

Creating and Simulating Skeletal Muscle from the Visible Human Data Set

J. Teran,¹ E. Sifakis,¹ S. Blemker,¹ V. Ng Thow Hing,² C. Lau¹ and R. Fedkiw³

¹ Stanford University, {jteran,sifakis,ssblemker,cindabin}@stanford.edu

² Honda Research Institute USA, vng@honda-ri.com

³ Stanford University, fedkiw@cs.stanford.edu

Abstract

Simulation of the musculoskeletal system has important applications in biomechanics, biomedical engineering, surgery simulation and computer graphics. The accuracy of the muscle, bone and tendon geometry as well as the accuracy of muscle and tendon dynamic deformation are of paramount importance in all these applications. We present a framework for extracting and simulating high resolution musculoskeletal geometry from the segmented visible human data set. We simulate 30 contact/collision coupled muscles in the upper limb and describe a computationally tractable implementation using an embedded mesh framework. Muscle geometry is embedded in a non-manifold, connectivity preserving simulation mesh molded out of a lower resolution BCC lattice containing identical, well-shaped elements leading to a relaxed time step restriction for stability and thus reduced computational cost. The muscles are endowed with a transversely isotropic, quasi-incompressible constitutive model that incorporates muscle fiber fields as well as passive and active components. The simulation takes advantage of a new robust finite element technique that handles both degenerate and inverted tetrahedra.

Index Terms

Finite volume methods, constructive solid geometry, physically based modeling.

I. INTRODUCTION

Simulation of anatomically realistic musculature and flesh is critical for many disciplines including biomechanics, biomedical engineering and computer graphics where it is becoming an increasingly important component of any virtual character. Animated characters must have skin that deforms in a visually realistic manner. However, the complexity of the interaction of muscles, tendons, fat and other soft tissues with the enveloping skin and our familiarity with

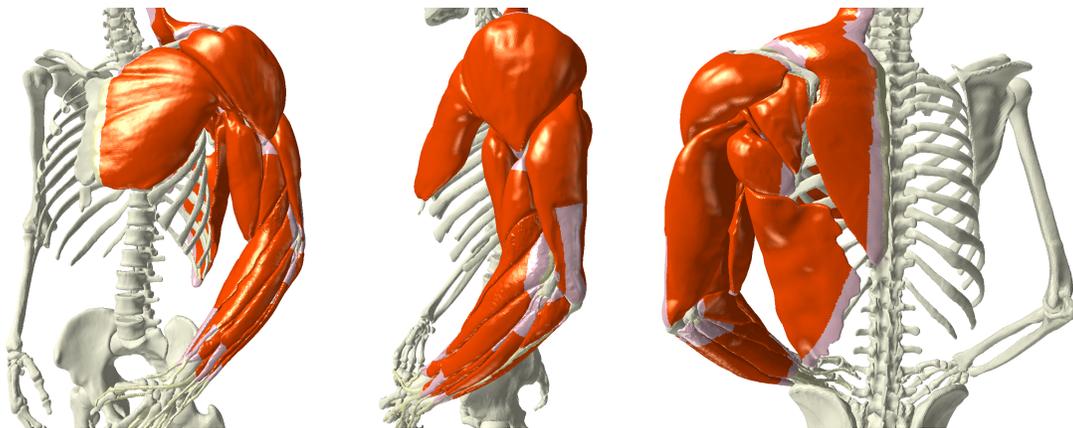


Fig. 1. Musculoskeletal model created from the visible human data set. Tendons are shown in pink. There are about 10 million tetrahedra in the approximately 30 muscles depicted.

this type of motion make these animations difficult if not impossible to create procedurally. In biomechanics and biomedical engineering, accurate descriptions of muscle geometry are needed to characterize muscle function. Knowledge of such quantities as muscle length, line of action and moment arm is essential for analyzing a muscle's ability to create forces, produce joint moments and actuate motion [32]. For example, many studies use knowledge of muscle lengths [1] and moment arms [2] to analyze muscle function for improving diagnosis and treatment of people with movement disabilities.

In order to create realistic flesh deformation for computer graphics characters, anatomy based modeling techniques of varying resolutions are typically applied. These models are generally composed of an underlying skeleton whose motion is prescribed kinematically (from motion capture or traditional animation) and a model that transmits motion of the underlying skeleton to tissue deformation. The model for this interaction can have varying levels of detail. For example, [25] maps joint configurations to skin deformers that procedurally warp the surface of the character. The work in [43] and [35] used anatomically based models of muscles, tendons and fatty tissue to deform an outer skin layer. The deformation of the muscle and tendon primitives was based on muscle characteristics such as incompressibility, but dynamic effects were not included. An obvious improvement to this approach is to include dynamic effects based on muscle mechanics as in [9], [22], [37], which incorporated theoretical muscle dynamic models (e.g. the relation between force, length and velocity in muscle) using the equations of solid mechanics to simulate muscle contraction. However in [9], [37], computational complexity restricted the application of their techniques to only a few muscles at a time. [22] simulated more tissues in the knee, but the dynamics were simulated quasi-statically ignoring the visually appealing effects of ballistic motion and inertia.

Musculoskeletal simulations in biomechanics typically fall into two categories: simulations of simple models for many muscles composing a large region of the body (e.g. the upper limb or lower extremity), or highly detailed muscle models that can only be simulated a few muscles at a time. Common muscle models compute accurate muscle moment arms and muscle/tendon lengths, but only resolve the average muscle line of action [13], [18]. However, it is difficult to represent the path of a muscle with complex geometry because it requires knowledge of how, as joints move, the muscle changes shape and interacts with underlying muscles, bones and other structures. These simplified models typically require the construction of elaborate "wrapping" surfaces and "via points" to resolve contact with other muscles and bones in compensation for simplifying muscles as piecewise linear bands. These simplified models of contact are difficult to construct robustly, as they require an a priori knowledge of the contact environment that is not always available. More detailed muscle models do not suffer from these difficulties, but are burdened by computational complexity. Typical examples are [44] and [19] which used modern nonlinear solid mechanics to recreate the stress and deformation, although only a few muscles with simplified geometry were considered and the simulations were carried out quasi-statically to avoid the stringent time step restrictions characteristic of explicit schemes.

We present a framework that can be used to create highly realistic virtual characters while still allowing for biomechanically accurate simulation of large muscle groups. We present a pipeline for creating musculoskeletal models from the segmented visible human data set that allows for the creation of highly detailed models of muscle, tendon and bone. We demonstrate this by creating a musculoskeletal model of the upper limb. Then we embed each high resolution muscle geometry in a non-manifold, uniform simulation mesh. The embedding mesh is comprised of identical, well conditioned elements thus significantly relaxing the time step restriction allowing us to avoid quasi-static simulation. Since the elements in each mesh are identical, we only need to store the material coordinates of a single undeformed tetrahedron per muscle as opposed to storing material information for every element in the mesh. Contact is treated directly based on muscle geometry as opposed to procedurally created, error prone wrapping surfaces. The inclusion of inertia forces

while performing the simulations in [37] illustrated the importance of the tendonous connective tissue networks that wrap muscle groups. In response to this phenomenon, we incorporate the effects of these tissues in a contact/collision algorithm that works between the high-resolution geometry and the low resolution simulation mesh.

II. RELATED WORK

[39], [38] simulated deformable materials including the effects of elasticity, viscoelasticity, plasticity and fracture. Although they mentioned that either finite differences or the FEM could be used, they seemed to prefer a finite difference discretization. Subsequently, [20] advocated the FEM for simulating a human hand grasping a ball, and since then a number of authors have used the FEM to simulate volumetric deformable materials.

[9] used the FEM, brick elements, and the constitutive model of [45] to simulate a few muscles including a human bicep. Due to computational limitations at the time, very few elements were used in the simulation. [43] built an entire model of a monkey using deformed cylinders as muscle models. Their muscles were not simulated but instead deformed passively as the result of joint motions. [35] carried out similar work developing a number of different muscle models that change shape based on the positions of the joints. They emphasized that a plausible tendon model was needed to produce the characteristic bulging that results from muscle contraction. A recent trend is to use the FEM to simulate muscle data from the visible human data set, see e.g. [46], [22], [14], [15].

In order to increase the computational efficiency, a number of authors have been investigating adaptive simulation. [10] used a finite difference method with an octree for adaptive resolution. This was later improved in [12] using a weighted finite difference integration technique (which they mistakenly referred to as “finite volume”) to approximate the Laplacian and the gradient of the divergence operators. [11] used FEM with a multiresolution hierarchy of tetrahedral meshes, and [21] refined basis functions instead of elements.

III. MODEL CREATION

Geometrically accurate musculoskeletal models are desired in graphics, biomechanics and biomedical engineering. However, the intricacy of the human anatomy makes it difficult to procedurally create models of the musculature and skeleton. As a consequence, researchers have turned to volume data from actual human subjects as a source for geometry. One such source is the visible human data set which consists of high resolution images of millimeter-spaced cross sections of an adult male [41]. We use a segmented version of this data to create the muscle, tendon and skeleton geometry for our simulations. Using the segmented anatomy information, we first create level set representations of each tissue intended for simulation. Unfortunately, the segmented data often contains imperfections or is unfit for creating a reasonable simulation mesh. We repair each tissue using simple level set smoothing techniques (see e.g. [31]), and/or CSG operations. A tetrahedralized volume is then produced for each muscle (including tendon), and a triangulated surface is produced for each bone. Both of these are created using the implicit surface meshing framework of [27], [7].

Once the muscle, tendon and bone geometries have been created, we encode necessary additional information into each muscle representation including material heterogeneity (tendon is stiffer than muscle and does not undergo active contraction) as well as spatially varying muscle fiber directions. Additionally, the kinematic structure of the underlying skeleton must be created to drive skeletal motion. Finally, boundary conditions are specified to attach muscle and tendon to bone.

A. Level Set Extraction

Due to the large amount of noise and occasional inaccuracies present in the segmented data, creating our model begins with examining and fixing such problems. We rely on a dual explicit/implicit representation of the muscle geometry to facilitate the repair process. We first create a level set representation of each tissue we wish to simulate using the visible human data. This data consists of gray-scale images of 1.0 mm axial slices of the entire body with individual tissues and bones assigned different values. Information of this type naturally converts to Heaviside descriptions of each individual tissue. The meshing algorithm we use to create the explicit geometric representations (tetrahedralized volume or triangulated surface) as well as the level set procedures we use to smooth noisy data require a signed distance function which we generate using the fast marching method [40], [36].

After the level sets are generated, slice-by-slice contour sculpting is used to repair problem regions. First, each slice of a generated level set is viewed graphically to check for and eliminate errors that would otherwise interfere with either the anatomical accuracy of our model or the algorithm for the subsequent meshing process. We then use basic level set smoothing techniques such as motion by mean curvature (see e.g. [31]) to eliminate any further noise automatically.

B. Meshing Bone And Muscle

Once the level sets are free of the inaccuracies and noise present in the original data, we use them to construct a triangulated surface representation of each bone and a tetrahedralized volume representation of each muscle [27], [7]. The tetrahedral mesh generation algorithm begins by partitioning all of space with a body-centered cubic (BCC) tetrahedral lattice, and extracting the subset of the tetrahedra that intersect with the object volume defined by the level set. Then a red green mesh subdivision algorithm is used to refine the initial mesh to an appropriate level of detail using both curvature and surface information as refinement criteria. Extra care is taken with elements near the boundary in order to obtain a well conditioned simulation mesh. Finally, using either a mass spring or finite element model, the boundary nodes of the mesh are compressed towards the zero isocontour of the signed distance function. For the triangulated surfaces used for the rigid bodies this procedure is carried out with the surface of the BCC lattice.

C. Tendon and Bone Attachment Designation

A major flaw in the segmented data set is that a large amount of tendon tissue is absent. For example, the segmented biceps data lacks any information about the distal tendon and its proximal tendons are under-resolved. In order to add missing tendon tissue to each muscle mesh, we make use of both explicit and implicit representations of each muscle. While explicit representations allow for more efficient and accurate graphical rendering of objects, implicit representations are advantageous for Boolean operations. Our method for regenerating missing tendon tissue for a given muscle mesh makes use of simple CSG methods on graphically positioned tendon primitives. After a set of tendon primitives is positioned in relation to a muscle mesh where its missing tendon tissue should be, the union of the tendon primitives and the muscle mesh is calculated and converted into a new level set (see figure 2). This new level set then undergoes another iteration of the editing, smoothing, and meshing processes described above. Due to the efficiency of the level set creation and tetrahedral meshing algorithms, the cost of this second iteration is reasonable. The result of this step is an improved tetrahedralized volume representation for each muscle that includes both the muscle tissue and all of its associated tendon tissue.

To improve the accuracy of our model during simulation, it is necessary not only to include tendons in the tetrahedron meshes, but also to differentiate between muscle and tendon tissue as

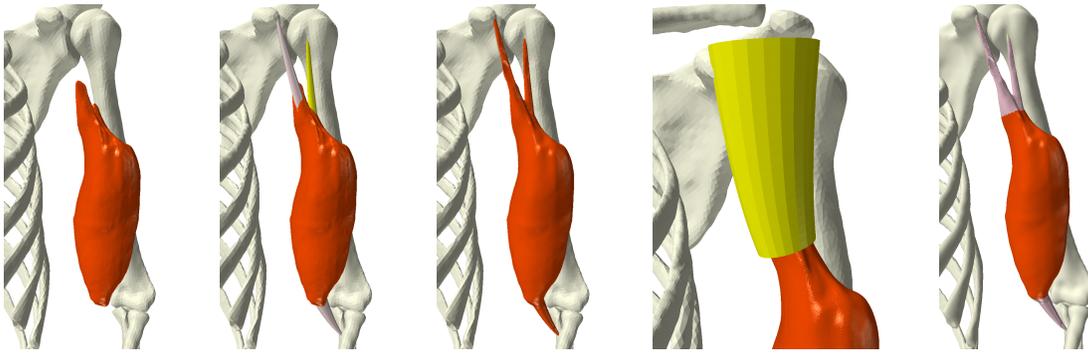


Fig. 2. Muscletendon mesh creation using CSG to repair errors in the biceps tendons. Heterogeneous tendon tetrahedra are selected using the triangulated surface primitive (large yellow cone in the fourth image). The rightmost image shows the finished product.

well as to define muscle-bone attachment regions. Therefore, we define subregions within each muscle mesh to represent muscle tissue, tendon tissue, and bone attachment regions. Tetrahedra designated as muscle are influenced by muscle activations whereas those designated as tendon remain passive during simulation. Furthermore, tendon tissue is an order of magnitude stiffer than muscle tissue. Tendon often extends into the belly of certain muscles forming an internal layer of passive tissue to which the active muscle fibers attach. This layer of connective tissue is known as an aponeurosis and can play a large role in many muscle functions [33], [16]. We take extra care to model this layer when selecting the regions of the muscle/tendon geometry to designate as tendon. Additionally, we rigidly attach tetrahedrons in the origin and insertion regions of each muscle mesh to their corresponding bones. Tetrahedrons that are designated as attached to bone are used to set Dirichlet boundary conditions during simulations.

Our method for defining the subregions described above involves graphically selecting portions of the mesh to be tendon or bone attachment tetrahedra leaving the remaining tetrahedra designated as muscle. In general, we use closed triangulated surfaces to select groups of tetrahedra making use of anatomy texts for anatomical accuracy. However, a good initial guess can be calculated by simply using a proximity threshold of the tetrahedra to a particular bone. We correct this guess by growing regions initially selected based on mesh connectivity as well as by graphical selection. See figure 3.

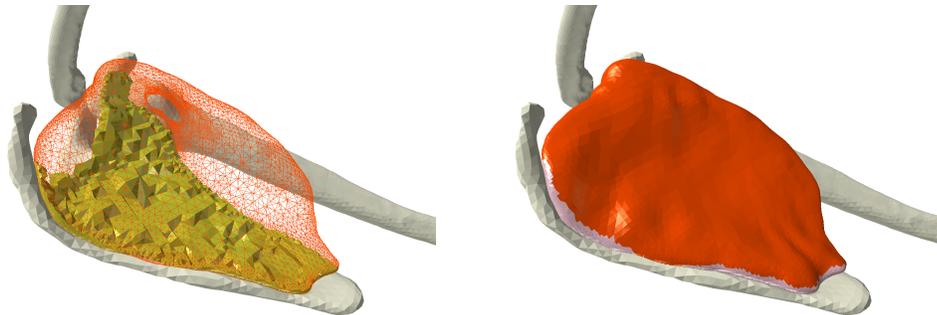


Fig. 3. Bone attachment process for the subscapularis and scapula. Constrained tetrahedra are shown in yellow, tendon tetrahedra are shown in pink. Bone attachment regions are determined by proximity and from anatomy texts.

D. B-spline Fiber Representation

Muscle tissue fiber arrangements vary in complexity from being relatively parallel and uniform to exhibiting several distinct regions of fiber directions. We use B-spline solids to assign fiber directions to individual tetrahedrons of our muscle simulation meshes querying the B-spline solid's local fiber direction at a spatial point corresponding to the centroid of a tetrahedron as in [29].

B-spline solids have a volumetric domain and a compact representation of control points, \mathbf{q}_{ijk} , weighted by B-spline basis functions $B^u(u), B^v(v), B^w(w)$:

$$\mathbf{F}(u, v, w) = \sum_i \sum_j \sum_k B_i^u(u) B_j^v(v) B_k^w(w) \mathbf{q}_{ijk}$$

where \mathbf{F} is a volumetric vector function mapping the material coordinates (u, v, w) to their corresponding spatial coordinates. Taking the partial derivatives of \mathbf{F} with respect to one of the three material coordinates $\partial\mathbf{F}/\partial u, \partial\mathbf{F}/\partial v, \partial\mathbf{F}/\partial w$ produces an implicit fiber field defined in the material coordinate direction. In [29], one of these directions always coincided with the local tangent of the muscle fiber located at the spatial position corresponding to the material coordinates. The inverse problem of finding the material coordinates for a given spatial point can be solved using numerical root-finding techniques to create a fiber query function

$$\mathbf{X}(\mathbf{x}) = \frac{\partial\mathbf{F}(\mathbf{F}^{-1}(\mathbf{x}))/\partial m}{\|\partial\mathbf{F}(\mathbf{F}^{-1}(\mathbf{x}))/\partial m\|}$$

with $m = \{u, v, w\}$ depending on the parameter chosen and the fiber directions normalized. The function \mathbf{X} describes an operation that first inversely maps the spatial points back to their corresponding material coordinates (u, v, w) and then computes the normalized fiber direction at that point.

We created these B-spline solids based on anatomy texts, however working with anatomy experts as in [29] or using fiber information from scanning technologies would improve accuracy. Additionally, using a fiber primitive template as was done in [3] would also improve accuracy and simplify the process.

E. Skeletal Motion

Bones are naturally articulated by ligaments and other soft tissues that surround them. However, we consider the inverse problem: a kinematic skeleton that drives the motion and contraction of the muscles and tendons attached to it. The joint spaces used to create a realistic kinematic structure involve intricate couplings of revolute and prismatic components resulting from the geometric complexity and redundancy of the muscles, tendons and ligaments that articulate the bones. Fortunately, there is much existing literature dedicated to the joint structures in the human body. We turned to the results of [17] to create the kinematic structure of the upper limb. In [17] the visible male was used to create a skeleton model of the right shoulder, elbow and wrist. Anatomical landmarks were then used to identify joint centers and to set up local coordinate frames for each of the bones. State of the art joint models with thirteen overall degrees of freedom were used to describe the relative motion of the sternum, clavicle, scapula, humerus, radius and ulna. Using the same virtual anatomy, we were able to directly incorporate their results.

Additional work was done in [18] to create a muscle model in the upper limb based on the Obstacle Set method for computing musculo-tendon paths, see figure 4 (left). This model for muscle length and moment arm computation assumes constant cross-sectional stress and simplifies the muscles to average lines of action. Basic geometric primitives like cylinders and spheres are used as collision objects to compute the paths of muscles as they collide with bones

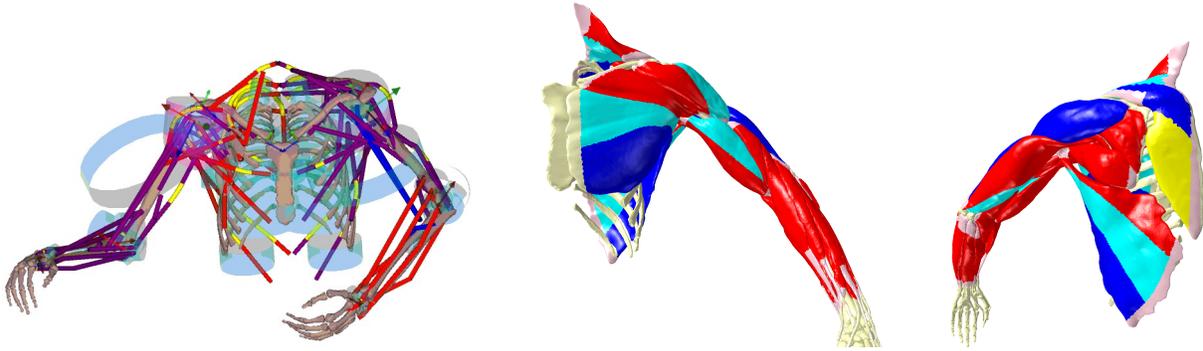


Fig. 4. The leftmost figure shows the piecewise linear muscle models with wrapping surfaces to model muscle contact in inverse dynamics calculations. Larger muscles have multiple contractile regions with individual activations and these must be embedded in the tetrahedron meshes for simulation (rightmost figures).

and other tissues. With this infrastructure in place, we use an inverse dynamics analysis with the results of [37] to compute activations for the muscles in the right upper limb. These techniques work with both motion capture and traditional animation.

IV. FINITE VOLUME METHOD

A. Force Computation

The Finite Volume Method provides a simple and geometrically intuitive way of integrating the equations of motion, with an interpretation that rivals the simplicity of mass-spring systems. However, unlike masses and springs, an arbitrary constitutive model can be incorporated into the FVM.

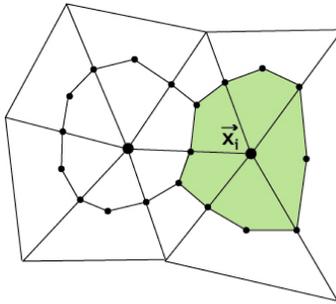


Fig. 5. FVM integration regions.

In the deformed configuration, consider dividing up the continuum into a number of discrete regions each surrounding a particular node. Figure 5 depicts two nodes each surrounded by a region. Suppose that we wish to determine the force on the node \mathbf{x}_i surrounded by the shaded region Ω . Ignoring body forces for brevity, the force can be calculated as

$$\mathbf{f}_i = \frac{D}{Dt} \int_{\Omega} \rho \mathbf{v} d\mathbf{x} = \oint_{\partial\Omega} \mathbf{t} dS = \oint_{\partial\Omega} \boldsymbol{\sigma} \mathbf{n} dS$$

where ρ is the density, \mathbf{v} is the velocity, and \mathbf{t} is the surface traction on $\partial\Omega$. The last equality comes from the definition of the Cauchy stress $\boldsymbol{\sigma} \mathbf{n} = \mathbf{t}$. Evaluation of the boundary integral requires integrating over the two segments interior to each incident triangle. Since $\boldsymbol{\sigma}$ is constant in each triangle and the integral of the local unit normal over any closed region is identically zero (from the divergence theorem), we have

$$\oint_{\partial\Omega_1} \boldsymbol{\sigma} \mathbf{n} dS + \oint_{\partial\Omega_2} \boldsymbol{\sigma} \mathbf{n} dS + \oint_{\partial T_1} \boldsymbol{\sigma} \mathbf{n} dS + \oint_{\partial T_2} \boldsymbol{\sigma} \mathbf{n} dS = 0$$

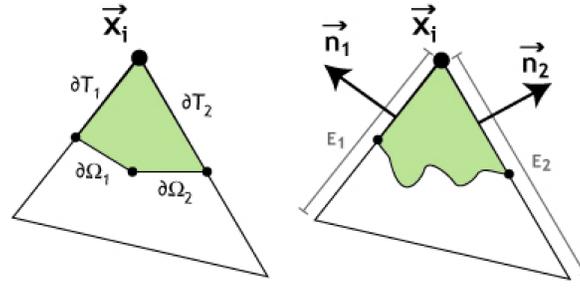


Fig. 6. Integration over a triangle.

where ∂T_1 and ∂T_2 are depicted in figure 6 (left).

More importantly, we have

$$\oint_{\partial\Omega_1} \sigma \mathbf{n} dS + \oint_{\partial\Omega_2} \sigma \mathbf{n} dS = - \oint_{\partial T_1} \sigma \mathbf{n} dS - \oint_{\partial T_2} \sigma \mathbf{n} dS$$

indicating that the integral of $\sigma \mathbf{n}$ over $\partial\Omega_1$ and $\partial\Omega_2$ can be replaced by the integral of $-\sigma \mathbf{n}$ over ∂T_1 and ∂T_2 . That is, for each triangle, we can integrate over the portions of its edges incident to \mathbf{x}_i instead of the two interior edges $\partial\Omega_1$ and $\partial\Omega_2$. Moreover, even if $\partial\Omega_1$ and $\partial\Omega_2$ are replaced by an arbitrary path inside the triangle, we can replace the integral over this region with the integral over ∂T_1 and ∂T_2 . We choose an arbitrary path inside the triangles that connects the *midpoints* of the two edges incident on \mathbf{x}_i as shown in figure 6 (right). Then the surface integrals are simply equal to $-\sigma \mathbf{n}_1 e_1/2$ and $-\sigma \mathbf{n}_2 e_2/2$ where e_1 and e_2 are the edge lengths of the triangles. Thus, the force on node \mathbf{x}_i is updated via

$$\mathbf{f}_i+ = -\frac{1}{2}\sigma (e_1 \mathbf{n}_1 + e_2 \mathbf{n}_2).$$

In three spatial dimensions, given an *arbitrary* stress σ , regardless of the method in which it was obtained, we obtain the FVM force on the nodes in the following fashion. Loop through each tetrahedron interpreting $-\sigma$ as the outward pushing “ multidimensional force”. For each face, multiply by the outward unit normal to calculate the traction on that face. Then multiply by the area to find the force on that face, and simply redistribute one third of that force to each of the incident nodes. Thus, each tetrahedron will have three faces that contribute to the force on each of its nodes, e.g. the force on node \mathbf{x}_i is updated via

$$\mathbf{f}_i+ = -\frac{1}{3}\sigma (a_1 \mathbf{n}_1 + a_2 \mathbf{n}_2 + a_3 \mathbf{n}_3).$$

Note that the cross product of two edges is twice the area of a face times the normal, so we can simply add one sixth of $-\sigma$ times the cross product to each of the three nodes.

B. Piola-Kirchhoff Stress

A deformable object is characterized by a time dependent map ϕ from undeformed material coordinates \mathbf{X} to deformed spatial coordinates \mathbf{x} . We use a tetrahedron mesh and assume that the deformation is piecewise linear, which implies $\phi(\mathbf{X}) = \mathbf{F}\mathbf{X} + \mathbf{b}$ in each tetrahedron. For simplicity, consider two spatial dimensions where each element is a triangle. Figure 7 depicts a mapping ϕ from a triangle in material coordinates to the resulting triangle in spatial coordinates. We define edge vectors for each triangle as $\mathbf{d}_{m_1} = \mathbf{X}_1 - \mathbf{X}_0$, $\mathbf{d}_{m_2} = \mathbf{X}_2 - \mathbf{X}_0$, $\mathbf{d}_{s_1} = \mathbf{x}_1 - \mathbf{x}_0$ and $\mathbf{d}_{s_2} = \mathbf{x}_2 - \mathbf{x}_0$. Note that $\mathbf{d}_{s_1} = (\mathbf{F}\mathbf{X}_1 + \mathbf{b}) - (\mathbf{F}\mathbf{X}_0 + \mathbf{b}) = \mathbf{F}\mathbf{d}_{m_1}$ and likewise $\mathbf{d}_{s_2} = \mathbf{F}\mathbf{d}_{m_2}$ so that \mathbf{F} maps the edges of the triangle in material coordinates to the edges of the triangle in spatial coordinates. Thus, if we construct 2×2 matrices \mathbf{D}_m with columns \mathbf{d}_{m_1} and \mathbf{d}_{m_2} , and \mathbf{D}_s with columns \mathbf{d}_{s_1} and \mathbf{d}_{s_2} , then $\mathbf{D}_s = \mathbf{F}\mathbf{D}_m$ or $\mathbf{F} = \mathbf{D}_s \mathbf{D}_m^{-1}$. The matrix \mathbf{F} is known as

the deformation gradient and conveys all the necessary information to determine the material response to deformation, since the translational component of ϕ does not induce any stress. In three spatial dimensions, \mathbf{D}_m and \mathbf{D}_s are 3×3 matrices with columns equal to the edge vectors of the tetrahedra. Note that \mathbf{D}_m^{-1} can be precomputed and stored for efficiency.

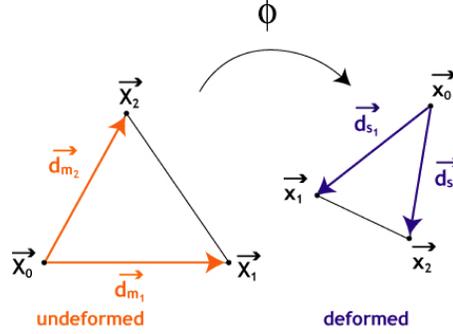


Fig. 7. Undeformed and deformed triangle edges.

Often, application of a constitutive model will result in a second Piola-Kirchoff stress, \mathbf{S} , which can be converted to a Cauchy stress via $\sigma = J^{-1}\mathbf{F}\mathbf{S}\mathbf{F}^T$ where $J = \det(\mathbf{F})$. Using this equality and the identity $a\mathbf{n} = J\mathbf{F}^{-T}\mathbf{A}\mathbf{N}$, we can write

$$\mathbf{f}_i+ = -\frac{1}{3}\mathbf{P} (A_1\mathbf{N}_1 + A_2\mathbf{N}_2 + A_3\mathbf{N}_3)$$

where $\mathbf{P} = \mathbf{F}\mathbf{S}$ is the first Piola-Kirchoff stress tensor, the A_i are the areas of the undeformed tetrahedron faces incident to \mathbf{X}_i and the \mathbf{N}_i are the normals to those undeformed faces.

Since the A_i and \mathbf{N}_i do not change during the computation, we can precompute and store these quantities. Then the force contribution to each node can be computed as $\mathbf{g}_i = \mathbf{P}\mathbf{b}_i$, where the \mathbf{b}_i are precomputed and the force on each node is updated with $\mathbf{f}_i+ = \mathbf{g}_i$. Moreover, we can save 9 multiplications by computing $\mathbf{g}_0 = -(\mathbf{g}_1 + \mathbf{g}_2 + \mathbf{g}_3)$ instead of $\mathbf{g}_0 = \mathbf{P}\mathbf{b}_0$. We can compactly express the computation of the other \mathbf{g}_i as $\mathbf{G} = \mathbf{P}\mathbf{B}_m$ where $\mathbf{G} = (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$ and $\mathbf{B}_m = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$. Thus, given an *arbitrary* stress \mathbf{S} in a tetrahedron, the force contribution to all four nodes can be computed with two matrix multiplications and 6 additions for a total of 54 multiplications and 42 additions. A similar expression can be obtained for the Cauchy stress, $\mathbf{G} = \sigma\mathbf{B}_s$ where \mathbf{B}_s is computed using deformed (instead of undeformed) quantities. Unfortunately, \mathbf{B}_s cannot be precomputed since it depends on the deformed configuration.

C. Comparison with FEM

Using constant strain tetrahedra, linear basis functions N_i , etc., an Eulerian FEM derivation [5] leads to a force contribution of

$$\mathbf{g}_i = \int_{tet} \sigma \nabla N_i^T d\mathbf{v}.$$

A few straightforward calculations lead to

$$\mathbf{G} = \int_{tet} \sigma \mathbf{D}_s^T d\mathbf{v} = \sigma \mathbf{D}_s^T v = \sigma \hat{\mathbf{B}}_s$$

using our compact notation. Here, v is the volume of the deformed tetrahedron and $\hat{\mathbf{B}}_s = v\mathbf{D}_s^{-T}$.

Now consider $\mathbf{D}_s^T\mathbf{B}_s$ from the FVM formulation. Since the rows of \mathbf{D}_s^T are edge vectors and the columns of \mathbf{B}_s are each the sum of three cross-products of edges divided by 6, we obtain a number of terms that are triple products of edges divided by 6. Each of these terms is equal to either 0 or $\pm v$, and the final result is $\mathbf{D}_s^T\mathbf{B}_s = v\mathbf{I}$. That is, $\mathbf{B}_s = v\mathbf{D}_s^{-T} = \hat{\mathbf{B}}_s$, and in this case

of constant strain tetrahedra, linear basis functions, etc. (see e.g. [30], [28]), FVM and FEM are identical methods.

\mathbf{D}_s^{-T} is the cofactor matrix of \mathbf{D}_s^T divided by the determinant, and since \mathbf{D}_s^T is a matrix of edge vectors, its determinant is a triple product equal to $6v$. That is, $\hat{\mathbf{B}}_s = v\mathbf{D}_s^{-T}$ computes the volume twice even though it cancels out resulting in a cofactor matrix times $1/6$. Thus, \mathbf{B}_s can be computed with 27 multiplications and 18 additions, for a total of 54 multiplications and 42 additions to compute the force contributions using the Cauchy stress.

Muller et al. [28] point out that a typical FEM calculation such as in O'Brien and Hodgins [30] requires about 288 multiplications. Instead, they use QR-factorization, loop unrolling, and the precomputation and storage of 45 numbers per tetrahedron to reduce the amount of calculation to a level close to our 54 multiplications. However, in the second Piola-Kirchhoff stress case that they consider, we only need to store 9 numbers per tetrahedron (as opposed to 45). Moreover, in the Cauchy stress case that they do not consider, it is not clear that their optimizations could be applied without an expensive calculation to transform back to a second Piola-Kirchhoff stress. On the other hand, using the geometric intuition we gained from FVM that led to the cancellation of v (that other authors have not noted [30], [28]), we once again need only 54 multiplications and this time do not need to precompute and store any extra information at all.

D. Invertible Finite Elements

Motivated by our geometric FVM formulation, [23] introduced a strategy that allows one to robustly treat inverted or degenerate tetrahedra via a new *polar SVD* technique that expresses the deformation gradient in a space that makes it a diagonal matrix. In this doubly rotated space, one can readily extend any constitutive model into the degenerate and inverted regime in a fashion that results in smooth force behavior that opposes degeneracy and inversion.

To extend constitutive models to degenerate elements, [23] makes use of the newly proposed *polar SVD* of $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$ where \mathbf{U} and \mathbf{V} are rotation matrices and $\hat{\mathbf{F}}$ is a diagonal matrix. The inverting elements framework is applied in the following fashion. First, \mathbf{V}^T rotates the tetrahedron from material coordinates into a coordinate system where the deformation gradient is conveniently a diagonal matrix. Similarly, \mathbf{U}^T rotates the tetrahedron from spatial coordinates into this same space. Typically researchers work to find the polar decomposition that gives the rotation relating material space to world space. Removing this rotation produces a still difficult to work with symmetric deformation gradient. In contrast, the polar SVD gives two rotations, one for the material space tetrahedron and one for the world space tetrahedron. After applying these, the deformation gradient has a much more convenient diagonal form. In practice, the polar SVD is used to find the diagonal deformation gradient, to apply the constitutive model and the FVM forces in diagonal space in standard fashion, and then map the forces on the nodes back to world space using \mathbf{U} . The beauty of working in a space that has a diagonal deformation gradient is that it is trivial to extend constitutive models to work for degenerate and inverted elements.

We display the robustness of the inverting FVM algorithm which was developed from the geometric FVM framework. An exceptionally soft torus is dropped to the ground and crushed flat by its own weight. The Young's modulus is then substantially increased causing it to jump from the ground and into the air demonstrating that simulation can proceed despite large numbers of inverted and degenerate elements. The results are shown in figure 8.

The simulation environment for large muscle groups can be considerably volatile. In regions like the shoulder girdle, muscles are constantly in contact with other muscles, tendons and bones. In addition, the kinematic skeleton subjects them to an extreme range of boundary conditions. An additional complication comes from the errors in modeling the complex structure of the glenohumeral and sternoclavicular joints that determine the motion of the clavicle, scapula and humerus relative to the sternum. Errors inherent in modeling these joints can cause spurious

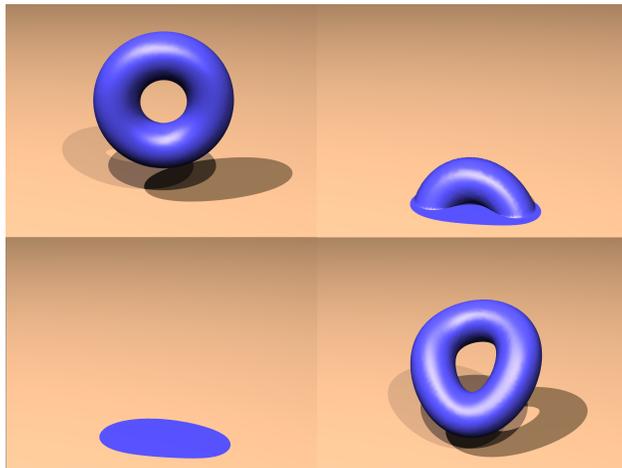


Fig. 8. Deformable torus simulated with the inverting FVM. A torus with near zero strength collapses into a puddle. When the strength is increased, the torus recovers.

configurations of the musculature that can cause tetrahedra in the computational domains involved to invert. Perfectly recreating the joint kinematics in the region might alleviate these issues, however it is prohibitively difficult. Rather, we employ the inverting FVM/FEM framework. This algorithm allows elements to arbitrarily invert and return to more reasonable configurations later in the simulation, enabling simulations to progress that would have otherwise ground to a halt.

V. CONSTITUTIVE MODEL FOR MUSCLE

Muscle tissue has a highly complex material behavior—it is a nonlinear, incompressible, anisotropic, hyperelastic material and we use a state-of-the-art constitutive model to describe it with a strain energy of the following form

$$W(I_1, I_2, \lambda, \mathbf{a}_o, \alpha) = F_1(I_1, I_2) + U(J) + F_2(\lambda, \alpha)$$

where I_1 and I_2 are deviatoric isotropic invariants of the strain, λ is a strain invariant associated with transverse isotropy (it equals the deviatoric stretch along the fiber direction), \mathbf{a}_o is the fiber direction, and α represents the level of activation in the tissue. F_1 is a Mooney-Rivlin rubber-like model that represents the isotropic tissues in muscle that embed the fascicles and fibers, $U(J)$ is the term associated with incompressibility, and F_2 represents the active and passive muscle fiber response. F_2 must take into account the muscle fiber direction \mathbf{a}_o , the deviatoric stretch in the along-fiber direction λ , the nonlinear stress-stretch relationship in muscle, and the activation level. The tension produced in a fiber is directed along the vector tangent to the fiber direction. The relationship between the stress in the muscle and the fiber stretch has been established using single-fiber experiments and then normalized to represent any muscle fiber [45]. This strain energy function is based on [42] and is the same as that used in [37].

This model does have some notable limitations. Muscle undergoes history dependent changes in elasticity such as strain hardening and has a force/velocity relationship in addition to force/length dependence [45], [34]. Additionally, we neglect any model for anisotropic shear behavior relative to the fiber axis. Our model includes only what is necessary to produce bulk length based contraction along the muscle fiber directions. Given the large number of colliding and contacting muscles we wish to simulate, the effects of these phenomena on the bulk muscle deformation are subtle at best. However, when focusing on more specific behavior in a more localized region of muscle, e.g. non-uniform contraction of the biceps as in [33], it would be useful to add the

effects of these phenomena. Note that our framework readily allows for a more sophisticated constitutive model such as that proposed in [4].

The diagonalized FEM framework of [23] is most naturally formulated in terms of a first Piola-Kirchoff stress. A stress of this type corresponding to the above constitutive model has the form

$$\begin{aligned}\mathbf{P} &= w_{12}\mathbf{F} - w_2\mathbf{F}^3 + (p - p_f)\mathbf{F}^{-1} + 4J_{cc}T(\mathbf{F}\mathbf{f}_m)\mathbf{f}_m^T \\ J_c &= \det(\mathbf{F})^{-\frac{1}{3}}, J_{cc} = J_c^2, I_1 = J_{cc}\mathbf{C}, \lambda = \sqrt{\mathbf{f}_m^T \mathbf{C} \mathbf{f}_m} \\ w_1 &= 4J_{cc}mat_{c1}, w_2 = 4J_{cc}^2mat_{c2}, w_{12} = w_1 + I_1W_2 \\ p &= K\log(J), p_f = \frac{1}{3}(w_{12}Tr(\mathbf{C}) - w_2Tr(\mathbf{C}^2) + T\lambda^2)\end{aligned}$$

Here, \mathbf{F} is the deformation gradient, $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ is the Cauchy strain and \mathbf{f}_m is the local fiber direction (in material coordinates). mat_{c1} and mat_{c2} are Mooney-Rivlin material parameters and K is the bulk modulus. T is the tension in the fiber direction from the force length curve (see [45]). Typical values for these parameters are:

$$\begin{aligned}mat_{c1} &= 30000Pa \text{ (muscle)}, mat_{c1} = 60000Pa \text{ (tendon)}, \\ mat_{c2} &= 10000Pa \text{ (muscle and tendon)}, \\ K &= 60000Pa \text{ (muscle)}, K = 80000Pa \text{ (tendon)}, \\ T &= 80000Pa.\end{aligned}$$

This formula holds throughout both the muscle and tendon tetrahedra, however the tendons are passive (no active stress). Note that tendon is considerably stiffer than muscle. Modeling this inhomogeneity is essential for generating muscle bulging during contraction (as well as for accurately computing the musculotendon force generating capacity). Also, large muscles like the deltoid, trapezius, triceps and latissimus dorsi have multiple regions of activation. That is, muscle contraction and activation is non-uniform in the muscle. In general, the effects of varying activation within a muscle can be localized to a few contractile units in each muscle. For example, each head of the biceps and triceps receive individual activations (see figure 4).

Fascia tissues wrap individual muscles and muscle groups and are made up of fibrous material with a stiffness similar to that of tendon. These elastic sheaths hold the muscles together and as a result keep the muscle near the underlying skeleton during motion. The stiffness of these connective tissues must be incorporated into the muscle constitutive model. One approach is to make each muscle inhomogeneously stiff near the muscle boundary (i.e. similar material to tendon). However, we simply add an additional resistance to elongation in the constitutive model to encourage resistance to stretching on the boundary of the muscles. This is done by adding in an additional linearly elastic stress into the diagonalized form of the constitutive model during elongation. The problematic effects of large rotations associated with linear elasticity are naturally removed in the diagonalized setting, see [23]. Elongation is identified when the diagonalized deformation gradient values are greater than 1.

VI. EMBEDDING FRAMEWORK

The human musculature is geometrically complex and creating a visually realistic model requires many degrees of freedom. Our upper limb model has over thirty muscles made up of over 10 million tetrahedra. The simulation of such a model is hindered by both its overall size and the time step restriction imposed by the smallest tetrahedron in the mesh. To reduce the computational cost, our system uses a dynamic Free Form Deformation embedding scheme. The simulation mesh is created by overlaying a BCC lattice on the high resolution geometry (as in [27], [7]). For each particle on the surface of the initial high resolution tetrahedralized volume,

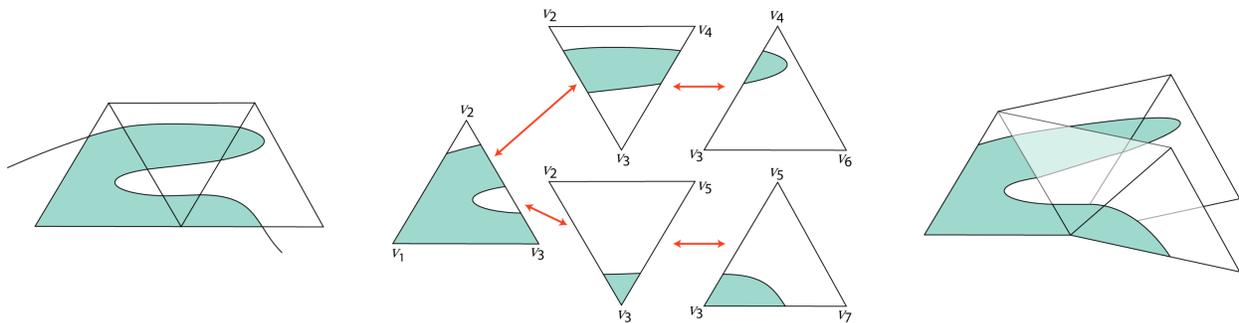


Fig. 9. Illustration of a topology-offending embedding scenario (left). Individual connected components of material within the same element are assigned to distinct copies of the original (middle). Vertices on common boundaries of elements that exhibit material continuity (indicated by arrows) are considered equivalent. Collapsing equivalent vertices leads to the final non-manifold simulation mesh (right).

we compute its barycentric coordinates in the low resolution tetrahedron that contains it and use these to update the high resolution geometry during subsequent simulation.

Our BCC embedding approach gives rise to several substantial benefits. The BCC grid size we used led to a tenfold reduction in the size of the simulation mesh, from about ten million to about one million tetrahedra. Most importantly, the time step restriction for stability was relaxed by a factor of 25 owing to the regular structure of the BCC tetrahedra and the elimination of poorly shaped elements. These combined facts enabled the full finite element simulation of the whole upper limb musculature at rates of 4 minutes per frame on a single CPU Xeon 3.06Ghz workstation. Substantial RAM savings are also achieved, since all simulation tetrahedra are identical up to a rigid body transform eliminating the need to store the rest state matrix on an individual tetrahedron basis. Only one rest state tetrahedron is stored per muscle.

The embedding process can potentially change the topology of the original high resolution geometry, since the original connectivity of the input geometry is projected to the connectivity of the embedding coarse tetrahedra. Cases where parts of the high resolution geometry attempt to separate but cannot since they are embedded in the same coarse tetrahedron (see figure 9) are particularly frequent in our musculoskeletal simulation, for example in the concavity between the two heads of the bicep. To some extent, this change of topology is inevitable as we are reducing the number of degrees of freedom. Nevertheless, we propose to limit the undesirable topology changes by relaxing some constraints on the embedding mesh. In particular we allow it to be non-manifold and to possess multiple copies of nodes corresponding to the same location in space in a fashion similar to the “virtual node algorithm” of [26].

Consider a coverage of our high resolution geometry by a manifold tetrahedral mesh as illustrated in figure 9 (left). We note that the fragment of the high resolution geometry that is contained within each tetrahedron might consist of several disjoint connected components as is the case in the two rightmost elements of our example. In order to avoid connecting such disjoint material fragments by embedding them in the same tetrahedral element, we create a copy of the original tetrahedron for each one of them as shown in figure 9 (middle). All tetrahedra thus created are completely disjoint, in the sense that we assign a different copy of each vertex of the original mesh to each duplicate tetrahedron that contains it. We subsequently assign each connected material fragment within an original tetrahedron to a different one of its newly created copies.

In the second phase of our algorithm, we rebuild the connectivity of our mesh by collapsing vertices on adjacent tetrahedra that should correspond to the same degree of freedom. In particular, when two of the newly created tetrahedra exhibit material continuation somewhere across their common face, their corresponding vertices are identified. Such pairs are indicated

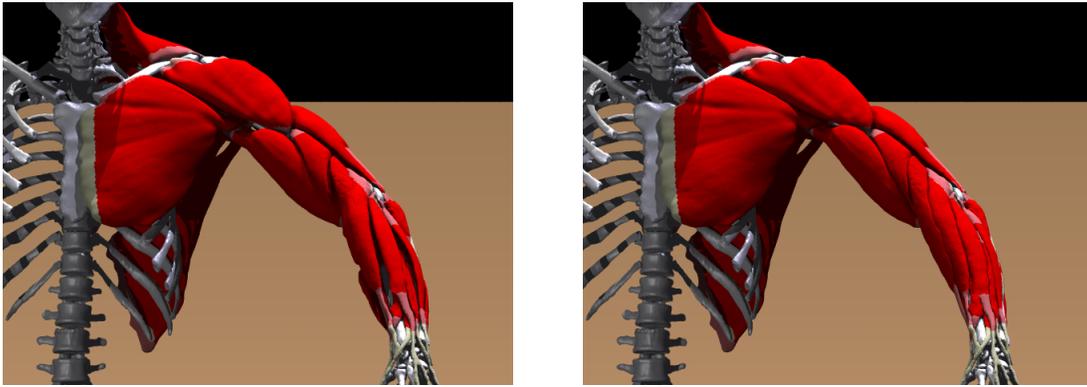


Fig. 10. Muscle simulations without fascia (left) and with fascia (right) show the effects of inertia forces in the absence of connective tissue. Note, for example, the spurious separation of muscles in the forearm (left).

with double arrows in figure 9 (middle). For each corresponding pair of vertices on the common boundary of two materially contiguous tetrahedra, we collapse the two vertices onto a single one using a union-find data structure for the vertex indices. The resulting tetrahedron mesh is non-manifold in general, as illustrated in figure 9 (right).

After our mesh generation process we project the fiber directions, inhomogeneities (tendon material) and boundary conditions (origins and insertions) from the high resolution mesh to the coarse simulation mesh. Using a BCC covering of space as our generator mesh provides for an efficient implementation as most point location or tetrahedron intersection queries can be performed in constant and linear time respectively. We note that in our current implementation the mesh generation is a static process performed prior to the beginning of the simulation, although the described technique extends to a dynamic context if the topology of our input geometry is changing in time.

VII. FASCIA AND CONNECTIVE TISSUES

Skeletal muscles are contained in a network of connective tissue, much of which is fascia, that keeps them in tight contact during motion. Without modeling these constraints, dynamic models will have difficulties with ballistic motion and exhibit spurious separation as shown in figure 10. Our fascia model enforces a state of frictionless contact between muscles. It is similar in spirit to [24], [8], [6] which all used “sticky” regions in one sense or another to create (possibly temporary) bonds between geometry in close proximity.

The fascia framework works in the context of the embedding techniques presented in section VI. First, we find all intersections between the high resolution muscle surface and the edges of the BCC simulation mesh and label these embedded nodes. The primitives in our fascia model are line segments (*links*) that connect each embedded node to its closest point (*anchor*) on the high resolution surface of each nearby muscle. Links between each embedded node and *all* its neighboring muscles within a certain distance are initially created and their anchors are maintained as the closest points of the corresponding muscle surfaces during simulation. Each time step, the link corresponding to the closest anchor is selected as the active one and dictates the contact response.

Every time step, the fascia links are used to adjust the BCC mesh. Each embedded node has a position \mathbf{x}_{emb} and velocity \mathbf{v}_{emb} which can be compared to the position \mathbf{x}_a and velocity \mathbf{v}_a of its anchor. Ideally, we need the positions and velocities along the normal direction to the surface at the anchor position to match closely. Thus, we compute a new desired position and velocity

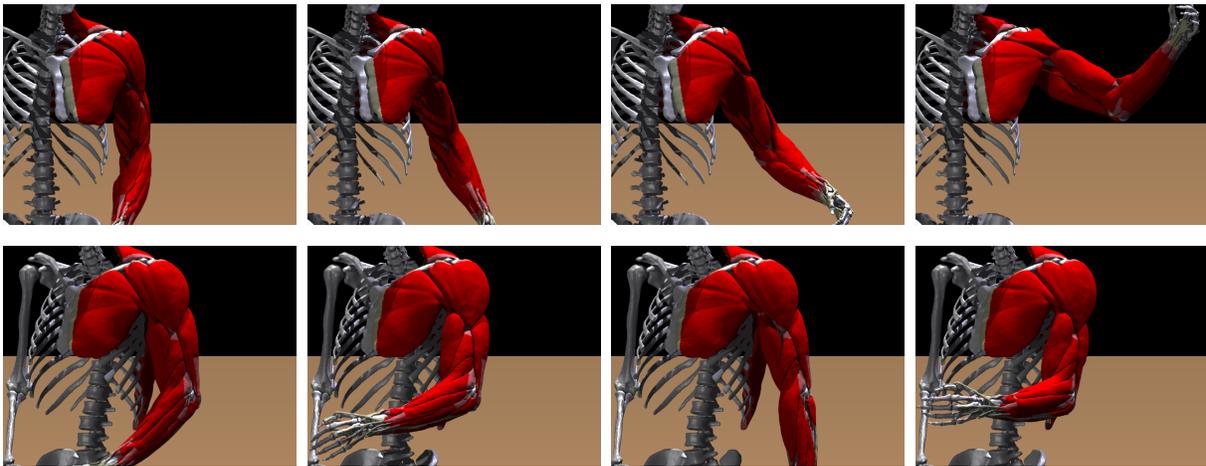


Fig. 11. Simulation of muscles in the upper limb.

for the embedded point:

$$\begin{aligned}\vec{\mathbf{x}}'_{emb} &= \vec{\mathbf{x}}_{emb} + \alpha[(\vec{\mathbf{x}}_a - \vec{\mathbf{x}}_{emb}) \cdot \vec{\mathbf{N}}]\vec{\mathbf{N}} \\ \vec{\mathbf{v}}'_{emb} &= \vec{\mathbf{v}}_{emb} + \beta[(\vec{\mathbf{v}}_a - \vec{\mathbf{v}}_{emb}) \cdot \vec{\mathbf{N}}]\vec{\mathbf{N}}.\end{aligned}$$

The embedded particle and the anchor should optimally meet halfway with $\alpha = \beta = .5$, although we cannot move either of these points since they are both enslaved by their embedding BCC lattices. Thus, we first compute the desired position and velocity changes for all embedded particles and map these to the BCC mesh in a second step. The anchor end of each link does not inflict any correction on the neighboring muscle as that effect is accomplished by the links originating on that particular muscle. We found values of $\alpha = .1$ and $\beta = .5$ to work well in practice and attenuate them as the length of a link surpasses a given threshold.

In the second step we map the desired state of the embedded nodes to the BCC mesh. For each node on the BCC mesh, we look through all its edges to find embedded nodes, and change the position and velocity of this BCC node using the average desired change recorded by the embedded nodes. See [26] for more details.

Figure 10 shows a comparison of a simulation with and without fascia. The effects of the connective tissues and the problems that inertia forces can cause in their absence are evident in the muscles of the forearm that wobble around, unnaturally separating from the bone.

VIII. SIMULATING SKELETAL MUSCLE

We demonstrate the strength of our pipeline with a series of skeletal animations of the upper limb (see figure 11). The bones in the shoulder and the arm are animated through a series of key-frames and thirty muscles are simulated with FVM. Inverse dynamics were used with the results of [37] to compute muscle activations at each one of the key-frame poses in the animation. The activations obtained were interpolated at key-frames (just as for the bone positions) throughout the simulation.

IX. CONCLUSIONS AND FUTURE WORK

Unfortunately, computational complexity and limitations in existing algorithms limit the scope and accuracy of musculoskeletal models in both graphics and biomechanics. In computer graphics, the emphasis is on the visual nature of the musculature and particularly the effect that it has on the skin. As a result, models in the field have focused mainly on generating plausible

muscle geometry at the expense of other quantities. However, muscle geometry, fascicle length, stress, force generating properties, etc. are all coupled together. As technology and algorithms improve and demands for realism are met in both graphics and biomechanics, the models used for examining the respective quantities will become more and more similar. Our framework is a step in this direction.

The presented framework allows for the creation of highly detailed geometry as well as for realistic anisotropies and heterogeneities. Additionally, realistic dynamic deformations are produced from a transversely isotropic muscle constitutive model. The computational burden of simulating large muscle groups is ameliorated by our embedding framework while preserving high resolution geometry for rendering. The volatile simulation environment, inherent in the complex coupling of intricately articulated rigid bodies and dozens of contacting deformable objects, is handled by the extremely robust diagonalized FEM. In addition, our fascia model both robustly recreates the effects of the connective tissues that surround the muscles as well as efficiently resolving the unique contact environment inherent in the musculoskeletal system.

However, many aspects of the pipeline could be improved. More realistic muscle constitutive models that include the force/velocity relationship, time dependent elasticity changes noted in [34] as well as anisotropic shear behavior relative to the fiber axis as in [4], [3] can be used when examining more specific phenomena on a smaller scale such as nonuniform contraction of the biceps.

While the geometry of the musculoskeletal system extracted from the segmented visible human is very well resolved, the tendon/aponeurosis and fiber information could be improved with the aid of scanning technologies or anatomy experts. In the future, subject specific models would be desirable using segmented data from MRI and CT. However, the resolution of the visible human data set is still greater than those that are attainable with scanning technologies. Thus, given the additional difficulty of segmenting the scanned data, a reasonable alternative approach is to use the model created from the visible human data set and to deform (or morph) it to match a specific subject or body type using anatomical landmarks similar to [15].

X. ACKNOWLEDGEMENTS

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ONR N00014-02-1-0720, ARO DAAD19-03-1-0331, NSF DMS-0106694, NSF ITR-0121288, NSF IIS-0326388, NSF ACI-0323866, NSF ITR-0205671, NIH HD38962 and by Honda Research Institute USA. In addition, J. T. and S. B. were each supported in part by a National Science Foundation Graduate Research Fellowship and E. S. was supported by a Stanford Graduate Research Fellowship.

REFERENCES

- [1] A. Arnold, S. Blemker, and S. Delp. Evaluation of a deformable musculoskeletal model for estimating muscle-tendon lengths during crouch gait. *Comput. Aided Surg.*, 29:263–274, 2001.
- [2] A. Arnold and S. Delp. Rotational moment arms of the medial hamstrings and adductors vary with femoral geometry and limb position: Implications for the treatment of internally rotated gait. *J. Biomech.*, 34:437–447, 2001.
- [3] S. Blemker. *3D Modeling of Complex Muscle Architecture and Geometry*. PhD thesis, Stanford University, June 2004.
- [4] S. Blemker, P. Pinsky, and S. Delp. A 3d model of muscle reveals the causes of nonuniform strains in the biceps brachii. *J. Biomech.*, In Press., 2004.
- [5] J. Bonet and R. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, Cambridge, 1997.
- [6] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 28–36, 2003.
- [7] R. Bridson, J. Teran, N. Molino, and R. Fedkiw. Adaptive physics based tetrahedral mesh generation using level sets, (in press). 2004.
- [8] J. T. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *Proc. ACM SIGGRAPH Symposium on Computer Animation*, pages 77–80, 2002.

- [9] D. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Comput. Graph. (SIGGRAPH Proc.)*, pages 89–98, 1992.
- [10] G. DeBunne, M. Desbrun, A. Barr, and M-P. Cani. Interactive multiresolution animation of deformable models. In *Proc. of the Eurographics Wrkshp on Comput. Anim. and Sim. 1999*. Springer Verlag, 1999.
- [11] G. DeBunne, M. Desbrun, M. Cani, and A. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proc. SIGGRAPH 2001*, volume 20, pages 31–36, 2001.
- [12] G. DeBunne, M. Desbrun, M.-P. Cani, and A. Barr. Adaptive simulation of soft bodies in real-time. In *Comput. Anim. 2000, Philadelphia, USA*, pages 133–144, May 2000.
- [13] S. Delp and J. Loan. A computational framework for simulating and analyzing human and animal movement. *IEEE Computing In Science And Engineering*, 2(5):46–55, 2000.
- [14] F. Dong, G. Clapworthy, M. Krokos, and J. Yao. An anatomy-based approach to human muscle modeling and deformation. *IEEE Trans. Vis. Comput. Graph.*, 8(2), 2002.
- [15] J. Fernandez, F. Mithraratne, S. Thrupp, M. Tawhai, and P. Hunter. Anatomically based geometric modeling of the musculo-skeletal system and other organs. *Biomech Mod. Mechanobio.*, 3, 2003.
- [16] T. Fukunaga, Y. Kawakami, S. Kuno, K. Funato, and S. Fukashiro. Muscle architecture and function in humans. *J. Biomech.*, 30, 1997.
- [17] B. Garner and M. Pandy. A kinematic model of the upper limb based on the visible human project (vhp) image dataset. *Comput. Meth. Biomech. Biomed. Eng.*, 2:107–124, 1999.
- [18] B. Garner and M. Pandy. Musculoskeletal model of the upper limb based on the visible human male dataset. *Comput. Meth. Biomech. Biomed. Eng.*, 4:93–126, 2001.
- [19] A.J.W. Gielen, P.H.M. Bovendeerd, and J.D. Janssen. A three dimensional continuum model of skeletal muscle. *Computer Methods in Biomechanics and Biomedical Engineering*, 3:231–244, 2000.
- [20] J.-P. Gourret, N. Magnenat-Thalmann, and D. Thalmann. Simulation of object and human skin deformations in a grasping task. *Comput. Graph. (SIGGRAPH Proc.)*, pages 21–30, 1989.
- [21] E. Grinspun, P. Krysl, and P. Schroder. CHARMS: A simple framework for adaptive simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:281–290, 2002.
- [22] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In *Comput. Anim.*, 2001.
- [23] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 131–140, 2004.
- [24] S. Jimenez and A. Luciani. Animation of interacting objects with collisions and prolonged contacts. In B. Falcidieno and T. L. Kunii, editors, *Modeling in computer graphics—methods and applications*, Proc. of the IFIP WG 5.10 Working Conference, pages 129–141. Springer-Verlag, 1993.
- [25] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics*, 22(3):562–568, 2003.
- [26] N. Molino, J. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:385–392, 2004.
- [27] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable*, pages 103–114, 2003.
- [28] M. Muller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Comput. Anim. and Sim. '01*, Proc. Eurographics Workshop, pages 99–111. Eurographics Assoc., 2001.
- [29] V. Ng-Thow-Hing and E. Fiume. Application-specific muscle representations. In W. Sturzlinger and M. McCool, editors, *Proc. of Gr. Inter. 2002*, pages 107–115. Canadian Information Processing Society, 2002.
- [30] J. O’Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH 99*, volume 18, pages 137–146, 1999.
- [31] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY.
- [32] M. Pandy. Moment arm of a muscle force. *Ex. Sprt. Sci. Rev.*, 27, 1999.
- [33] G. Pappas, D. Asakawa, S. Delp, F. Zajac, and J. Drace. Nonuniform shortening in the biceps brachii during elbow flexion. *J. Appl. Physio.*, 92(6), 2002.
- [34] R. Schachar, W. Herzog, and T. Leonard. Force enhancement above the initial isometric force on the descending limb of the force/length relationship. *J. Biomech.*, 35, 2002.
- [35] F. Scheepers, R. Parent, W. Carlson, and S. May. Anatomy-based modeling of the human musculature. *Comput. Graph. (SIGGRAPH Proc.)*, pages 163–172, 1997.
- [36] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.*, 93:1591–1595, 1996.
- [37] J. Teran, S. Blemker, V. Ng, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 68–74, 2003.
- [38] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Comput. Graph. (SIGGRAPH Proc.)*, pages 269–278, 1988.
- [39] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Comput. Graph. (Proc. SIGGRAPH 87)*, 21(4):205–214, 1987.
- [40] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*, 40:1528–1538, 1995.
- [41] U.S. National Library of Medicine. The visible human project, 1994. <http://www.nlm.nih.gov/research/visible/>.
- [42] J. Weiss, B. Maker, and S. Govindjee. Finite-element implementation of incompressible, transversely isotropic hyperelasticity. *Comput. Meth. in Appl. Mech. and Eng.*, 135:107–128, 1996.

- [43] J. Wilhelms and A. Van Gelder. Anatomically based modeling. *Comput. Graph. (SIGGRAPH Proc.)*, pages 173–180, 1997.
- [44] C. A. Yucesoy, B. H. Koopman, P. A. Huijing, and H. J. Grootenboer. Three-dimensional finite element modeling of skeletal muscle using a two-domain approach: Linked fiber-matrix mesh model. *J. of Biomech.*, 35:1253–1262, 2002.
- [45] F. Zajac. Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control. *Critical Reviews in Biomed. Eng.*, 17(4):359–411, 1989.
- [46] Q. Zhu, Y. Chen, and A. Kaufman. Real-time biomechanically-based muscle volume deformation using FEM. *Comput. Graph. Forum*, 19(3):275–284, 1998.