoted effort to understanding the optimality criteria [Crowninshield 1978; Pandy et al. 1990]. Recently, some researchers have adopted robot control theories for human motor control. Sapio et al. [2005] proposed a task-level feedback control framework in the simulation of goal-directed human motion.

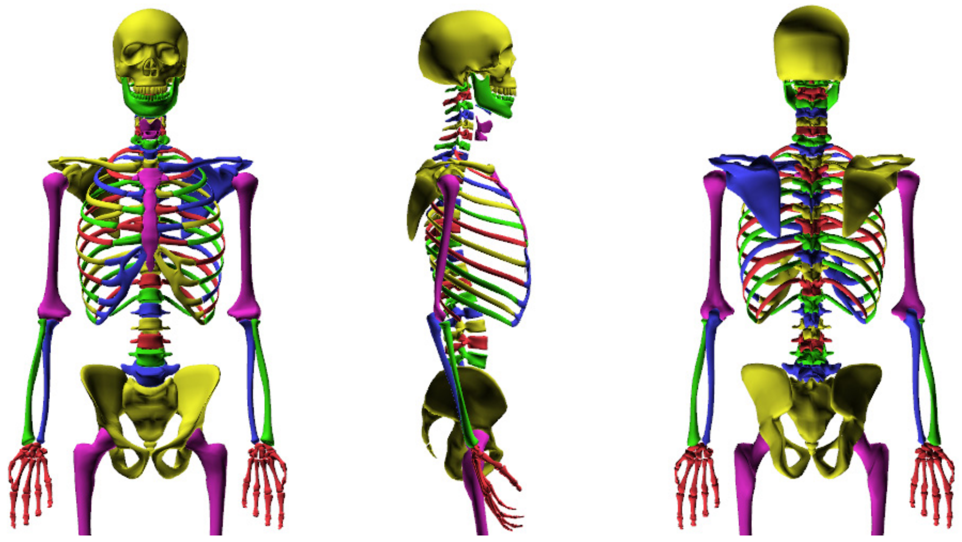


Figure 2: *Ventral (a), lateral (b), and dorsal (c) views of the modeled skeleton of the body. Most of the articular bones are individually modeled. There is a total of 68 bones articulated by 147 DOFs, of which 133 DOFs are in the head-neck-trunk region. Neighboring bones with the same color are treated as a single rigid body. The parameters of the skeleton are given in [Lee 2008].*

Thelen et al. [2003] used static optimization along with feedforward and feedback controls to drive the kinematic trajectory of a musculoskeletal leg model toward a set of desired kinematics, and reported that the muscle excitations computed by their method were similar to measured electromyographic patterns. Similarly, our motion controller computes muscle activation levels in the feedforward controller, but with a novel method for computing the feedforward signal.

We model almost all the relevant articular joints, including joints that cannot be controlled by muscles (i.e., the sternum and the costal cartilages). This prevents us from using the popular, recursive Newton-Euler inverse dynamics algorithm. Instead, we apply a hybrid system dynamics algorithm introduced by Featherstone [1987]. When input torques are given to some joints and desired accelerations are specified for other joints, the hybrid system dynamics algorithm computes the resulting accelerations for the torque-specified joints and the required torques for the acceleration-specified joints in linear time for open loop systems.[1] Kokkevis and Metaxas [1998] introduced a similar control algorithm, but it has $O(nm+m^3)$ complexity, where n is the number of DOFs and m is the number of acceleration-specified joints.

3 Modeling

This section presents each of the major components of our biomechanical model of the human upper body, namely the skeleton, the muscle-based actuation model, and the soft tissue modeling and simulation.

3.1 Skeletal Model

The skeleton is modeled as an articulated, multi-body dynamic system. As shown in Fig. 2, we individually modeled most of the articular bones in human upper body (the hands remain kinematically articulated). The system has a total of 68 bones with 147 DOFs. Among these, 133 DOFs are associated with the head-neck-trunk region. In particular, all the vertebrae in the lumbar, thoracic, and cervical regions are modeled as individual rigid bodies interconnected with 3-DOF joints. The first 10 ribs are able to rotate independently from the spine along axes running through costotransverse and costovertebral joints [Kapandji 1974] while the floating ribs (the 11th and 12th ribs) are rigidly attached to their parent vertebrae. Although costal cartilages are flexible bodies, we model them as rigid bodies, and the flexibility of the cartilages is emulated by 3-DOF joints connecting to the sternum and springs connecting to the ribs. By not modeling the joints between the costal cartilages and ribs, we maintained the skeleton as an open-loop system so that we can use a fast method for simulating articulated body dynamics. The hyoid bone and the thyroid cartilages are also modeled as rigid bodies. Even though the actual bones are not jointed to vertebrae, we modeled them for simplicity as child links of nearby vertebrae, with rotational joints positioned about 5cm posterior to the parent bones.

Since we are less concerned about highly coordinated motions of the clavicle, scapula, and humerus, we simplified the model of the shoulder; the clavicle has 2 DOFs and the scapula is rigidly attached to the clavicle. Nevertheless, this allows plausible movement for a modest range of upper arm motions.

The inertial properties of the skeleton are approximated from the dense volumetric mesh of the surrounding soft tissues. We associated the inertial parameters of each volumetric element to the near-

[1]In robotics, the hybrid system dynamics algorithm has been applied to mechanisms with a (zero-torque) free-floating root and acceleration-specified joints, such as robotic animals [Albro et al. 2000; Hu et al. 2005]. We present an inverse dynamics technique for the hybrid system, particularly when simulated using an implicit time-integration scheme. Note that our algorithm differs from other so-called “hybrid” control approaches, which are mostly concerned with switching between kinematic and dynamic simulation [Shapiro et al. 2003; Zordan and Hodgins 2002].

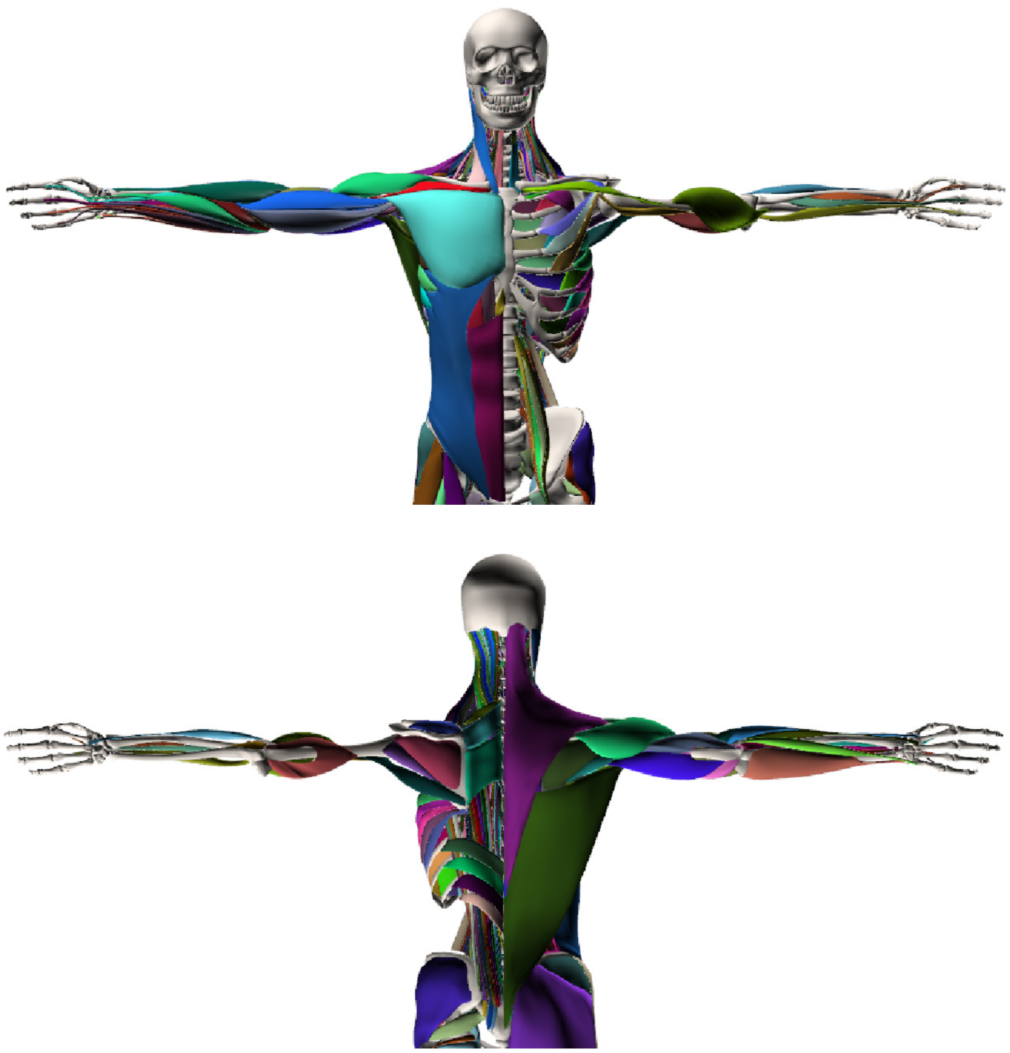


Figure 3: *Source geometry data of the modeled muscles. The superficial muscles are shown on the right side of the body, while deeper muscles are revealed on the left side.*

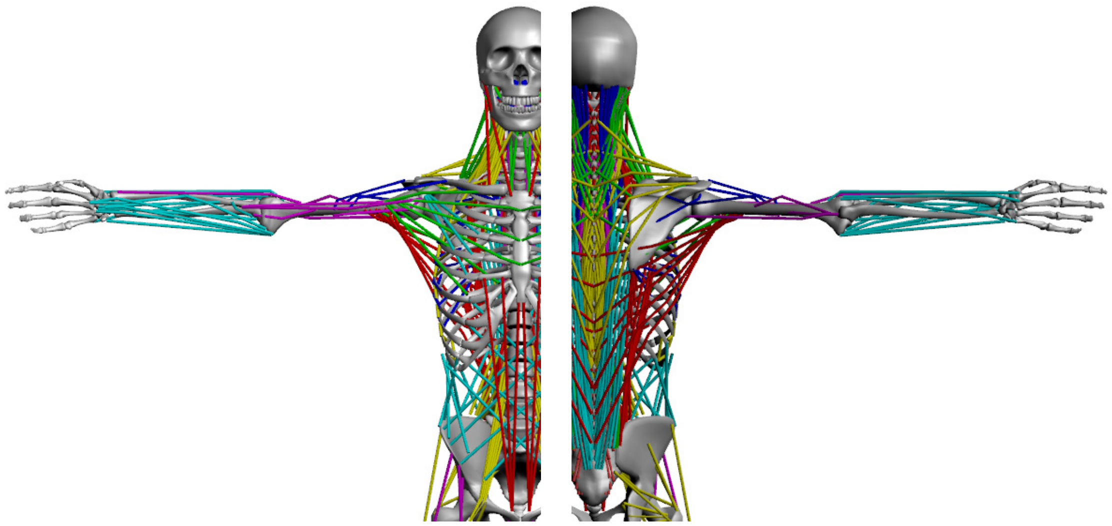


Figure 4: *The lines-of-action of the uniaxial muscle actuators in the body. A total of 814 actuators are modeled. Table 1 lists the modeled muscles.*

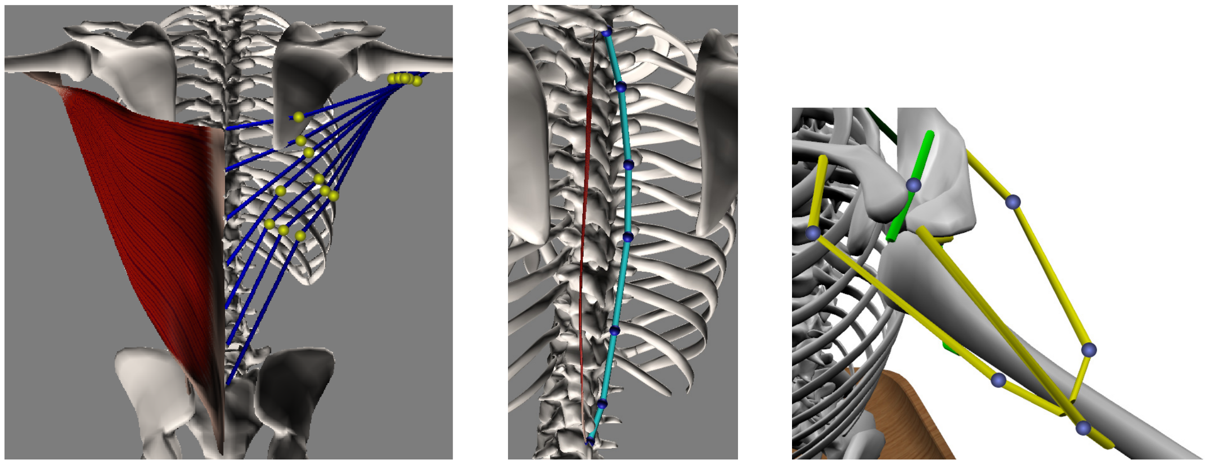


Figure 5: *(a) Broad muscles such as the Latissimus dorsi are modeled using multiple line segments. Yellow spheres indicate the position of via points. (b) When a deep muscle spans many vertebrae, one in every two or three vertebrae is assigned a fixation point (blue sphere) to conform the muscle to the movement of the vertebrae. (c) The via points of muscles that have a large range of motion are determined subjectively such that the resulting PLS muscle model deforms convincingly as the skeleton moves.*

Head/Neck muscles (300)
Iliocostalis thoracis (2), Interspinalis (12), Intertransversi (28), Rotatores (10), Semispinalis thoracics (4), Trapezius (4), Masseter (2), Iliocostalis cervicis (10), Longissimus capitis (16), Longissimus cervicis (10), Semispinalis cervicis (12), Splenius capitis (10), Semispinalis capitis (18), Longus capitis (8), Geniohyoid (2), Longus colli (14), Obliquus capitis (4), Omohyoid (2), Rectus capitis posterior (4), Rectus capitis anterior (2), Scalenes (50), Sternocleidomastoid (4), Sternohyoid (2), Sternothyroid (2), Stylohyoid (2), Rectus capitis lateralis (2), Levator scapulae (16), Multifidus (36), mylohyoid (2), Splenius cervicis (8), Rhomboid minor (2)
Trunk muscles (432)
External/Internal obliques (22), Rectus abdominis (6), Iliocostalis lumborum (24), Iliocostalis thoracis (18), Interspinalis (14), Intertransversi (34), Multifidus (116), Rotatores (30), Semispinalis thoracics (8), Spinalis thoracis (16), Trapezius (10), External intercostal (20), Pectoralis minor (6), Serratus anterior (16), Rhomboid major (2), Quadratus lumborum (10), Subclavius (2), Iliocostalis cervicis (2), Longissimus thoracis (48), Serratus posterior inferior (8), Internal intercostal (20)
Arm muscles (41)
Pectoralis major (6), Latissimus dorsi (7), Biceps brachii (2), Brachioradialis (1), Brachialis (1), Coracobrachialis (1), Triceps Brachii (3), Infraspinatus (1), Deltoid (3), Supraspinatus (1), Teres major (1), Extensor carpi radialis brevis (1), Extensor carpi radialis longus (1), Extensor carpi ulnaris (1), Supinator (1), Flexor carpi ulnaris (1), Flexor digitorum superficialis (4), Palmaris longus (1), Pronator quadratus (1), Pronator teres (2), Extensor pollicis longus (1)

Table 1: *A total of 814 muscles are modeled. The numbers in parentheses indicate the number of muscle actuators of each kind.*

est bone so that each bone's inertial tensor is augmented by the inertial parameters of the associated soft tissues. The parameters of the skeleton, among them the inertial properties and the location and axis of the joints, are provided in [Lee 2008].

3.2 Muscle Actuation Model

Ideally, to compute the correct activation levels of individual muscles, the parameters of all the muscles of the skeleton should be known. Since many muscles are highly correlated, assigning inexact parameter values to one muscle prevents the computation of the correct activation level of other muscles. Furthermore, if certain muscles are omitted, other muscles should be activated more highly in order to compensate for the missing muscles. These considerations present a challenge. Unfortunately, although the biomechanics literature provides accurate parameters for many muscles (e.g., [Holzbaur et al. 2005]), no available model provides a full set of parameter values for all the muscles in the upper body. Notably absent are the parameters of many of the spinal muscles that are important for maintaining the stability of the upper body.

As an alternative, we use the comprehensive muscle geometry information from the Ultimate Human model. Fig. 3 shows the detailed source geometry of the muscles. However, the model provides no information regarding the force generating properties of muscles, making it necessary to estimate these properties from the muscle geometry.

We model most of the skeletal muscles available in the source data—a total of 814 muscles, modeled using piecewise line segment (PLS) models (Fig. 4). Table 1 lists the modeled muscles. It may at first seem that we model more muscles than necessary, but given the large number of DOFs of our skeletal model, this is only about 2.8 times the minimum number of muscles required to actuate the system with one agonist/antagonist muscle pair per DOF.

We model broad muscles such as the trapezius and latissimus dorsi using multiple PLS models (Fig. 5(a)). Although the external obliques and internal obliques are abdominal muscles organized in different layers, we construct PLS models of the left (right) external obliques and right (left) internal obliques as if they were connected.

With regard to the deep muscles, we can determine the via points for the PLS model without much difficulty, because deep muscles are positioned close to bones and the distance from a muscle to neigh-

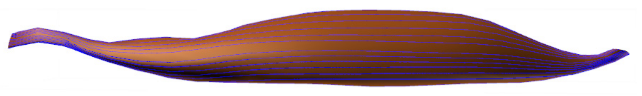


Figure 6: *The PCSA of a muscle is calculated as its volume divided by its mean fiber length. Blue lines indicate the fiber direction of the biceps brachii.*

boring bones does not change much as the bones move. When a deep muscle spans many vertebrae, via points are set only to some of the bones for computational efficiency (Fig. 5(b)). It is more difficult to model the superficial muscles using PLS models, since these muscles are rather far from bones, making it harder to determine via points, and because their positions relative to the bones vary significantly as the bones move. For superficial muscles, we subjectively selected a few (typically 2 to 4) via points, making sure that the resulting PLS muscle model deforms convincingly as the skeleton moves (Fig. 5(a),(c)).

We did not model the diaphragm as a grid of PLS muscles, but it is not important for pose control. As a result, respiratory movement is accomplished mainly by the intercostal muscles. We also omitted the transversus abdominis muscles to avoid the complication of modeling them as parallel PLS models.

The force generating characteristic of the PLS muscle is modeled as a *linearized Hill-type model* [Lee and Terzopoulos 2006]. Assuming that the length of the tendon is constant, we model a muscle force as the sum of the forces from a contractile element (CE) and a parallel element (PE).

The PE force is modeled as an unidirectional exponential spring; i.e.,

$$f_P = \max(0, k_s(\exp(k_c e) - 1) + k_d \dot{e}), \tag{1}$$

where k_s, k_c, and k_d are elastic and damping coefficients, $e = (l - l_0)/l_0$ is the strain of the muscle, with l and l_0 its length and slack length, respectively, and $\dot{e} = \dot{l}/l_0$ is the strain rate.

The CE force is expressed as

$$f_C = aF_l(l)F_v(\dot{l}), \tag{2}$$

where $0 \leq a \leq 1$ is the activation level of the muscle. The force-length relation is $F_l(l) = \max(0, k_{\max}(l - l_m))$, where $k_{\max}$ is the maximum stiffness of a fully activated muscle and l_m is the minimum length at which the muscle can produce force, and $F_v(\dot{l}) = \max(0, 1 + \min(\dot{l}, 0)/v_m)$, where $v_m(\geq 0)$ is the maximum contraction velocity under no load. We set $l_m = 0.3l_0$, $v_m = 8l_0$ sec^{-1}, and $k_c = 6$ for all the muscles. The coefficients k_s, k_d, and $k_{\max}$ for each muscle are scaled by the physiological cross sectional area (PCSA) of the muscle. We used $k_s = 0.8A$, $k_d = 0.1k_s$, and $k_{\max} = A/(l - l_m)$, where A is the PCSA of a muscle, which is calculated by dividing its geometric volume by its mean fiber length (Fig. 6). The parameters of each muscle, such as the PCSA, rest length, and the attachment points are listed in [Lee 2008].

It is important to note that even if we had faithfully modeled the majority of the skeletal muscles, the sternum and the costal cartilages cannot be controlled by muscles. This is natural because they are in fact moved only passively by the motion of the neighboring ribs; hence, we dub them *passive joints*. This does not pose any problem in forward dynamics simulation, but it does complicate inverse dynamics; i.e., computing the muscle forces needed to generate desired accelerations. We will address this problem in Section 4.3.

Even though some muscles exert forces on passive joints, we assume that such forces contribute negligibly to the accelerations of the passive joints compared to the forces exerted by the connecting tissues. Consequently, the equations of motion of the skeletal system are as follows:

$$\mathbf{M}(\mathbf{q}) \begin{bmatrix} \ddot{\mathbf{q}}_m \\ \ddot{\mathbf{q}}_p \end{bmatrix} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{P}(\mathbf{q})\mathbf{f}_C \\ \mathbf{0} \end{bmatrix} + \mathbf{J}^T \mathbf{f}_e, \tag{3}$$

where $\mathbf{q} = [\mathbf{q}_m^T, \mathbf{q}_p^T]^T$ is the state vector with $\mathbf{q}_m$ the generalized coordinates of the muscle-driven joints and $\mathbf{q}_p$ those of the passive joints, $\mathbf{M}$ is the mass matrix, and $\mathbf{c}$ accounts for the forces from connecting tissues and muscle parallel elements (f_P) as well as gravity, Coriolis forces, and centrifugal forces. The Jacobian matrix $\mathbf{J}(\mathbf{q})$ transforms the applied external force $\mathbf{f}_e$ into joint torques. The moment arm matrix $\mathbf{P}$ transforms the muscle force $\mathbf{f}_C$ to the joint space torque and is defined as $\mathbf{P}^T = d\mathbf{l}/d\mathbf{q}$, where $\mathbf{l}$ is the vector of the lengths of each muscle. The algorithm to compute $\mathbf{P}$ is detailed in [Delp and Loan 1995].

To simulate the skeletal dynamics, we use the implicit Euler time-integration method with linearized equations of motion. The equations of motion (3) can be written as

$$\ddot{\mathbf{q}} = \phi(\mathbf{q}, \dot{\mathbf{q}}, \tau). \tag{4}$$

Rather than computing ϕ from (3), we perform forward dynamics. Then the implicit Euler method computes $\dot{\mathbf{q}}(t+h)$ by solving

$$\dot{\mathbf{q}}(t+h) - \dot{\mathbf{q}}(t) = h\phi(\mathbf{q}(t+h), \dot{\mathbf{q}}(t+h), \tau), \tag{5}$$

which requires the use of iterative root-finding methods. We simplify the problem by using the first-order approximation

$$\Delta\dot{\mathbf{q}} = h\left[\phi(\mathbf{q}(t), \dot{\mathbf{q}}(t), \tau) + \frac{\partial \phi}{\partial \mathbf{q}}\Delta\mathbf{q} + \frac{\partial \phi}{\partial \dot{\mathbf{q}}}\Delta\dot{\mathbf{q}}\right] \tag{6}$$

$$= h\left[\phi(\mathbf{q}(t), \dot{\mathbf{q}}(t), \tau) + \frac{\partial \phi}{\partial \mathbf{q}}h(\dot{\mathbf{q}}(t) + \Delta\dot{\mathbf{q}}) + \frac{\partial \phi}{\partial \dot{\mathbf{q}}}\Delta\dot{\mathbf{q}}\right], \tag{7}$$

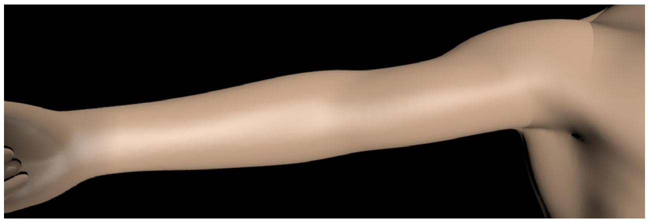
where $\Delta\dot{\mathbf{q}} = \dot{\mathbf{q}}(t+h) - \dot{\mathbf{q}}(t)$. We use a simulation time step $h \approx 7$ms.

3.3 Soft Tissue Model

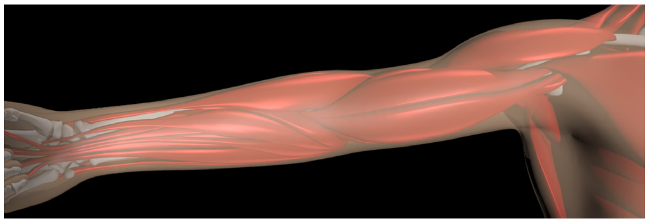
In our human body model, the muscle activation parameters are computed using a PLS idealization of the musculature. Driven by these muscle actions, a companion volumetric, finite element simulation of the musculoskeletal structure introduces the visual richness of more detailed, 3D musculature models (Fig. 7).

3.3.1 Skin surface model creation

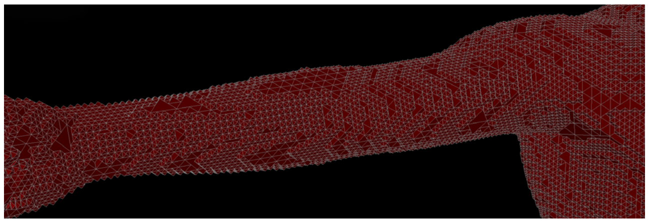
The first step in the construction of our physics-based soft tissue model is the creation of a high quality skin surface geometry. The initial high-resolution skin surface mesh that serves as our prototype is created by subdividing the original mesh of the Ultimate Human model. Although rich in geometric detail, it does not clearly define the volumetric boundary of the body, since it is not a closed, intersection-free surface due to openings at the eye sockets, spurious intersections near the ears, etc. Additionally, the mesh is rather poorly conditioned for the purposes of a volumetric physics-based simulation, with elements exhibiting aspect ratios as high as 60:1, while the ratio of the longest to the shortest edge in the mesh is in excess of 1000:1. This would hinder the time integration of the resulting physics-based model and collision processing. We ameliorate these shortcomings as follows:



(a) Skin Visualization Geometry



(b) Anatomical Bone and Soft Tissue Geometry



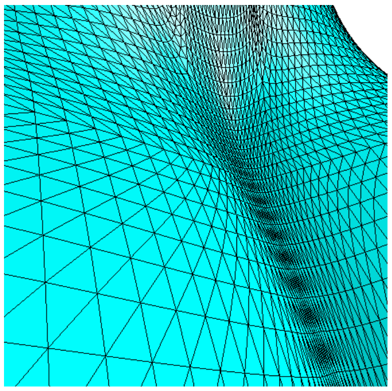
(c) Simulation Mesh

Figure 7: *Our volumetric human body model incorporates (a) detailed aspects of skin geometry and (b) active muscle tissues, passive soft tissues, and skeletal substructure. The skin surface is discretized into a 300K-triangle mesh. Resolving this surface detail with a fully tetrahedralized mesh through the bounded volume would make any form of finite element simulation impractical. To overcome this difficulty, we decouple the visualization geometry from the simulation geometry by creating an embedded model. To this end, an adaptive, BCC tetrahedralized mesh is superimposed on the soft tissue volume (c). This mesh embeds the high-resolution surface representation by means of barycentric interpolation of the surface nodes from the nodes of the tetrahedral simulation mesh.*

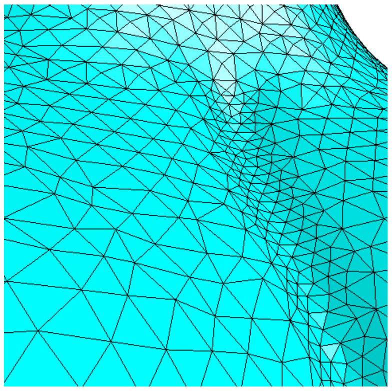
- Holes in the skin mesh (e.g. eyes) were procedurally closed.

- The closed skin mesh is rasterized into a level set implicit surface [Osher and Fedkiw 2002]. A grid size of 1.5mm is used for most of the body, while certain areas with thin features, such as the ears, are additionally rasterized into local level sets with a grid size of 0.5mm. This implicit surface is slightly dilated (by 1mm) and smoothed to eliminate excessively thin flesh features (some of which were present in the ears) and artifacts caused by spurious self-intersection of the original surface. These subtle corrections are hardly noticeable, even upon close inspection.

- The meshing algorithm of [Molino et al. 2003] is utilized to convert the level set implicit model of the flesh geometry into a well-conditioned tetrahedralized volumetric mesh. We create a moderately adaptive tetrahedral mesh with element diameters ranging from 1mm to 10mm, resolving the surface geometry at an average resolution of 3–5mm. The resulting mesh has an overwhelming 6.2 million tetrahedral elements; however, instead of directly using this model for simulation, we keep only the *triangulated surface* of this tetrahedralized volume (i.e., its topological boundary), which we use in the context of an "embedded simulation" framework described below. The interior structure of this tetrahedral mesh is discarded. Constructing the surface mesh as the topological boundary of a well-conditioned tetrahedralized volume imposes even more stringent mesh quality standards; e.g., we penalize bad dihedral angles and narrow parts in the volume enclosed by the computed surface, in addition to enforcing good conditioning along the surface itself. Although this construction may be more conservative than it strictly needs to be, this is a tolerable, one-time modeling cost. Alternative tetrahedral meshing schemes (e.g., [Labelle and Shewchuk 2007]) or mesh optimization schemes (e.g., [Hoppe et al. 1993]) can be used for this task, as long as they are tuned to produce a high-quality result.



(a)



(b)

Figure 8: *Detail from the chest of the anatomical model. The Ultimate Human model skin surface, riddled with sliver elements when subdivided for smoothness (a), compared to our well-conditioned surface mesh (b).*

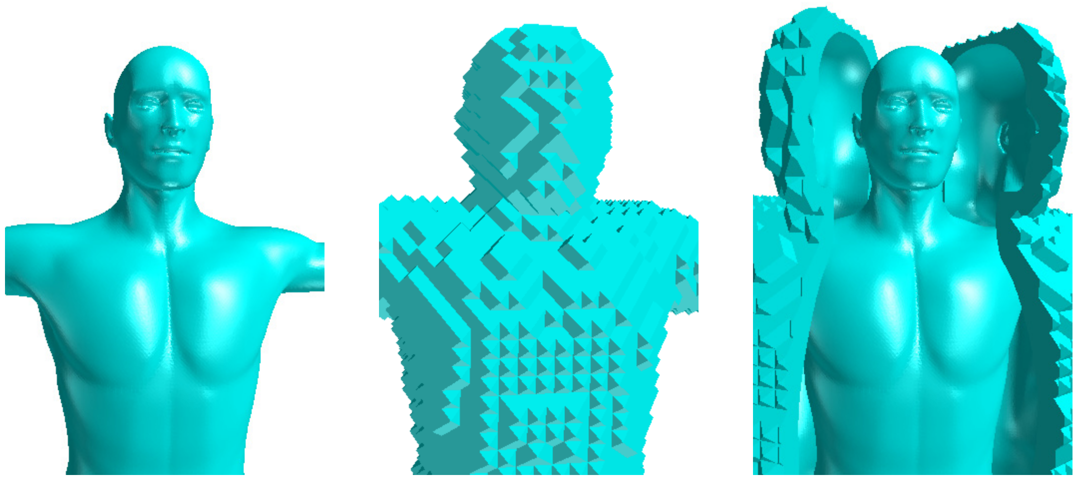

(a) (b) (c)

Figure 9: *(a) Triangulated skin geometry. (b) Background BCC-tetrahedralized material volume. (c) After cutting along the skin surface, the material volume has been separated into an embedded volumetric body model, and an outer mold of unused material (sliced and peeled open for illustration purposes).*

The resulting triangulated skin geometry has 302K triangles, with an average diameter of 4–5mm and a maximum aspect ratio of 3.8:1 (Fig. 8). This well-conditioned mesh provides an excellent starting point for physics-based flesh modeling and greatly helps collision handling.

3.3.2 Generation of the embedded simulation mesh

Having created a high-quality surface representation, our next goal is to generate a *volumetric* simulation mesh on which the governing equations of the soft tissue will be defined. One possibility would be simply to create a tetrahedral mesh directly from the implicit surface representation of the flesh volume that was previously created. In fact, we have already described the creation of such a tetrahedral mesh with the purpose of using its boundary triangle mesh as the skin surface representation. However, there are problems with using such a mesh for simulation. The mesh resolves the skin surface at a resolution of 3–5mm, with an interior element size of 10mm. At this resolution, the mesh comprises 6.2 million tetrahedra, impractically many for a full nonlinear finite element flesh simulation. One possible remedy would be a more aggressive adaptivity; increasing the maximum element size to 20–30mm could lead to 8 to 27 times fewer *interior* elements. Unfortunately, at the given level of surface resolution, as many as 1.0–1.5 million tetrahedra would be incident to the surface where high refinement would be necessary. In practice, we found that even with very aggressive adaptivity in the interior of the flesh, the minimum mesh size attainable without coarsening near the surface would be approximately 2 million tetrahedra. Finally, even if we tolerated a lower surface resolution for most of the skin surface, there are certain regions that need to be adequately resolved due to high-curvature features (e.g., fingers, face) or to facilitate collision handling (e.g., inner thighs, elbow, armpit).

Our goal is to reduce the size of our simulation model to a few hundred thousand tetrahedral elements. This represents a reduction of the number of simulation elements by roughly an order of magnitude from the aforementioned approaches, but our experience indicates that the coarser resolution can require two or three orders of magnitude less computation time. This is attributable to the decreased number of solver iterations—both for the outer Newton iteration for the solution of the nonlinear problem, as well as for the conjugate gradients solver of the linearized problem at each Newton step—as a result of using fewer, larger tetrahedral elements. Note that the computational cost of every conjugate gradients iteration scales faster than linearly in practice as the state variable storage requirements of million-element models begin to outgrow the size of processor cache, further impeding performance due to memory bandwidth limitations.

We address these issues with a hybrid simulation technique, adapting the framework of [Sifakis et al. 2007b] to our soft tissue simulation task. We use an embedded simulation scheme which decouples the geometric representation of the skin surface from the volumetric simulation mesh. Thus, we can benefit from the higher resolution of the triangulated skin surface mesh for rendering and collision handling, while simulating the elastic flesh deformation on a coarser adaptive tetrahedral mesh in which the detailed skin surface is embedded.

We start by generating a Body-Centered-Cubic (BCC) tetrahedral lattice (see [Molino et al. 2003] for the details), which completely covers the volume bounded by the human body (Fig. 9(a)), as shown in Fig. 9(b). We use a uniform size of 7mm for the tetrahedral elements at this step. Subsequently, we use the algorithm of [Sifakis et al. 2007a] to *cut* this background tetrahedral along the triangulated surface of the skin into two separate parts—the fragment interior to the skin surface, which corresponds to the human body volume, and the exterior part, which forms a negative “mold” enclosing the body, as shown in Fig. 9(c). We discard this exterior volume as it is irrelevant to our simulation. The interior volume comprises the soft tissue model that we wish to simulate. The cutting algorithm of [Sifakis et al. 2007a] provides the subset of the original tetrahedral mesh that intersects this volume (Fig. 10(b)) plus an embedded skin surface geometry in terms of a triangle mesh whose vertices are barycentrically embedded into the tetrahedra of the embedding volume.

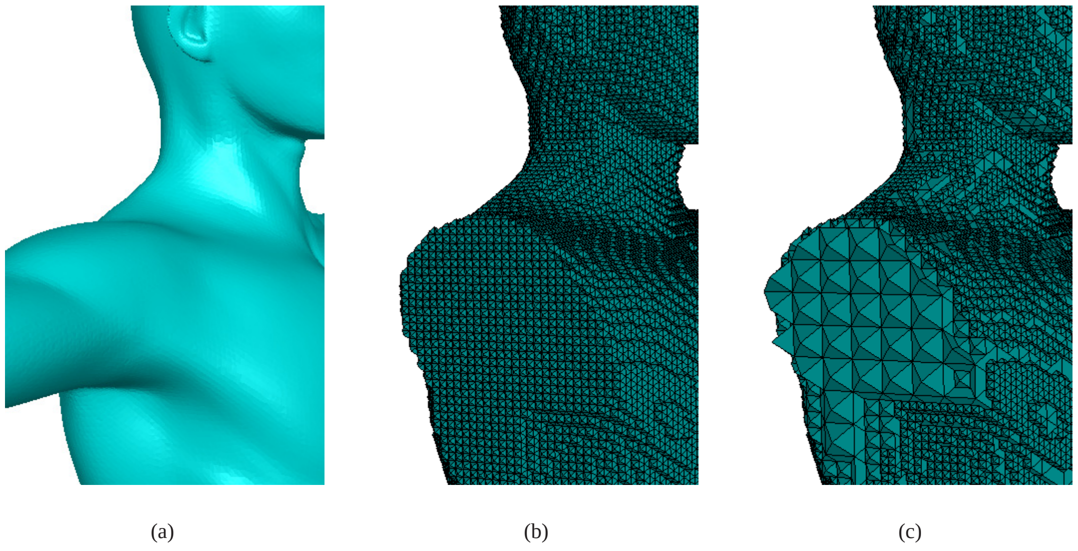


Figure 10: *(a) Visual reference. (b) Uniform embedding mesh at a resolution of 7mm. (c) The embedding mesh after 2 steps of adaptive coarsening, down to an element resolution of 28mm. T-junctions are visible at the boundaries between refinement levels.*

Our decision to create the embedding geometry using the cutting method of [Sifakis et al. 2007a] is influenced by the ability of this algorithm to create new degrees of freedom to better resolve parts of the embedded material exhibiting branching or narrow separation. For example, in the vicinity of the fingers, a single embedding tetrahedron from our background BCC mesh will often touch two neighboring fingers. A naive strategy that simply embeds every part of the surface into the tetrahedron in which it lies would effectively "join" these otherwise separate parts of flesh, whereas the use of the aforementioned cutting method automatically introduces new degrees of freedom to separate the parts and to better resolve the topology of the embedded material.

Since the embedding mesh thus created originates from a uniform resolution lattice, its total number of elements (3.8 million) is still prohibitively high. Leveraging the highly regular structure of the underlying BCC lattice, we proceed to coarsen this mesh adaptively by reversing the process of a *red refinement* as defined in [Molino et al. 2003]. The inverse of this process, *red coarsening*, collapses eight child tetrahedra into one, similar to each of the child tetrahedra with an edge ratio of 2:1. The criterion for coarsening is that all eight child tetrahedra must be present in the embedding mesh and that none of them has been duplicated by the cutting algorithm of [Sifakis et al. 2007a]. After recursively coarsening for a maximum of two levels (i.e., a tetrahedron size of 28mm), we obtain the final simulation mesh consisting of 354,000 tetrahedra (Fig. 10(c)). Due to the nature of our refinement process, T-junctions are present at the boundaries between different levels of refinement. These special points are simulated in a straightforward fashion using the framework of [Sifakis et al. 2007b].

The individual challenges presented by our biomechanical simulation can be addressed by a number of techniques documented in the literature. For example, adaptivity and T-junctions are accommodated in approaches such as [Grinspun et al. 2002; Wicke et al. 2007], large nonlinear deformation and element inversion is treated in [Irving et al. 2004] while the work of [Bridson et al. 2002] can be used for robust treatment of collisions. In our work, we opted for the hybrid simulation framework of Sifakis et al. [2007b], which provides a simple, unified infrastructure for accommodating embedding, adaptivity, and complex boundary conditions, in combination with established robust techniques for large, nonlinear deformation and collision processing.

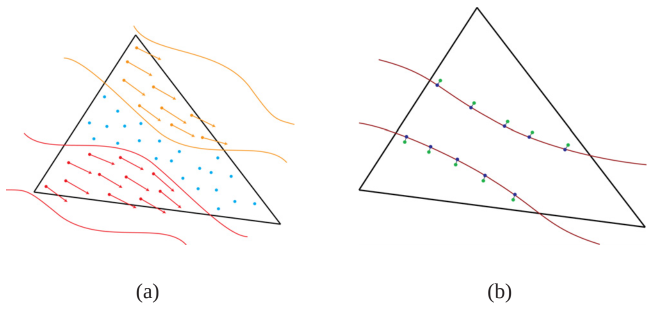


Figure 11: *(a) Randomly scattered point samples in a simulation element are classified as belonging to active muscles (red and orange points, with their associated fiber directions) or passive flesh (blue points). (b) A rigid bone (the area between the brown outlines) is sampled on its surface (blue points). These sample points are connected with zero length springs to barycentrically embedded locations (green points) in the simulation mesh.*

3.3.3 Modeling musculature and skeletal structure

The tetrahedral simulation mesh created in the previous step does not strictly conform to the geometry of muscles or bones. In analogy to the treatment of the high-resolution skin surface, these geometric features are *embedded* into the simulation mesh. As a first step, we use the geometry of the muscles to modulate the material properties assigned to each simulation element. A number of randomly generated sample points (we used between 10^3 and 10^4 points, depending on element size) are uniformly distributed in each tetrahedron of the simulation mesh. We check whether each of these sample points, indicated as colored dots in Fig. 11(a), is located inside any muscle volume, in which case the direction of the muscle fiber field at the given location is associated with the sample point. In the figure, these samples are depicted as red and orange vector fields corresponding to the two distinct muscles intersecting the simulation element. Points outside any muscle volume, displayed as blue dots in the figure, are regarded as locations of passive flesh or fatty tissue. Using these sample points, we compute a *muscle density* $d_m \in [0,1]$, denoting the fraction of the simulation element covered by muscle m, while d_p denotes the fraction of the simulation element covered by passive flesh outside any muscle. Consequently, these densities satisfy $d_\text{p} + \sum_m d_m = 1$. Finally, we average the fiber directions of the sample points inside muscle m and normalize the result to unit length to obtain a representative fiber direction $\mathbf{f}_m$ for this muscle with respect to the simulation element in question.

We describe the constitutive model of each simulation tetrahedron in terms of the strain energy density $\Psi(\mathbf{F})$, which is defined at each point as a function of the deformation gradient $\mathbf{F} = \partial\phi/\partial\mathbf{X}$. Here, ϕ is the *deformation function* that maps a point $\mathbf{X}$ in the undeformed configuration of the body to its deformed position $\mathbf{x} = \phi(\mathbf{X})$. The total strain energy E is obtained by integrating the energy density $\Psi(\mathbf{F})$ over the entire deformable body. Subsequently, this energy can be used to compute nodal forces by taking the negative gradient $\mathbf{f} = -\partial E/\partial\mathbf{x}$ of the strain energy with respect to the nodal position $\mathbf{x}$. Refer to [Bonet and Wood 1997] for a detailed discussion of hyperelastic constitutive models and methods for their numerical discretization, and [Teran et al. 2005b] for a specialized exposition in the context of musculoskeletal simulation.

Our constitutive model is defined as a weighted average of the constitutive models for passive flesh and active muscles, using the previously computed muscle densities d_m, as follows:

$$\Psi(\mathbf{F}) = d_\text{p}\Psi_\text{p}(\mathbf{F}) + \sum_m d_m \Psi_m(\mathbf{F}). \tag{8}$$

The passive flesh is modeled as an isotropic, quasi-incompressible Mooney-Rivlin material [Bonet and Wood 1997], leading to the following formula for its strain energy density Ψ_p:

$$\Psi_\text{p} = \mu_{10}(\text{tr}\hat{\mathbf{C}} - 3) + \frac{1}{2}\mu_{01}[(\text{tr}\hat{\mathbf{C}})^2 - \hat{\mathbf{C}} : \hat{\mathbf{C}} - 6] + \frac{1}{2}\kappa \log^2 J, \tag{9}$$

where $J = \det\mathbf{F}$ is the volume change ratio and $\hat{\mathbf{C}} = \hat{\mathbf{F}}^T\hat{\mathbf{F}}$ is the deviatoric Cauchy strain tensor with $\hat{\mathbf{F}} = J^{-1/3}\mathbf{F}$ the deviatoric component of the deformation gradient. We use the values $\mu_{01} = 0.06$MPa and $\mu_{10} = 0.02$MPa for the elasticity moduli, and the bulk modulus (a measure of incompressibility of the tissue) $\kappa = 10$MPa. The constitutive model for active muscles is the sum of the isotropic contribution Ψ_p and an anisotropic muscle term Ψ_m:

$$\Psi_m = \Psi_\text{p} + \Psi_\text{m}(\lambda_m), \tag{10}$$

where $\lambda_m = \|\hat{\mathbf{F}}\mathbf{f}_m\|$ is the along-fiber contraction ratio of muscle m in the simulation element, and the function Ψ_m is defined via its first derivative:

$$\frac{\partial\Psi_\text{m}(\lambda)}{\partial\lambda} = \frac{\sigma_\text{max}}{\lambda_\text{opt}} f_\text{tot}(\lambda), \tag{11}$$

where $\sigma_\text{max} = 0.3$MPa is the peak isometric stress of skeletal muscle, $\lambda_\text{opt} = 1.4$ is the optimal fiber contraction ratio for force generation, and f_tot is the normalized force-length function for the passive and active component. We define f_tot in accordance with a standard Hill-type model [Zajac 1989]. We refer the interested reader to [Blemker 2004; Teran et al. 2005b; Sifakis 2007] for a further discussion of the constitutive model used in our system, alternative models offering even higher biomechanical accuracy, and details on the numerical discretization and implementation of these models.

Next, we address the issue of integrating the rigid skeleton with our soft tissue simulation model. Our volumetric simulation mesh does not resolve the rigid bones; in fact, the simulation mesh *overlaps* with the skeleton, requiring special treatment of the interface between the hard and soft tissues. One possibility would be to constrain any node of the simulation mesh that lies *inside* or *near* a bone to a fixed position within the local coordinate frame of that bone. However, this approach leads to issues with (a) simulation nodes that are near more than one bone, (b) bones located very close to the skin surface leading to odd-looking patches of skin that move rigidly with the bones underneath, and (c) thin bones (e.g., ribs) located deeper inside the flesh where the simulation mesh is coarser, which may be inadequately constrained unless an unnaturally large constraint radius is used.

We circumvent these problems by using *soft constraints* and applying them to *embedded* locations rather than true nodes of the simulation mesh, as follows: A set of points is uniformly sampled on the surface of each bone. These samples, displayed as blue dots in Fig. 11(b), move with the bone. We then duplicate each of these samples with the locations that they have in the undeformed configuration of the soft tissue. The duplicated samples, illustrated with a slight displacement as green dots in Fig. 11(b), are barycentrically embedded into the simulation element with which they overlap. Finally, the samples attached to a bone are connected with their duplicate embedded counterparts using zero rest-length elastic springs, which apply traction forces on the surrounding soft tissue as the bone moves, thus inducing an appropriate soft tissue movement and deformation. This embedded treatment of skeletal attachments allows us to decouple the resolution of the simulation mesh from the resolution of the skeletal geometry and define the attachment regions as arbitrarily point-sampled surfaces.

3.3.4 Numerical simulation and time-integration

Our musculoskeletal simulation model contains features such as T-junctions, hybrid descriptions using embedded collision geometries, and embedded point sets for skeletal attachments, as well as soft constraints implemented as zero-length elastic springs. We use the hybrid simulation framework of [Sifakis et al. 2007b], which accommodates such simulation elements in the context of either explicit or implicit time-integration schemes. In particular, T-junctions and points embedded in the simulation mesh are naturally handled without compromising the symmetry or definiteness of the linear systems arising from the finite element discretization of the simulation mesh. Additionally, the elastic springs used to enforce soft constraints are handled fully implicitly in the context of Newmark-type time-integration schemes, alleviating timestep restrictions that could arise from stiff constraint springs. For the examples illustrated later in this paper, we used the quasi-static time-integration scheme of [Teran et al. 2005c], which provides the robust handling of extreme deformation and element inversion, both of which are frequent occurrences in our application. Depending mostly on the rate of change of muscle activation and the velocity of the skeleton, we obtained simulation times of 1–4 minutes per frame using a single core of a 3.0Ghz Intel Xeon CPU workstation.

4 Control

To animate our model, we take a *computed muscle force control* approach; i.e., we first employ the equations of motion (3) of the musculoskeletal system to compute the muscle forces necessary to produce some desired motion—say, a sequence of target key poses set by an animator or from motion capture data specifying the de-

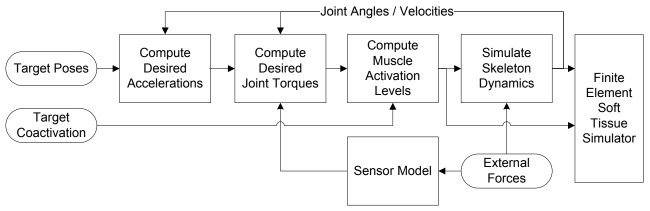

Joint Angles / Velocities
Target Poses
Compute Desired Accelerations
Compute Desired Joint Torques
Compute Muscle Activation Levels
Simulate Skeleton Dynamics
Finite Element Soft Tissue Simulator
Target Coactivation
Sensor Model
External Forces

Figure 12: *Overview of the motion controller.*

sired joint angles over time—subject to external forces, and then we apply the computed muscle forces to produce the final forward dynamics simulation. To this end, we develop a physics-based animation controller that computes accelerations to drive the musculoskeletal system toward the sequence of target poses, and then computes the required activation signal for each muscle through inverse dynamics.

4.1 Overview

Fig. 12 shows a schematic of our motion controller. The inputs to the controller are the target pose and muscle coactivation of agonist and antagonist muscles of the body. Although the pose/coactivation inputs can be dense time series, such as motion capture data, we will use sparse key frames, at an approximate rate of 1 frame/sec, as inputs in our experiments. Given the inputs and the current state, the motion controller determines the desired acceleration of the joints, and then computes the required muscle activation levels in order to achieve the desired accelerations and target coactivation. Then the articulated-body dynamics of the skeletal system are simulated using the implicit Euler method, and the joint angles and muscle activation levels are made available to the soft tissue simulator. Finally, given the position of the bones and the activation level of each muscle, the soft tissue is displaced and deformed by the elastic spring traction forces coupling the soft tissue to the bones, as well as the muscle forces along the fiber directions, as was described in the previous section.

4.2 Computing the Desired Accelerations

Given target positions and orientations of the head and chest (the T1 bone) as well as target joint angles of the arms and legs, the motion controller first performs inverse kinematics to determine the desired angles of the joints. Specifically, we iteratively update the joint angles in a gradient-descent manner such that the differences between the current and target poses of the head and T1 bone are minimized.

At each animation time step, the controller determines the desired acceleration to reach the target pose, using feedback information about the joint angle and velocity. In our experiments, we compute the desired accelerations $\ddot{\mathbf{q}}^*$ as follows:

$$\ddot{\mathbf{q}}^* = k_p(\mathbf{q}^* - \mathbf{q}) + k_v(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}), \tag{12}$$

where $\mathbf{q}^*$ and $\dot{\mathbf{q}}^*$ are the desired joint angles and angular velocities, respectively, and k_p and k_v determine the characteristics of the acceleration given the differences between the desired and actual values.

4.3 Generalized Force Computation

Prior to computing the muscle activation levels, it is convenient to compute the equivalent generalized forces; i.e., to solve the inverse dynamics problem in the joint space. The efficient, recursive Newton-Euler inverse dynamics method is predominantly used for open-loop systems, but we cannot use this method because some joints (the sternum and costal cartilages) cannot be controlled by muscles. In other words, even if we compute generalized forces for these joints, no muscle can generate the generalized force. Actually, it would be unnatural to specify a desired motion for these *passive joints* as they are articulated passively by connected bones, while we compute the muscle forces required to achieve the specified accelerations of the *muscle-driven joints*.

The *hybrid dynamics algorithm* [Featherstone 1987] is an efficient algorithm that serves this purpose. Using this algorithm, we provide desired accelerations for *acceleration-specified joints* and input torques for *torque-specified joints* and, in linear time complexity in the number of joints, we can compute the required joint torques for the former and the resulting accelerations of the latter.

To perform inverse dynamics, we set the passive joints to be torque-specified joints with zero generalized forces, and set the muscle-driven joints to be acceleration-specified joints with the desired accelerations. Then, the hybrid dynamics algorithm computes the required generalized forces for the muscle-driven joints and the resulting acceleration of the passive joints. Mathematically, the algorithm solves for τ^* and $\ddot{\mathbf{q}}_p$ in

$$\begin{bmatrix} \tau^* \\ \mathbf{0} \end{bmatrix} = \mathbf{M}(\mathbf{q}) \begin{bmatrix} \ddot{\mathbf{q}}_m^* \\ \ddot{\mathbf{q}}_p \end{bmatrix} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{J}^T \mathbf{f}_e, \tag{13}$$

where $\ddot{\mathbf{q}}_m^*$ is the desired accelerations of the muscle-driven joints.

When applying the implicit time-integration scheme with rather large time steps, evolving the system using the computed torque from (13) creates a large error and the system can fail to achieve the desired motion. This is because the implicit time-integration method effectively uses (5) as the equations of motion, in which the system state is described with respect to the next time step. The solution is to perform inverse dynamics at the *next* time step, according to the following procedure:

```
1: Set the muscle-driven joints as acceleration-specified joints,
   while the passive joints remain torque-specified joints.
2: Set the desired accelerations for the muscle-driven joints.
3: Run the hybrid dynamics algorithm to compute the resulting
   acceleration of the passive joints.
4: Advance to the next time step with the forward Euler method
   using the accelerations computed above.
5: Run the hybrid dynamics algorithm to compute the required
   torques for the muscle-driven joints.
6: Return to the state of the current time step.
7: Reset the muscle-driven joints to torque-specified joints.
```

We can employ this scheme with reasonably large time-steps even if we use equation (7), which is the linearization of (5).

4.4 Computation of the Muscle Activation Levels

Next, we will determine the muscle activation levels required to generate the generalized forces. One of the most distinguishing features of muscles is that their stiffness varies according to the activation level. In addition to controlling pose, humans readily control the stiffness of their bodies by exploiting the redundancy of their muscle actuators. Stiffness is an important aspect of stylistic motions, such as dance.

We introduce a new method to compute the muscle activation levels. Our method achieves the desired stiffness by explicitly computing agonist and antagonist muscle activations. The agonist muscle activation level $\mathbf{a}_g$ is determined by solving the following optimization problem:

$$\mathbf{a}_g = \operatorname*{argmin}_{\mathbf{a}} \frac{1}{2}\sum_i (w_i a_i)^2 \quad \text{such that} \quad \mathbf{P}\mathbf{f}_C = \tau^*,\ \mathbf{a} \geq 0, \tag{14}$$

where w_i is the muscle weight.[2] We did not enforce an upper bound inequality constraint, i.e., $\mathbf{a} \leq 1$, because muscles should not reach their maximum state in normal situations. We define the antagonist muscle activation as generating an opposing force; i.e., we perform the optimization only to change the sign of the generalized force:

$$\mathbf{a}_n = \operatorname*{argmin}_{\mathbf{a}} \frac{1}{2}\sum_i (w_i a_i)^2 \quad \text{such that} \quad \mathbf{P}\mathbf{f}_C = -\tau^*,\ \mathbf{a} \geq 0. \tag{15}$$

Naturally, the total muscle activation level $\mathbf{a}_g + \mathbf{a}_n$ generates zero net generalized force, but it increases the stiffness of the system. Using a non-negative coactivation parameter γ, we determine the activation level as

$$\mathbf{a} = (1+\gamma)\mathbf{a}_g + \gamma\mathbf{a}_n. \tag{16}$$

For computational efficiency, we divide the muscles into four groups and perform separate optimizations for the muscles in each of the two arms, the head-neck complex, and the torso. First, we optimize the activation levels of the arm and head-neck muscles. Naturally, muscles that cross the torso to other parts of the body apply torques on joints in the torso. Therefore, we determine the activation level of the torso muscles so that the net torque is the same as the desired torque. In this way, the coactivation parameter can also be specified on a per body part basis.

Lee and Terzopoulos [2006] also solved two optimization problems in order to control both pose and stiffness, but their optimization problems are different—one computes agonist muscle activation similarly to ours, while the other finds a vector in the null-space of the moment arm matrix. In contrast, our method solves the same optimization problem twice, but with different equality constraints. Our new method promises to be better for implementing a learning machine (say, neural networks) that can determine muscle activation levels, potentially enabling a single trained machine to determine both agonist and antagonist muscle activation levels. Additionally, in our method the stiffness is determined by the desired acceleration as well as the state of the system, while in [Lee and Terzopoulos 2006] the maximum stiffness signal is determined solely from the pose.

4.5 Sensor Modeling

Since we impose exact control, the resulting motion follows the target motion very well. However, this is not always desirable in the context of computer animation. Since this method computes the required control input to create the desired motion under any given external force, the resulting body motion can easily lack its natural response to rapidly varying external forces, making the movement appear too stiff. For example, when a ball unexpectedly impacts the character, the body motion would not be affected, since our method computes the control inputs that nullify the external force from the ball.[3]

Sensor modeling can avoid this problem by discriminating between the external force input to the controller and the actual applied external force. The response time delay of biological control systems yields a natural response to the external forces. We use the following simple delay model which produces plausible reactive motion:

$$\frac{d}{dt}\tilde{\mathbf{f}}_e = \sigma(\mathbf{f}_e - \tilde{\mathbf{f}}_e), \tag{17}$$

where $\tilde{\mathbf{f}}_e$ is the *sensed* external force that is used to compute the muscle force and σ is the time lag coefficient. To apply the sensor model, we replace $\mathbf{f}_e$ in (13) with $\tilde{\mathbf{f}}_e$ computed above. While simplistic, our sensor model does enable intuitive control of the character's response.

5 Experiments and Results

Our experiments with our biomechanical upper-body model range from simulating dumbbell curls to creating respiratory movement. We have produced several simulations demonstrating that our muscle control algorithms can readily actuate the elaborate musculoskeletal system in a controlled manner in order to track various input key-pose sequences, such as arm flexing motions with dumbbell loads (Fig. 1). We also demonstrate an autonomous breathing animation in which plausible respiratory movement is produced by the intercostal muscles. Finally, we created a "pectoral dance" animation in which the activation signals to the pectoralis muscles are modulated through time to make them bulge rhythmically.

Fig. 13 shows a sample still frame from one of our animation experiments. As the figure shows, the soft tissue exhibits natural deformations due to the activation of the muscles and the motion of the skeleton. The embedded volumetric muscles also demonstrate credible, volume-preserving bulging, which suggests that our embedded model approach is a promising technique for accurately estimating muscle deformation.[4]

Fig. 14 shows a close-up of the shoulder region with different coactivations of agonist/antagonist muscle pairs. Naturally, high coactivation produces more muscle bulging, as is clear from the deformations of the skin surface.

A "dumbbell curl" animation illustrated in Fig. 15 demonstrates sensor modeling in the controller. When the mass of the dumbbells suddenly increases, the biomechanical character exhibits transient pose control failure before recovering the target pose. Without sensor modeling, the target pose would have been perfectly maintained in an unnatural manner, despite the unexpected change in the external load.

Figure 16 shows snapshots of an autonomous breathing animation in which plausible respiratory movement is produced by the intercostal muscles. For the purposes of this demonstration, we have specified a simple periodic function $\rho = \rho_{\max}(1-\cos\theta)/2$, where θ is the phase angle, and set $q_j = c_j\rho$ as target joint angles for the ribs. Here, c_j linearly increases until the fifth rib and stays constant for the remaining ribs.

5.1 Sensitivity Analysis

We have measured the sensitivity of our method to various muscle modeling parameters. Fig. 17(a)–(b) show the effect of the PCSA

[2]The muscle weights serve to regularize the muscle activation levels. When solving (14), if the activation level for some muscle i exceeds 1, we increase w_i in order to reduce the activation level, which will commensurately increase the activation levels of the associated synergistic muscles.

[3]The computed muscle force approach can also lead to implausible results when excessively large muscle forces are computed in order for the system to achieve unnatural target poses as a result of external forces. We do not consider this possibility, assuming instead that the animator provides realistically achievable target poses.

[4]Since we use static optimization to compute muscle activation levels, their temporal smoothness is not guaranteed. Therefore, although in most cases the optimal muscle activation levels change smoothly to produce smooth target motions, we observe occasional muscle twitches in our experiment. Had we used a full dynamic simulation of the soft tissue rather than quasi-static simulation, these discontinuities would have been smoothed out.

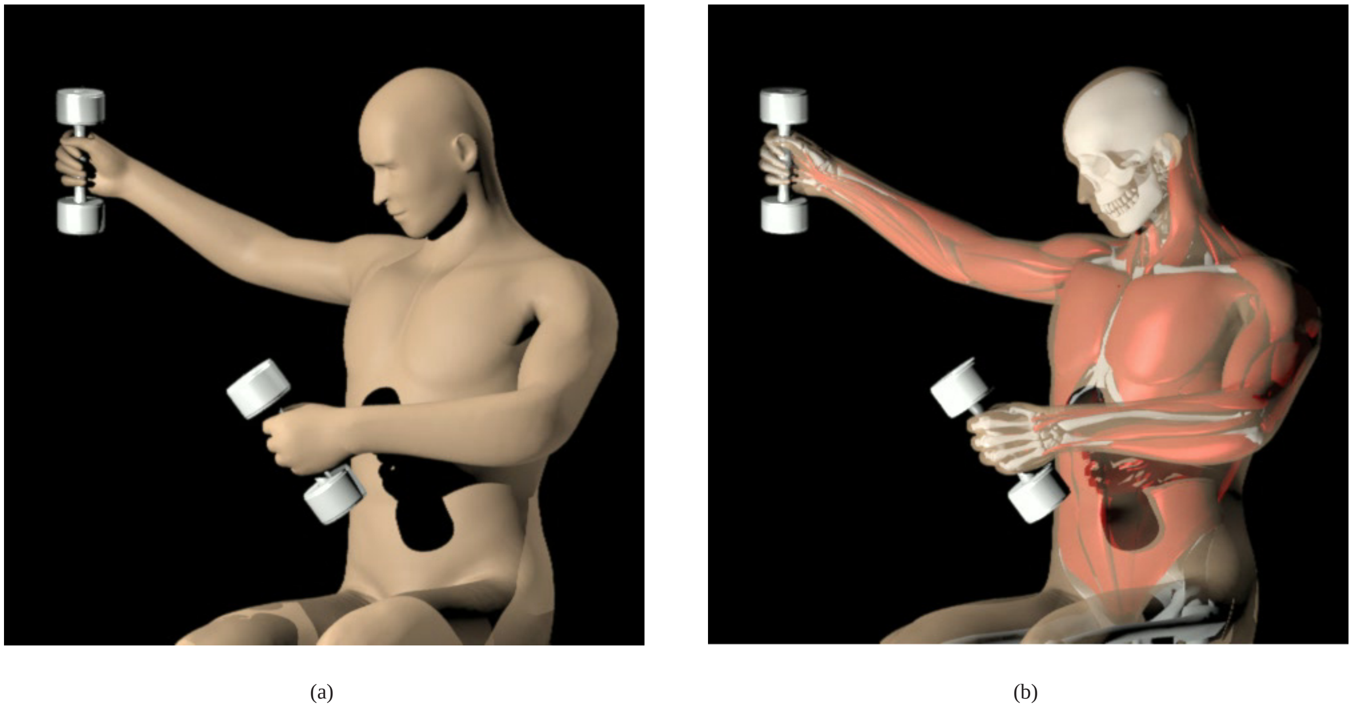


Figure 13: *The soft tissue simulator produces realistic deformations of the visualization geometry (a) and embedded volumetric muscles (b).*

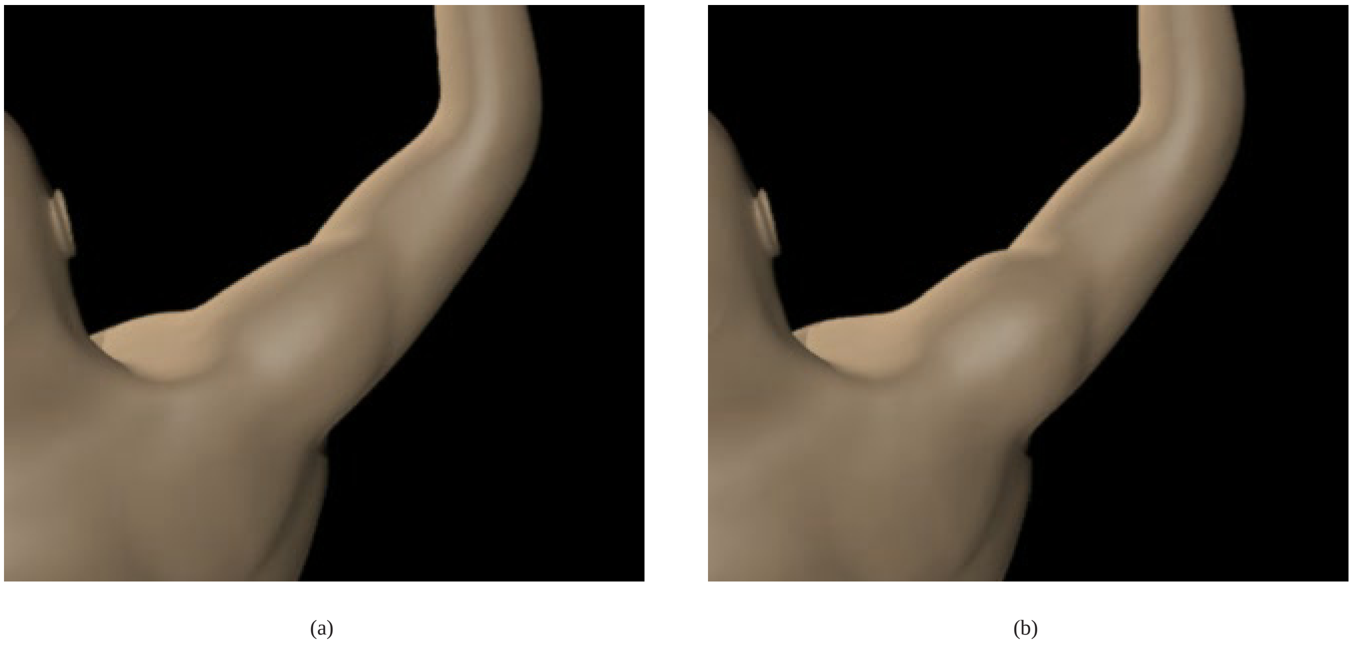


Figure 14: *Compared to zero muscle coactivation (a), higher coactivation (b) results in greater muscle bulging and stiffness in the shoulder.*

of muscles on the computation of muscle activation levels. In each test, we randomly perturbed the PCSAs of every muscle by up to $\pm5\%$ and measured the change of activation levels for 6 different poses. We executed a total of 10 tests and Fig. 17(a) shows a histogram of the mean change of activation levels of each muscle. We excluded non-activated muscles from the histograms. The figure indicates that more than 90% of the muscles show an average activation level change of less than 0.03. Fig. 17(b) shows the histogram of maximum change of activation levels of each muscle throughout the test.

We performed the same experiment with respect to the via points of muscles. In this case, we randomly perturbed the position of every via point by up to 3% of their rest length while the origin and insertion points remain fixed. Fig. 17(c) shows the average change of muscle activation levels due to the perturbation of via points and

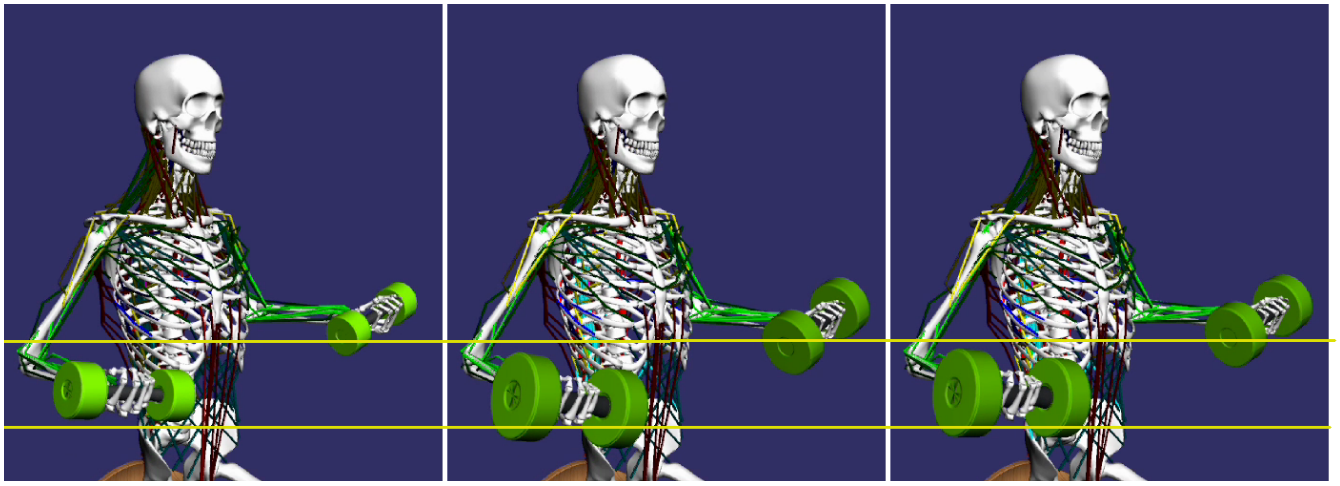


Figure 15: *When the mass of the dumbbells increases suddenly, the arms show a natural failure to maintain the pose and are lowered for an instant (center). Soon, they return to the desired, original pose (right). The horizontal lines highlight the displacement.*

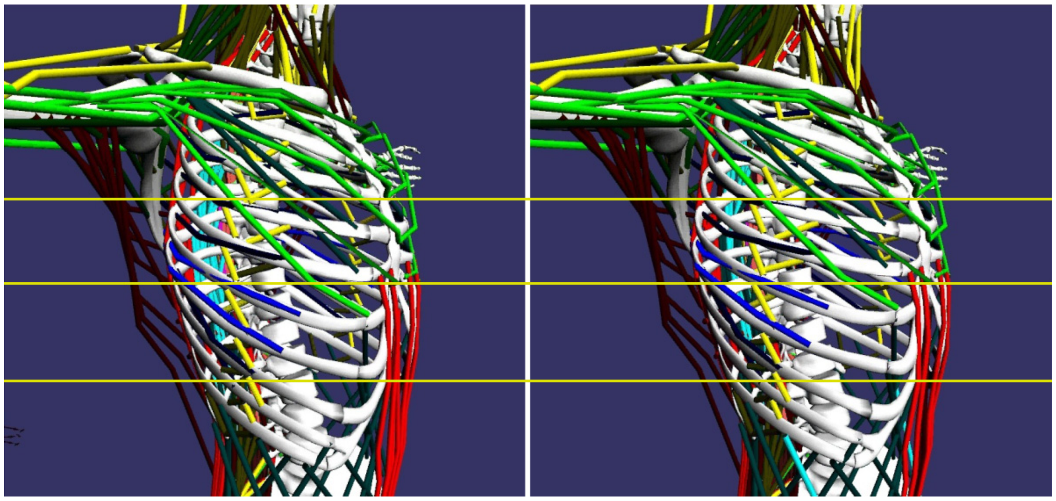


Figure 16: *Normal breathing. The ribs in inspiration (right) are elevated relative to those in expiration (left). The horizontal lines highlight the displacement.*

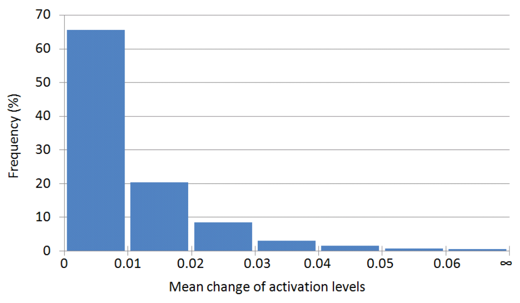
Fig. 17(d) shows the histogram of maximum change of activation level of each muscle throughout the test.

Fig. 17(b) and (d) reveal that the resulting muscle activation level can vary considerably due to changes in the muscle parameters. Actually, it has been known in the biomechanics community that the estimated muscle force from optimization is highly sensitive to modeling parameters such as the PCSA and the moment arm as well as with respect to the optimization method employed [Nussbaum et al. 1995; Raikova and Prilutsky 2001]. This stresses the importance of employing more accurate modeling parameters and better criteria for computing muscle forces, which remains an active research area in biomechanics. It is worth noting, however, that the muscle force patterns are not as sensitive to the muscle modeling parameters as are the muscle force magnitudes [Raikova and Prilutsky 2001].

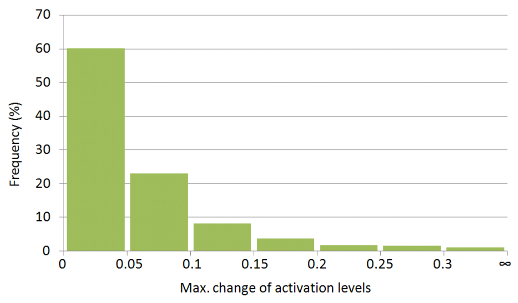
6 Conclusion and Future Work

We have introduced a highly-detailed, biomechanical model of the human upper body that comprises a dynamic, articulated skeleton, numerous Hill-type muscle actuators, and a realistic finite element simulation of soft tissues. We were able to achieve reasonably fast performance in soft tissue simulation by decoupling the visualization geometry from the simulation geometry, using an embedded model. To tackle the complexity of controlling the skeletal model in the presence of both active and passive joints, we developed an inverse dynamics algorithm for this hybrid system in conjunction with an implicit time-integration scheme. Additionally, we presented an improved method to compute muscle activation levels by explicitly computing agonist and antagonist muscle forces in order to control the stiffness of the body.

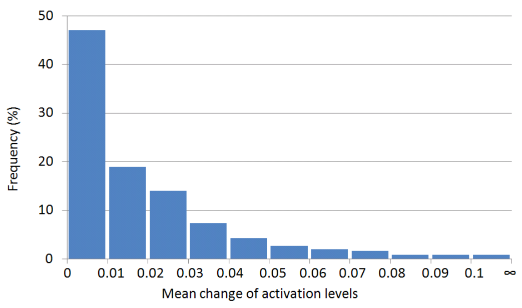
Despite the rich history of biomechanical modeling and control research, we have yet to see a truly integrated muscle-controlled human animation system (except perhaps in the case of facial animation), where volume-preserving 3D soft-tissue muscles actuate hard tissues (bones) and deform the surrounding skin. Our detailed musculoskeletal and soft tissue model represents an important stride toward this challenging long-term objective. Since our soft tissue model approximates the actual deformation of each muscle, rather than approximating muscle forces using simple line segment mod-



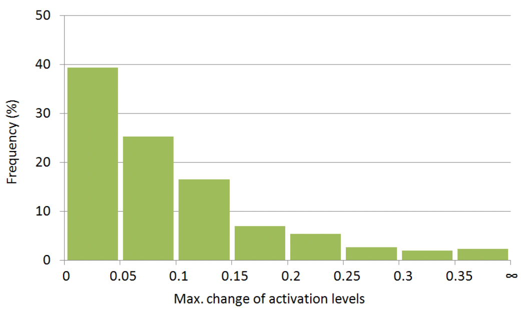
(a)



(b)



(c)



(d)

Figure 17: *The sensitivity of muscle activation levels to muscle modeling parameters. Histogram (a) shows the mean and (b) the maximum change of activation levels when the PCSAs of the muscles are randomly perturbed by up to* $\pm 5\%$. *Histogram (c) shows the mean and (d) the maximum change of activation levels when the via points of muscles are randomly perturbed by up to 3% of their rest lengths (origins and insertion points remain unchanged). Non-activated muscles are excluded from the histograms.*

els, we can directly compute the moment arm of each muscle from the soft tissue simulator. Moreover, by avoiding the urge to lump all the inertial properties of the surrounding tissues into the bones, the soft tissues can retain their inertial properties, enabling a more accurate dynamics simulation.

The embedding technique that we employed for soft tissue modeling enables the robust and efficient simulation of soft tissue deformation within the finite element framework. An important benefit of this embedding approach is the avoidance of small or ill-conditioned elements that might be necessary to resolve intricate anatomical detail (e.g., tendons and connective tissue) in the simulation mesh. We do, however, incur the compromise that such anatomical features are only represented at the resolution of the simulation mesh. Our treatment effectively computes a weighted average of the material properties of the tissues contained in every simulation element, without any subsequent attention to their relative placement. This is a source of inaccuracy that we necessarily tolerate at present, and that we expect will vanish with refinement. Another aspect missing from our current embedded model is the ability for muscles in contact to slide along one another and relative to the passive tissue surrounding them. This is in contrast with systems such as [Sueda et al. 2008] and [Teran et al. 2005a] that modeled muscles individually. In fact, the embedded simulation of individual muscles is an option (as demonstrated in [Teran et al. 2005a]) that we would like to investigate in the future, even at the cost of requiring explicit handling of collision and contact between individual muscles and tissue [Pai et al. 2005].

For some applications, it would be necessary to model not only additional individual muscles, but also the ligaments and disks (cartilage filled with a gelatinous substance) that deform to cushion the vertebrae. A more complete model would enable us to simulate cervical injuries such as whiplash and other injuries of the spinal column.

Regarding the modeling of joints, since we modeled the scapula as rigidly attached to the clavicle, the shoulder complex is restricted to a moderate range of motion, which also limits the richness of the resulting soft tissue deformation. In future work we will aim to improve the modeling of this region using the spline joint modeling technique proposed in [Lee and Terzopoulos 2008].

Our animations employed a quasi-static time-integration scheme for soft tissue simulation, which lacks the proper resolution of inertial motion effects. This was a choice motivated by the lower computational cost of a quasi-static simulation. Nevertheless, the use of an implicit backward Euler scheme or a semi-implicit Newmark integrator is readily supported within our framework. In our future work, we expect to leverage the performance offered by parallel and multi-core platforms to compensate for the higher overall cost of dynamic integration schemes.

In our work to date, we have treated each muscle force as an independent actuator when computing activation levels. However, muscles show highly correlated patterns of activity during normal human movements. We have applied principal components analysis (PCA) to a set of sample activation levels of the 422 torso muscles and satisfied ourselves that only 50 to 100 basis vectors can approximate the sample activations reasonably well. An opportunity for future work would be to develop a more efficient muscle controller for the entire body that utilizes a lower-dimensional control space through dimensionality reduction and, ultimately, a machine learning approach which generalizes the one already demonstrated for the neck [Lee and Terzopoulos 2006].

Another important future effort would be to extend our modeling framework to include a full biomechanical model of the hands and, indeed, to encompass the entire human body (see [Lee 2008]). As far as the skeletal and soft tissue modeling are concerned, a full-body model would not introduce substantial new difficulties. Indeed, our current model already includes finite element soft tissue simulation in the lower body, albeit no muscle actuation. However, it would surely be a significant control challenge to enable such a comprehensive biomechanical model to, say, locomote autonomously while maintaining dynamic balance in gravity.

Additional interesting venues of future work would be to develop associated algorithms for creating person-specific biomechanical models for use in, say, surgery simulation, as well as to adapt and extend our framework to the modeling of nonhuman primates and other lower animals.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. IIS-0326388 and by a UCLA Dissertation Year Fellowship to SHL.

References

ALBRECHT, I., HABER, J., AND SEIDEL, H.-P. 2003. Construction and animation of anatomically based human hand models. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA'03)*, 98–109.

ALBRO, J. V., SOHL, G. A., BOBROW, J. E., AND PARK, F. C. 2000. On the computation of optimal high-dives. In *Proc. IEEE International Conf. on Robotics and Automation*, 3958–3963.

BLEMKER, S. S. 2004. *3D Modeling of Complex Muscle Architecture and Geometry*. PhD thesis, Stanford University, Mechanical Engineering Department.

BONET, J., AND WOOD, R. 1997. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, Cambridge.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics 21*, 3, 594–603. Proc. ACM SIGGRAPH 2002.

CHADWICK, J. E., HAUMANN, D. R., AND PARENT, R. E. 1989. Layered construction of deformable animated characters. *Computer Graphics 23*, 3 (July), 243–252. Proc. ACM SIGGRAPH 89.

CHEN, D. T., AND ZELTZER, D. 1992. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics 26*, 2 (July), 89–98. Proc. ACM SIGGRAPH 92.

CROWNINSHIELD, R. D. 1978. Use of optimization techniques to predict muscle forces. *Journal of Biomechanical Engineering 100*, 88–92.

DELP, S. L., AND LOAN, J. P. 1995. A software system to develop and analyze models of musculoskeletal structures. *Computers in Biology and Medicine 25*, 21–34.

DILORENZO, P. C., ZORDAN, V. B., AND SANDERS, B. L. 2008. Laughing out loud: Control for modeling anatomically inspired laughter using audio. *ACM Transactions on Graphics 27*, 5 (Dec.), 125:1–8.

DONG, F., CLAPWORTHY, G. J., KROKOS, M. A., AND YAO, J. 2002. An anatomy-based approach to human muscle modeling and deformation. *IEEE Transactions on Visualization and Computer Graphics 8*, 2, 154–170.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 251–260.

FEATHERSTONE, R. 1987. *Robot Dynamics Algorithms*. Kluwer Academic Publishers.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *ACM Transactions on Graphics 21*, 3, 281–290. Proc. ACM SIGGRAPH 2002.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of ACM SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 71–78.

HOLZBAUR, K. R. S., MURRAY, W. M., AND DELP, S. L. 2005. A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Annals of Biomedical Engineering 33*, 6, 829–840.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. In *Proc. ACM SIGGRAPH 93*, 19–26.

HU, W., MARHEFKA, D. W., AND ORIN, D. E. 2005. Hybrid kinematic and dynamic simulation of running machines. *IEEE Transactions on Robotics 21*, 3, 490–497.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA'04)*, 131–140.

KÄHLER, K., HABER, J., YAMAUCHI, H., AND SEIDEL, H.-P. 2002. Head shop: Generating animated head models with anatomical structure. In *ACM SIGGRAPH/EG Symposium on Computer Animation*, 55–64.

KAPANDJI, I. A. 1974. *The Physiology of the Joints. Vol. 3: The Trunk and the Vertebral Column*. Churchill Livingstone, Edinburgh.

KOKKEVIS, E., AND METAXAS, D. 1998. Efficient dynamic constraints for animating articulated figures. *Multibody System Dynamics 2*, 89–114.

KOMURA, T., SHINAGAWA, Y., AND KUNII, T. L. 1997. A muscle-based feed-forward controller of the human body. *Computer Graphics Forum 16*, 3 (Aug.), 165–176.

KOMURA, T., SHINAGAWA, Y., AND KUNII, T. L. 2000. Creating and retargeting motion by the musculoskeletal human body model. *The Visual Computer 16*, 5, 254–270.

LABELLE, F., AND SHEWCHUK, J. R. 2007. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics 26*, 3, 57. Proc. ACM SIGGRAPH 07.

LEE, S.-H., AND TERZOPOULOS, D. 2006. Heads up! Biomechanical modeling and neuromuscular control of the neck. *ACM Transactions on Graphics 25*, 3 (July), 1188–1198. Proc. ACM SIGGRAPH 06.

LEE, S.-H., AND TERZOPOULOS, D. 2008. Spline joints for multibody dynamics. *ACM Transactions on Graphics 27*, 3 (Aug.), 22:1–8. Proc. ACM SIGGRAPH 2008.

LEE, Y., TERZOPOULOS, D., AND WATERS, K. 1995. Realistic modeling for facial animation. In *Proceedings of ACM SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 55–62.

LEE, S.-H. 2008. *Biomechanical Modeling and Control of the Human Body for Computer Animation*. PhD thesis, University of California, Los Angeles, Computer Science Department.

MOLINO, N., BRIDSON, R., TERAN, J., AND FEDKIW, R. 2003. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable*, 103–114.

MONHEIT, G., AND BADLER, N. I. 1991. A kinematic model of the human spine and torso. *IEEE Computer Graphics & Applications 11*, 2 (Mar.), 29–38.

NAKAMURA, Y., YAMANE, K., FUJITA, Y., AND SUZUKI, I. 2005. Somatosensory computation for man machine interface from motion-capture data and musculoskeletal human model. *IEEE Transactions on Robotics 21*, 1, 58–66.

NG-THOW-HING, V. 2001. *Anatomically-Based Models for Physical and Geometrical Reconstruction of Humans and Other Animals*. PhD thesis, University of Toronto, Department of Computer Science.

NUSSBAUM, M. A., CHAFFIN, D. B., AND RECHTIEN, C. J. 1995. Muscle lines-of-action affect predicted forces in optimization-based spine muscle modeling. *Journal of Biomechanics 28*, 4, 401–409.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag. New York, NY.

PAI, D. K., SUEDA, S., AND WEI, Q. 2005. Fast physically based musculoskeletal simulation. In *Proceedings of Sketches & Applications of ACM SIGGRAPH 2005*.

PANDY, M. G., ZAJAC, F. E., SIM, E., AND LEVINE, W. S. 1990. An optimal control model for maximum-height human jumping. *Journal of Biomechanics 23*, 12, 1185–1198.

RAIKOVA, R. T., AND PRILUTSKY, B. I. 2001. Sensitivity of predicted muscle forces to parameters of the optimization-based human leg model revealed by analytical and numerical analyses. *Journal of Biomechanics 34*, 10, 1243–1255.

SAPIO, V. D., WARREN, J., KHATIB, O., AND DELP, S. 2005. Simulating the task-level control of human motion: A methodology and framework for implementation. *The Visual Computer 21*, 5, 289–302.

SCHEEPERS, F., PARENT, R. E., CARLSON, W. E., AND MAY, S. F. 1997. Anatomy-based modeling of the human musculature. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 163–172.

SHAPIRO, A., PIGHIN, F., AND FALOUTSOS, P. 2003. Hybrid control for interactive character animation. In *PG'03: Proc. 11th Pacific Conference on Computer Graphics and Applications*, 455–461.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Transactions on Graphics 24*, 3 (Aug.), 417–425. Proceedings of ACM SIGGRAPH 2005.

SIFAKIS, E., DER, K., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proc. ACM SIGGRAPH/EG Symposium on Computer Animation*, 73–80.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. ACM SIGGRAPH/EG Symposium on Computer Animation*, 81–90.

SIFAKIS, E. 2007. *Algorithmic aspects of the simulation and control of computer generated human anatomy models*. PhD thesis, Stanford University, Computer Science Department.

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. *ACM Transactions on Graphics 27*, 3 (Aug.), 83:1–83:8.

TERAN, J., SIFAKIS, E., BLEMKER, S. S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 3, 317–328.

TERAN, J., SIFAKIS, E., BLEMKER, S. S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualizaton and Computer Graphics 11*, 3, 317–328.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proc. ACM SIGGRAPH/EG Symp. on Computer Animation.*, 181–190.

THELEN, D. G., ANDERSON, F. C., AND DELP, S. L. 2003. Generating dynamic simulations of movement using computed muscle control. In *Journal of Biomechanics*, vol. 36, 321–328.

TSANG, W., SINGH, K., AND FIUME, E. 2005. Helping hand: An anatomically accurate inverse dynamics solution for unconstrained hand motion. In *SCA'05: Proc. 2005 ACM SIGGRAPH/EG Symposium on Computer Animation*, 319–328.

VAN NIEROP, O. A., VAN DER HELM, A., OVERBEEKE, K. J., AND DJAJADININGRAT, T. J. 2008. A natural human hand model. *The Visual Computer 24*, 1 (Jan.), 31–44.

WATERS, K. 1987. A muscle model for animating three-dimensional facial expression. *Computer Graphics 22*, 4, 17–24. Proc. ACM SIGGRAPH 87.

WICKE, M., BOTSCH, M., AND GROSS, M. 2007. A Finite Element Method on Convex Polyhedra. *Computer Graphics Forum 26*, 3, 355–364.

WILHELMS, J., AND GELDER, A. V. 1997. Anatomically based modeling. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 173–180.

ZAJAC, F. 1989. Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control. *Critical Reviews in Biomed. Eng. 17*, 4, 359–411.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/EG Symposium on Computer Animation*, ACM, New York, NY, USA, 89–96.

ZORDAN, V. B., CELLY, B., CHIU, B., AND DILORENZO, P. C. 2004. Breathe easy: Model and control of simulated respiration for animation. In *2004 ACM SIGGRAPH/EG Symposium on Computer Animation*, 29–37.