ALGORITHMIC ASPECTS OF THE SIMULATION AND CONTROL OF COMPUTER GENERATED HUMAN ANATOMY MODELS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Eftychios D. Sifakis
June 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ronald Fedkiw) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Leonidas Guibas)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Matthew West)

Approved for the University Committee on Graduate Studies.

Preface

Computer aided simulation of the appearance and function of the human body has found compelling applications in entertainment, biomechanics and medicine. Furthermore, recreating realistic humanlike behavior with a synthetic face or body model constitutes one of the most challenging benchmarks for physics-based simulation algorithms, due to the complexity and resolution of the utilized models. As a result, the quest for visual realism and biomechanical accuracy in virtual human simulation has often inspired novel algorithms with broader impact in the field of physics-based simulation. Using realistic character animation as the underlying motivation, this thesis describes a host of new techniques that helped bring physics-based face and body models to life. These include a robust quasistatic finite element solver able to simulate meshes with over a million elements in the presence of inverted or degenerate elements, an optimization algorithm for the automatic extraction of facial muscle activations from motion captured performances, a hybrid solids simulation framework that allows the utilization of distinct representations for elastic simulation, collision handling and constraint resolution, and a flexible geometric algorithm for placing cracks and incisions on deformable structures during simulation. Geometrical and constitutive modeling of active musculature is addressed for musculoskeletal and facial simulation tasks. Finally, the usability of these algorithms and models is illustrated both in human anatomy simulation scenarios as well as in more general physics-based simulation tasks for computer graphics applications.

Acknowledgements

The support, compassion and unconditional love of my fiancée, Demetra Makris, have been a constant source of strength and inspiration for me. From the very beginning, she embraced my hopes and dreams, personal as well as professional, and made them her own. It is only appropriate that I dedicate this degree to her. My parents, Dimitrios and Irini, spared no sacrifice to avail me and my sister of the best education possible and I know this accomplishment fills them with pride. They are my role models and I want to thank them for making all of this possible.

My advisor, Ron Fedkiw, had a profound impact on my attitude towards research and life alike. He taught me not to compromise my dreams and goals for the fear of the pain that goes into realizing them. I am honored to have had him as my mentor, and my friend. I would also like to thank the other members of my reading committee and PhD oral committee, Leo Guibas, Sebastian Thrun, Matthew West and Sandy Napel, for their valuable feedback.

During my time as a PhD student at Stanford, I was privileged to work with a number of extremely talented people. My interaction with my co-authors Joey Teran, Geoffrey Irving, Andrew Selle, Igor Neverov, Tamar Shinar, Kevin Der, Avi Robinson-Mosher, Padma Sundaram, Cynthia Lau and Sylvia Salinas-Blemker was absolutely inspirational. I am also grateful to Robert Bridson, Jiayi Chong, Frederic Gibou, Eran Guendelman, Jeong-Mo Hong, Sergey Koltakov, Ranjitha Kumar, Nipun Kwatra, Frank Losasso, Neil Molino, Duc Ngyuen, Robert Strzodka, Jonathan Su, Jerry Talton, Michael Turitzin, Rachel Weinstein for making Ron Fedkiw's group such an exciting place work in!

In conjunction with my studies at Stanford, I was given the opportunity to collaborate with Intel's Applications Research Lab and work with the following talented people: Radek Grzeszczuk, Chris Hughes, Daehuyn Kim, Sanjeev Kumar, Y.K. Chen, Jatin Chuugani, Pradeep Dubey and Jim Hurley. I am grateful to all of them for making this collaboration enjoyable and productive and for educating me in the computer architect's perspective of physics-based simulation.

This research would not have been possible without the aid of companies such as Cyberware (Monterey, CA), XYZrgb (Ottawa, Canada) and Motion Analysis (Santa Rosa, CA). Their help and dedication to our facial modeling project truly went above and beyond. I am grateful to our collaborators from the medical field, including Court Cutting, Aaron Oliker, Garry Gold, Gary Glover, Sabine Girod and Sam Most for lending us their unique insight in the function and structure of the human face, as well as for their guidance in pursuing real-world applications that address important needs in surgery and medicine.

Finally, I wish to thank SAP Labs for sponsoring the Stanford Graduate Fellowship that supported my work for the majority of my time at Stanford.

Contents

List of Figures

Chapter 1

Introduction

Simulation of virtual characters is becoming an essential component of many motion pictures, while emerging applications in biomechanics, medicine and surgery highlight the value of accurate simulation of human anatomy. Innovative simulation algorithms and modern hardware have recently led to leaping advances in the quality of physics-based simulation, allowing applications in computer graphics and visual effects a level of realism previously unattainable with finite element simulation techniques. Moreover, the ability to combine variable-resolution anatomical geometry with accurate constitutive models has unveiled the potential for next-generation surgical planning and training tools, which could combine the usability of an interactive virtual surgery environment with the predictive power of an accurate physics-based simulator.

Beyond the practical value of physics-based human anatomy simulation technology, the pursuit of such tasks often motivates the development of novel simulation techniques. Anatomical models of human bodies and faces require discretizations with hundreds of thousands, or even millions of degrees of freedom to resolve the complex action of musculature, the wrinkling and folding of skin and the forces that determine skeletal motion. Beyond the large number of degrees of freedom, models of human anatomy exhibit substantial material inhomogeneity and anisotropy, incorporate elements with significant impact on flesh deformation despite their small size (such as thin muscle sheets, tendons and connective tissue) and are subject to elaborate collision and contact constraints. Traditional simulation techniques that perform satisfactorily for simpler, non-actuated physical objects are often proven inadequate for simulating these elaborate muscle-driven anatomical models. This thesis presents a survey of novel simulation techniques that were developed to address the unique challenges of virtual human simulation. Although the exposition employs the photo-realistic simulation of facial motion and speech as the central benchmark, examples are given to demonstrate the applicability of the developed techniques to a broader spectrum of physics-based simulation tasks.

This thesis is structured as follows. Chapter 2 describes the geometrical and constitutive modeling process for musculoskeletal and facial simulation systems [167, 24] and presents a formalization of the finite element simulation framework employed throughout this thesis. Chapter 3 presents a method that brings the simulation of tetrahedralized models with millions of elements into the realm of feasibility. Operating under a quasistatic assumption (with the physical accuracy limitations it entails) this novel static solver [166] is shown to handle complex models robustly even in the presence of inverted or degenerate elements. In chapter 4 this technology is used in an optimization framework [155] that solves the inverse problem of determining the muscle activations that give rise to a motion captured facial expression. Such muscle activation signals obtained from speech samples are used in chapter 5 to synthesize new speech animations, which can be augmented by editing the apparent emotion or through interaction with objects of the environment [156]. The practical restriction of a quasistatic simulation is alleviated in chapter 6 where an embedded simulation framework [157] is used to decouple elasticity simulation from collision detection and attachment handling, reducing the required degrees of freedom to levels that enable full dynamic simulation. Applications of this hybrid simulation in general physical simulation tasks are also illustrated, including the ability to simulate composite systems with rigid and deformable components. An additional use of this framework is highlighted in chapter 7 where an embedded simulation is used to animate the processes of cutting and fracture [154] and specific applications in computer aided surgery are discussed.

Chapter 2

Muscles

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.

2.1 Introduction

Simulation of anatomically realistic musculature and flesh is critical for many disciplines including biomechanics, biomedical engineering and computer graphics where it

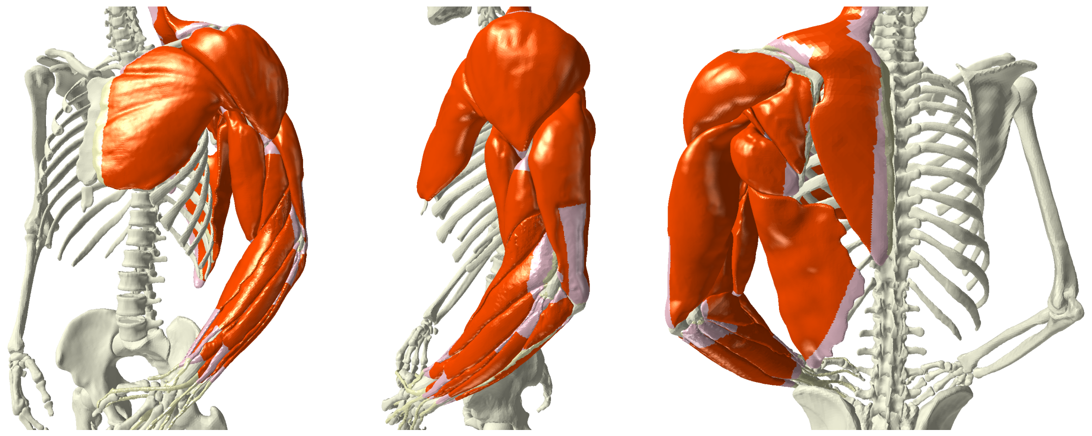

Figure 2.1: Musculoskeletal model created from the visible human data set.

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 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 [136]. For example, many studies use knowledge of muscle lengths [6] and moment arms [7] 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, [111] maps joint configurations to skin deformers that procedurally warp the surface of the character. The work in [188] and [151] 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 [41, 80, 165], 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 [41, 165], computational complexity restricted the application of their techniques to only a few muscles at a time. [80] 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 [52, 66]. 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 [192] and [67] 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 [165] 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.

2.2 Related Work

[170, 169] 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, [72] 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.

[41] used the FEM, brick elements, and the constitutive model of [193] to simulate a few muscles including a human bicep. Due to computational limitations at the time, very few elements were used in the simulation. [188] 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. [151] 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. [196, 80, 54, 60].

In order to increase the computational efficiency, a number of authors have been investigating adaptive simulation. [48] used a finite difference method with an octree for adaptive resolution. This was later improved in [50] 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. [49] used FEM with a multiresolution hierarchy of tetrahedral meshes, and [74] refined basis functions instead of elements.

2.3 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 [178]. 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. [135]), 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 [113].

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.

2.3.1 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 [177, 153].

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. [135]) to eliminate any further noise automatically.

2.3.2 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 [113]. 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.

2.3.3 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 2.2). 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 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

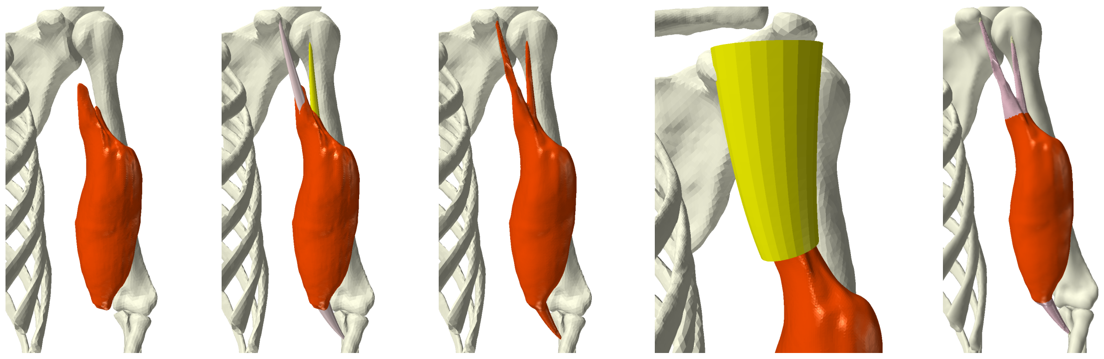

Figure 2.2: Musculotendon mesh creation using Constructive Solid Geometry.

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 [137, 63]. 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 2.3.

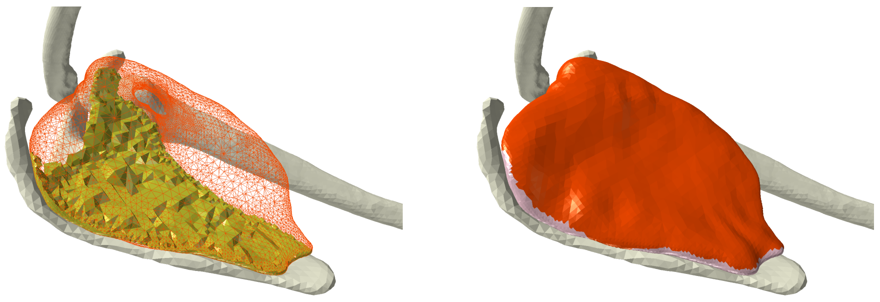

Figure 2.3: Bone attachment specification for the subscapularis and scapula.

2.3.4 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 [126].

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 [126], 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 [126] or using fiber information from scanning technologies would improve accuracy. Additionally, using a fiber primitive template as was done in [22] would also improve accuracy and simplify the process.

2.3.5 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 [65] to create the kinematic structure of the upper limb. In [65] 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 [66] to create a muscle model in the upper limb based on the Obstacle Set method for computing musculo-tendon paths, see figure 2.4 (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 and other tissues. With this infrastructure in place, we use an inverse dynamics analysis with the results of [165] to compute activations for the muscles in the right upper limb. These techniques work with both motion capture and traditional animation.

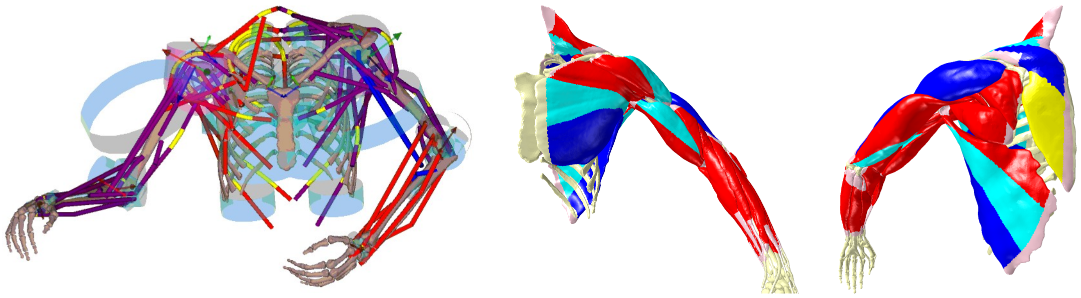

Figure 2.4: Piecewise linear muscle models with wrapping surfaces.

2.4 Finite element models

2.4.1 Hyperelasticity

Consider a volumetric object occupying a region $\Omega \in R^N$ in its rest state. We model an elastic deformation of this object as a function $\phi : \Omega \to R^N$ which maps the undeformed object into its deformed state $\phi[\Omega]$. We denote the independent variable of the mapping function ϕ by $\mathbf{X} \in \Omega$ and associate it with the rest position of a material point. The respective deformed position is denoted by $\mathbf{x} = \phi(\mathbf{X}) \in \phi[\Omega]$.

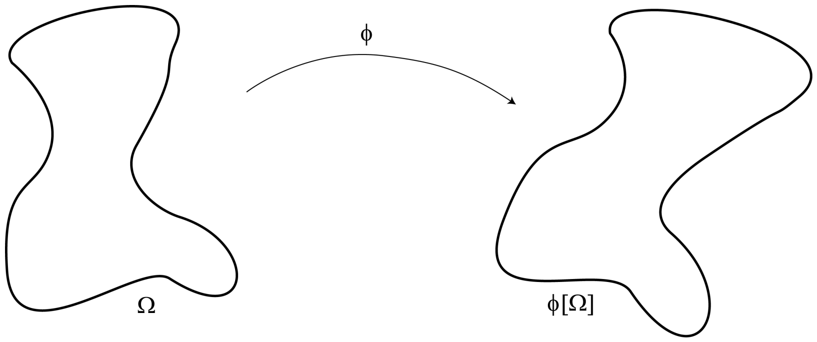

Figure 2.5: Elastically deforming solid

A deformed *hyperelastic* material stores potential energy, which is defined via a constitutive law as $\hat{\Psi} = \hat{\Psi}(\phi[\cdot]; \mathbf{X})$. $\hat{\Psi}(\mathbf{X})$ corresponds to the hyperelastic energy *density* at $\mathbf{X}$, i.e. it is the ratio of the energy stored in an infinitesimal volumetric region $d\Omega$ around $\mathbf{X}$ normalized by the volume dV of $d\Omega$. Consequently, the total energy stored in a deformed hyperelastic object is

$$\Psi = \int_\Omega \hat{\Psi}(\mathbf{X}) d\mathbf{X}. \tag{2.1}$$

The specific formula for the energy density $\hat{\Psi}$ is the defining property of different materials. However, the nature of a hyperelastic material mandates a number of important properties for the energy density function:

- $\hat{\Psi}(\mathbf{X})$ depends only on the local properties of the deformation map ϕ around $\mathbf{X}$. For smooth functions the dependence of $\hat{\Psi}(\mathbf{X})$ on ϕ reduces to the Taylor expansion of the latter around $\mathbf{X}$. In typical constitutive models (such as all the models addressed in this thesis) $\hat{\Psi}(\mathbf{X})$ only depends on the local linearization of ϕ around $\mathbf{X}$, i.e. the values of $\phi(\mathbf{X})$ and $\partial\phi(\mathbf{X})/\partial\mathbf{X}$.

- The hyperelastic energy density is independent of the deformation history. As a result, hyperelastic forces are conservative, i.e. the energy difference between two deformed states is independent of the path that was followed to transition from one to the other.

- The energy density is zero at locations where ϕ is locally a rigid body transformation and nonnegative everywhere else. Consequently, rigid body transformations of a deformable object (including the rest state) are global minima of the total hyperelastic energy.

In analogy with equation 2.1 the total momentum $\mathbf{P}$ and kinetic energy K of the deformable object are given by

$$\mathbf{P} = \int_\Omega \rho(\mathbf{X}) v(\mathbf{X}) d\mathbf{X}, \quad \text{and} \quad K = \int_\Omega \rho(\mathbf{X}) \|v(\mathbf{X})\|^2 d\mathbf{X} \tag{2.2}$$

2.4.2 Linear finite elements

As a first step to modeling an elastically deforming object on the computer one has to discretize its state into a finite-dimensional representation. The techniques presented in this thesis utilize a linear finite element discretization on simplicial meshes. A formal definition of a Lagrangian FEM discretization using linear basis functions is given in [25] while [167] illustrates the equivalence of this discretization with the Finite Volume Method for the case of constant strain elements. Here, we present an equivalent intuitive definition which is best adapted to the formulations in the remaining of this work.

The continuous geometry of a volumetric deformable object is discretized into a tetrahedral mesh via a mesh-generation process. The degrees of freedom of this discrete deformable geometry are identified with the nodes of the tetrahedral mesh. Thus the deformation map ϕ is constrained to satisfy $\mathbf{x}_i = \phi(\mathbf{X}_i)$ for every node $\mathbf{X}_i$ in the rest state of the deformable mesh and its counterpart $\mathbf{x}_i$ in the deformed mesh. In addition the deformation map ϕ is defined to be affine within every tetrahedron of the mesh.

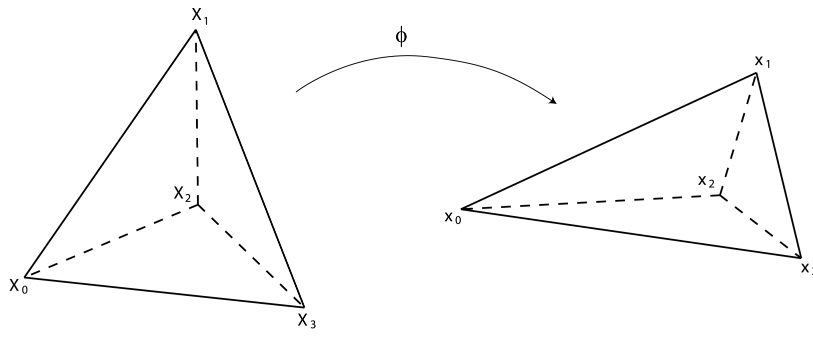

Figure 2.6: Action of an affine deformation map on a tetrahedral element

These conditions are sufficient to fully determine the deformation function from the positions $\mathbf{x}_i$ of the deformed mesh nodes. Within each tetrahedron $T(\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3)$ the deformation map is an affine function $\mathbf{x} = \phi(\mathbf{X}) = \mathbf{F}\mathbf{X} + \mathbf{t}$. The 3×3 matrix $\mathbf{F} = \partial x/\partial X$ is called the *deformation gradient* and is constant within each tetrahedron.

It is computed as:

$$\begin{aligned}\mathbf{x}_i - \mathbf{x}_0 &= (\mathbf{F}\mathbf{X}_i + \mathbf{t}) - (\mathbf{F}\mathbf{X}_0 + \mathbf{t}) \\ &= \mathbf{F}(\mathbf{X}_i - \mathbf{X}_0)\end{aligned}$$

$$\Big(\mathbf{x}_1 - \mathbf{x}_0 \;\mid\; \mathbf{x}_2 - \mathbf{x}_0 \;\mid\; \mathbf{x}_3 - \mathbf{x}_0 \Big) = \mathbf{F} \Big(\mathbf{X}_1 - \mathbf{X}_0 \;\mid\; \mathbf{X}_2 - \mathbf{X}_0 \;\mid\; \mathbf{X}_3 - \mathbf{X}_0 \Big)$$

or $\mathbf{D}_s = \mathbf{F}\mathbf{D}_m$ where $\mathbf{D}_m = (\mathbf{X}_1 - \mathbf{X}_0, \mathbf{X}_2 - \mathbf{X}_0, \mathbf{X}_2 - \mathbf{X}_0)$ and $\mathbf{D}_s = (\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0)$ are the *material* (undeformed) and *spatial* (deformed) 3×3 shape matrices for the tetrahedron T. The deformation gradient can subsequently be computed as $\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1}$, while the translation $\mathbf{t}$ can be computed from any of the relations $\mathbf{x}_i = \mathbf{F}\mathbf{X}_i + \mathbf{t}$.

Since $\mathbf{F}$ and $\mathbf{t}$ fully define the deformation map ϕ within each tetrahedron, the energy density function $\hat{\Psi}$ is fully defined as a function of these. Furthermore, since the elastic energy is invariant under rigid body transformations, $\hat{\Psi}$ does not depend on the translation $\mathbf{f}$ and can be written solely as a function of the deformation gradient $\hat{\Psi} = \hat{\Psi}(\mathbf{F})$. We note that due to the invariance of hyperelastic energy under rigid rotation, the energy density function satisfies $\hat{\Psi}(\mathbf{F}) = \hat{\Psi}(\mathbf{R}\mathbf{F})$ for any rotation matrix $\mathbf{R}$. Finally, due to the deformation gradient $\mathbf{F}$ being constant across T we can compute the hyperelastic energy stored in the deformed tetrahedron $\phi[H!]$ from equation (2.1):

$$\begin{aligned}\Psi_T &= \int_T \hat{\Psi}(\mathbf{F})d\mathbf{X} = \frac{1}{6}\det\mathbf{D}_m\hat{\Psi}(\mathbf{F}_T) \\ &= b_T\Psi(\mathbf{F}_T) \qquad (2.3)\end{aligned}$$

where in equation (2.3) $b_T = \frac{1}{6}\det\mathbf{D}_m$ is the volume of the undeformed tetrahedron and the simplified notation $\Psi(\mathbf{F})$ is used to denote the energy density associated with a deformation gradient $\mathbf{F}$ (Ψ_T will refer to the total energy stored in tetrahedron T).

Finally, the piecewise affine nature of the deformation map ϕ implies that the kinematic properties of any point within a tetrahedron can be computed via barycentric averaging. Specifically, if the undeformed position $\mathbf{X}$ of a point interior to the tetrahedron $(\mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3)$ yields the barycentric coordinates w_1, w_2, w_3 such that

$$\begin{aligned}
\mathbf{X} &= (1 - w_1 - w_2 - w_3)\mathbf{X}_0 + w_1\mathbf{X}_1 + w_2\mathbf{X}_2 + w_3\mathbf{X}_3 \\
&= \mathbf{X}_0 + w_1(\mathbf{X}_1 - \mathbf{X}_0) + w_2(\mathbf{X}_2 - \mathbf{X}_0) + w_3(\mathbf{X}_3 - \mathbf{X}_0) \\
\mathbf{F}\mathbf{X} + \mathbf{t} &= \mathbf{F}\mathbf{X}_0 + \mathbf{t} + w_1\mathbf{F}(\mathbf{X}_1 - \mathbf{X}_0) + w_2\mathbf{F}(\mathbf{X}_2 - \mathbf{X}_0) + w_3\mathbf{F}(\mathbf{X}_3 - \mathbf{X}_0) \\
\mathbf{x} &= \mathbf{x}_0 + w_1(\mathbf{x}_1 - \mathbf{x}_0) + w_2(\mathbf{x}_2 - \mathbf{x}_0) + w_3(\mathbf{x}_3 - \mathbf{x}_0) \\
&= (1 - w_1 - w_2 - w_3)\mathbf{x}_0 + w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + w_3\mathbf{x}_3
\end{aligned}$$

thus the deformed position of the interior point is interpolated from the deformed positions of the tetrahedron nodes using the barycentric weights computed from its undeformed position. Differentiating this equation, we see that the velocity and acceleration of an interior point is interpolated from the respective velocities and accelerations of the tetrahedron nodes using the same barycentric weights.

2.4.3 Mass discretization and lumping

The ability to define the velocity of any point interior to an element via barycentric interpolation from the velocities of the element nodes allows us to write the kinetic energy of the element from equation (2.2) in closed form. For simplicity, we consider a deformable object in two dimensions, which has been discretized into a triangle mesh. Using the relation $v = \sum w_i v_i$ we write the kinetic energy of a triangle $T(\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3)$ as

$$\begin{aligned}
K &= \int_T \rho \|v(\mathbf{X})\|^2 d\mathbf{X} \\
&= \frac{1}{2}\begin{pmatrix} v_1^T & v_2^T & v_3^T \end{pmatrix}\begin{pmatrix} M/6 & M/12 & M/12 \\ M/12 & M/6 & M/12 \\ M/12 & M/12 & M/6 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \\
&= \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}
\end{aligned}$$

where the velocity vector $\mathbf{v}$ is the concatenation of the nodal velocities v_1, v_2, v_3 and $\mathbf{M}$ is a symmetric, positive semi-definite mass matrix. The density ρ is assumed constant throughout each element. Although this representation of mass yields an accurate measure for the kinetic energy of the element, it gives rise to an important algorithmic complication: Computation of nodal accelerations by application of Newton's law $\mathbf{a} = \mathbf{M}^{-1}\mathbf{f}$ would require inversion of the mass matrix. The typical remedy is to *lump* the mass matrix into a diagonal matrix by redistributing the continuous mass of the elements into discrete masses at the nodes of the simulation mesh. This is accomplished by computing the total mass of each element and distributing it equally among the vertices of the element. In the example above, this would yield the lumped diagonal mass matrix $\hat{\mathbf{M}} = \mathrm{diag}(M/3, M/3, M/3)$. Consequently, the global mass matrix is diagonal as is its inverse which can be precomputed prior to simulation.

Mass lumping induces a perturbation on FEM forces in comparison to using non-lumped symmetric mass matrices, however this perturbation vanishes under refinement and is perfectly acceptable for the anatomical modeling applications described in this thesis, where they are easily outweighed even by subtle geometrical or constitutive modeling inaccuracies. Notably, both the lumped and non-lumped mass matrix yield the same value of the kinetic energy when all the vertices of the element share the same velocity. Likewise, the total momentum (computed by adding the nodal momenta $\mathbf{p} = \mathbf{M}\mathbf{v}$) of the triangle is accurately measured when using either mass matrix, for any value of the nodal velocities (a consequence of the fact that momentum, in contrast with kinetic energy, has a linear dependence on velocity).

2.4.4 Force computation

Given a deformable object with nodal positions $\mathbf{x}$ and nodal velocities $\mathbf{v}$ we can derive the definition of hyperelastic forces $\mathbf{f}$ via application of the conservation of energy. In the absense of damping or external forces, the total energy of the object is the sum of the kinetic (K) and potential (Ψ) energy and remains constant. Therefore

$$\begin{aligned} K + \Psi &= const \\ K_t + \Psi_t &= 0 \\ \frac{d}{dt}\left(\frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}\right) + \frac{\partial\Psi}{\partial\mathbf{x}}\frac{d\mathbf{x}}{dt} &= 0 \end{aligned}$$

$$
\begin{aligned}
\mathbf{v}^T\mathbf{M}\mathbf{a} + \mathbf{v}^T\left(\frac{\partial \Psi}{\partial \mathbf{x}}\right)^T &= 0 \\
-\left(\frac{\partial \Psi}{\partial \mathbf{x}}\right)^T &= \mathbf{M}\mathbf{a} = \mathbf{f}
\end{aligned}
$$

This definition of force as the negative gradient of potential energy is characteristic of a conservative force.

Although forces may be computed via direct evaluation of $\mathbf{f} = -\nabla^T\Psi$ on a per-node basis, it is computationally beneficial to perform this evaluation on a per-element basis instead. This is facilitated by introducing for each tetrahedron $T(\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3)$ the force matrix $\mathbf{G} = (\mathbf{f}_1\ \mathbf{f}_2\ \mathbf{f}_3)$. Note that the force on $\mathbf{x}_0$ can be computed as $\mathbf{f}_0 = -\mathbf{f}_1 - \mathbf{f}_2 - \mathbf{f}_3$ since the internal elastic forces cannot affect the total momentum of the element, thus must sum to zero on each respective element. $\mathbf{G}$ can be computed as

$$
\begin{aligned}
\mathbf{G} &= \begin{pmatrix} -(\partial\Psi_T/\partial\mathbf{x}_1)^T & -(\partial\Psi_T/\partial\mathbf{x}_2)^T & -(\partial\Psi_T/\partial\mathbf{x}_3)^T \end{pmatrix} \\
&= -\frac{\partial \Psi_T}{\partial(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)} \\
&= -\frac{\partial \Psi_T}{\partial(\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0, \mathbf{x}_3 - \mathbf{x}_0)} = -\frac{\partial \Psi_T}{\partial \mathbf{D}_s} \qquad (2.4) \\
&= -\frac{\partial \Psi_T}{\partial \mathbf{F}} : \frac{\partial \mathbf{F}}{\partial \mathbf{D}_s} \\
&= -b_T \frac{\partial \Psi(\mathbf{F}_T)}{\partial \mathbf{F}} : \frac{\partial \mathbf{F}}{\partial \mathbf{D}_s} \qquad (2.5)
\end{aligned}
$$

where equation (2.4) is due to the translational invariance of the hyperelastic energy. Ψ_T denotes the total energy stored in tetrahedron T while $\Psi(\mathbf{F})$ is the energy density associated with a deformation gradient $\mathbf{F}$. The fourth order tensor $\partial\mathbf{F}/\partial\mathbf{D}_s$ has entries $[\partial\mathbf{F}/\partial\mathbf{D}_s]_{ijkl} = \delta_{ik}(\mathbf{D}_m^{-T})_{jl}$, thus after some tensor algebra equation (2.5) becomes

$$
\mathbf{G} = -b_T \left.\frac{\partial \Psi}{\partial \mathbf{F}}\right|_{\mathbf{F}_T} \mathbf{D}_m^{-T} = \mathbf{P}(\mathbf{F}_T)\mathbf{B}_m \qquad (2.6)
$$

where $\mathbf{B}_m = (1/6)\det\mathbf{D}_m \cdot \mathbf{D}_m^{-T}$ and $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$ is the *First Piola-Kirchhoff stress* *tensor*. Since by virtue of equation (2.6) nodal forces may be computed using only the first Piola-Kirchhoff stress tensor and the precomputed $\mathbf{B}_m$ matrix, constitutive laws are often formulated as a relation that defines $\mathbf{P}$ (instead of the energy density Ψ) as a function of $\mathbf{F}$.

2.4.5 Isotropic materials

As noted in section 2.4.1 any hyperelastic constitutive model will not be affected by spatial rotations of the deformation gradient, that is $\Psi(\mathbf{RF}) = \Psi(\mathbf{F})$ for any rotation matrix $\mathbf{R}$. We can isolate the subset of degrees of freedom within $\mathbf{F}$ that affect the energy Ψ by considering the Singular Value Decomposition of the deformation gradient $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are rotation matrices. In this setting, the degrees of freedom of the deformation gradient are factored into three degress of freedom for each of the rotations $\mathbf{U}$ and $\mathbf{V}$ and three degrees of freedom in the diagonal matrix $\mathbf{\Sigma}$.

Isotropic materials carry the additional property that the hyperelastic energy Ψ does not depend on either of the material ($\mathbf{V}$) or spatial ($\mathbf{U}$) rotations, but only on the singular values of the deformation gradient ($\mathbf{\Sigma}$). Such materials do not have any preferred directions and a rotation of the deformation function simply leads to the same deformation of resulting elastic forces. The independence of isotropic materials on spatial or material rotations can be formalized as

$$\Psi(\mathbf{R}_1\mathbf{F}\mathbf{R}_2) = \Psi(\mathbf{F}) \tag{2.7}$$

for any rotation matrices $\mathbf{R}_1, \mathbf{R}_2$. Equation (2.7) can be regarded as the definition of an isotropic constitutive model.

Although the constitutive law for an isotropic material could express the energy as a function of the singular values $\mathbf{\Sigma}$ of the deformation gradient (and this form of definitions is used in some cases) it is often more convenient to express it in terms of algebraic quantities that can readily be computed from $\mathbf{F}$ without explicit computation of the SVD. Such algebraic quantities must be constant when evaluated over the family of matrices $\{\mathbf{R}_1\mathbf{F}\mathbf{R}_2\}$ where $\mathbf{F}$ is a given 3×3 matrix and $\mathbf{R}_1, \mathbf{R}_2$ are rotations.

Consequently such quantities are known as *isotropic invariants.* The three isotropic invariants most commonly in use are:

$$I_1(\mathbf{F}) = \mathrm{tr}\left(\mathbf{F}^T\mathbf{F}\right), \quad I_2(\mathbf{F}) = \mathrm{tr}\left(\mathbf{F}^T\mathbf{F}\mathbf{F}^T\mathbf{F}\right), \quad I_3(\mathbf{F}) = \det\left(\mathbf{F}^T\mathbf{F}\right)$$

The invariance of these tensors is easily verified. Although they can be computed from the deformation gradient algebraically, they can also be expressed in terms of its singular values (by simply substituting $\mathbf{\Sigma}$ for $\mathbf{F}$, using the invariance property) as:

$$I_1 = \sum \sigma_i^2, \quad I_2 = \sum \sigma_i^4, \quad I_3 = \prod \sigma_i^2$$

If an isotropic constitutive model is defined in terms of the isotropic invariants I_1, I_2, I_3, the first Piola-Kirchhoff stress can be computed as $\mathbf{P} = \partial\Psi/\partial\mathbf{F} = \sum \Psi_{I_k}\partial I_k/\partial\mathbf{F}$. The derivatives of the three invariants with respect to the deformation gradient are

$$\frac{\partial I_1}{\partial \mathbf{F}} = 2\mathbf{F}, \quad \frac{\partial I_2}{\partial \mathbf{F}} = 4\mathbf{F}\mathbf{F}^T\mathbf{F}, \quad \frac{\partial I_3}{\partial \mathbf{F}} = 2I_3\mathbf{F}^{-T}$$

leading to the final expression for the first Piola-Kirchhoff stress

$$\mathbf{P} = 2\Psi_{I_1}\mathbf{F} + 4\Psi_{I_2}\mathbf{F}\mathbf{F}^T\mathbf{F} + 2I_3\Psi_{I_3}\mathbf{F}^{-T} \tag{2.8}$$

In the remaining of this thesis the symbols I_1, I_2, I_3 will be used interchangeably with I, II, III where notational convenience favors the latter. Two additional invariants that are used in certain constitutive models are

$$I_2^* = \frac{1}{2}\left(I_1^2 - I_2\right) = \sum_{i<j} \sigma_i^2\sigma_j^2 \quad \text{and} \quad J = \sqrt{I_3} = \det\mathbf{F} = \prod \sigma_i$$

Notably, J is the volume change ratio ($J > 1$ for expanded elements, $J < 1$ for compressed ones). Locally incompressible materials satisfy $J = 1$ at all times.

Certain constitutive models choose to separate the hyperelastic energy due to volumetric compression or expansion from the energy due to deformation modes orthogonal to volume change. In the interest of this separation we define the *deviatoric* invariants of $\mathbf{F}$ as the respective isotropic invariants evaluated on the scaled deformation gradient $\hat{\mathbf{F}} = J^{-1/3}\mathbf{F}$ which satisfies $J(\hat{\mathbf{F}}) = \det \hat{\mathbf{F}} = 1$. The expressions for the deviatoric counterparts of I_1 and I_2 are

$$\hat{I}_1(\mathbf{F}) = I_1(\hat{\mathbf{F}}) = J^{-2/3} I_1 \quad \text{and} \quad \hat{I}_2(\mathbf{F}) = I_2(\hat{\mathbf{F}}) = J^{-4/3} I_2.$$

Constitutive models that separate the dependence of the elastic energy Ψ on the deviatoric invariants $\hat{I}_1, \hat{I}_2$ and the volume change J are used extensively in this thesis where the approach of quasi-incompressibility is used to promote volume preservation.

2.4.6 Invertible Finite Elements

Motivated by our geometric FVM formulation, [82] 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, [82] 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 **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.

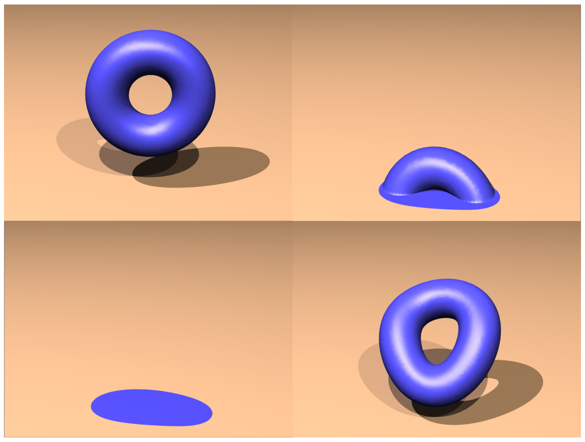

Figure 2.7: Deformable torus simulated with the inverting FVM.

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 2.7.

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 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.

2.5 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

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

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 fasicles 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 [193]. This strain energy function is based on [186] and is the same as that used in [165].

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 [193, 150]. 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 [137], 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 [23].

The diagonalized FEM framework of [82] 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

$$\begin{aligned}
\mathbf{P} &= w_{12}\mathbf{F} - w_2\mathbf{F}^3 + (p - p_f)\mathbf{F}^{-1} + 4J_{cc}T(\mathbf{F}\mathbf{f}_m){\mathbf{f}_m}^T \\
J_c = det(\mathbf{F})^{-\frac{1}{3}}, J_{cc} &= J_c^2, I_1 = J_{cc}\mathbf{C}, \lambda = \sqrt{{\mathbf{f}_m}^T\mathbf{C}\mathbf{f}_m} \\
w_1 = 4J_{cc}mat_{c1}, w_2 &= 4J_{cc}^2 mat_{c2}, w_{12} = w_1 + I_1 W_2 \\
p = Klog(J), p_f &= \frac{1}{3}(w_{12}Tr(\mathbf{C}) - w_2 Tr(\mathbf{C}^2) + T\lambda^2)
\end{aligned}$$

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 [193]). Typical values for these parameters are:

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

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 2.4).

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 [82]. Elongation is identified when the diagonalized deformation gradient values are greater than 1.

2.6 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 [113]). For each particle on the surface of the initial high resolution tetrahedralized volume, 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 2.8) 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 [112].

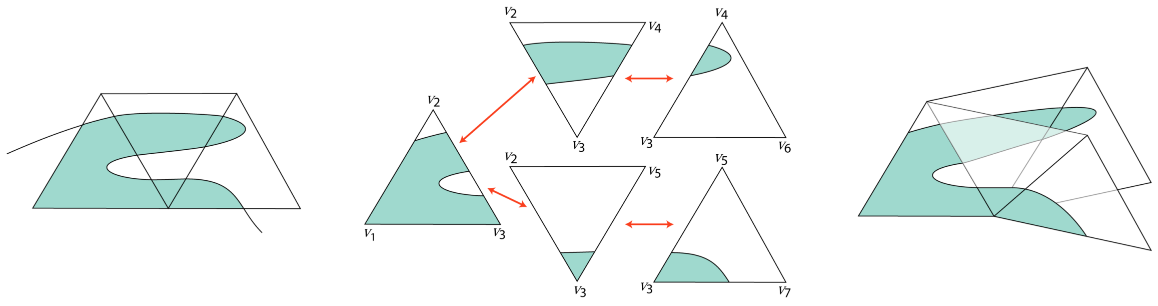

Figure 2.8: Resolution of a topology-offending embedding scenario.

Consider a coverage of our high resolution geometry by a manifold tetrahedral mesh as illustrated in figure 2.8 (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 2.8 (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 with double arrows in figure 2.8 (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 2.8 (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.

2.7 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 2.9. Our fascia model enforces a state of frictionless contact between muscles. It is similar in spirit to [86, 39, 30] which all used "sticky" regions in one sense or another to create (possibly temporary) bonds between geometry in close proximity.

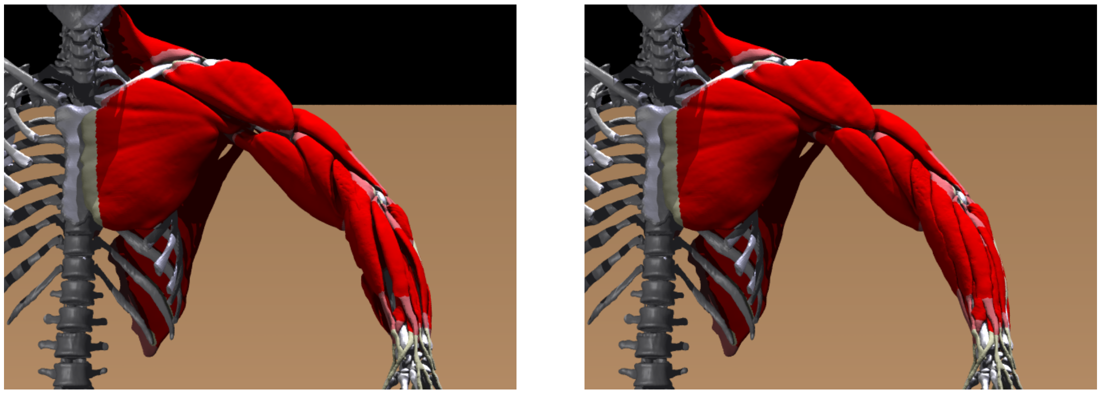

Figure 2.9: Muscle simulations with and without fascia.

The fascia framework works in the context of the embedding techniques presented in section 2.6. 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 for the embedded point:

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

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 [112] for more details.

Figure 2.9 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.

2.8 Simulating Skeletal Muscle

We demonstrate the strength of our pipeline with a series of skeletal animations of the upper limb (see figure 2.10). 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 [165] 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.

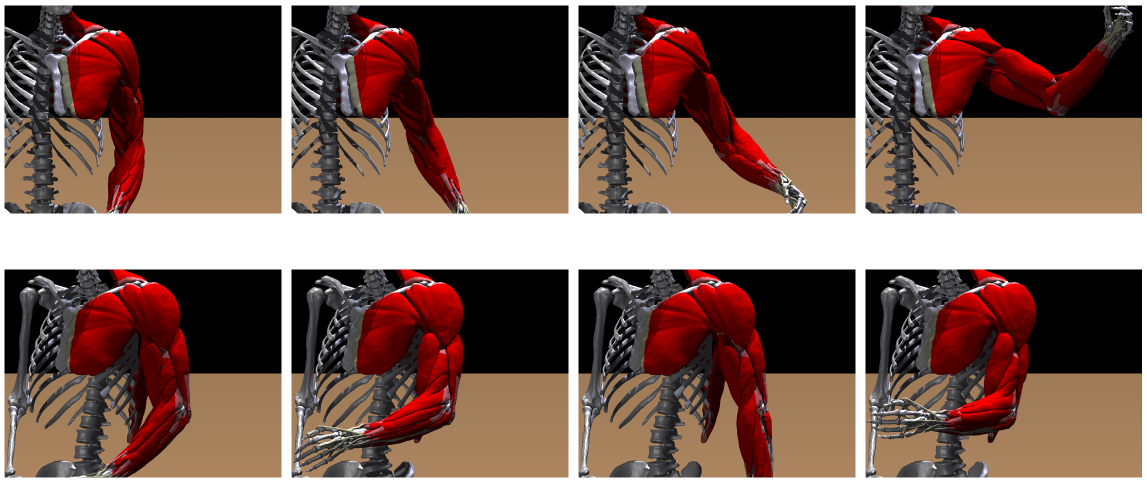

Figure 2.10: Simulation of muscles in the upper limb.

2.9 Summary

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, fasicle 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 [150] as well as anisotropic shear behavior relative to the fiber axis as in [23, 22] 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 [60].

Chapter 3

Quasistatics

Quasistatic and implicit time integration schemes are typically employed to alleviate the stringent time step restrictions imposed by their explicit counterparts. However, both quasistatic and implicit methods are subject to hidden time step restrictions associated with both the prevention of element inversion and the effects of discontinuous contact forces. Furthermore, although fast iterative solvers typically require a symmetric positive definite global stiffness matrix, a number of factors can lead to indefiniteness such as large jumps in boundary conditions, heavy compression, etc. We present a novel quasistatic algorithm that alleviates geometric and material indefiniteness allowing one to use fast conjugate gradient solvers during Newton-Raphson iteration. Additionally, we robustly compute smooth elastic forces in the presence of highly deformed, inverted elements alleviating artificial time step restrictions typically required to prevent such states. Finally, we propose a novel strategy for treating both collision and self-collision in this context.

3.1 Introduction

Fast and robust simulations of elastic solids are becoming increasingly important in computer graphics applications due largely to the prominence of virtual characters. Feature films such as Van Helsing, Spiderman, The Lord of the Rings and countless others benefit from the use of humanoid characters in scenes that would be difficult and expensive if not impossible to create with live actors, see e.g. [92, 162]. Typical models are composed of an underlying skeleton whose motion is prescribed kinematically (from motion capture or traditional animation) and a mechanism for transmitting the skeletal motion to skin deformation. Physics based simulations of musculature and fleshy tissues are becoming increasingly popular for producing these deformations, especially when virtual characters undergo contact and collision with the surrounding environment. Moreover, faithfully depicting the artist's conception of the character requires reasonably high resolution tetrahedral meshes placing additional demands for efficiency on the simulation algorithm.

Since explicit time integration schemes can often have stringent time step restrictions, various authors have investigated the use of semi-implicit (e.g. [30]), fully implicit (e.g. [169, 10]) and quasistatic (e.g. [80, 122]) time integration schemes. Quasistatic schemes ignore inertial effects and thus are not suitable for simulating less constrained phenomena such as ballistic motion. However, in applications where inertial effects are relatively small compared to the deformation caused by contact, collision, and time varying boundary conditions, quasistatic solvers can often provide a speedup of one to two orders of magnitude over explicit schemes. For example, quasistatic simulations are well suited for flesh deformation where the flesh is rigidly attached to bones and heavily influenced by contact, collision and self-collision.

Although implicit and quasistatic schemes remove the time step restriction associated with wave propagation, the Newton-Raphson method used to solve the resulting nonlinear equations may produce inverted elements during iteration when large time steps are used, bringing the algorithm to a halt. For example, large displacement boundary conditions tend to invert elements unless steps are taken to distribute the effects to surrounding elements, and the typical approach is to impose an artificial time step restriction even in the quasistatic case. This has been discussed in both the computer graphics (e.g. [80]) and the computational physics (e.g. [77]) literature. Even in the case where the final mesh will be inversion free, artificially small time steps are required to ensure that every intermediate state considered during Newton-Raphson iteration is also inversion free restricting the speed at which one can converge

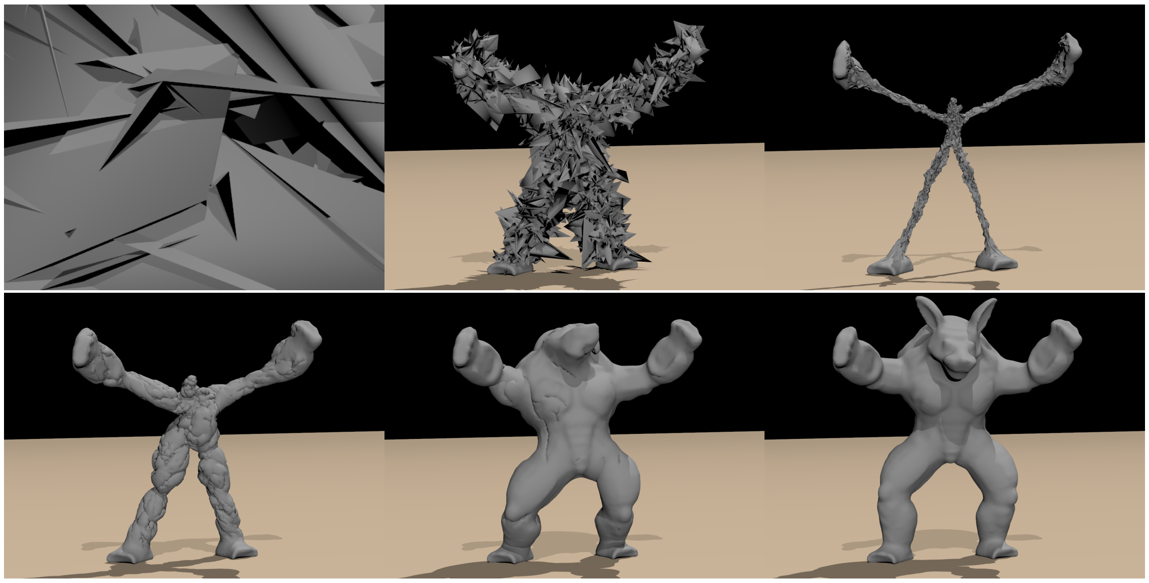

Figure 3.1: Quasistatic evolution of highly distorted model towards equilibrium

to the desired solution. Recently, researchers have aimed at handling inversion using altitude springs [113], volume preservation terms [175], rotated linear models [119], etc. However, these methods change or limit the underlying partial differential equation, whereas [82] allows for general nonlinear constitutive models with forces that are smooth enough to be used in conjunction with iterative methods. Thus, we adopt the approach of [82] and extend it to the quasistatic regime removing the artificial time step restriction required by other schemes making our solution method extremely efficient.

In each Newton-Raphson iteration, the nonlinear system of equations is reduced to a linear system that must be solved to advance to the next iteration. This linear system is guaranteed to be symmetric and positive definite in the vicinity of equilibrium states, enabling the use of fast conjugate gradient solvers. Unfortunately, the use of large time steps produces substantial divergence from a steady state, leading to a symmetric linear system that is often indefinite. State of the art finite element packages such as NIKE3D still use direct solvers such as that proposed in [164], even though such methods are much slower and require considerably more memory than iterative methods. [77] first try a fast iterative solver switching to a slower direct method when it fails. [80] discussed these issues in the context of quasistatic simulation pointing out the erratic behavior of conjugate gradient methods and a preference against direct methods. By adding an artificial "viscosity" to their simulations, they were able to obtain reasonable results with a GMRES iterative scheme. In the context of implicit time integration, [44] pointed out that extra damping forces such as those applied in [10, 181] can help to overcome indefiniteness, but not guarantee it. Furthermore, they point out that this damping degrades the realism of the simulation. Instead, they take a closer look at the problem in the case of springs identifying compression as a source of indefiniteness and proposing a technique to guarantee definiteness in the special case of cloth simulation with springs. A key contribution of our paper is a new and general method for guaranteeing positive definiteness, thus allowing for the use of fast conjugate gradient solvers under all circumstances (including inversion) for arbitrary constitutive models in the finite element framework. Our method modifies the search path followed towards equilibrium without altering the set of equilibrium solutions or the governing equations.

3.2 Previous Work

[170, 169, 168] pioneered deformable models in computer graphics including early work on plasticity and fracture. Finite element simulations have been used to model a hand grasping a ball [72], for virtual surgery [141], fracture [132, 122, 131, 112], etc. Other work includes the adaptive frameworks of [49, 74, 35], the rotation based approaches in [118, 119, 45] (see also [173]), the bending models in [30, 73], the precomputed data driven models of [83], and the point based methods in [121].

The construction of muscles and/or flesh deformation is important for computer graphics characters, and anatomy based modeling techniques of varying resolutions have been applied. [188, 151] used anatomically based models of muscles, tendons and fatty tissue to deform an outer skin layer. [126] fit deformable B-spline solids to anatomic data in order to create volumetric, anisotropic representations of muscles and their internal structures. [1] used a variety of techniques to model a human hand. More biomechanically accurate techniques for muscle simulation were proposed in

[41, 80, 165], and a number of researchers are working to simulate data from the NIH visible human dataset, e.g. [196, 80, 54, 165].

Instead of creating an explicit model for muscle and fatty tissue, one can place an articulated skeleton inside the character skin and formulate correspondences between each vertex on the skin mesh and the various joints in the skeleton. This is typically called enveloping or skinning and can suffer from a number or artifacts especially near joints such as elbows and shoulders. A number of techniques have been proposed to overcome these difficulties, see for example [103, 158, 182, 111]. [2] used these techniques in conjunction with range scan data, and [99] used them to model a human hand. [97] proposed a similar method that used principal component analysis and a library of deformations precomputed with nonlinear static finite element analysis. Although these techniques are fast and do not require one to build an underlying muscle model for each character, they can lead to lower quality results than full finite element simulations. A physically based approach was taken in [84] to add ballistic motion to character skins in otherwise kinematically constructed motions. [34] approaches this problem by embedding the character in a coarse finite element mesh which deforms rigidly with the bones, but obeys a linear finite element model locally to each bone.

3.3 Quasistatic Formulation

Using Newton's second law of motion we can describe the evolution of a deformable body using the equations $\vec{\mathbf{x}}_t = \vec{\mathbf{v}}$ and $\vec{\mathbf{v}}_t = M^{-1}\vec{\mathbf{f}}(t, \vec{\mathbf{x}}, \vec{\mathbf{v}})$ where $\vec{\mathbf{x}}$, $\vec{\mathbf{v}}$ and $\vec{\mathbf{f}}$ denote the positions, velocities and aggregate forces of *all* the nodes of the tetrahedral mesh. (We use $\mathbf{x}$ as the vector valued position of a single node.) M is the mass matrix, which is diagonal in our lumped mass formulation. The nodal forces can be decomposed into internal and external forces, $\vec{\mathbf{f}} = \vec{\mathbf{f}}_{int} + \vec{\mathbf{f}}_{ext}$, the latter being supplied as time varying input to the simulation.

We apply a quasistatic assumption that both the accelerations and velocities are zero to obtain $\vec{\mathbf{f}}(t, \vec{\mathbf{x}}, \vec{\mathbf{0}}) = \vec{\mathbf{0}}$ which states that the externally supplied time varying input must be balanced by the internal resistance of the material. In particular, we use a nonlinear finite element method to solve for the internal forces, and thus we must invert a nonlinear equation to find the time varying positions $\vec{\mathbf{x}}(t)$ at any time t. This is accomplished with a Newton-Raphson iterative solver, and each step towards the steady state solution begins with the linearization of the nodal forces about the current solution estimate $\vec{\mathbf{x}}_k$, i.e. $\vec{\mathbf{f}}(\vec{\mathbf{x}}_k + \Delta\vec{\mathbf{x}}_k) \approx \vec{\mathbf{f}}(\vec{\mathbf{x}}_k) + (\partial\vec{\mathbf{f}}/\partial\vec{\mathbf{x}})\Big|_{\vec{\mathbf{x}}_k} \Delta\vec{\mathbf{x}}_k$ where $\Delta\vec{\mathbf{x}}_k = \vec{\mathbf{x}}_{k+1} - \vec{\mathbf{x}}_k$. Since we desire force equilibrium with $\vec{\mathbf{f}}(\vec{\mathbf{x}}_{k+1}) = \vec{\mathbf{f}}(\vec{\mathbf{x}}_k + \Delta\vec{\mathbf{x}}_k) = \vec{\mathbf{0}}$, we solve the linear system

$$-\left.\frac{\partial\vec{\mathbf{f}}}{\partial\vec{\mathbf{x}}}\right|_{\vec{\mathbf{x}}_k} \Delta\vec{\mathbf{x}}_k = \vec{\mathbf{f}}(\vec{\mathbf{x}}_k) \tag{3.1}$$

to find the next iterate $\vec{\mathbf{x}}_{k+1}$.

Although the quasistatic assumption does not apply to free falling, unconstrained, lightly damped objects whose richness of deformation is largely enhanced by the effects of inertia, it is a viable modeling strategy for a range of applications in which boundary conditions and external forces predominantly determine the material state (e.g. skeletal muscles under a variety of conditions).

3.4 Strain Energy

For a hyperelastic material, the nodal forces can be defined via the energy as $\vec{\mathbf{f}} = -\partial\Psi/\partial\vec{\mathbf{x}}$, and thus we can rewrite equation (3.1) as

$$\left.\frac{\partial^2\Psi}{\partial\vec{\mathbf{x}}^2}\right|_{\vec{\mathbf{x}}_k} \Delta\vec{\mathbf{x}}_k = -\left.\frac{\partial\Psi}{\partial\vec{\mathbf{x}}}\right|_{\vec{\mathbf{x}}_k} . \tag{3.2}$$

That is, the global stiffness matrix $-\partial\vec{\mathbf{f}}/\partial\vec{\mathbf{x}}$ is *always* symmetric, as a result of the hyperelastic energy having continuous second derivatives with respect to the spatial configuration. Furthermore, a steady state corresponds to a local minimum of the hyperelastic energy indicating that the energy Hessian, $\partial^2\Psi/\partial\vec{\mathbf{x}}^2$, (or equivalently the global stiffness matrix) is positive definite in the vicinity of an *isolated* steady state. Moreover, systems that possess steady states along a continuous manifold in configuration space, such as underconstrained bodies with rigid degrees of freedom (e.g. a single spring with only one fixed endpoint that is otherwise free to rotate), still exhibit semi-definite stiffness matrices at their steady state. Thus, such systems can be reduced to the fully constrained case by factoring out the manifold of the configuration space that does not affect the hyperelastic energy.

Symmetry of the coefficient matrix in the linear system (3.2) allows for the use of symmetric solvers, and direct methods are commonly used. However, the fact that the stiffness matrix is positive definite close to the steady state suggests that symmetric positive definite solvers such as the conjugate gradient method *might* be applicable. This would alleviate the drawbacks of direct methods including the need to explicitly form the stiffness matrix, the memory demands incurred by matrix fill during the direct solve, and the excessive computational expense of direct solvers as opposed to iterative methods.

Our method modifies the coefficient matrix in equation (3.2) into a *positive definite* symmetric matrix and proceeds to compute the next iterate $\Delta\vec{\mathbf{x}}_k$ using this modified system. We emphasize that this modification only alters individual steps towards a minimum of the strain energy and not those minima themselves. These modifications are localized to regions of the simulation mesh that contribute to this indefiniteness. This practice of modifying the Hessian of the optimization functional is common in the optimization literature (see e.g. [68]) and is usually referred to as a modified Newton method.

3.5 Finite Element Forces

We follow the notation of [165], and their geometric interpretation of the finite element method. Consider a time dependent map ϕ from the undeformed material coordinates $\mathbf{X}$ to world coordinates $\mathbf{x}$. The stress at a point $\mathbf{X}$ in the material depends on the deformation gradient $\mathbf{F}(\mathbf{X}) = \partial\mathbf{x}/\partial\mathbf{X}$ of this mapping. We use constant strain tetrahedral elements where $\mathbf{F}$ is a constant 3×3 matrix in each tetrahedron. We define edge vectors for each tetrahedron in both material coordinates, $\mathbf{d}_{m_1} = \mathbf{X}_1 - \mathbf{X}_0$, $\mathbf{d}_{m_2} = \mathbf{X}_2 - \mathbf{X}_0$, $\mathbf{d}_{m_3} = \mathbf{X}_3 - \mathbf{X}_0$, and world coordinates, $\mathbf{d}_{s_1} = \mathbf{x}_1 - \mathbf{x}_0$, $\mathbf{d}_{s_2} = \mathbf{x}_2 - \mathbf{x}_0$, $\mathbf{d}_{s_3} = \mathbf{x}_3 - \mathbf{x}_0$, and construct 3×3 matrices $\mathbf{D}_m$ and $\mathbf{D}_s$ using the edge vectors as

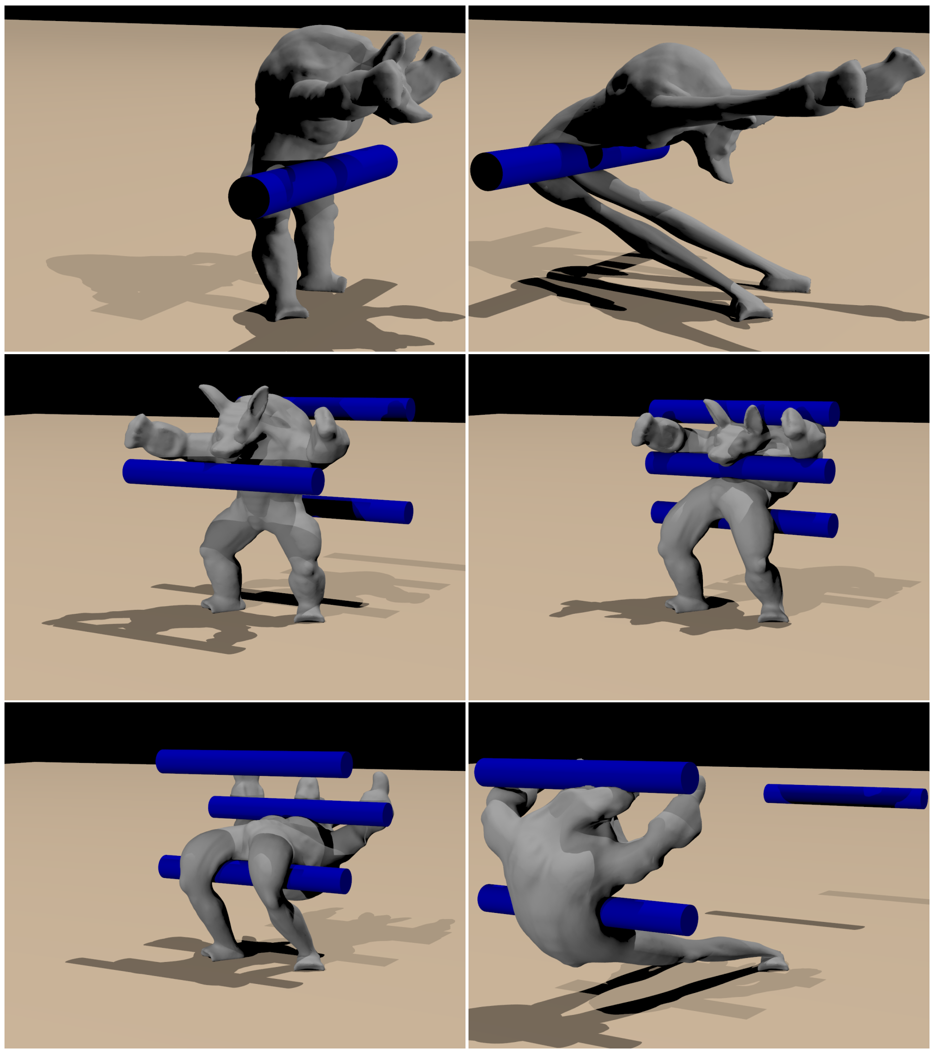

Figure 3.2: Illustration of large deformation in conjunction with collision.

columns. Then $\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1}$, and $\mathbf{D}_m^{-1}$ is constant and can be precomputed and stored for efficiency.

For hyperelastic materials, stress is defined as the derivative of a strain energy typically constructed from various strain invariants, and we use the first Piola-Kirchhoff stress which is the gradient of the strain energy with respect to the deformation gradient, $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$. $\mathbf{P}$ maps area weighted normals in material space to forces in world space. The force on a node i due to a single tetrahedron incident to it is

$\mathbf{g}_i = -\mathbf{P}\left(A_1\mathbf{N}_1 + A_2\mathbf{N}_2 + A_3\mathbf{N}_3\right)/3$, where the $A_j\mathbf{N}_j$ are the area weighted normals of the faces of the tetrahedron incident to node i. Since these do not change during the simulation, we can precompute a vector $\mathbf{b}_i$ such that $\mathbf{g}_i = \mathbf{P}\mathbf{b}_i$. For efficiency, we compute $\mathbf{g}_0 = -(\mathbf{g}_1 + \mathbf{g}_2 + \mathbf{g}_3)$ and compactly express the other three $\mathbf{g}_i$ as $\mathbf{G} = \mathbf{P}\mathbf{B}_m$ where $\mathbf{G} = (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$ and $\mathbf{B}_m = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = -V\mathbf{D}_m^{-T}$ with V the volume of the tetrahedron in material space.

As noted in [82], the first Piola-Kirchhoff stress is invariant under rotations of either material or world space for isotropic materials. Furthermore, the deformation gradient can be transformed into a diagonal matrix, $\hat{\mathbf{F}}$, with an application of a material and a world space rotation, $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$. This decomposition is obtained from the standard singular value decomposition of $\mathbf{F}$ along with the subsequent removal of any reflections in the orthogonal $\mathbf{U}$ and $\mathbf{V}$. This requires the negation of the smallest singular value of $\hat{\mathbf{F}}$ for inverted tetrahedra. Combining the rotational invariance of the first Piola-Kirchoff stress with the diagonalization of the deformation gradient yields

$$\mathbf{P}(\mathbf{F}) = \mathbf{U}\mathbf{P}(\mathbf{U}^T\mathbf{F}\mathbf{V})\mathbf{V}^T = \mathbf{U}\mathbf{P}(\hat{\mathbf{F}})\mathbf{V}^T \tag{3.3}$$

where $\mathbf{P}(\hat{\mathbf{F}})$ is also diagonal for isotropic materials. This factorization is particularly convenient, because it allows for a simple extension of the constitutive model to inverted elements in a smooth manner. That is, one only needs to modify the diagonal $\mathbf{P}(\hat{\mathbf{F}})$ to be valid for a single negative entry in the diagonal $\hat{\mathbf{F}}$. For more details, see [82].

3.6 Element Stiffness Matrix

The global stiffness matrix in equation (3.1) is constructed from the additive contributions of the element stiffness matrices, $-\partial\mathbf{f}/\partial\mathbf{x}$, which are based on contributions from individual tetrahedra. As a result of this additive decomposition, definiteness of the element stiffness matrices is a sufficient condition for definiteness of the global stiffness matrix. Motivated by this fact, we manipulate the element stiffness matrix to ensure global definiteness. In section 3.8 we show that this elemental manipulation amounts to the solution of a single 3×3 symmetric eigenproblem and a few simple algebraic operations. In contrast, dealing with the global stiffness matrix directly can be prohibitively expensive, especially if eigenanalysis or Cholesky factorization of that matrix is required, as in most standard approaches to treating locally indefinite optimization problems [68].

In order to establish the positive definiteness of the element stiffness matrix, we must ensure that $\delta\mathbf{x}^T(-\partial\mathbf{f}/\partial\mathbf{x})\delta\mathbf{x} = -\delta\mathbf{x}^T\delta\mathbf{f} > 0$ for any increment $\delta\mathbf{x}$. Using the formulas from the last section and some tensor manipulations yields

$$\begin{aligned}\delta\mathbf{x}^T\delta\mathbf{f} &= \sum_{i=1}^{3}\delta\mathbf{x}_i^T\delta\mathbf{g}_i - \delta\mathbf{x}_0^T\sum_{i=1}^{3}\delta\mathbf{g}_i = \sum_{i=1}^{3}(\delta\mathbf{x}_i - \delta\mathbf{x}_0)^T\delta\mathbf{g}_i \\ &= \delta\mathbf{D}_s : \delta\mathbf{G} = \mathrm{tr}[\delta\mathbf{D}_s^T\delta\mathbf{G}] = -V\mathrm{tr}[\delta\mathbf{D}_s^T\delta\mathbf{P}\mathbf{D}_m^{-T}] \\ &= -V\mathrm{tr}[\mathbf{D}_m^{-T}\delta\mathbf{D}_s^T\delta\mathbf{P}] = -V\mathrm{tr}[\delta\mathbf{F}^T\delta\mathbf{P}] = -V\left(\delta\mathbf{F} : \delta\mathbf{P}\right).\end{aligned}$$

Since the material element volume V is always a positive constant, the positive definiteness condition reduces to $\delta\mathbf{F} : \delta\mathbf{P} > 0$ or $\delta\mathbf{F} : (\partial\mathbf{P}/\partial\mathbf{F}) : \delta\mathbf{F} > 0$. Therefore, the positive definiteness of the element stiffness matrix is equivalent to the positive definiteness of the fourth order tensor $\partial\mathbf{P}/\partial\mathbf{F}$. This result is in direct analogy with the energy based formulation of the Newton-Raphson iteration system (3.2), since by definition $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$ and thus $\partial\mathbf{P}/\partial\mathbf{F} = \partial^2\Psi/\partial\mathbf{F}^2$.

3.7 Diagonalization

Testing and enforcing positive definiteness of the fourth order tensor $\partial\mathbf{P}/\partial\mathbf{F}$ directly can be rather unwieldy. Instead, we start as in [82] by rotating both stresses and deformations into diagonal space (transforming our configuration using the rotation matrices that diagonalize the current $\mathbf{F}$ and $\mathbf{P}$). In order to do this, first note that $\delta\mathbf{P} = \left(\partial\mathbf{P}\left(\mathbf{F}\right)/\partial\mathbf{F}\right)|_{\mathbf{F}} : \delta\mathbf{F}$ where we explicitly stress the dependency of $\mathbf{P}$ on $\mathbf{F}$ with $\mathbf{P}\left(\mathbf{F}\right)$. We can manipulate this equality into

$$\delta\mathbf{P} \;=\; \left.\frac{\partial\mathbf{U}\mathbf{P}(\mathbf{U}^T\mathbf{F}\mathbf{V})\mathbf{V}^T}{\partial(\mathbf{U}^T\mathbf{F}\mathbf{V})}\right|_{\mathbf{F}} : \delta(\mathbf{U}^T\mathbf{F}\mathbf{V})$$

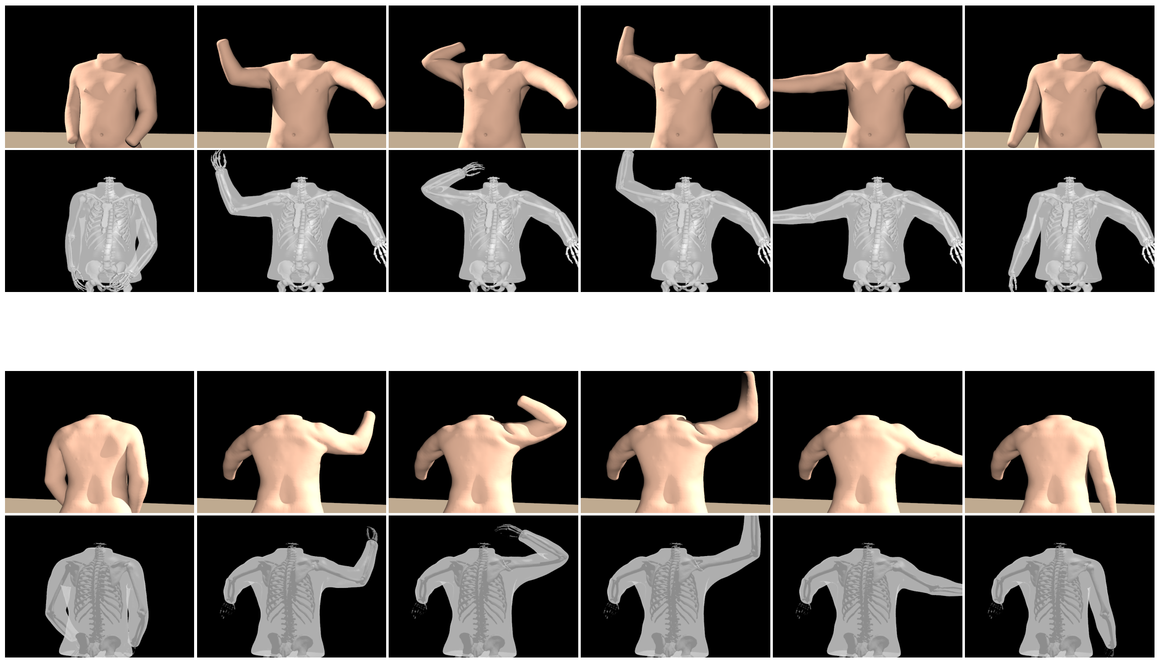

Figure 3.3: Quasistatic flesh deformation driven by a kinematic skeleton.

$$
\begin{aligned}
&= \mathbf{U}\left\{ \left.\frac{\partial \mathbf{P}(\mathbf{F})}{\partial \mathbf{F}}\right|_{\mathbf{U}^T\mathbf{F}\mathbf{V}} : \mathbf{U}^T \delta\mathbf{F}\mathbf{V} \right\} \mathbf{V}^T \\
&= \mathbf{U}\left\{ \left.\frac{\partial \mathbf{P}}{\partial \mathbf{F}}\right|_{\hat{\mathbf{F}}} : \mathbf{U}^T \delta\mathbf{F}\mathbf{V} \right\} \mathbf{V}^T \qquad (3.4)
\end{aligned}
$$

where the first equality comes from equation (3.3) and replacing $\delta\mathbf{F}$ with a rotated version, the second comes from a change of variables and the fact that $\mathbf{U}$ and $\mathbf{V}$ are chosen independent of $\mathbf{F}$, and the third comes from choosing $\mathbf{U}$ and $\mathbf{V}$ to be the rotation matrices that diagonalize the initial value of $\mathbf{F}$, i.e. where we evaluate $\partial\mathbf{P}/\partial\mathbf{F}$ to linearize for iteration. Also in the last equality, we drop the explicit dependence of $\mathbf{P}$ on $\mathbf{F}$.

Equation (3.4) provides all the information we need for solving the Newton-Raphson iteration system using a conjugate gradient solver, since the nodal force differentials can readily be computed from the stress differentials as $\delta\mathbf{G} = \delta\mathbf{P}\mathbf{B}_m$.

Furthermore we have

$$\begin{aligned}\delta\mathbf{P} : \delta\mathbf{F} &= \mathbf{U}\left\{ \left.\frac{\partial\mathbf{P}}{\partial\mathbf{F}}\right|_{\hat{\mathbf{F}}} : \mathbf{U}^T\delta\mathbf{F}\mathbf{V}\right\}\mathbf{V}^T : \delta\mathbf{F} \\ &= \mathbf{U}^T\delta\mathbf{F}\mathbf{V} : \left.\frac{\partial\mathbf{P}}{\partial\mathbf{F}}\right|_{\hat{\mathbf{F}}} : \mathbf{U}^T\delta\mathbf{F}\mathbf{V}\end{aligned}$$

illustrating that the condition for definiteness, $\delta\mathbf{P} : \delta\mathbf{F} > 0$, derived in section 3.6 is equivalent to positive definiteness of $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$. We might expect that applying the rotations that diagonalize the current deformation $\mathbf{F}$ to $\delta\mathbf{P}$ and $\delta\mathbf{F}$ would induce a simple structure for the tensor $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$. In fact this tensor turns out to have a block diagonal structure in the case of isotropic materials.

3.8 Enforcing Positive Definiteness

In order to reveal the block diagonal structure of $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$, we rewrite the $3\times3\times3\times3$ fourth order tensor as a 9×9 matrix. To do this, we consider the rearrangement of a 3×3 matrix $\mathbf{S}$ into the 9×1 vector $(s_{11}, s_{22}, s_{33}, s_{12}, s_{21}, s_{13}, s_{31}, s_{23}, s_{32})$. We can then represent $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$ as the 9×9 matrix that maps the vector equivalent of $\delta\mathbf{F}$ to the vector equivalent of $\delta\mathbf{P}$. For isotropic materials this matrix is block diagonal with diagonal components $\mathbf{A}$, $\mathbf{B}_{12}$, $\mathbf{B}_{13}$ and $\mathbf{B}_{23}$ where

$$\mathbf{A} = \begin{bmatrix} \alpha_{11}+\beta_{11}+\gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{12} & \alpha_{22}+\beta_{22}+\gamma_{22} & \gamma_{23} \\ \gamma_{13} & \gamma_{23} & \alpha_{33}+\beta_{33}+\gamma_{33} \end{bmatrix}, \quad \mathbf{B}_{ij} = \begin{bmatrix} \alpha_{ij} & \beta_{ij} \\ \beta_{ij} & \alpha_{ij} \end{bmatrix}$$

Here,

$$\alpha_{ij} = 2\Psi_I + 4(\sigma_i^2 + \sigma_j^2)\Psi_{II}$$

$$\beta_{ij} = 4\sigma_i\sigma_j\Psi_{II} - \frac{2III\Psi_{III}}{\sigma_i\sigma_j}$$

$$\gamma_{ij} = \begin{pmatrix} 2\sigma_i & 4\sigma_i^3 & \frac{2III}{\sigma_i} \end{pmatrix} \frac{\partial^2\Psi}{\partial\left(I, II, III\right)^2} \begin{pmatrix} 2\sigma_j \\ 4\sigma_j^3 \\ \frac{2III}{\sigma_j} \end{pmatrix} + \frac{4III\Psi_{III}}{\sigma_i\sigma_j}$$

where $\Psi = \Psi(I, II, III)$ is the strain energy written in terms of the invariants $I = \text{tr}\,\mathbf{C}$, $II = \mathbf{C} : \mathbf{C}$ and $III = \det \mathbf{C}$ with $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ and subscripts representing partial derivatives. Also, σ_1, σ_2 and σ_3 are the diagonal components that constitute $\hat{\mathbf{F}}$.

Positive definiteness of $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$ is equivalent to positive definiteness of each of the blocks $\mathbf{A}$, $\mathbf{B}_{12}$, $\mathbf{B}_{13}$ and $\mathbf{B}_{23}$. For $\mathbf{A}$ a simple 3×3 diagonalization is required, followed by the clamping of all negative eigenvalues to zero. For the 2×2 matrices $\mathbf{B}_{12}$, $\mathbf{B}_{13}$ and $\mathbf{B}_{23}$ no eigenanalysis is necessary since the negative eigenvalue, if present, can be clamped to zero analytically.

Our algorithm computes the stress differential $\delta\mathbf{P}$ as outlined in equation (3.4). First we compute the rotated deformation differential $\mathbf{U}^T\delta\mathbf{F}\mathbf{V}$, and then convert this 3×3 second order tensor into a 9×1 vector and multiply it by the 9×9 matrix for $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$ to carry out the contraction. Of course, we use the clamped positive definite version of $(\partial\mathbf{P}/\partial\mathbf{F})|_{\hat{\mathbf{F}}}$. The result is then converted from a 9×1 vector back to a 3×3 second order tensor, before being premultiplied by $\mathbf{U}$ and postmultiplied by $\mathbf{V}^T$.

Since we clamp eigenvalues to zero, the element stiffness matrices are only positive semi-definite, not positive definite, which raises the issue of whether the resulting global stiffness matrix could be semi-definite or ill-conditioned itself. In practice, the additive contributions of neighboring elements and boundary conditions always lead to a positive definite global stiffness matrix, even for configurations as extreme as shown in Figure 3.1. (Note that one *could* clamp to a small positive value as well.) The effect of boundary conditions on the definiteness of the stiffness matrix is analogous to that observed in the matrix resulting from the discretization of the Poisson equation. When all Neumann boundary conditions are specified, the resulting matrix is positive semi-definite. In this case a special version of Conjugate Gradients is still applicable, since an analytic description of the null space is available and, similarly, the global stiffness matrix of an elastic object has a null space corresponding to global translation and linearized rotation. Specification of one or more Dirichlet boundary conditions makes the Poisson matrix strictly positive definite, with positional constraints having the same effect on the definiteness of the global stiffness matrix for elasticity.

Anisotropic materials need special treatment in order to enforce definiteness. One

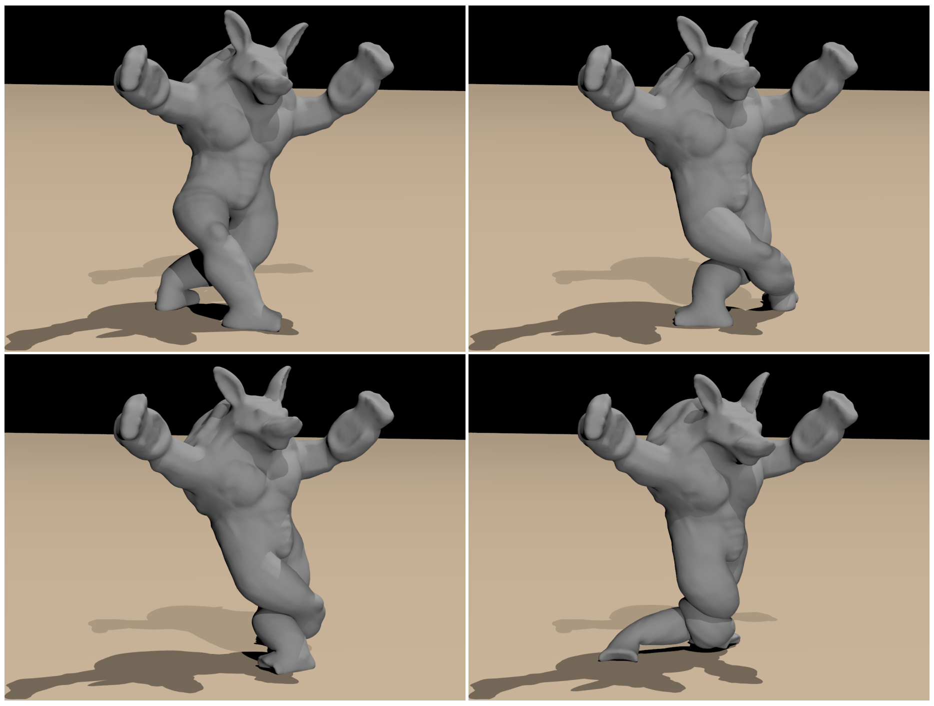

Figure 3.4: Illustration of self-collision handling.

possibility would be to perform the explicit eigenanalysis on the 12×12 elemental stiffness matrix and perform the definite projection by clamping its eigenvalues to positive numbers. A more efficient treatment is presented in appendix A.3 and analyzed for the special case of transverse isotropy.

3.9 Inverted Elements

Typically, realistic constitutive models have infinite strain energy as the volume of an element approaches zero, and this discourages element inversion when the equations of motion are integrated with a small enough time step to resolve the stiff material response. Nevertheless, each Newton-Raphson iteration of a quasistatic solver begins with a linearization of the elastic forces after which only a finite amount of energy is required to invert the element. In order to efficiently solve the equations without artificial limits on the allowable time step, we adopt the approach of [82] smoothly extending the definition of forces past a maximum compression threshold. Constant, linear, or smoother extrapolations can be used for this purpose. In our work constant extrapolation proved to be both simple and sufficient. To implement constant extrapolation we threshold the diagonal values of $\hat{\mathbf{F}}$ and compute both forces and force differentials using the thresholded deformation gradient. The resulting force differentials are then treated for indefiniteness.

3.10 Collisions

For volumetric collisions one could use the method in [29] applied to the triangulated boundary surface of the tetrahedron mesh as was done in [82]. There is also the self-collision untangling strategy of [11]. But we prefer a penalty based formulation that can more readily be incorporated into the quasistatic formulation. We use a penalty force for collision of our objects with themselves, other deformable tetrahedral bodies and rigid bodies. As a consequence of using penalty forces, the steady state may exhibit slight interpenetration of the colliding surfaces, an effect that is rather subtle and acceptable for our line of applications. The penetration depth can also be adjusted by changing the stiffness of the penalty forces. A penetrating node receives a force in the form of the gradient of a penetration potential defined as $\Psi_p(\mathbf{x}) = k\phi^2(\mathbf{x})/2$ where ϕ is the signed distance to the surface of the object for $\mathbf{x}$ interior to the body and zero otherwise. Then the force is $\mathbf{f}_p = -k\phi(\mathbf{x})\nabla\phi(\mathbf{x})$, and the force differential is $\delta\mathbf{f}_p = -k(\nabla\phi(\mathbf{x})\nabla\phi^T(\mathbf{x}) + \phi(\mathbf{x})\ \partial^2\phi/\partial\mathbf{x}^2|_{\mathbf{x}})\delta\mathbf{x}$. These forces can corrupt the definiteness of the linearized forces used with Newton-Raphson iteration. The potential for indefiniteness arises from isocontours of the signed distance function with curvatures of differing sign, see e.g. [4]. These curvatures are the eigenvalues of $\partial^2\phi/\partial\mathbf{x}^2$, and we assure definiteness by projecting this matrix to its positive definite component in the case of rigid body collisions. For deformable object collisions, we omit the last term altogether. As before, this modification does not change the equilibrium states, only the convergence path towards one of these states.

We take a level set approach (see e.g. [135]) to computing penetration depth as did [61, 109], but instead of updating the level set function as the object deforms we utilize a static level set in material space as in [80]. However, many key aspects of our algorithm are significantly different than that proposed in [80]. For each rigid and deformable object in the scene, we first precompute a signed distance function on a uniform Cartesian or octree grid as in [75]. This representation is computed in object space for rigid bodies and material space for deformable bodies and is not updated as the simulation progresses. Collecting the depth, normal and curvature information is straightforward for rigid bodies, but we propose a novel approach for deformable tetrahedral bodies.

To compute point collisions against deforming tetrahedral bodies, we maintain a bounding box hierarchy for the tetrahedra in each body. Then for each point, we use this hierarchy to find any tetrahedra that our candidate point may lie inside (inverted tetrahedra are ignored as they represent negative space). For each tetrahedron, we compute the barycentric coordinates of our candidate point to determine if the point is either inside or very close to the tetrahedron in question. We do not require robustness here as this computation is not used to determine whether a point is inside an object, but instead the barycentric coordinates are used to transform the point from world space to material space, i.e. the point is placed in material space keeping its same barycentric coordinates but using the nodal positions of the material space tetrahedron.

Then the material space position of the point is used to query the material space level set to see if the point is inside the object, and if so the local unit normal and level set value are used to estimate the closest point on the surface as $\mathbf{x}_c = \mathbf{x} - \phi\mathbf{N}$ (where ϕ is negative inside the object). If $\phi \neq 0$ at $\mathbf{x}_c$ this equation can be iterated on to find an $\mathbf{x}_c$ as close to the zero level set as is desired. Before the simulation begins, we also precompute a static bounding box hierarchy for the triangles on the surface of the object, and this is used to find the triangle closest to $\mathbf{x}_c$ as well as the barycentric coordinates of the point on this triangle closest to $\mathbf{x}_c$. Before proceeding, we check to make sure that the local level set value at this point on the triangle, $\mathbf{x}_t$, is larger than that at the original point $\mathbf{x}$, to ensure that $\mathbf{x}_t$ is actually farther outside the object than $\mathbf{x}$. This keeps us from incorrectly pulling points back towards the object (nonphysical stickiness), because of rasterization errors with the level set function that cause it to have a slightly different approximation to the object surface than as given by surface triangle mesh. Finally, the barycentric coordinates of $\mathbf{x}_t$ are used to find the corresponding point in world space, $\mathbf{x}_s$, on the surface of the deforming object.

In this fashion, we do not use the level set in material space to push points out of the object, which is important because this is unlikely to give us the proper directions for deformed objects. Instead, we merely use the level set to find a point that is truly on the surface of the object. Then the distance from $\mathbf{x}_s$ to $\mathbf{x}$ and the vector pointing between them are used to compute $\phi(\mathbf{x})$ and $\nabla\phi(\mathbf{x})$ for the penalty forces and differentials.

3.11 Examples

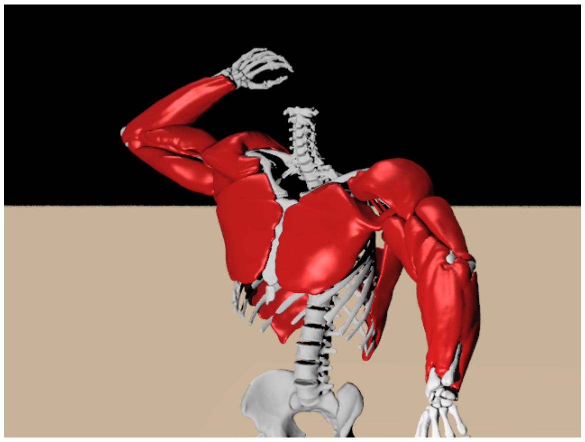

Figure 3.5: Quasistatic simulation of the upper torso musculature.

We demonstrate the applicability of our quasistatic algorithm in a number of complex scenarios. To illustrate the robustness of the extension of the elastic response to degenerate and inverted elements, we solve for elastic equilibrium with an armadillo mesh whose vertices are initially randomly distributed on a cube ten times the size of the armadillo mesh itself and whose hands and feet are constrained. Figure 3.1 shows a number of iterates in the solution process towards equilibrium. Figure 3.4

demonstrates our algorithm for deformable collision detection and response. In the simulation, the hands are held fixed while the feet twist on the ground plane causing the legs to self-collide. To demonstrate rigid body collisions, we deform the armadillo mesh with rigid cylinders as seen in figure 3.2. The interactions with the cylinders demonstrate the time coherency of the strain energy local minima achieved by using the previous equilibrium state as an initial guess for the Newton-Raphson solver.

Inertia effects are neglected when simulating quasistatic elasticity, and deformation is primarily driven by external time dependent forces due to contact, collision and boundary conditions. As a result, quasistatic simulations are particularly well suited for flesh deformation where the flesh is rigidly attached to bones and heavily influenced by contact, collision and self-collision. We demonstrate the applicability of our approach with several simulations of flesh and muscles in the upper torso, derived from the visible human data set as described in chapter 2.

In figure 3.3, we attach the deformable flesh directly to the underlying skeleton. The flesh naturally deforms from the influence of the skeleton as well as from self collision, providing realistic deformation and wrinkling of the outer skin. The flesh mesh consists of 600 thousand tetrahedral elements and was simulated with a neo-Hookean constitutive model extended to the inverted regime as in [82]. Figure 3.5 shows skeletal muscle in the upper limb simulated with the muscle constitutive model outlined in [82] and [165]. Although our quasistatic formulation was only presented for isotropic materials, it is readily extended to the case of simple transverse isotropy, since the strain energy is a sum of an isotropic and a transversely isotropic component with each term being a function of their respective associated invariants. This property leads to a stiffness matrix which is a sum of an isotropic term (which can be processed in the standard fashion) and a simple anisotropic term whose eigenstructure is easy to manipulate. The resulting simulations are enriched by muscle activations that are computed from the skeletal motion as in chapter 2 to produce realistic contractile motion. Finally, figure 3.6 shows a layered approach where we use the simulated motion of the skeletal muscles as kinematic boundary conditions for a second flesh only simulation to create more realistic muscle based skin deformation. During the second simulation, flesh nodes are constrained to follow the muscle motion if they are within a tolerance of the musculoskeletal surface.

The originally scattered armadillo geometry of figure 3.1 consists of 380K tetrahedra and converged to steady state in 80 Newton-Raphson iterations requiring 2-3 seconds each, under a neo-Hookean constitutive model (collision handling disabled). For the same 380K element armadillo mesh in figure 3.2 the computational cost was approximately 90 seconds per frame. The flesh mesh of figure 3.6 consisted of 600K tetrahedra and was simulated at 2 minutes per frame. All simulations were performed on a 3 GHz pentium 4 workstation. We stress that these are rather large simulation meshes, and meshes on the order of 10 thousand elements can be typically simulated at rates of 5–10 frames per second (computational cost scales nonlinearly). This is with tight bounds on the tolerance, where additional Newton-Raphson iterations lead to no visible changes. Additionaly, the authors of [155] use our method for a highly constrained face simulation application and report running times that translate to approximately 30 seconds per frame for a 370K tetrahedron mesh with full self and rigid body collision handling, as opposed to 50 minutes per frame, on average, for a fully dynamic simulation. Moreover, their use of quasistatic (as opposed to dynamic) simulation allows them to construct a full system Jacobian enabling the solution of an inverse problem to find muscle activations based on surface deformation.

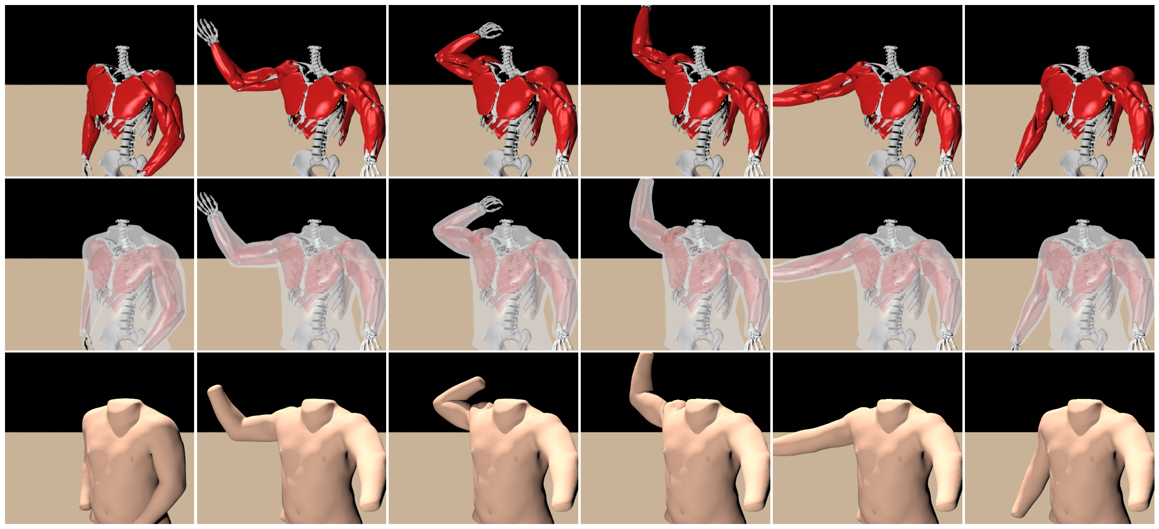

Figure 3.6: A layered, muscle-driven skinning example.

3.12 Summary

We presented a framework for efficient and robust quasistatic simulation of nonlinear elastic materials using a modified Newton-Raphson algorithm that can robustly iterate through configurations that give rise to mesh inversion and buckling instabilities. Fast conjugate gradient solvers can be used, since we enforce positive definiteness of the modified linear equilibrium equations at each iteration. This simulation technique is ideal for constrained objects influenced by the motion of their specified boundary conditions. In particular, it is useful for simulating deformable flesh and skin for virtual characters whose motion is driven by an underlying kinematic skeleton.

Chapter 4

Facial analysis

We built an anatomically accurate model of facial musculature, passive tissue and underlying skeletal structure using volumetric data acquired from a male subject (the author of this thesis). The tissues are endowed with a highly nonlinear constitutive model including controllable anisotropic muscle activations based on fiber directions. Detailed models of this sort can be difficult to animate requiring complex coordinated stimulation of the underlying musculature. We propose a solution to this problem automatically determining muscle activations that track a sparse set of surface landmarks, e.g. acquired from motion capture marker data. Since the resulting animation is obtained via a three dimensional nonlinear finite element method, we obtain visually plausible and anatomically correct deformations with spatial and temporal coherence that provides robustness against outliers in the motion capture data. Moreover, the obtained muscle activations can be used in a robust simulation framework including contact and collision of the face with external objects.

4.1 Introduction

Facial modeling and animation, enabled by recent advances in technology, is a vital new area in high demand. While this is especially true in the entertainment industry (e.g. [26]), it is also quite popular elsewhere including applications to lip reading and surgical planning. For example, [96] pointed out the utility of synthesizing expressions on a post-surgical face to determine the effects of the surgical modifications.

Starting with data from the visible human data set [178], we used the techniques proposed in chapter 2 to construct a highly detailed anatomically accurate model of the head and neck region. This includes a triangulated surface for each bone, a tetrahedralized volume and a B-spline fiber field representation for each muscle, and a single tetrahedral mesh for all the soft tissue. Then we morphed this anatomically accurate model to fit data obtained from both laser and MRI scans of a living subject constructing new meshes where necessary.

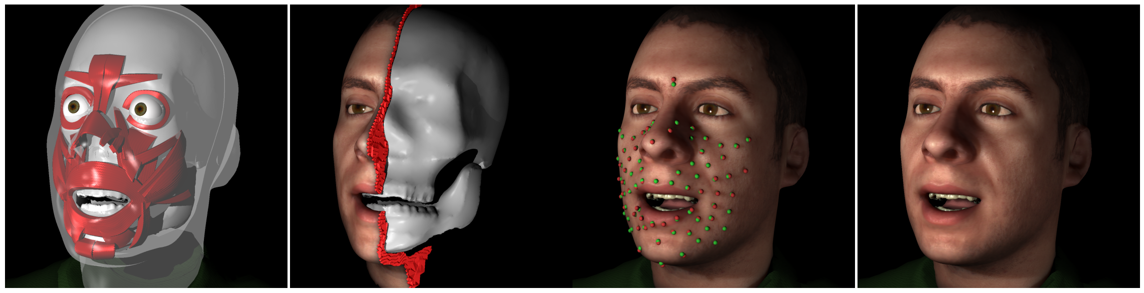

Figure 4.1: Physics based simulation of a muscle-driven facial model

Animating such a complex model can be rather difficult, so we propose using three dimensional sparse motion capture marker data (see e.g. [189, 76]) to automatically determine muscle activations. [172] took a similar approach early on estimating muscle actuation parameters based on the position of facial features tracked by snakes. Later, [115] contracted both individual and combinations of muscles in order to learn patterns, and then used two dimensional marker positions or optical flow as input for a neural network which estimated muscle contraction parameters. Both of these approaches aim to match a two dimensional projected image as opposed to our goal of matching the full three dimensional shape of the face.

A control-theoretic approach was used to estimate muscle contractions that match optical flow input in [56, 57]. Although this approach might work for more detailed anatomical models, they only considered a two dimensional finite element model for skin along with a simple muscle model for actuation. [13, 12] proposed avoiding the internal anatomy altogether constructing a two dimensional quasistatic finite element model for the lips that was used to minimize the strain as select nodes tracked motion data. This was done to *train* the lips, and PCA was used to reduce the subsequent degrees of freedom to about ten. This reduction mimics the fact that the actual degrees of freedom correspond to the muscles which are conveniently already in the *proper* lower dimensional space (as opposed to that obtained from PCA). Finally, they track lip motion automatically determining the parameters of their model that match the motion data using a steepest descent iterative solver.

More recently, [43] used a two dimensional linear quasistatic finite element model of the skin surface, along with underlying muscles that apply forces to the surface mesh based on linear activations. Their lack of anatomical structure led to the use of heuristic correction forces near the mouth. Given marker data, they used a steepest descent method to calculate the muscle activations that best track the data including penalty forces to constrain muscle activations to the physical regime. They point out that their model is linear greatly simplifying this problem. This formulation suffered from both a lack of anatomical accuracy and a lack of nonlinearity, and [42] pointed out that it could produce unnatural artifacts. Thus, they modified this procedure allowing the artist to sculpt basis elements to be combined linearly, and used the active set method to solve a constrained quadratic program to obtain the muscle activations (or weights). Typically, the basis elements need to be resculpted a number of times to obtain satisfactory results.

A major benefit of our approach is that we strive for anatomical accuracy which gives biomechanical meaning to our activations, and thus makes the problem tractable in the sense that a person's face *is* driven by muscle activations of this sort. Moreover, we take a biomechanically accurate fully nonlinear approach to both the constitutive model and the finite element method, which complicates the solution process but provides behavior not captured by approaches that linearly blend basis functions. We automatically solve for not only muscle activations, but the head position and jaw articulation parameters as well. Moreover, our approach uses a sophisticated simulation framework allowing us to place the animated face in complex environments including arbitrary contact and collision with external objects in the scene. During this interaction, the extracted muscle activation controls can still be applied providing a realistic combination of muscle contraction and external stimuli. Furthermore, if the collision events give rise to ballistic phenomena, we can readily replace the quasistatic simulation with a fully dynamic one while retaining the extracted muscle activation values.

4.2 Related Work

Three dimensional facial animation began with [138] (see [139] for a review). [146] built a face model using masses and springs including forces generated by muscles, and made use of the Facial Action Coding System (FACS) [55]. Other early work included [185, 108, 91]. [101] constructed an anatomically motivated facial model based on scanned data, and endowed it with a mass spring system driven by muscle contractions. [184] built a muscle model for speech animation, stressing that muscles animate faces and that it is more productive to focus on a muscle model rather than a surface model. A number of authors have used finite element simulations in the context of facial surgery, e.g. [142, 95, 93, 149] (see also, [174]).

[51] used variational modeling and face anthropometry techniques to construct smooth face models. [143] used a number of photographs to fit a three dimensional template mesh to a given facial pose, and then obtained animations by blending different poses. [145] used this technique to fit a face model to each frame of a video sequence estimating the pose for subsequent analysis, and [87] automatically segments the face into smaller regions for blending. See also, [195]. Starting from a database of face scans, [20] derive a vector space representation of shapes and textures such that any linear combination of examples gives a reasonable result. This framework has been used to transfer animations from one individual to another [18], for face identification [21], and to exchange a face from one image to another [19]. [88] built a mass spring model of a face and skull with a muscle model to drive the deformation, [90] proposed a method for morphing this model to other faces, and [89] extended this approach to forensic analysis.

Other work includes facial animation based on audio or text input [36, 28, 27, 37, 58, 33, 32], wrinkle formation [190], eye motion [100], and facial motion transfer

[129, 147, 124, 163]. [31] modified the MPEG-4 Facial Animation Parameters (FAPs) [117] to add expressiveness, [98] used PCA to deform the mouth during speech, and [38] used facial tracking to drive animations from a motion capture database. [183] used a multiresolution deformable mesh to track facial motion, and a low dimensional embedding technique to learn expression style. [194] proposed a method for tracking facial animation fitting a template mesh to the data, and then used linear combinations of basis shapes to create an inverse kinematics system that allows one to create expressions by dragging surface points directly.

4.3 Anatomical Model

Our model building effort started with the volumetric data from the visible human data set [178]. As in chapter 2, we constructed level sets for each tissue and used them to create a triangulated surface for each bone and a tetrahedralized volume for each muscle. Since many of the muscles in the face are quite thin and thus not amenable to robust and efficient tetrahedral mesh simulation, we took an embedded approach to muscle modeling. First, a single tetrahedral flesh mesh was created to represent *all* the soft tissue in the face, and then we calculated the fraction of overlap between each muscle and each tetrahedron of the flesh mesh storing that fraction locally in the tetrahedron. We also create fiber fields for each muscle and store a single vector direction *per muscle* in each tetrahedron with a nonzero overlap. One graduate student spent 6 months constructing this template face and muscle model from the visible human data, but with existing tools this could be accomplished in 2 weeks.

Subsequently, we obtained laser and MRI scans of a living subject. The laser scans gave a high-fidelity likeness of the subject, and we wanted to adhere to them closely. The MRI scan was of much lower quality presenting only an approximate guideline, and we needed to reuse the bone, muscle, and to a lesser extent flesh geometry from our visible human template model. To that end we developed a set of point correspondences between the two models and morphed the geometry from the first using radial basis functions. This morphed geometry required further manual editing to satisfy considerations of aesthetics and general anatomical knowledge. Once we had geometry for the surface of the flesh volume, we again used the meshing algorithm to create a high-quality tetrahedral mesh for the face flesh. To summarize, our model consists of a rigid articulated cranium and jaw with about 30 thousand surface triangles, flesh in the form of a tetrahedral mesh with about 850 thousand tetrahedra out of which 370 thousand (in the front part of the face) are simulated, Dirichlet boundary conditions corresponding to bone attachments, and an embedded representation of 32 muscles. This subject specific model was constructed in 2 months by 5 undergraduate students, but would only take a single person a few days with existing tools. For example, rebuilding the facial tetrahedral flesh takes only a few hours. A cross-section of the simulation volume is illustrated in figure 4.2.

In addition to the main functionality, a number of auxiliary features were considered. To provide realistic collisions of lips against the underlying rigid structure, we incorporated scans of teeth molds into the cranium and jaw. To achieve more realistic muscle action, we independently scaled the strength of each embedded muscle based on the amplitude and plausibility of their flexion. This biomechanically corresponds to adjusting the thickness of muscles, which we could not reliably infer from the MRI scan. Finally, we added eyes, teeth, shoulders, realistic rendering, etc.

4.4 Finite Element Method

The flesh mesh is governed by a Mooney-Rivlin constitutive model for the deviatoric deformation augmented by a volumetric pressure term for quasi-incompressibility. Tetrahedra which contain facial muscles have an additional anisotropic response for each muscle, which consists of both passive and active components scaled by the volume fraction. See [165, 167] for more details. The definition of nodal forces can be summarized as

$$\mathbf{f}(\mathbf{x}, \mathbf{a}) = \mathbf{f}_0(\mathbf{x}) + \sum_{i=1}^{M} a_i \mathbf{f}_i(\mathbf{x}) \tag{4.1}$$

were $\mathbf{f}$ and $\mathbf{x}$ denote the forces and positions of *all* nodes in the simulation mesh, and $\mathbf{a} = (a_1, a_2, \ldots, a_M)^T$ is the vector of activations of all M muscles. $\mathbf{f}_0$ corresponds to the elastic material response of the flesh including the passive anisotropic component present in muscle regions. Each force component $\mathbf{f}_i$ corresponds to the contribution of a fully activated muscle and is weighted by the corresponding current muscle activation level with $a_i \in [0, 1]$. The $\mathbf{f}_i$ depend on the spatial configuration $\mathbf{x}$ alone, making the total force an affine function of muscle activations. This linear dependence of force on activation is a fundamental property of the force-length curve of [193] that provides a useful simplification to our control framework.

We use a quasistatic simulation scheme where each input of muscle activations and skeletal configuration is directly mapped to the steady state expression it gives rise to. Such an assumption is fundamental to our control strategy, since it enables facial expressions to be defined as functions of the input control parameters without any dependence on the deformation history. We stress that this hypothesis is adopted only in the context of our optimization process to automatically determine muscle activations, and inertial effects can later be included in extracted expressions via a full dynamic simulation utilizing the same muscle control parameters.

Given a set of muscle activation parameters, $\mathbf{a}$, and appropriate boundary conditions, we substitute these into equation (4.1) and solve the resulting nonlinear equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. The boundary conditions are derived from the position of the cranium and jaw (see section 4.8), and we abstractly encode this state with a vector $\mathbf{b}$. Solving this equation leads to an equilibrium configuration for the mesh, $\mathbf{X}(\mathbf{a}, \mathbf{b})$. These steady state positions are defined implicitly with the aid of equation (4.1) as

$$\mathbf{f}(\mathbf{X}(\mathbf{a}, \mathbf{b}), \mathbf{a}) = \mathbf{0}. \tag{4.2}$$

Note that $\mathbf{b}$ does not explicitly appear in the definition of the finite element forces, but instead fully determines the value of some constrained nodes in the simulation mesh, which we denote $\mathbf{X}^C(\mathbf{b})$. Equation (4.2) can therefore be considered an implicit definition for the quasistatic positions of the unconstrained set of mesh nodes, denoted by $\mathbf{X}^U(\mathbf{a}, \mathbf{b})$.

We solve equation (4.2) with a Newton-Raphson iterative solver. At each step, the finite element forces are linearized around the current estimate $\mathbf{X}_k$ as $\mathbf{f}(\mathbf{X}_k + \delta\mathbf{X}) \approx$

$\mathbf{f}(\mathbf{X}_k) + \partial\mathbf{f}/\partial\mathbf{x}|_{\mathbf{X}_k}\,\delta\mathbf{X}$. Then we compute the displacement $\delta\mathbf{X}$ that would restore the linearized equilibrium $-\partial\mathbf{f}/\partial\mathbf{x}|_{\mathbf{X}_k}\,\delta\mathbf{X} = \mathbf{f}(\mathbf{X}_k)$, and define the next iterate as $\mathbf{X}_{k+1} = \mathbf{X}_k + \delta\mathbf{X}$. Unfortunately, $\partial\mathbf{f}/\partial\mathbf{x}|_{\mathbf{X}_k}$ is often indefinite leading to significant computational cost when solving for $\delta\mathbf{X}$. Thus, we utilize the enabling technology described in 3 that allows a fast conjugate gradient solver to be used to find $\delta\mathbf{X}$. Moreover, the method proposed in 3 allows for element inversion during the quasistatics solve speeding up our Newton-Raphson iteration by a significant amount. Overall, the convergence is particularly fast yielding an admissible solution to the nonlinear equilibrium problem within a few Newton-Raphson iterations even for drastic changes of activation levels. Although other solvers could be used, and solving equation (4.2) can be considered a "black box" as far as our method is concerned, this particular solver makes our estimation of muscle activations practical as opposed to just doable. Mesh collisions between the lips or the lips and the teeth or gums are handled with the penalty force formulation of chapter 3. See also [79], [176], etc. for more on collision handling.

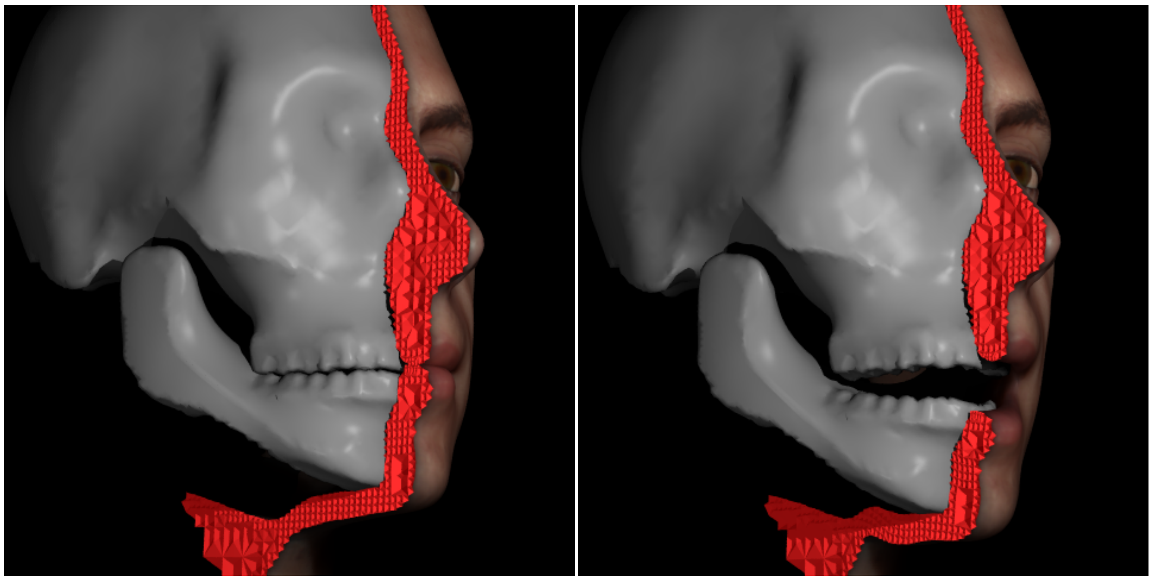

Figure 4.2: Finite element flesh mesh for face simulation.

4.5 Optimization Framework

We group all the muscle activations and kinematic parameters into a single set of controls $\mathbf{c} = (\mathbf{a}, \mathbf{b})$ writing the equilibrium positions as $\mathbf{X}(\mathbf{c})$. The input to our model consists of a sparse set of motion capture marker data, but markerless techniques or animator key framing could alternatively be used as long as the final inputs are converted to target locations for points on the surface mesh. In the rest pose, we find the surface triangle closest to each marker and compute the barycentric coordinates for the marker rest position. If the marker does not lie on the surface mesh, we subtract its vector offset from *all* the data for that marker so that it does lie on the surface mesh (and should continue to as it is animated). Given values for the control parameters, the vector of all our embedded landmark positions is given by $\mathbf{X}^L(\mathbf{c}) = \mathbf{W}\mathbf{X}(\mathbf{c})$ where $\mathbf{W}$ is a sparse matrix of barycentric weights. Our goal is then to find the set of controls that minimize the distance between our landmark positions $\mathbf{X}^L(\mathbf{c})$ and the motion capture marker data target positions $\mathbf{X}^T$, i.e.

$$\mathbf{c}_{\text{opt}}(\mathbf{X}^T) = \arg\min_{\mathbf{c}\in\mathbf{C}_0} \|\mathbf{X}^L(\mathbf{c}) - \mathbf{X}^T\|$$

where $\mathbf{c}_{\text{opt}}(\mathbf{X}^T)$ stresses that the optimal set of controls is a function of the target positions. Here, $\mathbf{C}_0$ is the *feasible* set of control configurations restricting the muscle activations (to the interval $[0, 1]$) as well as the positioning and articulation of the head and jaw. Although any geometric or statistical norm could be used, we use the Euclidean norm which leads to a nonlinear least squares optimization problem. The nonlinearity is from the dependence of $\mathbf{X}^L(\mathbf{c})$ on $\mathbf{X}(\mathbf{c})$ which is a complex nonlinear map defined implicitly by equation (4.2).

A standard Newton iterative approach to minimizing the functional $\phi(\mathbf{c}) = \|\mathbf{X}^L(\mathbf{c}) - \mathbf{X}^T\|_2^2$ consists of replacing $\phi(\mathbf{c})$ by its quadratic Taylor expansion about the current guess $\mathbf{c}_k$, i.e. $\phi(\mathbf{c}_k + \delta\mathbf{c}) \approx \phi(\mathbf{c}_k) + \delta\mathbf{c}^T\nabla\phi(\mathbf{c}_k) + \frac{1}{2}\delta\mathbf{c}^T H_\phi(\mathbf{c}_k)\delta\mathbf{c}$ where $H_\phi(\mathbf{c}_k) = 2\mathbf{J}_k^T\mathbf{J}_k + 2\mathbf{W}^T(\mathbf{X}^L(\mathbf{c}_k) - \mathbf{X}^T) : \partial^2\mathbf{X}/\partial\mathbf{c}^2|_{\mathbf{c}_k}$, $\mathbf{J}_k = \mathbf{W}\,\partial\mathbf{X}/\partial\mathbf{c}|_{\mathbf{c}_k}$ and $\delta\mathbf{c} = \mathbf{c} - \mathbf{c}_k$. Then the quadratic approximation is minimized by solving $-H_\phi(\mathbf{c}_k)\delta\mathbf{c} = \nabla\phi(\mathbf{c}_k)$ to find the next iterate $\mathbf{c}_{k+1} = \mathbf{c}_k + \delta\mathbf{c}$. In section 4.6 we illustrate how the Jacobian,

$\partial \mathbf{X}/\partial \mathbf{c}$, of the quasistatic configuration can be computed using an efficient and reliable process. However, computation of $\partial^2 \mathbf{X}/\partial \mathbf{c}^2$ is particularly expensive as well as error prone unless very stringent accuracy requirements on the computation of both the quasistatic solution and its Jacobian are satisfied. In light of this, we propose an alternative optimization technique linearizing about $\mathbf{c}_k$ to obtain

$$\mathbf{X}(\mathbf{c}) \approx \mathbf{X}(\mathbf{c}_k) + \partial \mathbf{X}/\partial \mathbf{c}\big|_{\mathbf{c}_k} \delta \mathbf{c} \tag{4.3}$$

which can be substituted into $\phi(\mathbf{c})$ to obtain $\hat{\phi}(\mathbf{c}) = \|\mathbf{X}^L(\mathbf{c}_k) + \mathbf{J}_k \delta \mathbf{c} - \mathbf{X}^T\|_2^2$. $\hat{\phi}(\mathbf{c})$ is minimized by the least squares solution of the linear system $-\mathbf{J}_k \delta \mathbf{c} \cong \mathbf{X}^L(\mathbf{c}_k) - \mathbf{X}^T$, and the normal equations approach to this requires solving $-\mathbf{J}_k^T \mathbf{J}_k \delta \mathbf{c} = \mathbf{J}_k^T \left(\mathbf{X}^L(\mathbf{c}_k) - \mathbf{X}^T\right)$ or $-2\mathbf{J}_k^T \mathbf{J}_k \delta \mathbf{c} = \nabla \phi(\mathbf{c}_k)$. Notably, this final equation is equivalent to the Newton approach where the Hessian has been approximated by only its first term, removing the problematic $\partial^2 \mathbf{X}/\partial \mathbf{c}^2$ term. This is known as the Gauss-Newton approach (see e.g. [68]).

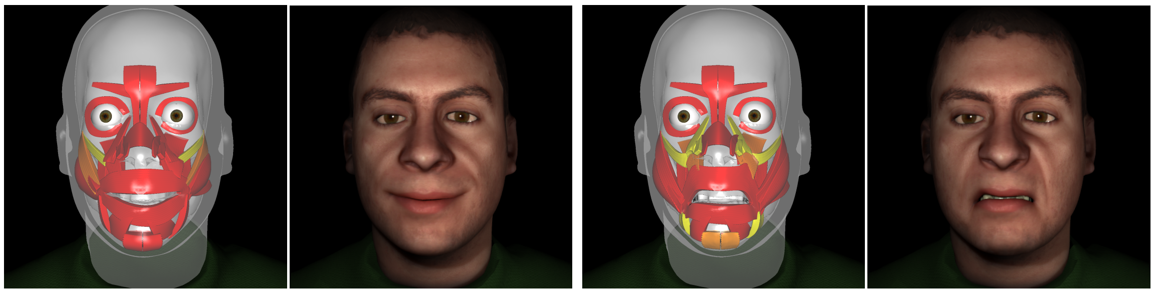

Figure 4.3: Synthetic expressions created by manual specification of activations.

When $\mathbf{X}^T$ is physically attainable, the Hessian is well approximated by its first term in the vicinity of the optimal value $\mathbf{c}_{\text{opt}}$. We have found this to be a very frequent case due to the expressive ability of our simulation model, especially in the context of expression tracking where the estimated control parameters at each frame constitute a very good initial guess to those at the next frame. However, large changes in activations or boundary conditions can make the solution to equation (4.2) unreliable causing the Gauss-Newton step to be suboptimal. This can also happen when low quality input causes $\mathbf{X}^T$ to be distant from the physically attainable configuration manifold. In order to safeguard against this suboptimal behavior, we use $\delta\mathbf{c} = \mathbf{c}_{k+1} - \mathbf{c}_k$ as a *search direction* and minimize $\phi(\mathbf{c})$ along the line segment connecting $\mathbf{c}_k$ and $\mathbf{c}_{k+1}$. Since $\phi(\mathbf{c})$ seemed to be unimodal in the vast majority of test cases, we used golden section search. Using linear interpolation to estimate the quasistatic configuration at internal points of the line segment provided a particularly good initial guess to the quasistatic solver making it typically converge in a single Newton-Raphson iteration for each golden section refinement. Given that no computation of Jacobians is necessary during the explicit line search, we found the incorporation of this process to incur only about a 10% performance overhead. As far as overall performance is concerned, remote initial guesses typically converge within an absolute maximum of 4-5 Gauss-Newton steps, while reasonable quality inputs typically led to convergence in a single step (notably with the full Gauss-Newton step to $\mathbf{c}_{k+1}$).

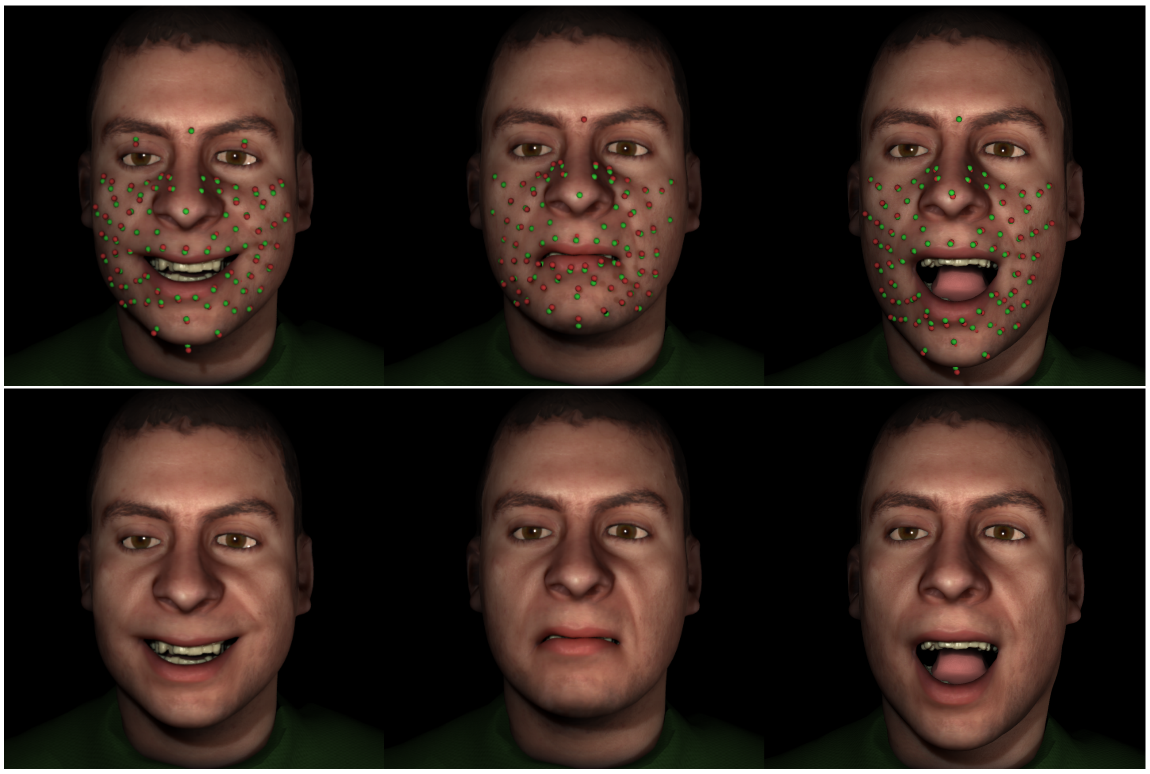
Figure 4.4: Expressions estimated from motion capture data.

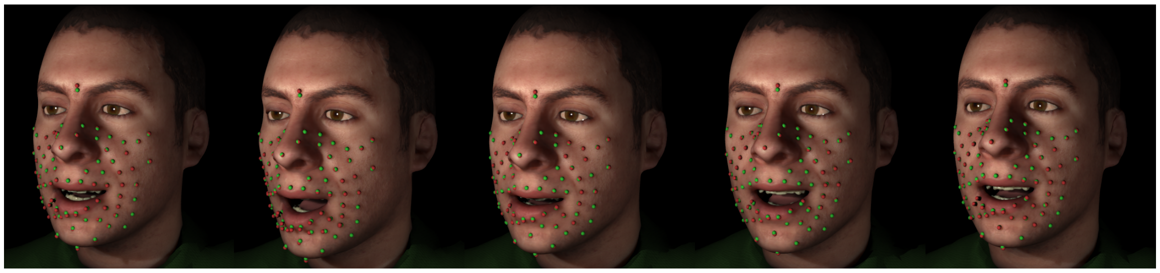

Figure 4.5: Tracking a motion captured narration sequence.

4.6 Jacobian Computation

Our optimization framework relies on the ability to compute both the equilibrium positions, $\mathbf{X}^* = \mathbf{X}(\mathbf{c}^*)$, and the Jacobians, $\partial\mathbf{X}/\partial\mathbf{c}|_{\mathbf{c}^*}$, for a given control configuration $\mathbf{c}^* = (\mathbf{a}^*, \mathbf{b}^*)$. The first of these is readily computed by solving equation (4.2), and the results of this can be used to (nontrivially) compute the Jacobians as well. To do this, we rewrite equation (4.2) to explicitly highlight the dependence on both constrained and unconstrained nodes

$$\mathbf{f}(\mathbf{X}^C(\mathbf{b}), \mathbf{X}^U(\mathbf{a}, \mathbf{b}), \mathbf{a}) = \mathbf{0} \tag{4.4}$$

stressing that there is still only one equation for each unconstrained node, since the net force at constrained nodes is trivially zero. Differentiating equation (4.4) with respect to an activation parameter a_i yields $\partial\mathbf{f}/\partial\mathbf{x}^U\big|_{\mathbf{X}^*,\mathbf{a}^*} \partial\mathbf{X}^U/\partial a_i\big|_{\mathbf{a}^*,\mathbf{b}^*} + \partial\mathbf{f}/\partial a_i|_{\mathbf{X}^*,\mathbf{a}^*} = \mathbf{0}$ or

$$-\partial\mathbf{f}/\partial\mathbf{x}^U\big|_{\mathbf{X}^*,\mathbf{a}^*} \partial\mathbf{X}^U/\partial a_i\big|_{\mathbf{a}^*,\mathbf{b}^*} = \mathbf{f}_i(\mathbf{X}^*) \tag{4.5}$$

where $\mathbf{f}_i$ was defined in equation (4.1) as the active force induced by a unit activation of the i-th muscle. This is a linear system of equations for the unknown partial derivatives $\partial\mathbf{X}^U/\partial a_i$. Additionally note that the activations have no effect on the constrained boundary nodes, and thus $\partial\mathbf{X}^C/\partial a_i$ is identically zero.

Some of the kinematic controls, $\mathbf{b}$, such as the base frame of reference for the position of the cranium, do not affect the strain of the deformable model. The Jacobian of the quasistatic positions with respect to such controls can be determined analytically since a rigid transformation of their associated boundary conditions simply induces the same rigid body transformation for the entire simulation mesh. Other kinematic parameters, such as those that articulate the jaw, nonrigidly change the quasistatic configuration. Differentiating equation (4.4) with respect to such a kinematic parameter b_i and rearranging gives

$$-\left.\partial\mathbf{f}/\partial\mathbf{x}^U\right|_{\mathbf{X}^*,\mathbf{a}^*} \left.\partial\mathbf{X}^U/\partial b_i\right|_{\mathbf{a}^*,\mathbf{b}^*} = \left.\partial\mathbf{f}/\partial\mathbf{x}^C\right|_{\mathbf{X}^*,\mathbf{a}^*} \left.\partial\mathbf{X}^C/\partial b_i\right|_{\mathbf{b}^*} \tag{4.6}$$

which is a linear system of equations for the unknown $\partial\mathbf{X}^U/\partial b_i$. Note that $\partial\mathbf{X}^C/\partial b_i$ is analytically known from the definition of the kinematic parameters. Also, $\partial\mathbf{f}/\partial\mathbf{x}^C$ is the stiffness of the forces on the unconstrained nodes with respect to the positions of the boundary conditions and is also analytically known from the definition of the finite element forces in our model. The entire right hand side of equation (4.6) can be interpreted as the linearized differential of the forces on the unconstrained nodes resulting from a displacement by $\partial\mathbf{X}^C/\partial b_i$ of *only* the boundary conditions.

Both equations (4.5) and (4.6) require solving a linear system with the coefficient matrix $-\partial\mathbf{f}/\partial\mathbf{x}^U$ which is the same coefficient matrix used in section 4.4, and thus the same solution techniques can be applied. Moreover, the coefficient matrix is symmetric positive definite near an equilibrium configuration, and thus special treatment is required only for element inversion (not definiteness). The Jacobians need to be computed before each Gauss-Newton step amounting to 44 applications of the conjugate gradient solver (for 32 activations and 12 kinematics parameters). In a sequence of expression tracking, an excellent initial guess to the conjugate gradient solver consists of using the Jacobians from the previous frame rotated according to the cranium motion. In fact, this allows us to update *all* the Jacobians at a cost approximately equal to that of a single quasistatic solve.

4.7 Muscle Activation Constraints

In order to restrict the optimization process to the allowable parameter set $\mathbf{C}_0$, we augment $\hat{\phi}(\mathbf{c})$ with a weighted penalty term $\rho\phi_p(\mathbf{c})$ which consists of piecewise quadratic penalty terms of the form $(\min\{0, a_i, 1-a_i\})^2$ for each activation. These C^2 continuous functions vanish within the allowable parameter space $\mathbf{C}_0$, and penalize the optimization functional when the control parameters drift away from $\mathbf{C}_0$. We typically initialize ρ with a value that is smaller than $\phi(\mathbf{c}_{\text{opt}})$ (for example the variance of the localization error of the motion capture system), and then progressively increase its value in multiplicative increments of 5% until it is 10^6 times larger than the current value of $\phi(\mathbf{c})$. This drives the activations to within a maximum distance of 10^{-4} of the allowed interval $[0, 1]$. At the maximum value of ρ, the contribution of $\rho\phi_p(\mathbf{c})$ to the overall optimization functional is typically on the order of .1% implying that our minimization effort is properly focused on the constrained minimum of the proximity error $\phi(\mathbf{c})$.

In each step of the Gauss-Newton approach, we replace $\phi(\mathbf{c})$ with $\hat{\phi}(\mathbf{c})$ and apply the standard Newton approach to obtain $-H_{\hat{\phi}}(\mathbf{c}_k)\delta\mathbf{c} = \nabla\hat{\phi}(\mathbf{c}_k)$ or $-2\mathbf{J}_k^T\mathbf{J}_k\delta\mathbf{c} = \nabla\hat{\phi}(\mathbf{c}_k)$. In our penalty term formulation, $\mathbf{c}_{k+1}$ is obtained as the limit of the solutions $\mathbf{c}_k^i$ to a *nested* sequence of unconstrained optimization problems minimizing $\hat{\phi}(\mathbf{c}) + \rho_i\phi_p(\mathbf{c})$ for an increasing procession of weights ρ_i, i.e. $\mathbf{c}_{k+1} = \mathbf{c}_k^N$ where the maximum ρ is ρ_N. Each step of this nested iteration is given by

$$-\left(2\mathbf{J}_k^T\mathbf{J}_k + \rho_i\, \partial^2\phi_p/\partial\mathbf{c}^2\big|_{\mathbf{c}_k^i}\right)\delta\mathbf{c}^i = \nabla\hat{\phi}(\mathbf{c}_k^i) + \rho_i\nabla\phi_p(\mathbf{c}_k^i) \tag{4.7}$$

where $\delta\mathbf{c}^i = \mathbf{c}_k^{i+1} - \mathbf{c}_k^i$. Note that all derivatives of ϕ_p can be computed analytically, Each iteration involves the solution of a *low* dimensional system (32 dimensions for the activations, but 56 total when the 24 kinematics parameters are included as outlined in section 4.8), and thus the overall computational cost is practically negligible.

4.8 Kinematics and Jaw Articulation

The kinematic parameters determine the placement of the cranium and mandible, and thus the position of specific mesh nodes of the interior flesh surface that have been rigidly attached to these bones. The frame of reference for the cranium determines the position and orientation of the entire head, while the frame of reference of the jaw is specified relative to the cranium and is subject to anatomical constraints that limit its relative placement and degrees of freedom. In order to define each frame of reference, a displacement vector for the origin of each system must be supplied together with a descriptor of the orientation. Typical descriptions of orientation are poor choices for our optimization framework, since equation (4.3) indicates that we need to linearize the quasistatic configuration with respect to the control parameters. Large linearized rotations induce significant erroneous nonrigid distortion leading to a poor approximation of the rotation slowing convergence of the Gauss-Newton iteration especially for a remote initial guess. Thus, we propose an atypical penalty term formulation.

We begin by describing the rigid body frame by a general *affine* transform, i.e. the frames that describe the cranium and mandible are $(\mathbf{M}_{\text{c}}, \mathbf{t}_{\text{c}})$ and $(\mathbf{M}_{\text{m}}, \mathbf{t}_{\text{m}})$ where $\mathbf{M}$ is *any* matrix (not necessarily orthogonal) and $\mathbf{t}$ is a translation. Then the vector of kinematic controls, $\mathbf{b}$, consists of the 24 coefficients specifying these two affine transforms. Under this parameterization the mapping from the coefficients of the global frame of reference $(\mathbf{M}_{\text{c}}, \mathbf{t}_{\text{c}})$ to the positions of *all* nodes in the flesh mesh is *linear*, when the matrix $\mathbf{M}_{\text{c}}$ is restricted to the set of rotation matrices. This implies that the linearization of equation (4.3) can capture exactly all rigid body transformations of the landmarks without any geometric distortion. Orthogonality of $\mathbf{M}_{\text{c}}$ and $\mathbf{M}_{\text{m}}$ (implying rigidity of the corresponding linear transform) can be enforced using the penalty term $\phi_{\text{rigid}}(\mathbf{M}) = \|\mathbf{M}^T\mathbf{M} - \mathbf{I}\|_F^2$ with F representing the Frobenius norm. Note that this does not penalize the rotation, since a polar decomposition of $\mathbf{M} = \mathbf{QS}$ into a rotation plus a symmetric matrix yields $\mathbf{M}^T\mathbf{M} = \mathbf{S}^T\mathbf{Q}^T\mathbf{QS} = \mathbf{S}^T\mathbf{S}$ which removes the rotation. Thus it only penalizes the symmetric nonrigid deformation to be the identity matrix, thus removing it. Under the progressive stiffening schedule for ρ described in section 4.7, this penalty term keeps the singular values of the affine matrices within 10^{-5} of unity. In order to ensure convexity of this penalty term we should project its Hessian to its positive definite component either through explicit eigenanalysis or through the process proposed in chapter 3. Furthermore, we only need to solve equation (4.6) for the jaw, as the partial derivative of the quasistatic positions with respect to a component of the cranium affine transform can be analytically computed as $\partial \mathbf{X}^j / \partial b_i = (\partial \mathbf{M}_\text{c} / \partial b_i)\, \mathbf{X}^j + \partial \mathbf{t}_\text{c} / \partial b_i$ for the quasistatic position of *any* node $\mathbf{X}^j$ assuming orthogonality of $\mathbf{M}_\text{c}$.

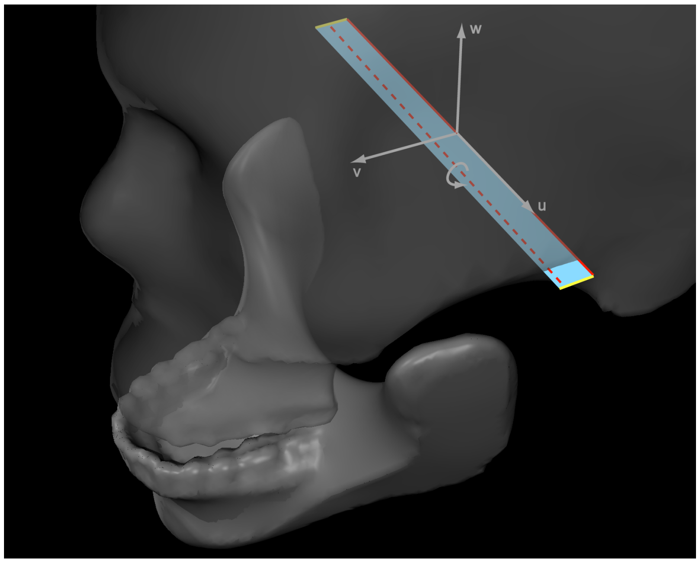

Figure 4.6: Configuration of the temperomandibular joint model.

We model the joint between the cranium and the mandible by a three degree of freedom articulation system as depicted in figure 4.6. During opening of the mouth, the lower jaw rotates around a horizontal axis passing through the mandibular condyles, which are located at the rear extreme of the jawbone and are free to slide a short distance along the temporal bone of the cranium. We model the allowable trajectories of the condyles with two parallel line segments. The condyles can slide symmetrically or asymmetrically along their designated tracks; the latter effectively results in rotation of the mandible about a vertical axis. We formalize these constraints by requiring the horizontal axis of rotation to always lie on the plane defined by the two sliding tracks and restricting the midpoint of the two condyles to positions on that plane that are equidistant from the two side tracks.

To provide algebraic descriptions for these anatomical constraints, we equip the jaw with three characteristic normalized vectors defining the geometry of the temperomandibular joint in its rest configuration (fully closed with horizontally aligned dentures). In the reference frame of the cranium, $\mathbf{u}$ points from the right to the left condyle, $\mathbf{v}$ is parallel to the sliding tracks of the condyles directed from back to front, and $\mathbf{w} = \mathbf{v} \times \mathbf{u}$. Labeling the initial location of the midpoint of the two condyles as $\mathbf{m}$, the algebraic constraints for anatomical validity are

$$\begin{aligned}
\psi_1(\mathbf{M}_{\mathrm{m}}, \mathbf{t}_{\mathrm{m}}) &= \mathbf{w}^T \mathbf{M}_{\mathrm{m}} \mathbf{u} \;=\; 0 \\
\psi_2(\mathbf{M}_{\mathrm{m}}, \mathbf{t}_{\mathrm{m}}) &= \mathbf{w}^T \left(\mathbf{M}_{\mathrm{m}}\mathbf{m} + \mathbf{t}_{\mathrm{m}} - \mathbf{m}\right) \;=\; 0 \\
\psi_3(\mathbf{M}_{\mathrm{m}}, \mathbf{t}_{\mathrm{m}}) &= \mathbf{u}^T \left(\mathbf{M}_{\mathrm{m}}\mathbf{m} + \mathbf{t}_{\mathrm{m}} - \mathbf{m}\right) \;=\; 0 \\
\psi_4(\mathbf{M}_{\mathrm{m}}, \mathbf{t}_{\mathrm{m}}) &= \mathbf{v}^T \left[\mathbf{M}_{\mathrm{m}}\left(\mathbf{m} - (l/2)\mathbf{u}\right) + \mathbf{t}_{\mathrm{m}} - \mathbf{m}\right] \;\in\; [0, d] \\
\psi_5(\mathbf{M}_{\mathrm{m}}, \mathbf{t}_{\mathrm{m}}) &= \mathbf{v}^T \left[\mathbf{M}_{\mathrm{m}}\left(\mathbf{m} + (l/2)\mathbf{u}\right) + \mathbf{t}_{\mathrm{m}} - \mathbf{m}\right] \;\in\; [0, d] \\
\psi_6(\mathbf{M}_{\mathrm{m}}, \mathbf{t}_{\mathrm{m}}) &= \mathbf{w}^T \mathbf{M}_{\mathrm{m}} \mathbf{v} \;\in\; [-s, 0]
\end{aligned}$$

where l is the distance between the two condyles, d is the length of the sliding tracks, and s is the sine of the maximum opening angle of the mouth. ψ_1 forces the horizontal rotation axis to be parallel to the plane of the sliding tracks, ψ_2 forces the midpoint to reside on the same plane, and ψ_3 forces it to be equidistant from the two sliding tracks. The additional constraints keep the three remaining degrees of freedom within their allowable range. ψ_4 and ψ_5 constrain the left and right condyle on their sliding tracks, and ψ_6 regulates the opening angle. Finally, the kinematic validity penalty term is

$$\begin{aligned}
\phi_{\mathrm{kin}}(\mathbf{b}) \;=\; & \phi_{\mathrm{rigid}}(\mathbf{M}_{\mathrm{c}}) + \phi_{\mathrm{rigid}}(\mathbf{M}_{\mathrm{m}}) \\
& + \psi_1^2 + \psi_2^2 + \psi_3^2 + \min\{0, \psi_4, d - \psi_4\}^2 \\
& + \min\{0, \psi_5, d - \psi_5\}^2 + \min\{0, -\psi_6, s + \psi_6\}^2
\end{aligned}$$

noting that all the piecewise quadratic terms based on ψ_1 to ψ_6 are convex functions, and therefore no adjustment of their Hessian is necessary. All the terms in ϕ_{kin} are included in ϕ_p and handled as in section 4.7.

4.9 Examples

We evaluate our system by estimating muscle activations and kinematic parameters from a set of test motion capture sequences. These include a 33 second long narration sequence (figure 4.5) and several individual examples of pronounced expressions typically 2-3 seconds long (figure 4.4), which can be compared with expressions obtained by manual specification of muscle activation levels (see figure 4.3). Our single mocap session used 79 markers, and we focused them on mouth and jaw movement as opposed to the forehead and eyes. The motion capture input was processed at a frame rate of 60 frames per second. With the possible exception of the first frame of each capture sequence, we typically used a single Gauss-Newton step followed by a golden section line search for estimating the muscle activations and kinematic parameters at each subsequent frame. The average processing time for our simulation model of 370K tetrahedral elements and 32 transversely isotropic muscles was 8 minutes per frame on a single Xeon 3.06Ghz CPU, which includes 10 quasistatic solves for the chosen search depth of the line search with full collision handling in addition to 44 linear solves for the update of the control jacobians by application of equations (4.5) and (4.6). Using linear interpolation for the initial guess given to the quasistatic solver during each golden section search refinement and using the transform of the global frame of reference to precondition the initial guesses for both the quasistatic positions and their jacobians proved to be the most important performance optimizations. The cost of computing the Gauss-Newton step itself, once the linearization of equation (4.3) had been updated, was less than one second per frame.

Between successive frames, the activations can change by as much as 20%-40% and the kinematics can experience rotations of 3-4 degrees for both the global frame of reference and that for the jaw. We stress that our approach is trivially parallelizable, as a result of our quasistatic formulation. At the expense of estimating the first of a sequence of expressions from a suboptimal guess (which typically requires 3-5 Gauss-Newton iterations), processing a long sequence of motion capture frames can be partitioned arbitrarily.

An important aspect of our approach is that the search for the optimal match

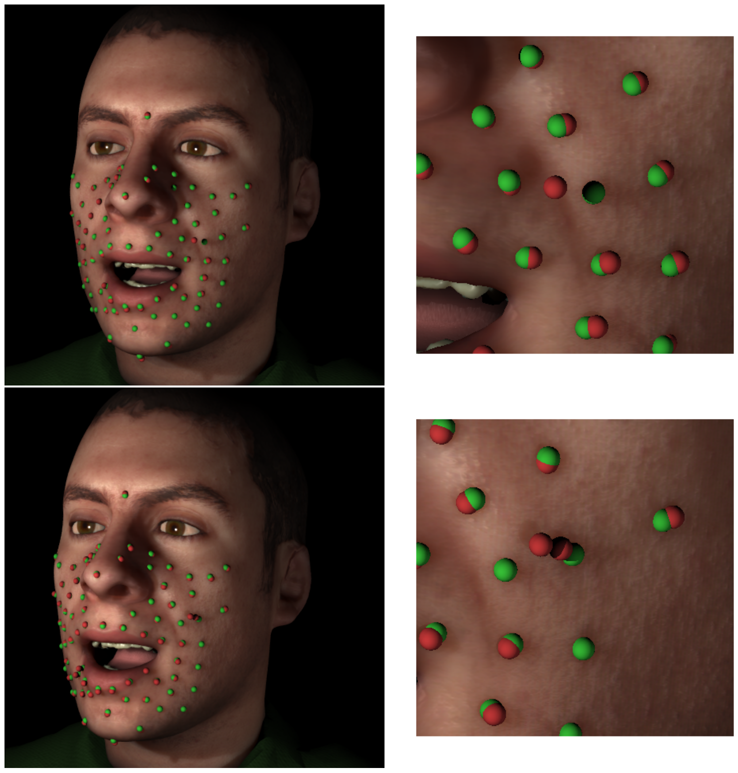

Figure 4.7: Robust handling of noisy motion capture markers.

for the motion capture markers is performed over the space of physically attainable configurations, as parameterized by the muscle activations. This results in robust handling of noisy input data or motion outliers as illustrated in figure 4.7, since their non-physical component is discarded through the optimization process. Free form deformation and shape based animation schemes do not exhibit this property, and unfiltered motion outliers incur nonphysical localized deformation.

Once the muscle activations and kinematic parameters for an input sequence have been computed, we can address a number of post-processing tasks using the extracted, physically based animation parameters. Interpolation between expressions can be performed in the muscle activation space with an automatic guarantee that the interpolated expressions are physically valid and attainable (figure 4.10). Furthermore, an expression can be exaggerated or deemphasized by multiplying the muscle activations by a scaling factor, and clamping the result within the valid activation range $[0, 1]$.

One can also exaggerate an expression (or sequence) beyond the physically attainable limits. It would be inadvisable to do so by extending the activation values beyond 1, since the force-length relationship is undefined for such values and heuristic extrapolations can be problematic. Instead, we scale the entire force-length curve effectively scaling the overall anisotropic behavior of the muscle while still maintaining plausible behavior as the muscle activation varies over the interval $[0, 1]$ (see figure 4.8). The same effect would be difficult to achieve using blending techniques as pointed out in [194].

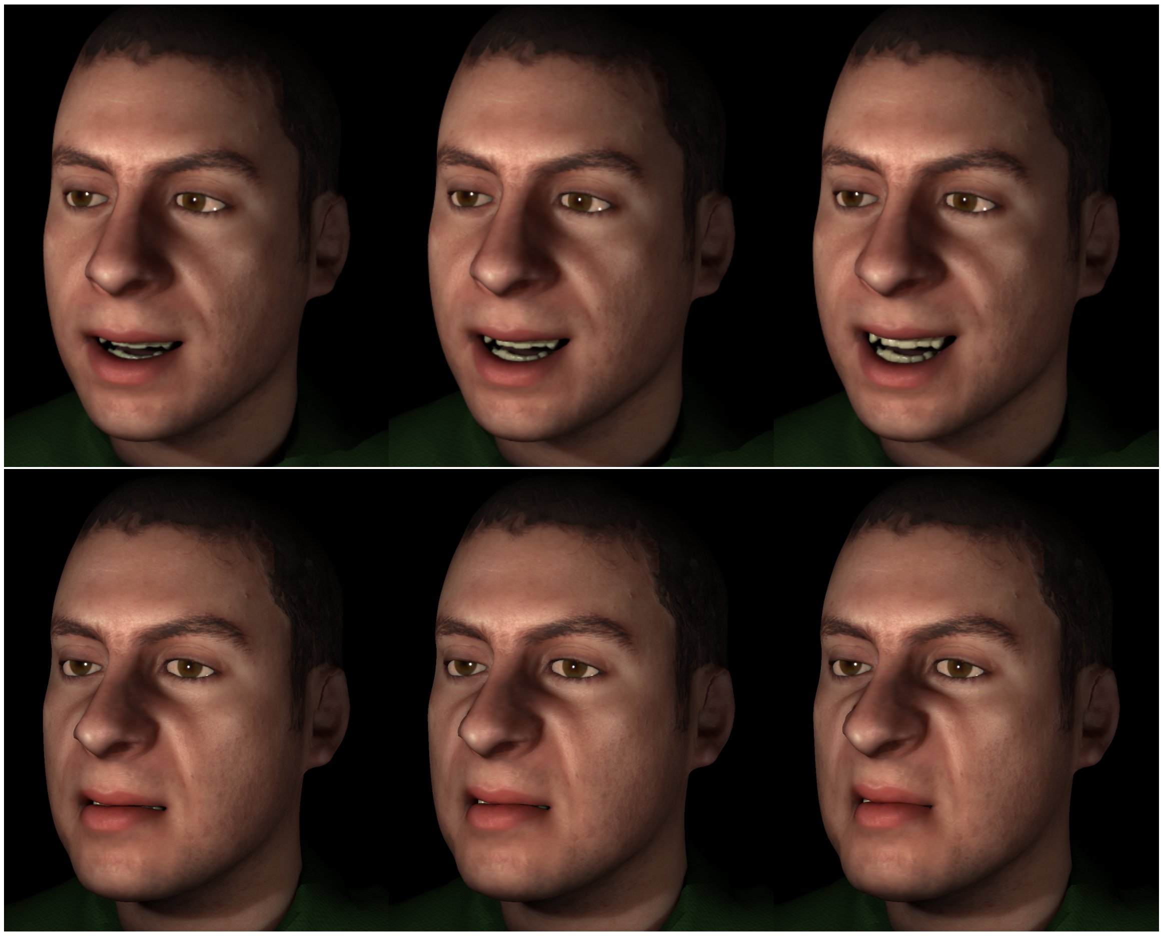

Figure 4.8: Accentuated expressions created by scaling of the force-length curve.

Our estimation process is based on a quasistatic simulation of the face which

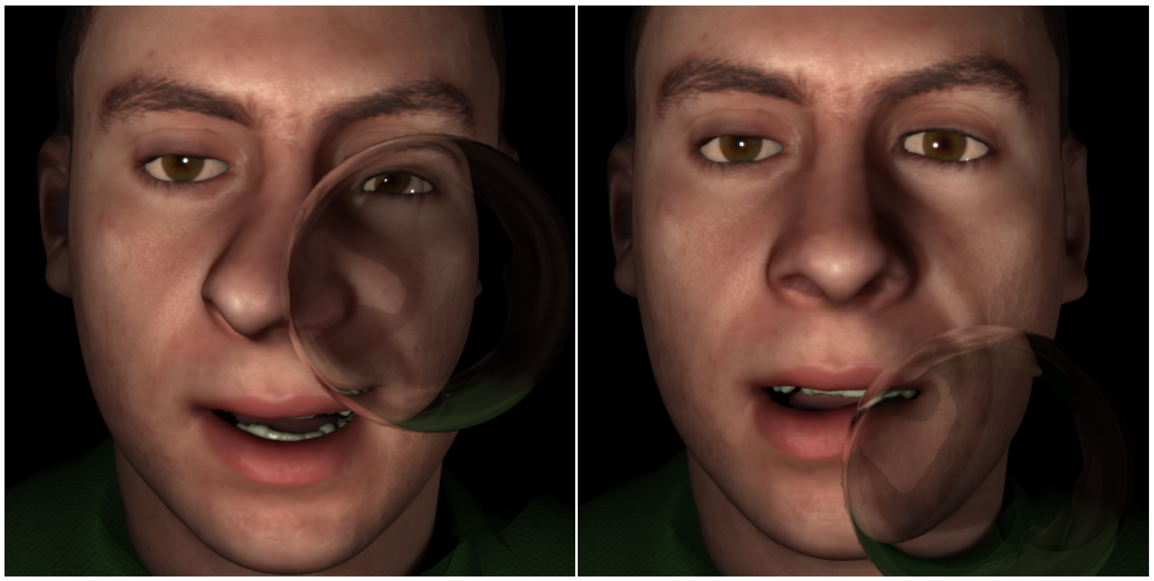

Figure 4.9: Interaction of the face with an external colliding object.

disregards inertial phenomena. The quasistatic hypothesis is consistent with the empirical fact that humans tend to avoid sudden ballistic motion of their head (e.g. this is how boxers knock each other out). When we compared our quasistatic simulation to a fully dynamic one using biologically realistic material parameters and the estimated activations and kinematic controls, the differences were unnoticeable. We had to loosen up the material parameters to get noticeable ballistic effects, e.g. softening the cartilage in the nose (as shown in the accompanying video). Even for highly dynamic motion such as a person jogging, one could still capture the actor's performance quasistatically and add the dynamics as a post process. Finally, external elements can be introduced into a quasistatic or dynamic simulation that uses the extracted parameters. For example, figure 4.9 illustrates a quasistatic simulation of the face interacting with a kinematic sphere.

4.10 Summary

We have presented an anatomically accurate face model controlled by muscle activations and kinematic bone degrees of freedom. A novel algorithm was developed to

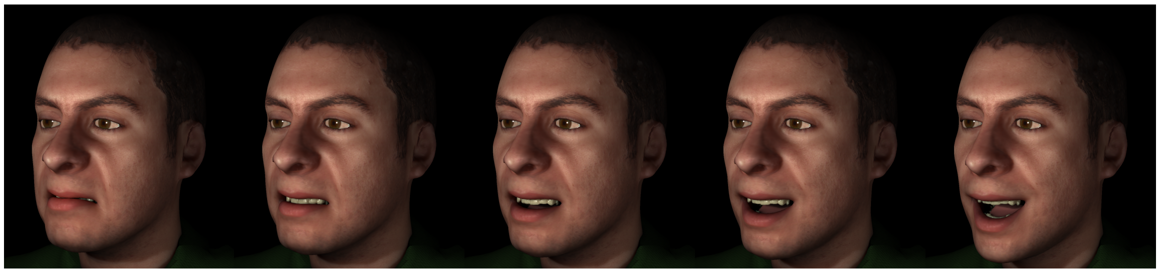

Figure 4.10: Interpolation between two expressions in activation space.

automatically compute control values that track sparse motion capture marker input. Once the controls are reconstructed, the model can be subjected to interaction with external objects, used in a dynamic simulation to capture ballistic motion, expressions can be edited in the activation space by combining multiple existing segments or making manual adjustments, etc. We are currently building an even more accurate face model from a more powerful MRI scanner and a laser scan of a highly detailed cast of the face. We are also working to obtain improved motion capture data including data for the forehead and eye region, as well as more detailed mouth and lip tracking (placing markers *on* the lips as opposed to only around them). An obvious extension would be to generalize the control estimation framework to accept markerless input data. In fact, we are currently undertaking a project that determines the muscle activations associated with the articulation of phonemes, and this will require more detailed lip motion and muscle data.

There are many application areas that we can now address including, for example, the ability to learn patient-specific muscle activations that can be used to predict the effect that surgical modifications will have on expression. A natural but highly promising research direction would be to estimate not just the controls, but also the model parameters including bone and flesh structure, material constitutive parameters, muscle locations and shapes in the rest state, etc. This would allow us to correct anatomical modeling errors and make the model more predictive in a data driven fashion. Finally, it would be interesting to analyze a large number of facial expressions, for example deriving correlations between muscle activations. In this vein, we are also working on validating our muscle activation results using electromyography.

Chapter 5

Speech synthesis

We present a physically based system for creating animations of novel words and phrases from text and audio input based on the analysis of motion captured speech examples. Leading image based techniques exhibit photo-real quality, yet lack versatility especially with regard to interactions with the environment. Data driven approaches that use motion capture to deform a three dimensional surface often lack any anatomical or physically based structure, limiting their accuracy and realism. In contrast, muscle driven physics-based facial animation systems can trivially integrate external interacting objects and have the potential to produce very realistic animations as long as the underlying model and simulation framework are faithful to the anatomy of the face and the physics of facial tissue deformation. We start with a high resolution, anatomically accurate flesh and muscle model built for a specific subject. Then we translate a motion captured training set of speech examples into muscle activation signals, and subsequently segment those into intervals corresponding to individual phonemes. Finally, these samples are used to synthesize novel words and phrases. The versatility of our approach is illustrated by combining this novel speech content with various facial expressions, as well as interactions with external objects.

5.1 Introduction

Photorealistic facial animation is both difficult to achieve and in high demand, as illustrated by [144], which discussed some of the challenges faced in recent blockbuster films and high profile research efforts. Many computer graphics practitioners are interested in animating conversation (see e.g.[36, 37]), and this has led to enormous interest in the key ingredients of speech and expression. Moreover, as stressed by [47], visible speech plays a large role in the interpretation of auditory speech.

Along the lines of talking presidents in "Forest Gump," [28] proposed a method that used existing video footage to create a new video of a person speaking novel words. [59] also proposed an image-based approach that relied on the morphing of the visemes associated with phonemes. These results were further improved using a multidimensional morphable model in [58] and used for retargeting in [40]. Although image based techniques produce animations of photo-real quality, they lack the versatility of some other approaches, e.g. it would be difficult to use them when the face has to interact with elements from the environment.

The idea of driving a three-dimensional character from text and audio (as in voice puppetry [27]) is quite compelling. Data driven approaches tend to use motion capture data (see e.g. [189, 76]) to drive a three dimensional surface mesh. [33] took this approach using independent component analysis to separate speech from expression (see also [32]). Similarly, [98] used PCA of marker data to determine facial movement parameters. [46] used a bilinear model that separates expression from speech in order to drive a three dimensional blend shape animation with video input. In a similar vein, [53] constructed a speech co-articulation model that can be mixed with keyframing in a manner that preserves expressiveness. [180] used multilinear models to separate expressions, visemes and identity in a three dimensional data set, enabling video to drive a three dimensional textured face model. While these methods have enjoyed recent popularity, especially for speech and visemes, they lack any anatomical or physically based structure, limiting their potential for accuracy and realism.

Even though [184] advocated the use of muscles rather than surfaces for speech animation early on, physically based simulation methods have not enjoyed popularity

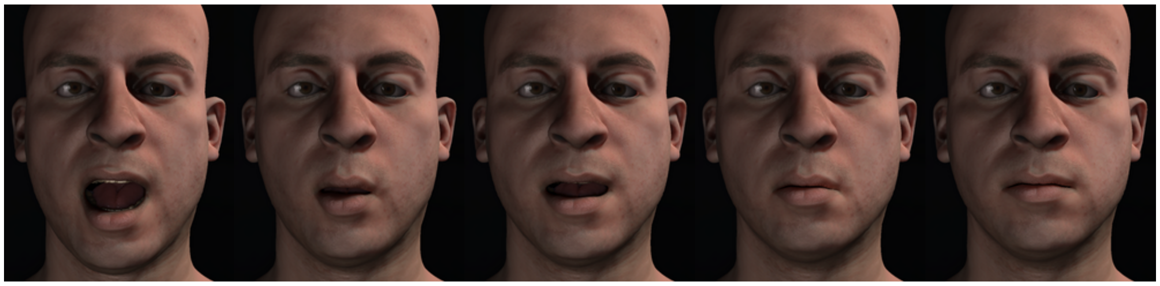

Figure 5.1: A synthesized utterance of the word “algorithm”

for phoneme or viseme research (as pointed out e.g. in [148]). This could be due to the high computational cost associated with the level of fidelity required to study speech. Although there is precedent for estimating muscle contraction parameters from video [171, 172] (see also [56, 57, 115]), [13, 12] avoided the internal anatomy altogether using only a surface based finite element model while studying lip motion. In fact, recent work in this area includes [43, 42] which conclude that it might be better to estimate linear combinations of sculpted basis elements rather than muscle activations. However, [155] argued that the limiting factor for fidelity was not the use of simulation per se, but rather the lack of *realistic* muscles with biomechanical nonlinearity and anatomical accuracy. Furthermore, they argue that a person’s face *is* driven by muscle activations, therefore an anatomically faithful model with the control granularity of actual human facial muscle exactly spans the space of facial expression.

Both image-based animation methods and data-driven surface deformation techniques have traditionally been preferred over physics-based approaches for facial animation. Both approaches operate directly on sample data without requiring an intricate anatomical facial model or the overhead of simulation for either analysis or synthesis. Yet, interacting with the simulated character in ways that are not spanned by the recorded training data is recognized as a task that lies beyond the scope of either approach. Physics-based approaches provide the unique ability to interact with the character in any way that can be physically prescribed while respecting the fundamental characteristics of a performance, namely motion style, expression and verbal content.

Following the approach of chapter 4 we build a high resolution, anatomically and biomechanically accurate flesh and muscle model of a subject's face. Then we automatically determine muscle activations based on three-dimensional sparse motion capture marker data. In particular, we focus on the capture of speech, constructing a phoneme database parameterized in muscle activation space. Notably, each phoneme is stored with temporal extent. We demonstrate that physically based approaches can be used for speech analysis *and* synthesis by creating animations of novel words and phrases from text and audio input. Moreover, we capture muscle activations representative of expression and show that these can be mixed with the speech synthesis to independently drive speech and emotion. Finally, we illustrate the versatility of a physically driven three dimensional model via interaction with foreign objects.

5.2 Previous Work

Early work on three dimensional facial animation includes [138, 146, 185, 108, 91] (see also [55]). [179] relate skin deformation of a physics based model to oral tract deformation while [107, 106] use a physics based model driven by muscle-control signals acquired by AMG and compare surface deformation against the human subject. Based on scanned data, [101] constructed an anatomically motivated, biomechanical facial model featuring a multilayer, deformable skin model with embedded muscle actuators. [96] used finite elements to predict emotions on a post-surgical face (see also [95, 149] for finite elements for facial surgery). [51] used variational modeling and face anthropometry techniques to construct smooth face models, [143, 145] animated faces based on photographs and video, and [87] worked on automatic segmentation for blending. The face was also divided into subregions for the facial animation in [195]. [20] proposed a vector space representation of shapes and textures for animation transfer [18] and face exchange in images [19]. [88] built a muscle based facial model and considered morphing to other faces [90] and forensic analysis [89]. [94] used a parametric muscle model with time varying visemes to extend the coarticulation algorithm of [47]. [31] added expressiveness to the MPEG-4 Facial Animation Parameters.

A number of authors have worked on facial motion transfer [129, 147, 124, 163]. [38] used tracking to drive animations from a motion capture database, [183] tracked facial motion with a multiresolution deformable mesh with the aim of learning expression style, and [194] proposed a face inverse kinematics system.

5.3 Data Capture

5.3.1 Model Building

We constructed a high-resolution volumetric model of facial flesh and musculature for both our analysis of speech samples and the synthesis of new utterances. First, we obtained an MRI scan which provided an approximation of the tissue extent and the shape of the interface between soft tissue and bone. Then a life-mask cast of the subject was scanned at a resolution of 100 microns, producing a 10 million triangle model and a fully registered texture map. The detail from this high resolution surface scan was integrated into our volumetric flesh model. The facial flesh volume was discretized into a 1,870,000 element tetrahedral mesh, with 1,080,000 elements in the frontal facial volume that was used to simulate deformation under action of the facial muscles. Due to the limited resolution of the MRI scan, much of the internal tissue structure was manually adjusted to create a muscle set that conforms to the anatomical prototypes published in the medical literature. Our model includes 39 of the muscles that are predominantly involved in facial expressions and speech. Muscles that have no effect or only a subtle effect on facial motion were excluded, as their behavior would not be reliably captured with our surface motion capture marker set.

5.3.2 Motion capture

We take a data-driven approach to speech synthesis constructing a database of prototypical subject-specific utterances of speech primitives (sample phonemes within a context of words or phrases). The motion component of these utterances was recorded with a motion capture system consisting of 8 cameras with 4MP CCD sensors. 250 thin circular patches of retroreflective material with a diameter of 3mm were placed on the subject's face at an average distance of 8-10mm apart. A small subset of markers were specifically placed on predominantly rigid parts of the head to capture the rigid head motion. The performance was sampled at 120Hz. See Figure 5.2.

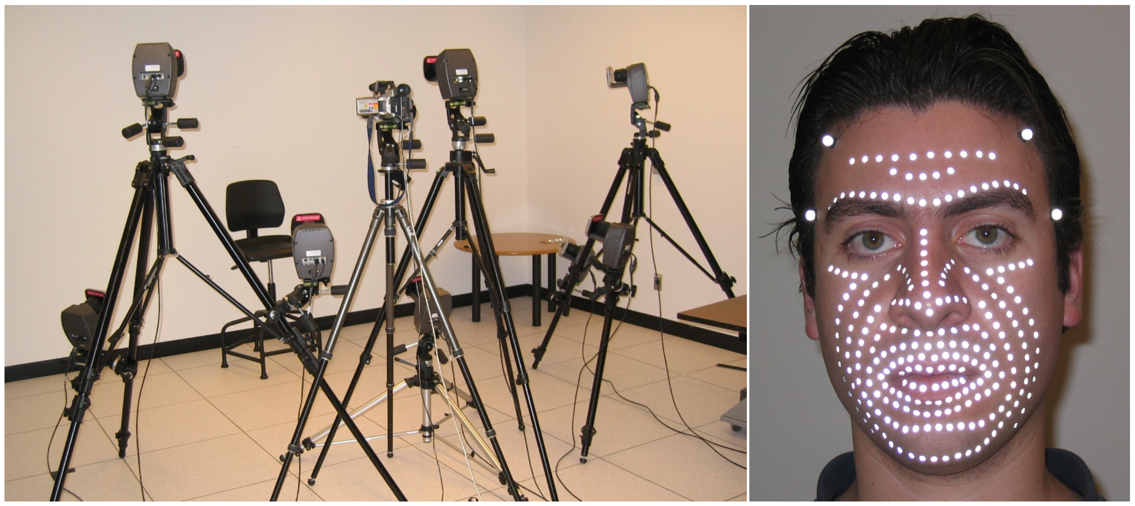

Figure 5.2: Eight camera, 250 marker optical motion capture layout

5.3.3 Inverse activations

Following [155] we model the isotropic response of passive fatty tissue by a hyperelastic Mooney-Rivlin constitutive model for the deviatoric component, with an additional volumetric pressure component for quasi-incompressibility. The parameters of the Mooney-Rivlin model are spatially adapted to the heterogeneity of the simulated tissues, yielding different stiffness values for areas occupied by collagen, cartilage, and tendinous structures. Areas of the flesh that are occupied by contractile muscle tissue are further assigned an anisotropic strain response corresponding to the passive or active behavior of muscle tissue along the direction of its fiber field. The inverse activation estimation framework employs the quasistatic simulation method of chapter 3. This formulation uses fast conjugate gradients solvers to evolve constrained deformable objects to an equilibrium state, and provides robust handling of mesh degeneracies such as element inversion, as well as rigid body and self-collision handling. We should point out that this quasistatic assumption is preferred for the estimation process as it greatly simplifies the inverse control problem. While it can also be used for the forward simulation of slow speech, a fully dynamic simulation method is superior for the simulation of faster speech from muscle activation controls.

5.4 Phonemes and Visemes

5.4.1 A Muscle Activation Basis for Speech

The inverse activation framework described in section 5.3.3 allows us to translate our database of motion captured speech samples into temporal sequences of control parameters for our deformable face model (i.e. muscle activations and kinematic configuration of the bones). We subsequently use these controls as the parameterization of facial motion for analysis and synthesis tasks.

A defining property of visual speech synthesis techniques is the choice of the feature space used to describe facial motion. Common examples found in the literature (cited above) include image-based descriptions and surface shape bases. Our approach provides the versatility to edit the animated performance affecting the emotion and expression of the character, as well as allowing physical interaction of the face with objects from the environment. In this context, the relevant feature is not the appearance or the shape of the face per se, but rather the *action* of speech articulation. Therefore, we follow the formulation described in chapter 4 using the activation signals that stimulate the facial muscles as our feature space, an approach that was pioneered in [171, 172].

Our approach is subject to a number of limitations. The quality of our parameterization and the fidelity of the resulting simulations are only as good as the detail and accuracy of our muscle-driven model as well as the physical consistency of the simulation method used. This highlights the need for detailed, nonlinear, volumetric finite element models of the anatomical components of the face. Additionally, our adoption of a quasistatic simulation scheme for analysis leads to a deviation from the true dynamic behavior expected of a physical system. However, for our training set

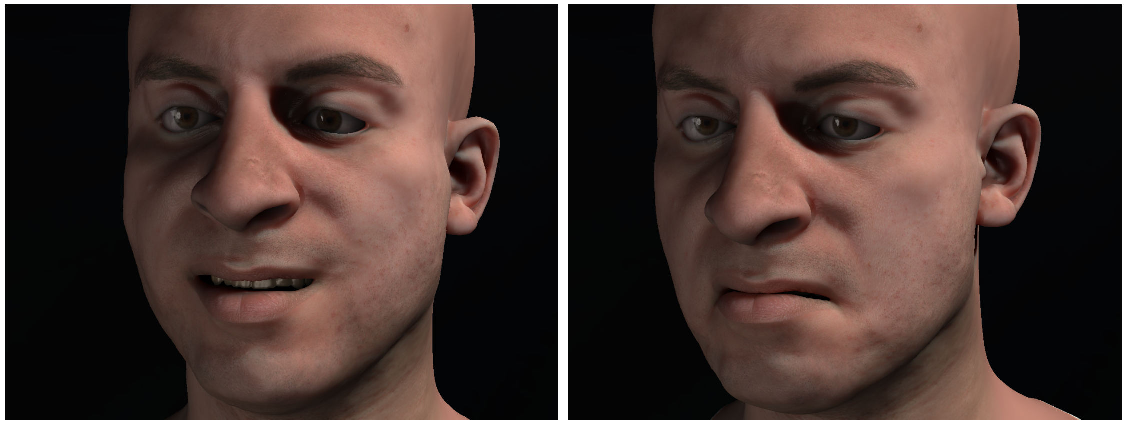

Figure 5.3: Estimated muscle activations of expressions.

of short words spoken at a casual pace, inputting the estimated muscle activation sequences into a forward quasistatic simulation produced a very close match to the original capture respecting almost all nuances of individual utterances. This supports our use of quasistatics for the estimation of muscle activations, although a full dynamic simulation would be superior for synthesis (especially for faster speech).

5.4.2 Primitives of Speech Simulation (Physemes)

We collected a database of motion capture data for the phoneme sets suggested by [5] and [59] where phonemes are presented within the context of sample words. We recorded 4-5 distinct captures of each phoneme set and used the inverse activations estimation process to convert each captured word into a short sequence of muscle activation signals and mandible articulation parameters. An examination of the signals corresponding to various phonemes revealed several important patterns. First, we observed a high degree of correlation between segments of words that contained the same phoneme, as illustrated in Figure 5.5 for samples of the phonemes **p** and **w** (we adopt the phoneme codes used in [17]). The temporal extent of this correlation varied with the particular phoneme being considered and its phonetic context. Phonemes with matching context (the phonemes immediately before and after the one in question) tended to correlate over a much longer time segment. In addition,

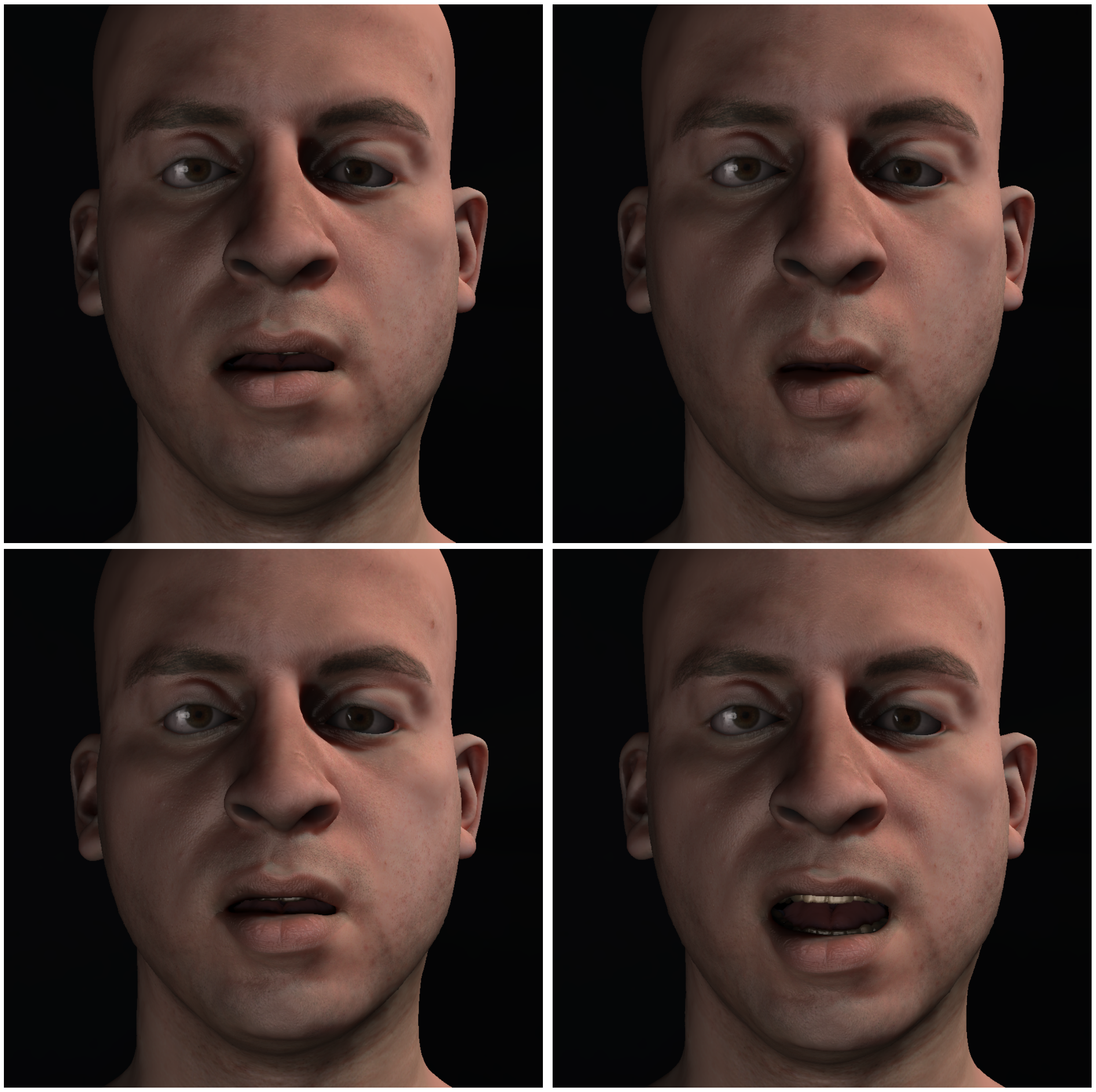

Figure 5.4: Estimated phonemes from motion captured examples.

several phonemes (such as the consonants **sh**, **v**, **z**) typically reached a steady state in activation space, surrounded by the transitions from the previous and to the next phoneme. Other phonemes (such as the diphthongs **ay**, **ey**) exhibited a characteristic *dynamic* pattern over their temporal extent, often marked by a distinctive transition. In Figure 5.5 we classify **p** as a dynamic phoneme (we can identify 3 distinct stages of mouth closure, lip retraction and mouth opening) while the static **w** appears to achieve a steady state in between transitions. We note that muscles in the oral region typically exhibited higher degree of correlation across utterances of the same phoneme than peripheral muscles.

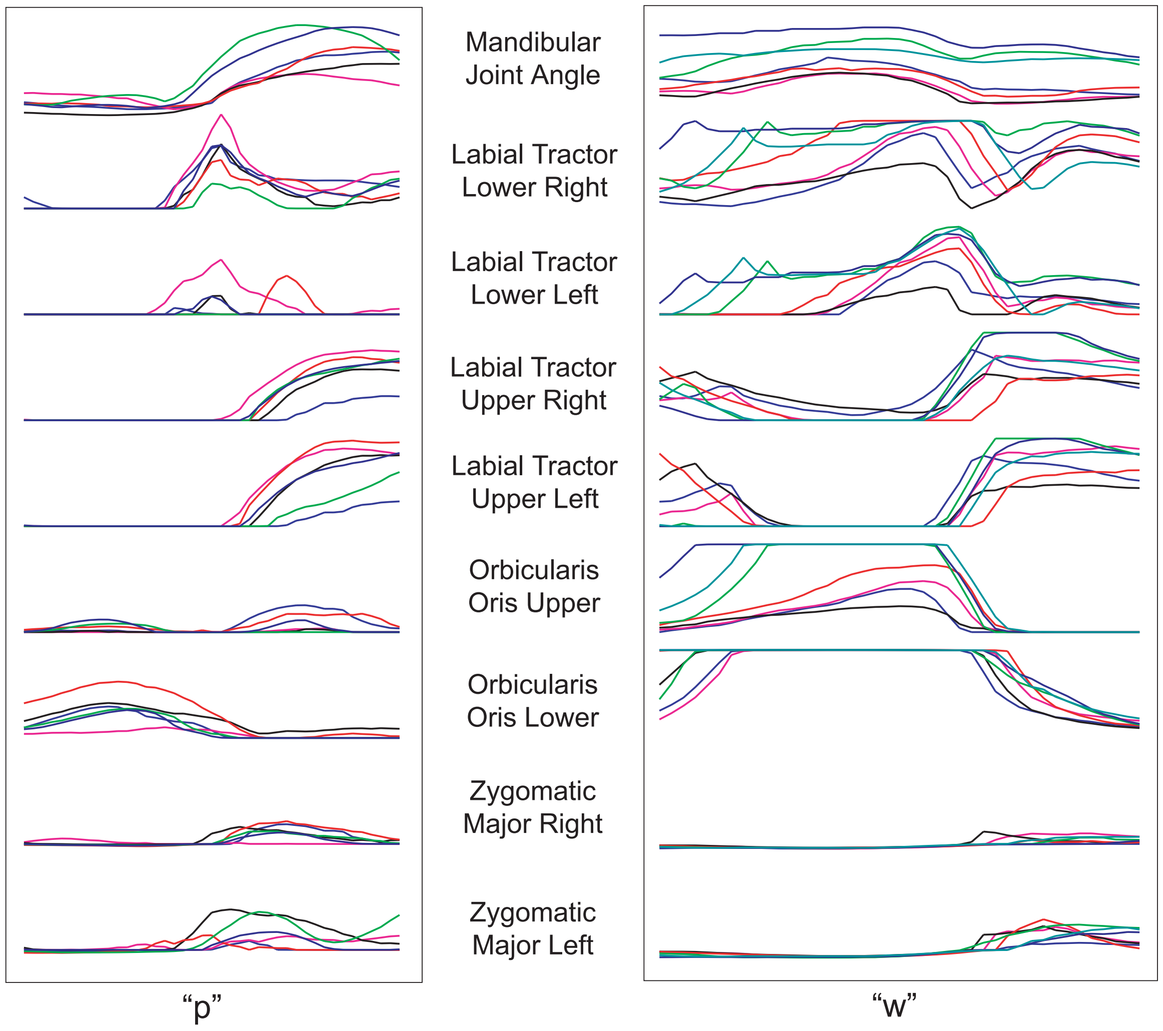

Figure 5.5: Comparison of the "p" and "w" physemes.

These observations support the hypothesis that *time-varying sequences of muscle activations* capture a large amount of information about phonemes and phoneme transitions. In particular, by recording the muscle activation signals over a time interval that extends beyond the duration of each phoneme and into its neighboring ones we capture the effect that the utterance of each phoneme has and receives from its context, formally known as *coarticulation*. Therefore, we associate each of these extended intervals of muscle activations and bone kinematics with its corresponding phoneme and use them as the primitives of our physics-based visual speech synthesis, labeling them as *physemes* (in analogy to phonemes and visemes).

To create a database of physemes we use the audio track to identify each phoneme, using the Festival Speech Synthesis System [17] to segment utterances into individual phrases and then into individual phonemes. The labelings were not completely accurate and sometimes manual annotation was also required. Once every word had been partitioned into time segments corresponding to different phonemes, physeme samples were collected by selecting the estimated muscle activation signals corresponding to the time range of each phoneme (see Figure 5.6) and padding the signal on each side of the time segment with enough data from its context in order to capture the coarticulatory effects.

5.5 Synthesis

5.5.1 Physeme-based Speech Synthesis

Our physeme database captures the motion signatures of phonemes and the transitional effects between them in the physically motivated space of muscle activation. We present a first approach to using this physeme basis directly for synthesis of visual speech. The input to our systems consists of an audio recording along with a transcript of its verbal content. We again use Festival [17] to segment this novel audio track into time intervals corresponding to distinct phonemes. The result is typically satisfactory for utterances of individual words or slow speech. However, sometimes with longer and faster speech passages it was necessary to manually adjust the phoneme annotation.

After we determine the constituent phonemes of the text to be synthesized, we assemble a matching temporal arrangement of physemes from our database. Each physeme contains muscle activation signals that extend beyond the duration of its associated phoneme, namely it starts with a *lead-in* from the previous phoneme, followed by the *body* corresponding to its base phoneme and a *lead-out* from the following phoneme. We place physemes in arrangements with their bodies contiguous and their lead-in and lead-out overlapping into the body of the adjacent physeme (see Figure 5.7). Silent intervals are modeled with a special arbitrary length "pause" physeme, with muscle activations corresponding to the neutral pose of the face. In general, the length of each phoneme in an audio recording will not match the length of the corresponding physeme in our database, so the muscle activation signal of the physeme is time-scaled to the appropriate length. A single, uniform time scaling is applied to the body as well as the lead-in/out of the physeme.

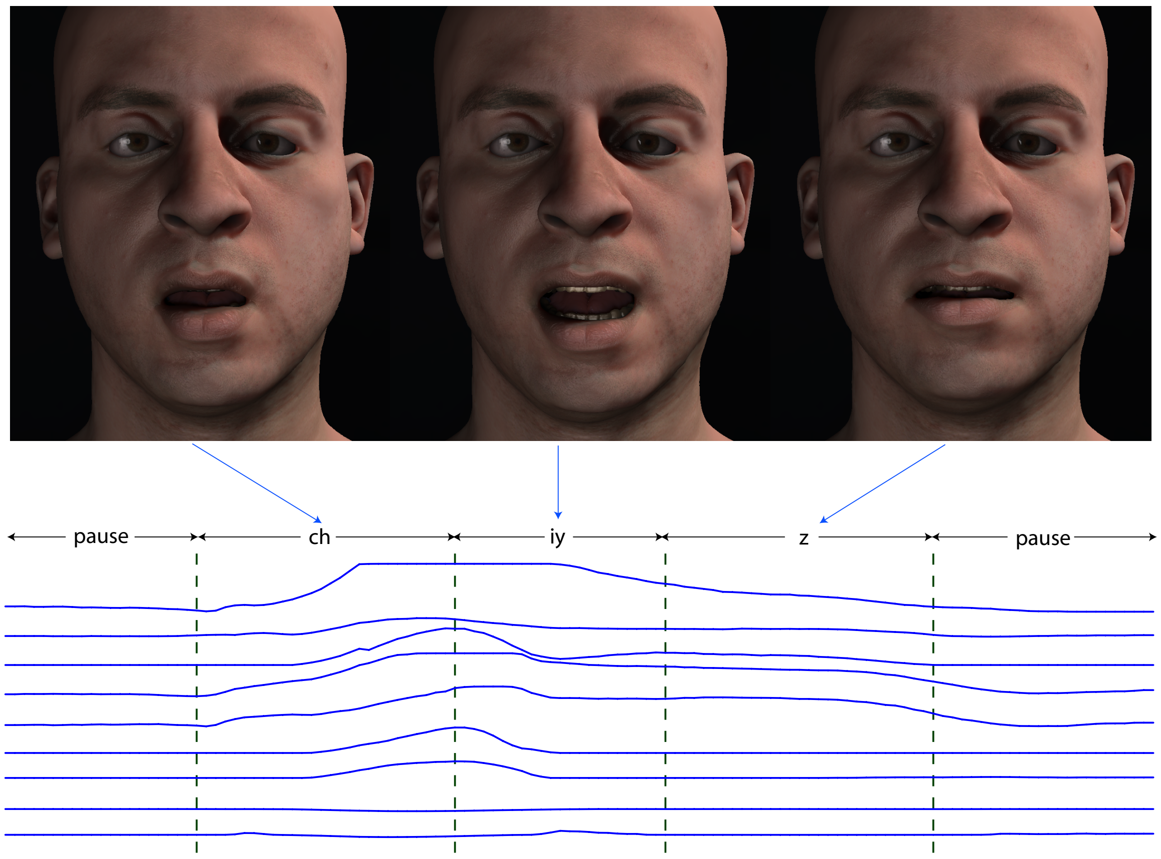

Figure 5.6: Segmentation of the word "cheese" into constituent physemes.

For each physeme inserted in an arrangement we use a blending curve that yields constant weights equal to unity throughout the body of the physeme and decays to zero at the outer endpoints of the lead-in and lead-out following a C^1 continuous sigmoid curve. We extract a single muscle activation signal from the complete physeme arrangement by performing weighted averaging of the signals overlapping at any instance in time using their corresponding blending weights, as illustrated in Figure 5.7.

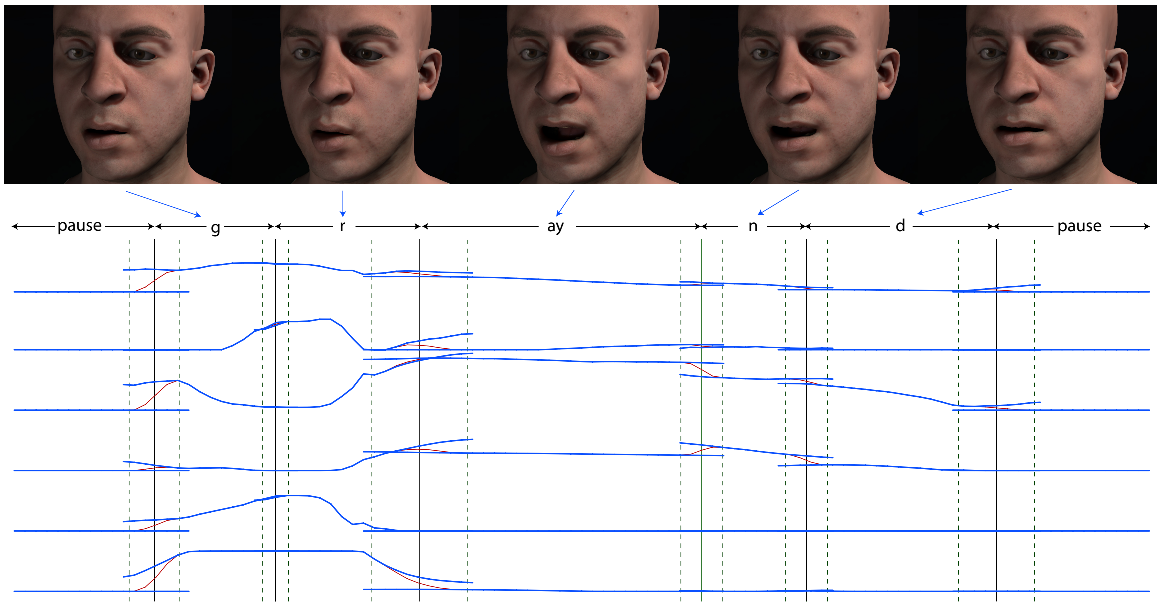

Figure 5.7: Arrangement of the word "grind" synthesized from computed physemes.

Among the kinematic parameters that are obtained through the inverse activation estimation process, the parameters that define the overall rigid body motion of the head (or the frame of reference of the cranium) receive special handling. Interpolating between different positions and orientations in such short time intervals as those corresponding to phoneme lengths would most likely incur sudden jumps and violent accelerations. Thus, instead of interpolating between frames of reference, we use the estimated rigid body motion for each individual physeme and use it to approximate the linear and angular velocity of the head at each frame. When blending phonemes we then proceed to blend linear and angular velocities instead of positions and orientations. The rigid body configuration is then obtained by integrating the linear and angular velocities forward in time. The resulting signals of muscle activations and rigid bone kinematics can be fed into a forward quasistatic or dynamic simulator to produce the final physically driven speech simulation.

5.5.2 Sequence Generation

We employ a semi-automatic interface for the creation of physeme-based simulations of speech including tools for the creation and refinement of physeme arrangements, preview of an approximate speech synthesis and final physics-based finite element simulation. Given the existence of tools such as Festival that simplify the segmentation of an audio speech signal into its constituent phonemes, we focus on the task of compiling a physeme arrangement to match a given, labeled phoneme sequence. The low dimensionality of our feature space (39 muscle activations and 3 kinematic parameters, a few tens of phonemes per sentence) makes optimization algorithms such as stochastic optimization attractive, i.e. since we avoid the overhead typically associated with them in higher dimensional spaces.

We adopted the constraints that the labels of the physemes have to match the labels of the phonemes that occupy the same time range in the audio recording, and we clamped the lead-in and lead-out of each physeme to 20% of the length of the body of the neighboring phoneme it overlaps with. Under these constraints, our free parameters are the choice of which of the 5-30 physemes from our database for each phoneme should be used to fill a particular time interval. We formulated a criterion for the quality of a particular phoneme arrangement and solved it using simulated annealing, using parameters that yield the global minimum with very high confidence. We obtain our quality criterion by computing the magnitude of the discrepancy between the muscle activation vectors for all frames where physeme extents would overlap and integrating over time. In order to prevent weak muscles or muscles that have little direct effect in the articulation of speech to dictate the quality of an arrangement, we scale the activation a_i value by the average magnitude of the quasistatic shape

Jacobian $\partial \mathbf{X}/\partial a_i$, computed over the entire range of motion captured visemes (as a by-product of the inverse activation estimation process). This biases the quality criterion towards accounting for muscles whose activation tends to have a more substantial effect on the shape of the face.

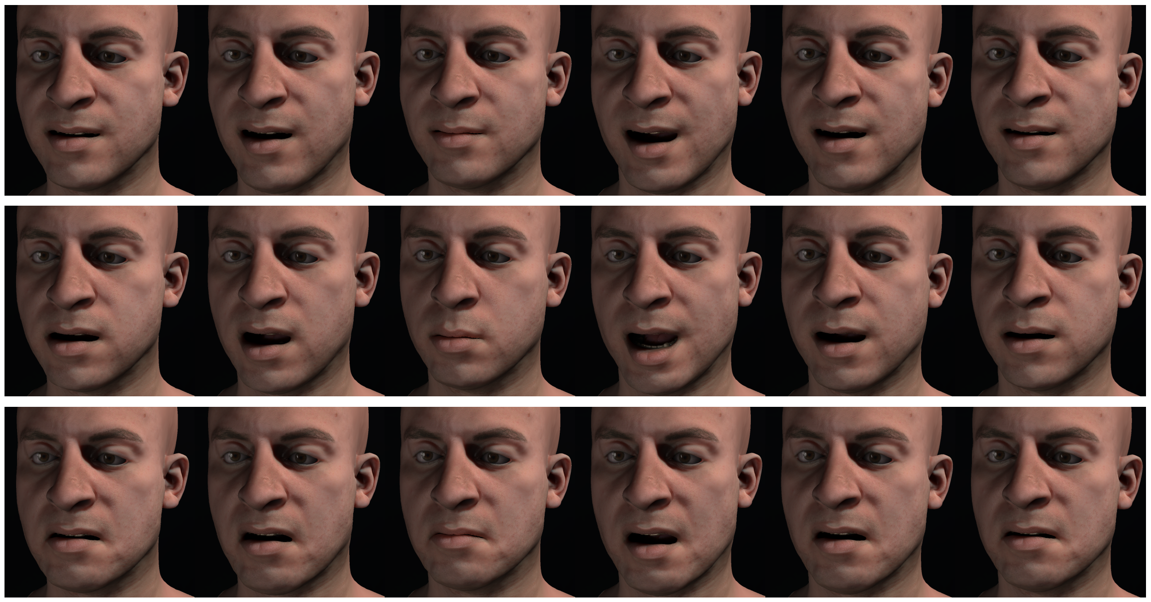

Figure 5.8: Synthesized speech segments blended with expressions.

Our optimization process provided convincing physeme arrangements for simple examples (such as single, slowly spoken words) requiring little to no manual intervention. However, with more complicated examples or faster speech this result often required manual adjustment, such as fine-tuning the length of the lead-in/out segments or the precise placement of phoneme boundaries. We created a graphical interface that provided us with the functionality to alter all these parameters, as well as the individual choice of physemes used at each moment in time. As a by-product of our inverse estimation process, we possess a quasistatic face shape approximation for each frame in our captured training set. By blending these face shapes with the same weights used for physeme blending we obtain a fast preview for our edits without the need for simulation, which is run only when our adjustments are complete. On average, editing a medium-sized sentence would typically entail 2-3 hours of manual processing. We note that quasistatic simulation contributed substantially to that cost, since the lack of damping and inertia made the final result more sensitive to the muscle activation input signal than it would be with a full dynamic simulation.

5.6 Speech and Expression

The versatility of a physically-based muscle driven face model for speech synthesis is highlighted by the ability to augment the simulation with elements that are secondary to the process of speech articulation. Facial expression and emotion are characteristic examples of such elements. Although there exists a correlation between the emotional appearance and the verbal content of human speech, a human speaker may adjust his facial expression independent of the words spoken. We simulate this process by motion capturing facial expressions and using our inverse activation process to convert these expressions into characteristic muscle contractions. Subsequently, we blend these muscle activation values into our synthesized physeme sequences and use physics-based simulation to obtain the final animation.

As illustrated in Figure 5.8, the integration of expressions such as a frown or a smile can be performed in a very natural manner, through simulation, without compromising either the articulatory content or the emotional response elicited by the expression that was blended in. Such an augmentation is straightforward and requires no manual adjustment of the physeme arrangement. It should be emphasized that it is much more challenging and labor intensive to obtain such a result, in regard to *both* speech and expression, with a technique based on blending images or face shapes. The nonlocal effects of pronounced facial expressions, in conjunction with anatomical phenomena that arise from these expressions (such as bulging of skin, deepening of facial furrows, or changes in the contact pattern of the lips) become particularly difficult to capture convincingly without a physically-based approach.

5.7 Speech and Physics

Beyond the task of enriching the facial motion with an expression, the real power of physics-based approaches is revealed when the face is required to interact with

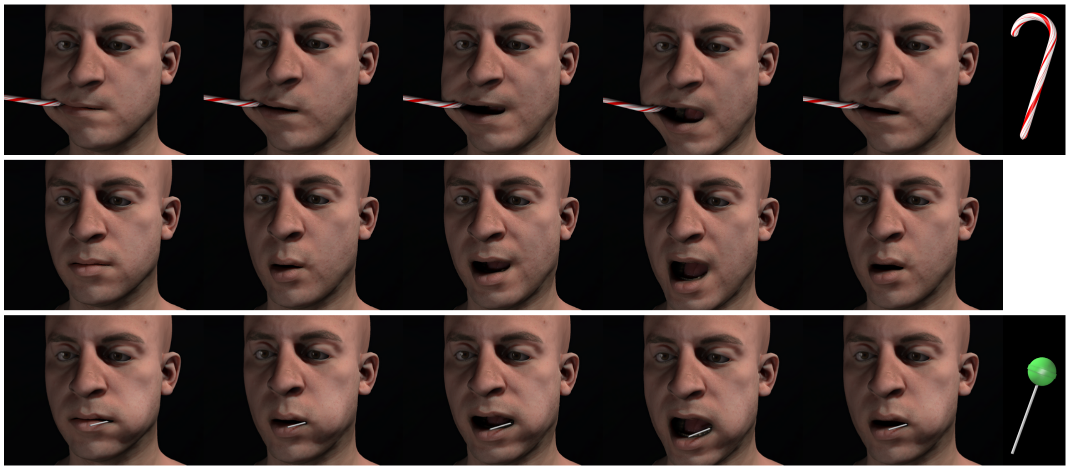

Figure 5.9: Synthesized speech segment augmented with external object collision.

the outside world. Physical simulation of a full volumetric facial musculature model allows us to produce effects that are difficult or impossible using image-based or data-driven surface deformation techniques. By keeping the muscle activation controls fixed and modifying the simulation environment, we can effortlessly produce a new facial deformation. In particular, once a speech database is created, reproducing such effects does not require additional motion capture data, analysis or modification of our synthesized physeme sequences. For example, we depict our virtual character speaking with a lollipop and candycane in his mouth in Figure 5.9, where the muscle activation signals were synthesized without regard to the object interaction.

5.8 Discussion

Using our quasistatic estimation framework, we processed approximately 10 minutes of speech at 120Hz in an average of 7 minutes per motion capture frame (including full analysis with full rigid body and self-collision handling) on a Xeon 3.8 Ghz CPU. The full processing of approximately 70,000 frames required the equivalent of a CPU year on clustered computer hardware. Although seemingly high in computational cost, this once-only process requires no human supervision and *all* the resulting muscle activation signals were of adequate quality for use in our database. Notably, our simulation model contained 1080K simulation elements, which is at least 3 times larger than typical high resolution finite element simulations in the computer graphics literature. Therefore, we expect our model to age much slower than the advance of computer hardware, making the computation affordable.

Once the facial model has been created and the physeme database has been assembled from the motion captured performance, the main labor-intensive effort is the manual adjustment of the physeme sequences to fine-tune the synthesized speech result. Currently at a cost of a few hours per sentence, the bulk of this effort is attributed to correcting mistakes of the speech analysis software (the Festival system) and adjustment of the transition intervals between successive phonemes. The latter would be substantially easier if a full dynamic framework was employed for the final forward simulation, as the physics of facial motion would handle physeme transitions in a smooth way alleviating issues with blending intervals. While this editing cost might seem high compared to data-driven methods, especially if one is only concerned with casual speech, it becomes much more competitive when one requires the added versatility of the face interacting with its environment. Our method can achieve physically based environmental interactions with almost no additional cost, whereas the cost increases substantially for data-driven approaches even though the quality incurs a significant decrease.

5.9 Summary

We presented a physics-based approach to visual speech synthesis using an anatomically accurate, muscle actuated finite element model of the human face. We collected motion capture data for the utterances of words in a training set and converted them into the time varying muscle activation signals that give rise to the captured face motion. We segmented these muscle activation signals into time segments corresponding to different phonemes and assembled them into a database of *physemes*. We create sequences of physemes with smooth transitions between them to match the phonemes of audio recordings of new speech, and use the resulting muscle activation signals to drive a finite element simulation of facial motion. The animation is readily enriched by blending in expressive emotions or by introducing external objects that interact with the talking face.

Our adoption of a quasistatic simulation scheme was motivated by the tractability of the muscle activation estimation framework of chapter 4. When our estimated activations were used in a quasistatic forward simulation, the results compared well to the original video visually validating our approach to estimation. However, for the reanimation of a novel synthesized physeme sequence, quasistatic simulation was only satisfactory for sequences of slow speech, where ballistic motion and pronounced inertial effects are not significant. When used with muscle activation signals created for faster speech, quasistatic simulation gave rise to rather abrupt and underdamped motion that lacked realism. Inertial effects have a profound effect on the motion of a real human face under such conditions, smoothing out phoneme transitions and smearing away the dynamics of individual utterances. We believe that a full dynamic simulation is the obvious choice for animations of such synthesized sequences of fast speech and discourage the use of quasistatics for forward simulation whenever possible. The practice of using quasistatics for the inverse problem and full dynamics for forward simulation is also a well established practice in the field of biomechanics, where it is well understood that even in cases where the estimated actuations are not smooth or even discontinuous, the simulated deformation using a full dynamic scheme is smooth and realistic due to the progressive fashion in which muscle activations translate to tissue deformation.

Our key objective for our future work is the illustration that a fully dynamic simulation of the synthesized muscle activation signals successfully treats demanding, fast-paced speech passages. Increased realism could be obtained by improving eyelid, eyebrow and forehead motion as well as modeling the effect of airflow in the oral cavity accommodating effects such as cheek puffing and improving the appearance of closed-mouth phonemes (such as p and b). Improved training sets will enable us to better capture the dynamics of speech in different contexts than that of short, slowly spoken phrases. The current work constitutes only a first step in using the muscle activation basis for speech analysis, and this compact, complete and physically motivated description provides the potential to improve several analysis techniques such as Principal Component Analysis by allowing them to operate in a space that is much more tightly bound to the physical process of speech articulation rather than in a space of facial appearances or skin shapes.

Chapter 6

Hybrid solids

Although mesh-based methods are efficient for simulating simple hyperelasticity, maintaining and adapting a mesh-based representation is less appealing in more complex scenarios, e.g. collision, plasticity and fracture. Thus, meshless or point-based methods have enjoyed recent popularity due to their added flexibility in dealing with these situations. Our approach begins with an initial mesh that is either conforming (as generated by one's favorite meshing algorithm) or non-conforming (e.g. a BCC background lattice). We then propose a framework for embedding arbitrary sample points into this initial mesh allowing for the straightforward handling of collisions, plasticity and fracture without the need for complex remeshing. A straightforward consequence of this new framework is the ability to naturally handle T-junctions alleviating the requirement for a manifold initial mesh. The arbitrarily added embedded points are endowed with full simulation capability allowing them to collide, interact with each other, and interact with the parent geometry in the fashion of a particle-centric simulation system. We demonstrate how this formulation facilitates tasks such as arbitrary refinement or resampling for collision processing, the handling of multiple and possibly conflicting constraints (e.g. when cloth is nonphysically pinched between two objects), the straightforward treatment of fracture, and sub-element resolution of elasticity and plasticity.

6.1 Introduction

Simulation of deformable models was pioneered in computer graphics by [170, 168, 169]. It has influenced research in areas such as cloth animation [10], muscle simulation [167], flesh deformation [35], facial motion [155], virtual surgery [160] and fracture [133].

Mesh-based methods require the generation of an initial simulation mesh, which can be conforming (e.g. [113, 3]), or non-conforming when used in conjunction with an embedded simulation technique (e.g. [34, 35, 112, 119, 123]) in which case a simple cube or BCC background mesh can be used. Generation of a conforming simulation mesh is typically non-trivial and is best suited to applications that require no changes to the initial mesh. However, if the mesh needs to be adapted on the fly, e.g. for fracture [133], remeshing can be prohibitively expensive and can introduce poor quality elements. [112] proposed a partial solution that uses embedded simulation technology to fracture tessellated objects without continuous remeshing, allowing elements to be modeled as partially full of material. Limitations, including the restriction that edges cannot be fractured twice, prevent this method from being completely general. Collision processing can often require more surface resolution than is present in the initial simulation mesh. While some have considered adaptive frameworks [49, 74, 35], it is wasteful to refine the full volumetric mesh in the vicinity of the boundaries solely for collisions. Although not as efficient as mesh-based methods for the simulation of elastic deformation, point-based methods provide added flexibility making them attractive for applications involving fracture, virtual surgery, resampling for collision handling, etc., see e.g. [121, 140, 161, 187, 120].

Starting with either a conforming or non-conforming mesh, we propose a method for embedding an arbitrary point in this mesh. We call this a *hard binding*. To derive the relationships between physical quantities of the embedded point and the mesh, we start by considering a hypothetical refinement of the mesh that resolves the embedded point. We compute internal finite element forces for these hypothetical subelements and illustrate how to redistribute these forces to the parent mesh. The mass of the embedded particle is redistributed to the parent mesh as well. This redistribution is

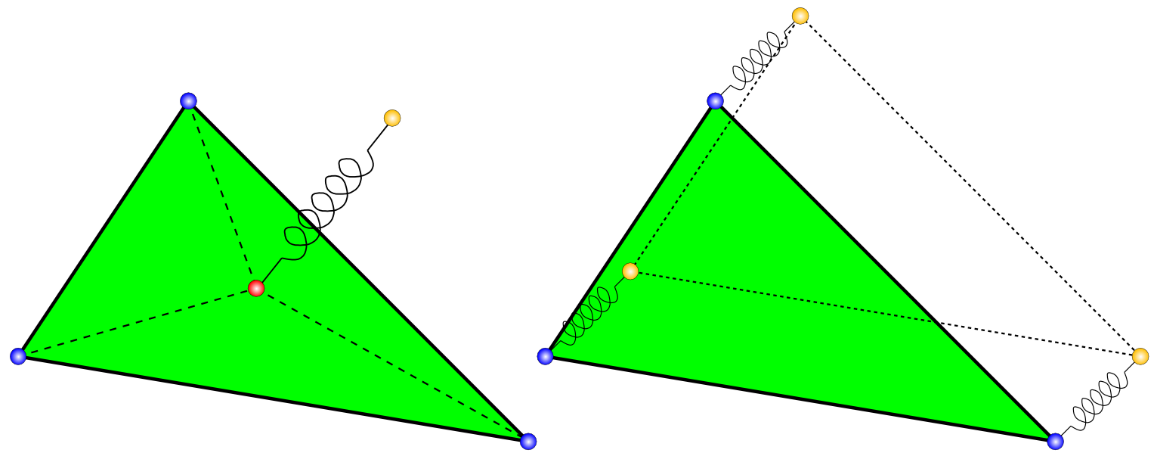
Figure 6.1: Parent particles, hard bound target locations and soft bound particles.

shown to be independent of the chosen refinement. Notably, this approach results in automatic and natural handling of T-junctions, as masses and forces of T-junction nodes can be redistributed to the parent mesh. This specific handling of T-junctions is similar to that in [105]. We then generalize this approach to arbitrary embeddings. Hard bound particles have their mass and any forces or impulses applied to them redistributed to the parents in a natural fashion while maintaining the notion of an *effective mass* so that they can fully participate in various numerical algorithms.

A hard binding constrains an embedded particle to its barycentrically determined location (a similar heuristic was given in [70, 71]). Thus, hard bound particles are not particles at all (i.e. they do not possess any degrees of freedom), but merely target locations that live on the parent mesh. In order to transition to a fully particle-centric simulation framework, we create the notion of a *soft binding*. A soft binding is an abstract connection between a real particle with full degrees of freedom and a hard binding target location enabling full two-way interaction. Soft bound particles are free to interact with each other and the parent mesh constituting a fully particle-centric framework. Soft bindings can be created between any particle and any target location even duplicating the original degrees of freedom in the mesh itself (see figure 6.1). The soft binding mechanism is responsible for the two-way interaction between the particle-based system and the mesh-based framework. Although one could implement this connection using simple springs, we have designed a more sophisticated soft binding interface which is notably fully implicit allowing one to stiffen the two-way interaction to the point where the soft bound particle always lies on its target location up to numerical precision, without stability issues or additional time step restrictions.

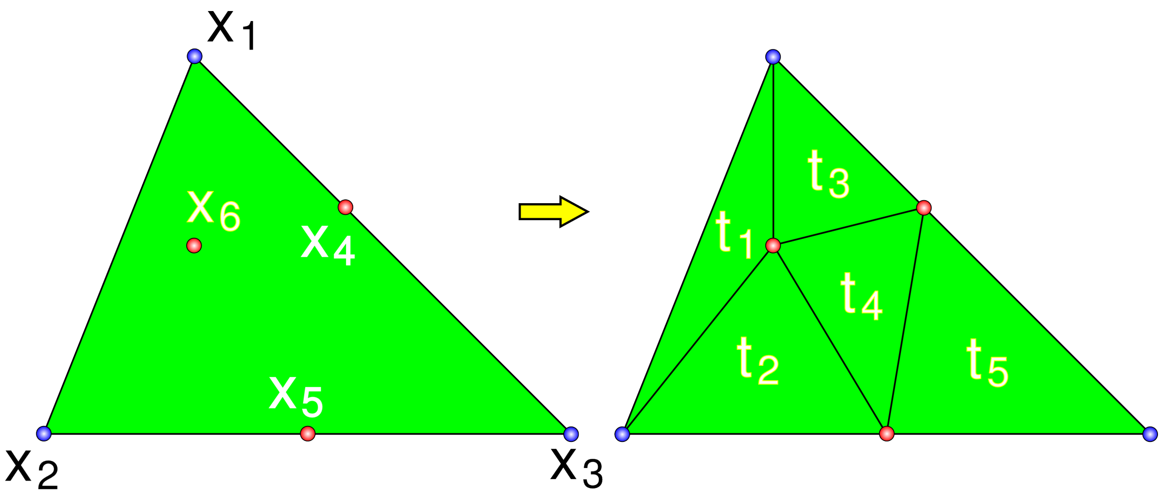

Figure 6.2: Resolution of hard bound particles via refinement of the parent element.

Our work shares the motivation of [70, 71] to incorporate particle constraints in the simulation of deformable objects, and the function of their geometric constraints is conceptually analogous to our hard bindings. However, their approach works by introducing external forces dependent on the integration method used whereas our hard bindings are enforced through the conjugate operations of force distribution and velocity interpolation. That is, instead of forcing our constrained degrees of freedom toward their target locations, we directly project them out of the equations of motion. This allows seamless incorporation of such constraints into any time integration scheme, including globally coupled implicit schemes. Another major difference is that our framework allows optional drift of particles away from their target locations through soft bindings.

Novel contributions of our work include the tight integration of hard binding constraints and unconditionally stable binding spring forces into the semi-implicit Newmark time integration scheme, lag-free duplication of degrees of freedom through soft bindings by force transfer from parent particles, and integration of rigid/deformable coupling into the Newmark scheme and conjugate gradient solver. We present these contributions as part of a broad hybrid simulation framework building on simple, physically motivated principles. We demonstrate the features of this framework with examples that include dynamically adapting the surface sampling density for collisions, duplicating parent mesh particles to resolve conflicting constraints caused by the nonphysical pinching of cloth, the facilitation of fracture and cutting algorithms, and an extension of our framework to two-way interactions with rigid bodies.

6.2 Hard Bindings

The simulation mesh is subject to internal forces, from for example finite elements, as well as external forces from gravity, friction, collisions, etc., and we want these forces to act on the hard bound particles as well. We motivate our approach for propagating forces to the parent mesh by considering a refinement that resolves all the hard bound particles. Figure 6.2 shows three hard bound particles along with an associated refinement. Although this refinement is not unique, it turns out that the propagation of physical properties from the hard bound particles to the parent mesh is independent of the tessellation used and can be performed without explicitly refining the parent element. Internal forces are defined on the subelements in standard fashion, but must be remapped to the parent particles since the hard bound particles are not free to move independently.

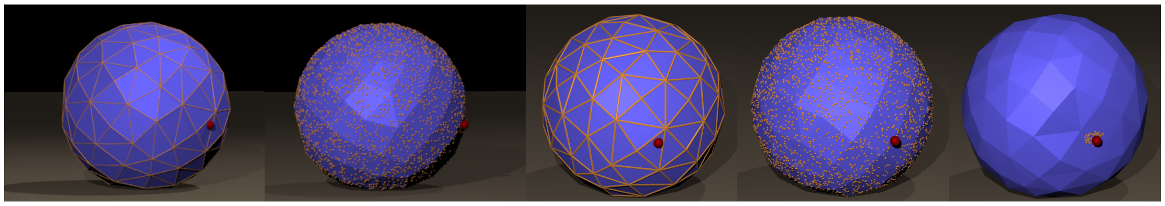
Figure 6.3: Hard bindings used to improve surface collision resolution

Conservative forces can be defined in terms of the gradient $\mathbf{f} = -\partial\Psi(\mathbf{x})/\partial\mathbf{x}$ of the potential energy $\Psi(\mathbf{x})$. The potential energy of the parent triangle is $\Psi = \sum \Psi_i$, where Ψ_i is the potential energy of subtriangle t_i. Each Ψ_i is naturally defined in terms of the positions of the vertices of triangle t_i rather than the positions of the parent particles, although the former are fully determined by the latter through the binding constraint. Let $\mathbf{x}_i$ denote the positions of the vertices of triangle t_i, e.g. $\mathbf{x}_2$ is composed of the positions of particles $\{x_2, x_5, x_6\}$ in figure 6.2. The parent triangle vertices $\{x_1, x_2, x_3\}$ are represented by $\mathbf{x}$. Using this notation, the three forces on the parent particles are

$$\mathbf{f} = -\sum_i \frac{\partial \Psi_i}{\partial \mathbf{x}} = -\sum_i \left(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}}\right)^T \frac{\partial \Psi_i}{\partial \mathbf{x}_i} = \sum_i \left(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}}\right)^T \mathbf{f}_i \tag{6.1}$$

where $\mathbf{f}_i = -\partial \Psi_i / \partial \mathbf{x}_i$ are the three vertex forces computed on each subtriangle t_i. The Jacobian $\partial \mathbf{x}_i / \partial \mathbf{x}$ is constant for each triangle t_i and contains the barycentric coordinates of $\mathbf{x}_i$ with respect to the vertices $\mathbf{x}$ of the parent triangle. Equation (6.1) can be extended to the computation of elastic force differentials for use in implicit or quasistatic time integration via

$$\delta \mathbf{f}|_{\delta \mathbf{x}} = \sum_i \left(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}}\right)^T \delta \mathbf{f}_i|_{\delta \mathbf{x}_i} \tag{6.2}$$

Equation (6.2) is obtained by taking the directional derivative of (6.1), assuming that the Jacobian $\partial \mathbf{x}_i / \partial \mathbf{x}$ is constant (i.e. the binding constraint is linear), which is always the case for the barycentric embeddings used here. Nonlinear binding constraints would result in additional terms in equation (6.2).

From equations (6.1) and (6.2) we can write the net force and force differential on a single parent particle as

$$f_k = \sum_{t_i} \sum_{x_j \in t_i} w_k^j f_j^i \quad \text{and} \quad \delta f_k|_{\delta \mathbf{x}} = \sum_{t_i} \sum_{x_j \in t_i} w_k^j \, \delta f_j^i|_{\delta \mathbf{x}_i} \tag{6.3}$$

where w_k^j is the barycentric weight of particle j with respect to particle k of the parent triangle and f_j^i is the force on particle j from subtriangle t_i. In practice we implement equation (6.3) by accumulating all forces on both parent and hard bound particles from all their incident elements and subsequently redistributing the force on each hard bound particle to its parents weighted by its barycentric weights. Note that the final force distribution does not depend on the tessellation used but only on the barycentric weights of the hard bound particles.

We use equation (6.3) for velocity dependent damping forces as well noting that it preserves the symmetry and definiteness of the linear damping forces allowing for our semi-implicit time integration framework. We consider the linear damping model $\bar{\mathbf{f}} = \bar{\mathbf{G}}\bar{\mathbf{v}}$ where $\bar{\mathbf{G}}$ is symmetric negative definite, and the force $\bar{\mathbf{f}}$ and velocity $\bar{\mathbf{v}}$ refer to *all* particles, including hard bound particles. The velocities of this extended set of particles are expressed in terms of the velocities of the parent particles, $\bar{\mathbf{v}} = \mathbf{W}\mathbf{v}$. Using this notation, equation (6.1) reduces to $\mathbf{f} = \mathbf{W}^T\bar{\mathbf{f}} = \mathbf{W}^T\bar{\mathbf{G}}\mathbf{W}\mathbf{v} = \mathbf{G}\mathbf{v}$ where $\mathbf{G} = \mathbf{W}^T\bar{\mathbf{G}}\mathbf{W}$ preserves both symmetry and negative definiteness of the original damping matrix $\bar{\mathbf{G}}$. A similar derivation shows that the distribution scheme for force differentials given in equation (6.3) preserves symmetry and negative definiteness of the elastic stiffness matrix, which is an important property for schemes that use an implicit or quasistatic treatment of elastic forces as well.

After forces have been computed and redistributed to the parent particles, accelerations are computed via $\mathbf{a} = \mathbf{M}^{-1}\mathbf{f}$ where $\mathbf{M}$ is diagonal in a typical lumped mass formulation. We store the lumped masses of the parent particles in a vector $\mathbf{m}$, which is the diagonal of $\mathbf{M}$. In analogy to our force derivation, we write the mass vector as the gradient of total momentum with respect to velocity, i.e. $\mathbf{m} = \partial P/\partial \mathbf{v}$. Let $\bar{\mathbf{m}}$ be the masses of all particles. As is typical, one could compute the initial mass of hard bound particles by refining the parent mesh as in figure 6.2 and assigning one third the mass of each triangle to its vertices, although any scheme for assigning mass (including uniform mass) is allowed. The total momentum is then given as $P = \bar{\mathbf{m}}^T\bar{\mathbf{v}}$, thus $\mathbf{m} = \frac{\partial}{\partial \mathbf{v}}P = \frac{\partial}{\partial \mathbf{v}}\left(\bar{\mathbf{m}}^T\bar{\mathbf{v}}\right) = \frac{\partial}{\partial \mathbf{v}}\left(\bar{\mathbf{m}}^T\mathbf{W}\mathbf{v}\right) = \mathbf{W}^T\bar{\mathbf{m}}$ indicating that masses of hard bound particles should be redistributed to the parents based on the barycentric weights, i.e. the same as for force redistribution. More generally in a non-lumped mass formulation, the mass matrix is computed as the Hessian of the kinetic energy

$$\mathbf{M} = \frac{\partial^2 K}{\partial \mathbf{v}^2} = \frac{\partial^2}{\partial \mathbf{v}^2}\left(\frac{1}{2}\bar{\mathbf{v}}^T\bar{\mathbf{M}}\bar{\mathbf{v}}\right) = \frac{\partial^2}{\partial \mathbf{v}^2}\left(\frac{1}{2}\mathbf{v}^T\mathbf{W}^T\bar{\mathbf{M}}\mathbf{W}\mathbf{v}\right) = \mathbf{W}^T\bar{\mathbf{M}}\mathbf{W}$$

Although hard bound particles have no mass or momentum, an *effective mass* is useful for many numerical algorithms. We define the effective mass as the ratio of an applied force to the resultant acceleration of the hard bound particle, i.e. $f_e = m_e a_e$. Denoting the binding weights by w_i, the applied force is distributed to the parent particles via $f_i = w_i f_e$ and the acceleration of the bound particle is $a_e = \sum w_i a_i$. Combining these equations gives

$$\frac{f_e}{m_e} = a_e = \sum w_i a_i = \sum w_i \frac{f_i}{m_i} = f_e \sum \frac{w_i^2}{m_i} \Rightarrow \frac{1}{m_e} = \sum \frac{w_i^2}{m_i} \tag{6.4}$$

The effective mass is used in the determination of stability restrictions for forces defined on the hard bound particle, e.g. if a hard bound particle is connected to a spring, the effective mass would be used to compute the harmonic mass.

One often needs to apply impulses to hard bound particles, e.g. for collisions. An impulse j_e applied to a hard bound particle is the result of a force f_e acting on the particle for an infinitesimal interval Δt, i.e. $j_e = f_e \Delta t$, and from equation (6.3) we obtain $j_i = w_i j_e$. Furthermore, changes in velocity are governed by $\Delta v_i = j_i/m_i = w_i(m_e/m_i)\Delta v_e$, and displacements follow $\Delta x_i = w_i(m_e/m_i)\Delta x_e$ assuming that $\Delta x_e = \Delta t v_e$.

Figure 6.3 illustrates the utility of hard bindings in processing collisions. A typical approach to collision processing is to collide the points (possibly only the surface points) of a deformable object with collision geometry, and we collide each tetrahedral mesh node from the sphere's surface with the ground and the kinematically controlled red sphere. Although ground collisions are sufficiently resolved, the kinematically controlled red sphere passes directly through the deformable object, missing any potential collisions with surface particles. Although one might adaptively refine tetrahedra near the colliding red sphere, this increases the total elements that need to be simulated and the smaller edge lengths lead to a stiffer time step restriction. Our hard binding framework allows us to simply sprinkle particles on the surface of the sphere at a higher density than the tetrahedral mesh for the sake of collisions. This can be done statically or even adaptively based on proximity to collision objects.

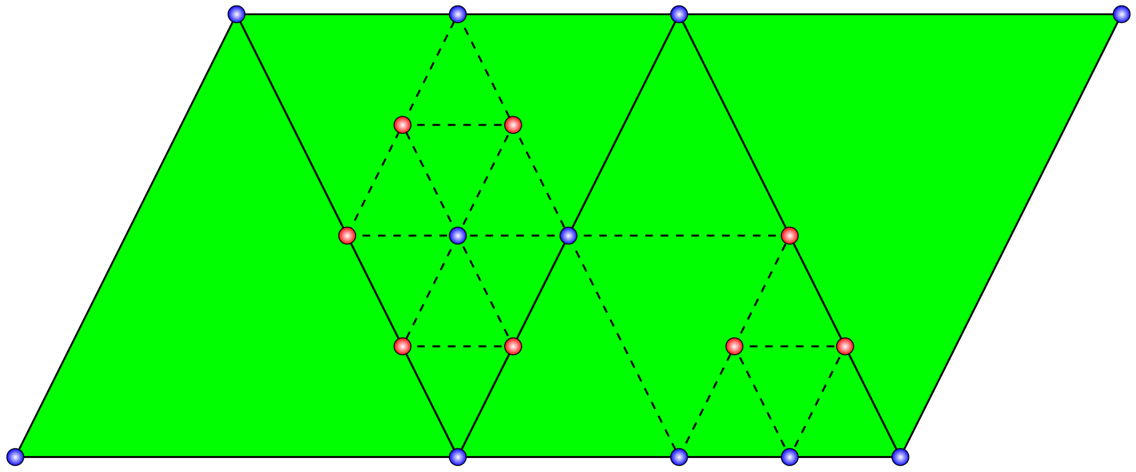

Figure 6.4: Non-graded red refinement leading to T-junctions.

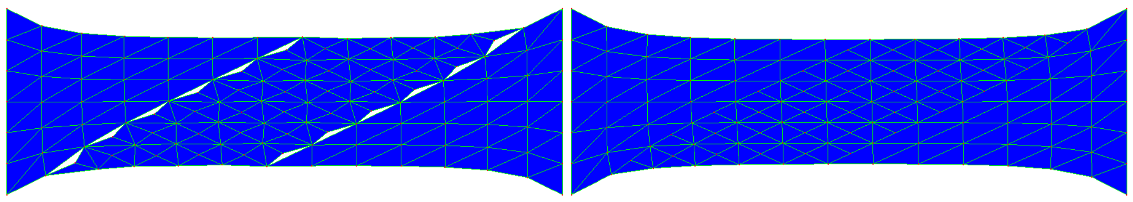

Figure 6.5: An elastic sheet with T-junctions is stretched with and without bindings.

6.2.1 T-junctions

Figure 6.4 depicts four coarse triangles undergoing up to two levels of non-graded red refinement leading to a number of T-junctions. Structured adaptive meshing schemes such as [113] resolve these T-junctions via a combination of red and green refinement. This produces both more elements increasing computational cost as well as lower quality green elements which adversely affect the time step restriction. Additional savings may be realized by exploiting the regularity of adaptive red-only refinements where all elements in the mesh are identical up to rotation and scaling requiring only the storage of a scale factor per element to encode the rest state (see appendix D.2. For every T-junction, we compute its parents by recursively tracking the endpoints of refined segments. This leads to the barycentric embedding of hard bound particles on segments or triangles in two spatial dimensions, and on segments, triangles or tetrahedra in three spatial dimensions. Figure 6.5 illustrates that the straightforward simulation of a T-junction mesh leads to gaps in the mesh (top), while treating T-junction particles as hard bound particles properly constrains them to their incident edges (bottom). We note that T-junctions have also been treated using hierarchical bases (see e.g. [74]).

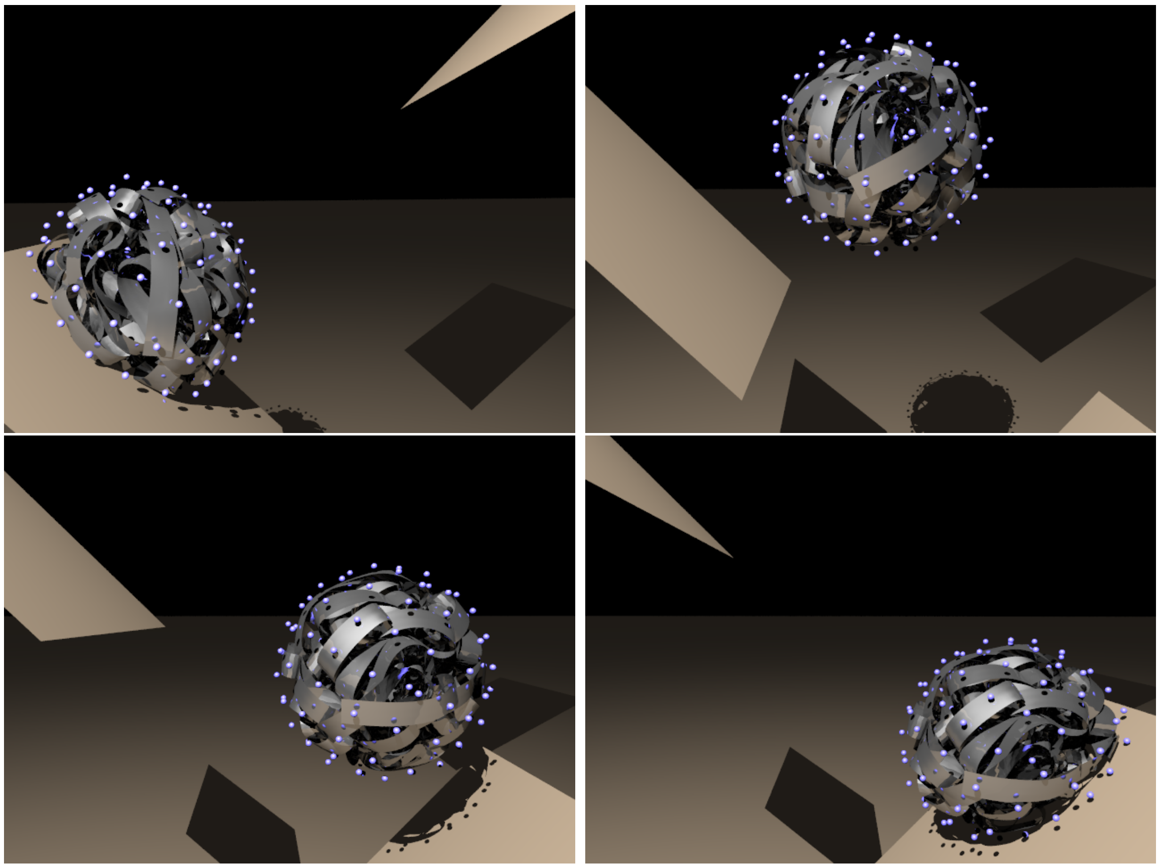
Figure 6.6: Triangulated surface embedded in point cloud.

6.2.2 Arbitrary embeddings

Although our framework is described in the context of a parent simulation mesh outfitted with a number of embedded particles that will be extended into a fully particle-centric simulation framework (in section 6.3), here we give an example to illustrate that our framework is actually more general than this. That is, we switch from the notion of a parent simulation mesh with an embedded particle-centric simulation system to a parent particle simulation system with an embedded mesh. In the context of free-form deformations, one might use a nearest neighbor approach to compute a mesh-free discretization of internal forces on the point cloud, and subsequently move the embedded geometry kinematically. Of course, this can lead to potentially severe distortions of the embedded geometry, especially since the point cloud sampling of deformation is very different from that perceived by the embedded geometry. This can be alleviated by computing the internal forces on the embedded mesh itself, as we do in the elastic ribbon shown in figure 6.6, and then mapping the resultant forces to the point cloud. Using these forces, we still time integrate the point cloud particles and kinematically enslave the embedded geometry, but the deformation of space is now sampled more adequately for the purpose of simulating the embedded geometry. Note that the kinematic motion of the ribbon is dictated by the k closest parent particles of the point cloud.

6.3 Soft Bindings

The hard bound particle framework is limited because the hard bound particles are kinematically constrained to the parent mesh rather than possessing real degrees of freedom. Thus, specialized algorithms would be required for collision processing, etc. We overcome these issues by introducing the notion of a *soft binding* which couples a new, fully simulation capable particle with an existing parent particle or hard bound particle target location as illustrated in figure 6.1. We can then ignore hard bound particles for the purpose of collisions and apply collision processing uniformly to parent particles and soft bound particles.

Under no external influence such as collisions or soft bound particle intrinsic forces, soft bound particles should follow their target positions. This is achieved by applying the force that leads to a matching acceleration between the soft bound particle and its target, i.e. the soft bound particles *inherit* elasticity and similar forces from the parent mesh. If the acceleration of a hard bound particle is $a_e = \sum w_i a_i = \sum w_i(f_i/m_i)$, the force that is applied to the soft bound particle in order to match this acceleration is $f_s = m_s \sum w_i(f_i/m_i)$ where m_s is the mass of the soft bound particle. Note that we only use this process to remap elastic forces, since applying it to damping forces would compromise the symmetry of the coupled damping matrix used in our implicit time integration scheme.

The mass of each soft bound particle is set to the *effective mass* of its target (which is just the mass for parent particles). This duplication of mass does not change the effective total mass of the object as measured by applied forces because our mapping of forces from the parent mesh to the soft bound particles properly accelerates those particles. However, a mass assignment stemming from a consistent physical principle would be desirable.

The coupling strategy between soft bound particles and their targets depends on whether short term or long term drift is desired. Therefore we give separate algorithms for short term and long term drift below.

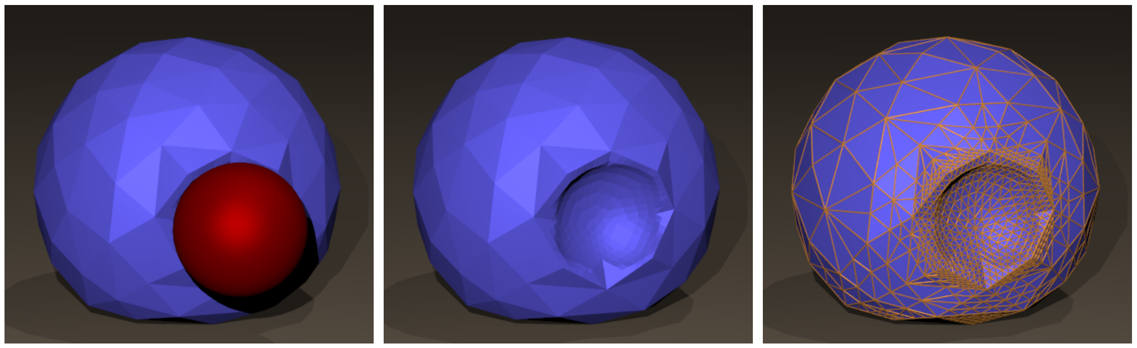
Figure 6.7: Soft bindings enable subtetrahedron elasticity in response to collision.

6.3.1 Synchronized coupling

The most straightforward way to transfer information from the target particle to the soft bound particle is by directly copying the target state. At first glance, this appears to nullify the desired properties of soft bound particles effectively reducing them to hard bound particles. However, in contrast with hard bound particles which are kept consistent with their target state at all times, soft bound particles are synchronized to their target state at specific points of the time integration loop allowing drift between two subsequent synchronizations. For example, if the soft bound particles are synchronized to the target states just before collision processing and subsequently allowed to deviate from that state as a result of collisions, we can use the unsynchronized state to obtain a detailed subtetrahedron visualization of the collision, as shown in figure 6.7. Apart from collision processing, additional forces may also be used to cause drift in soft bound particles, e.g. a soft bound copy of the surface of a volumetric object could be endowed with shell elastic forces. We subsequently propagate any drift from their target locations back to their parents, prior to the next synchronization of soft bindings with their target state.

To propagate drift from a soft bound particle to its target, we compute the discrepancy in position $\Delta x = x - x_e$ and velocity $\Delta v = v - v_e$, where x_e and v_e are the position and velocity of the (possibly hard bound) target particle. If the target particle is part of the parent mesh, its position and velocity are directly updated, while if it is a hard bound particle we use the formulas derived in section 6.2 to propagate this displacement and velocity change to the hard bound particle's parent particles.

6.3.2 Coupling through binding springs

The synchronized coupling scheme only allows for short term drift of the soft bound particles limiting their added simulation capabilities. If long term drift is desired, we instead employ a spring-like force between the soft bound particle and its respective target particle. This *binding spring* force is

$$f^b = -f^b_e = -k(x - x_e) - b(v - v_e). \tag{6.5}$$

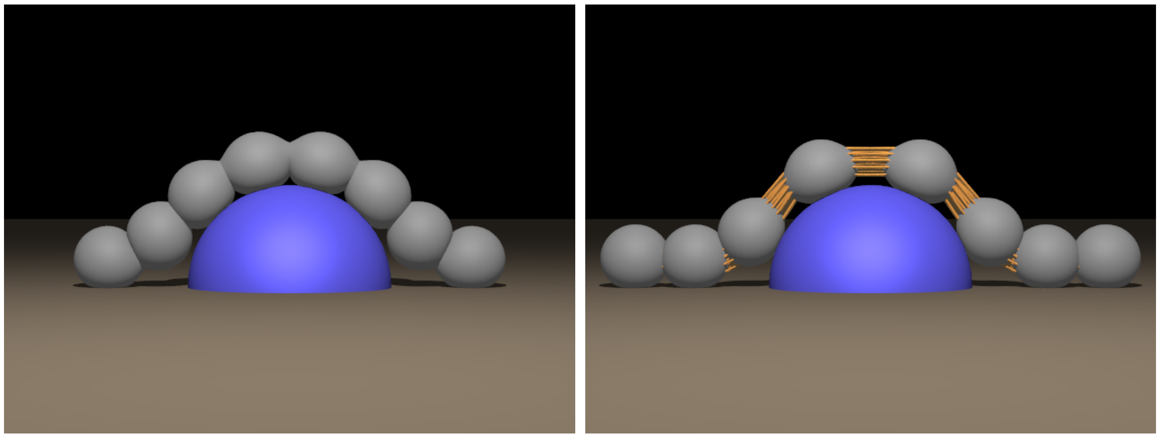

Figure 6.8: Spheres stitched together using binding springs of varying stiffness.

Note that the binding spring has zero rest length and thus linearizing about the rest state results in a multidirectional (as opposed to the typical directional) damping term. We will capitalize on this fact in our time integration scheme. Binding springs more readily allow for full simulation capabilities in the soft bound particles. One no longer needs to propagate information from the soft bound particles to the parents or to synchronize the soft bound particles, but rather the soft bound particles and parents are now implicitly coupled in a two-way fashion via these binding springs. We emphasize that the important properties of soft bindings depend on the combination of binding springs with force mapping, and would not be achieved through springs alone.

In figure 6.7 a coarse tetrahedralized volume has its surface mesh adaptively and dynamically refined based on proximity to collision geometry, and the soft bound particles present at every vertex of the surface mesh provide for subtetrahedron deformation.

6.3.3 Time integration

Our time integration scheme is a variant of the semi-implicit modified Newmark integration scheme of [82]. The deformable object is evolved from time t^n to t^{n+1} as follows:

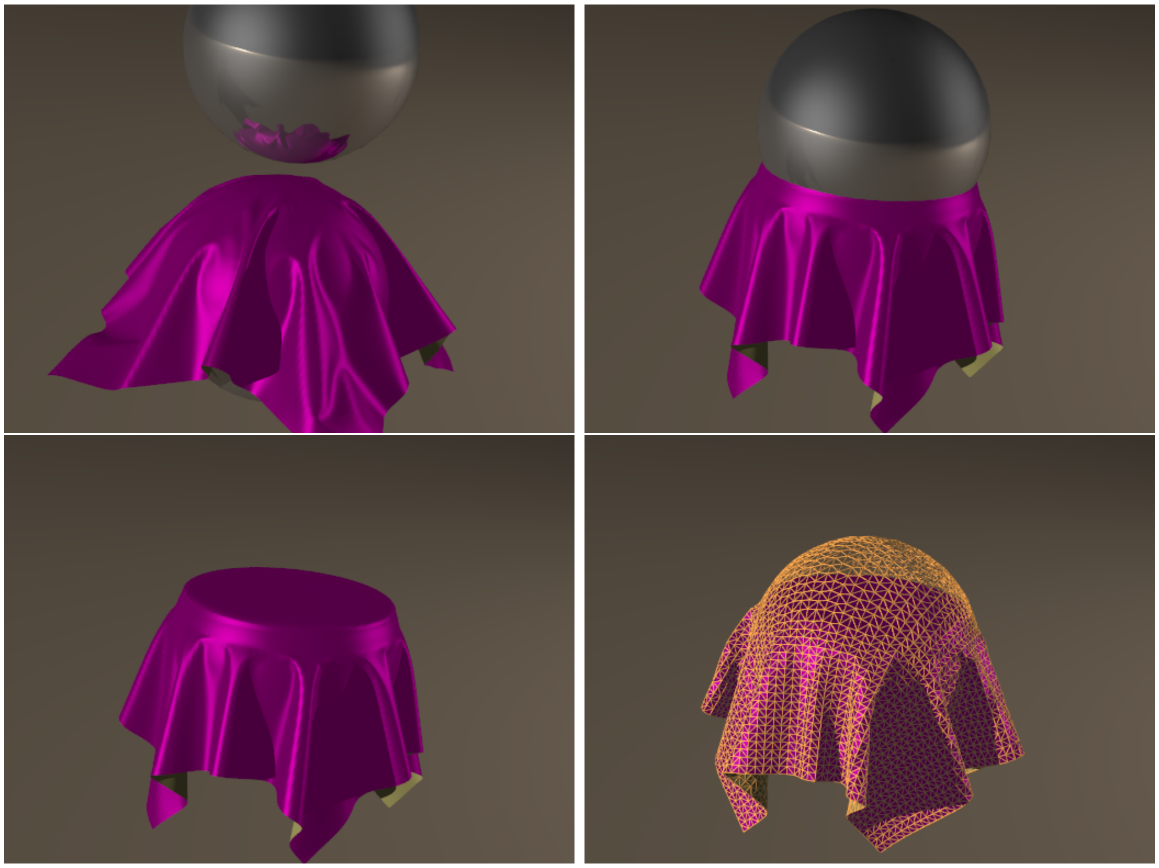

Figure 6.9: Cloth pinched between two colliding spheres.

1. $\tilde{\mathbf{v}}^{n+\frac{1}{2}} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^n, \tilde{\mathbf{v}}^{n+\frac{1}{2}})$

2. $\tilde{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t \tilde{\mathbf{v}}^{n+\frac{1}{2}}$

3. Process collisions $(\tilde{\mathbf{x}}^{n+1}, \mathbf{v}_n) \longrightarrow (\mathbf{x}^{n+1}, \tilde{\mathbf{v}}_n)$

4*. $\mathbf{v}^{n+1} = \tilde{\mathbf{v}}^n + \frac{\Delta t}{2}\Big(\mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^{n+\frac{1}{2}}, \tilde{\mathbf{v}}^n) + \mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^{n+\frac{1}{2}}, \mathbf{v}^{n+1})\Big)$

where $\mathbf{x}^{n+1/2} = (\mathbf{x}^n + \mathbf{x}^{n+1})/2$. Our scheme deviates from that of [82] in that the trapezoidal velocity update in step 4* uses $\mathbf{x}^{n+1/2}$. This substitution retains second order accuracy and allows for fully implicit integration of our binding spring forces. Under the common assumption that the velocity dependent forces are linear, we can rewrite step 4* as

4. $\mathbf{v}^{n+\frac{1}{2}} = \tilde{\mathbf{v}}^n + \frac{\Delta t}{2}\mathbf{a}\left(t^{n+\frac{1}{2}}, \mathbf{x}^{n+\frac{1}{2}}, \mathbf{v}^{n+\frac{1}{2}}\right)$
5. $\mathbf{v}^{n+1} = 2\mathbf{v}^{n+\frac{1}{2}} - \tilde{\mathbf{v}}^n$

The version of trapezoidal rule in step 4* can suffer from poor numerical conditioning when the two acceleration terms are rather large but of opposite sign, whereas steps 4 and 5 use a completely robust backward Euler update followed by extrapolation. That is, in the limit of a large time step the first acceleration term in step 4* pushes a positive coefficient c of an eigenvector toward $-\infty$ while the second acceleration term brings that value back from $-\infty$ to $-c$, possibly failing due to loss of precision. However, in the same large time step limit, step 4 damps c to 0 and step 5 robustly extrapolates c through 0 to $-c$.

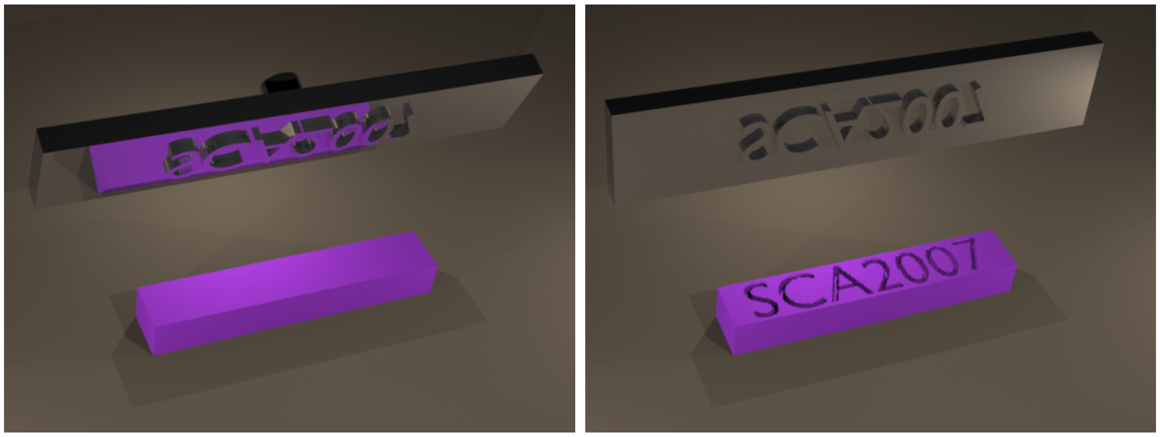

Figure 6.10: Allowing the hard bound targets to drift enables sub-element plaasticity

The time step restriction imposed by a single binding spring is $\Delta t < (b+\sqrt{b^2+4\mu k})/k$ where μ is the reduced mass (proof provided in appendix B). As is typical, this drives the time step to zero as the spring constant is increased, and thus we propose a fully implicit treatment of the binding springs leveraging the fact that our binding springs are fully linear in both position and velocity (which is not the case for non-zero rest length springs).

Equation (6.5) can be written in matrix form as $\mathbf{f}^b = -\mathbf{K}\mathbf{x} - \mathbf{B}\mathbf{v}$, where the stiffness and damping matrices $\mathbf{K}$ and $\mathbf{B}$ are symmetric positive semi-definite. We apply our time integration scheme to this force, with the modification that in step 1 we use the half time step position $\tilde{\mathbf{x}}^{n+1/2}$, making this step fully implicit:

$$\begin{aligned}\tilde{\mathbf{v}}^{n+\frac{1}{2}} &= \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}^b(t^{n+\frac{1}{2}}, \tilde{\mathbf{x}}^{n+\frac{1}{2}}, \tilde{\mathbf{v}}^{n+\frac{1}{2}}) &(6.6)\\ &= \mathbf{v}^n + \frac{\Delta t}{2}\left(-\mathbf{M}^{-1}\mathbf{K}\tilde{\mathbf{x}}^{n+\frac{1}{2}} - \mathbf{M}^{-1}\mathbf{B}\tilde{\mathbf{v}}^{n+\frac{1}{2}})\right). &(6.7)\end{aligned}$$

Substituting $\tilde{\mathbf{x}}^{n+1/2} = \frac{1}{2}(\mathbf{x}^n + \tilde{\mathbf{x}}^{n+1}) = \mathbf{x}^n + \frac{\Delta t}{2}\tilde{\mathbf{v}}^{n+1/2}$, where the last equality comes from step 2, gives

$$\left(\mathbf{I} + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{B} + \frac{\Delta t^2}{4}\mathbf{M}^{-1}\mathbf{K}\right)\tilde{\mathbf{v}}^{n+\frac{1}{2}} = \mathbf{v}^n - \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{K}\mathbf{x}^n \tag{6.8}$$

which is a true backward Euler step implicit in both position and velocity with no time step restriction. We emphasize that equation (6.6) only replaces step 1 in our time integration loop, and step 4 still requires the use of $\mathbf{x}^{n+1/2}$. Our modifications to [82] are crucial for unconditional stability, e.g. using $\mathbf{x}^{n+1}$ instead of $\mathbf{x}^{n+1/2}$ in step 4 leads to a time step restriction of $\Delta t < (b + \sqrt{b^2 + 8\mu k})/2k$. We stress that this implicit treatment of elastic forces is only used on the binding springs and that all other elastic forces are treated explicitly in order to more accurately resolve dynamic motion.

6.4 Examples

We used critical damping for the soft binding springs in our examples. The stiffness parameters were set experimentally.

Figure 6.8 shows eight deformable spheres stitched together using implicit binding springs as described in section 6.3.2. On the left the binding springs are made sufficiently stiff to retain coincidence of the connecting points without incurring additional time step restrictions. The binding springs are visible on the right where their stiffness is reduced.

In figure 6.10 we begin with a coarse deformable block. We then create a duplicate copy of the top surface, which is refined and hard bound to the block. Next, we use binding springs to attach a soft bound duplicate of this refined surface for collision with a stamp. When collisions stretch these binding springs to a length beyond a prescribed threshold the hard bound target particles are moved in material space (and a new barycentric embedding is computed) to bring the length back down to this threshold, analogous to typical plastic yield. We are careful not to move any hard bound target particle outside the material.

Using a variant of the virtual node algorithm [112], we cut a coarse cube mesh consisting of 320 tetrahedra into 289 sticks as shown in figure 6.11. Each stick is composed of a number of tetrahedra that are only partially filled with material and the polygonization of the material surface is augmented with soft bindings to resolve self-collisions and object collisions, see [154] which is enabled by our new hybrid simulation framework.

Figure 6.9 shows cloth pinched between two kinematically controlled spheres producing contradictory collision constraints (see [11]). For each particle in the cloth mesh that is pinched between the two spheres, we create two duplicate soft bound particles which each collide with only one of the two spheres. Collisions with the sphere pull these soft bound particles away from their hard bound target locations on the cloth surface resulting in binding spring forces that pull the cloth mesh in both directions equilibrating in a steady state. In the absence of any other forces, the soft bound particles would slide along the sphere in an attempt to minimize the length of the underlying binding springs. However, we map the mesh-based constitutive model forces of the cloth to the soft bound particles making them behave in a fashion similar to their hard bound counterparts, i.e. resisting stretching. Figure 6.9 (lower right) is a visualization of the effect of mapping the constitutive model forces from the hard bound target locations to their soft bound counterparts. That is, it is similar to creating a mesh which connects all the drifted soft bound particles to each other and to the parent mesh as depicted schematically by the orange wireframe. In fact, one could envision this as simulating three distinct meshes: the parent mesh given by the cloth itself, the orange wireframe mesh, and a second wireframe mesh which agrees with the bottom half of the orange wireframe but is concave up elsewhere resolving collisions with the top sphere. However, we do not need to construct these duplicate

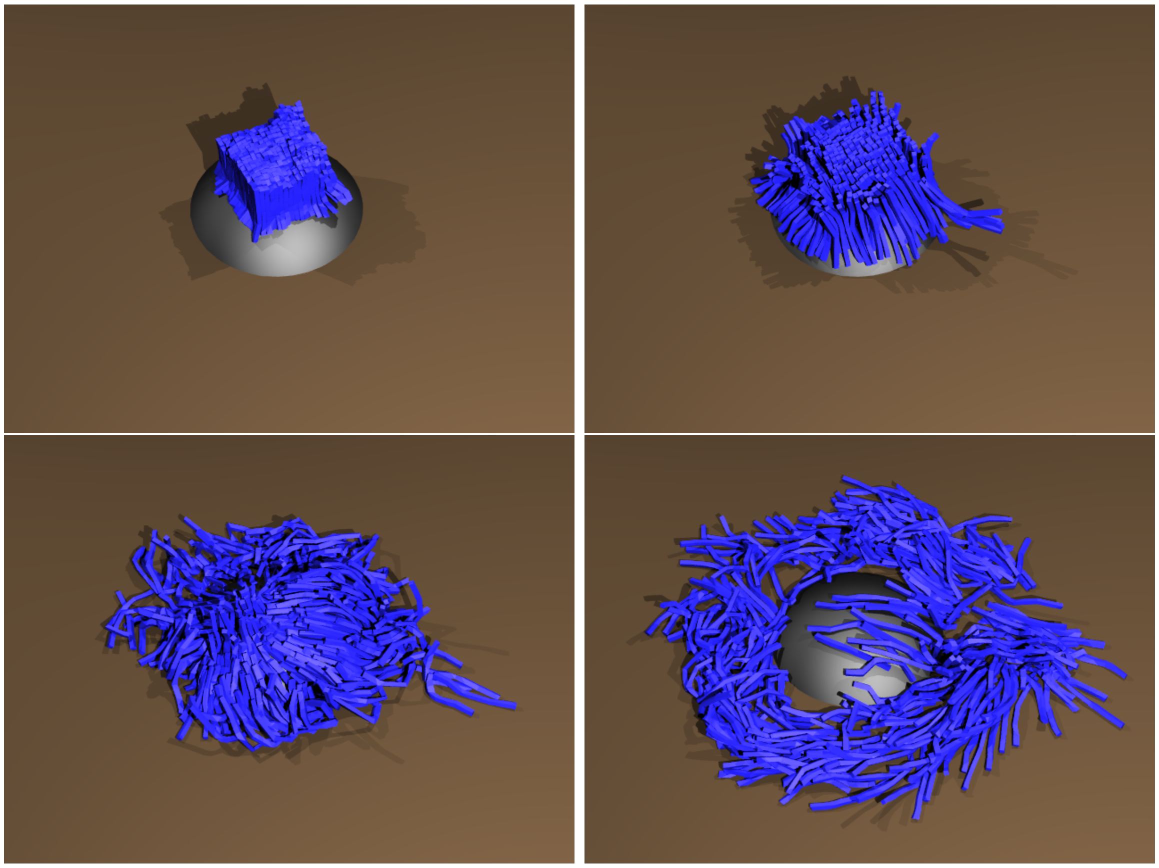

Figure 6.11: A coarse cube mesh is cut into 289 sticks.

meshes, and instead automatically inherit these properties from the parent simulation mesh.

Figure 6.12 depicts the dynamic simulation of a muscle-driven facial model for speech articulation based on [155]. The surface geometry is embedded in a non-graded adaptive red-only BCC mesh resulting in an 80% reduction in the number of simulation elements as compared with [156]. Hard bindings are used to simulate the resulting T-junctions, while soft bound particles are used to model the high resolution surface and also used for collisions, self-collisions and boundary conditions. Since the simulation mesh does not conform to the geometry of the cranium and jawbone, we use soft bound particles on the high-resolution surface to enforce bone attachments as well. A higher stiffness was used to attach the embedded free skin surface to its

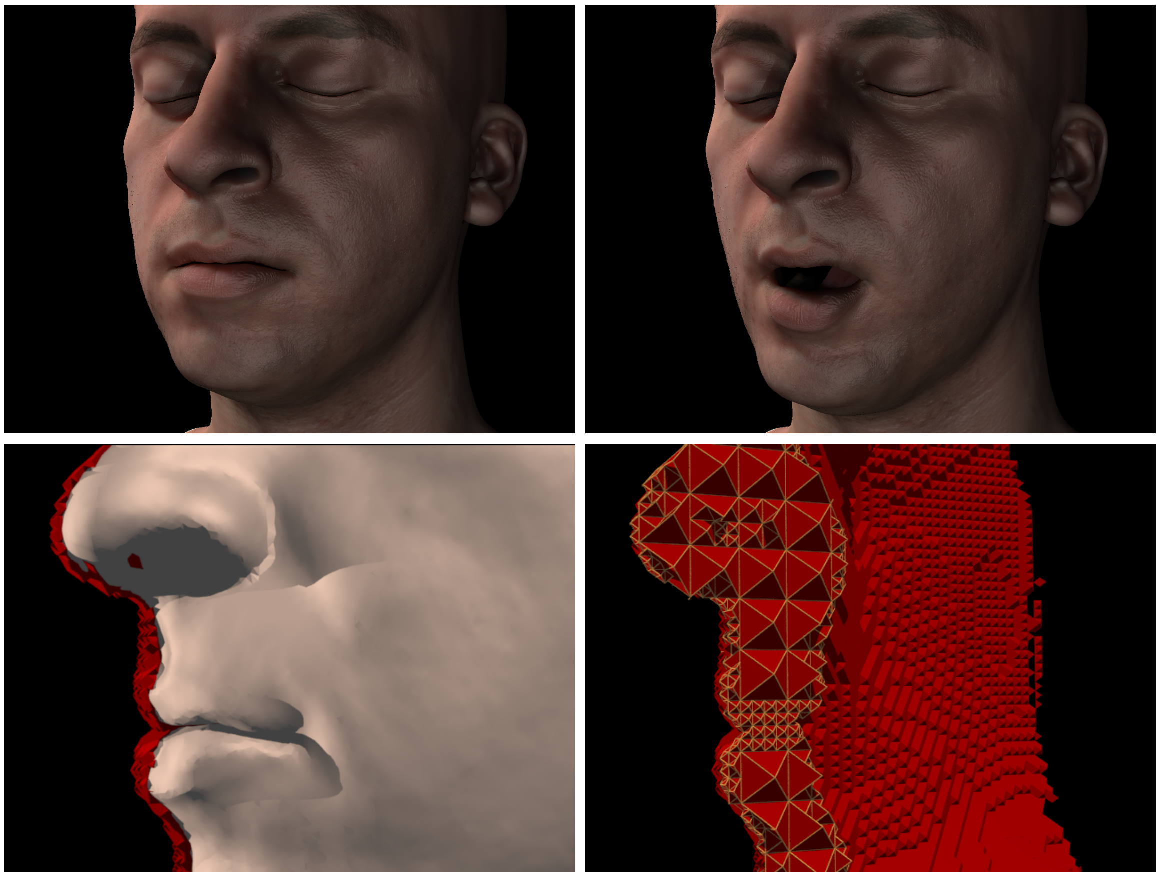

Figure 6.12: An adaptive red-only BCC mesh for an embedded face simulation.

hard bound counterpart, and a lower stiffness was used for the connections between soft-bound bone attachments and their hard bound targets.

6.5 Extensions to Rigid Body Coupling

Various authors have considered the two-way coupling of rigid and deformable bodies using a variety of schemes, see e.g. [9, 134, 85, 102]. We present preliminary results showing that our hybrid framework can be applied to these types of problems as well.

We consider the rigid body frame (x_c, q), twist (v_c, ω) and inertia tensor $I(q) = R(q)I_0R(q)^T$, where I_0 is a diagonal inertia tensor and $R(q)$ is the rotation matrix

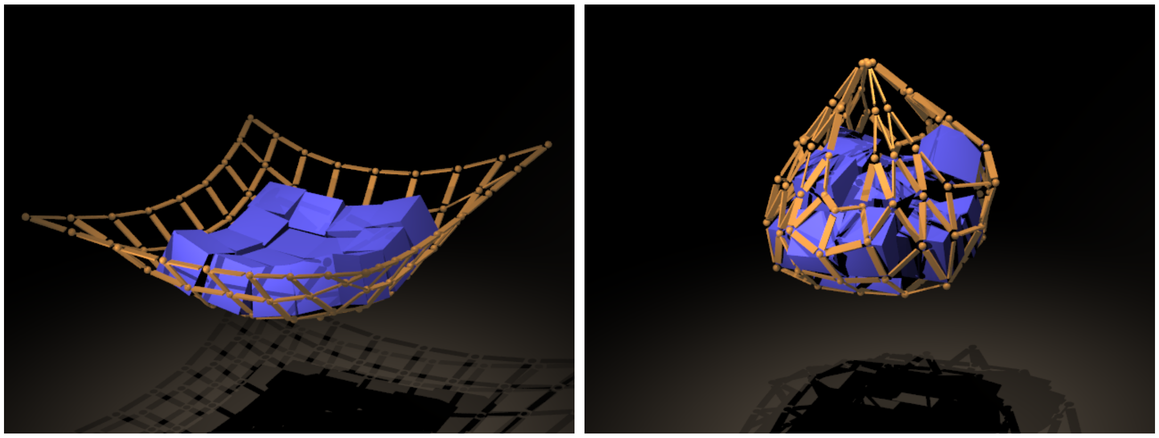

Figure 6.13: Binding springs enforcing articulation constraints.

associated with the current orientation. To facilitate our hybrid formulation, we define a *rigid body particle* with the same state as a rigid body. A rigid body hard binding is defined by embedding a particle onto a fixed object space location r_0. Consequently, the kinematic state of the bound particle is given as $x_e = x_c + r(q)$ and $v_e = v_c + \omega \times r(q)$ where $r(q) = R(q)r_0$ is the world space displacement of the embedded particle from the center of mass of the rigid body. We can write $v_e = W(q)(v_c, \omega)$ with the aid of the interpolation matrix $W(q) = (\mathbb{I} \;\; r(q)^{*T})$ where $\mathbb{I}$ is the 3×3 identity matrix and $r(q)^*$ is the cross product matrix. Although x_e cannot be written in similar form, a linearized change in the frame of the rigid body particle admits a similar mapping, i.e. $\Delta x_e = W(q)(\Delta x_c, \Delta q)$, where Δx_c is the displacement of the center of mass and Δq is the vector whose cross product matrix is the linearized rotation satisfying $R(q(t + \Delta t)) = R(q(t)) + \Delta q^* + O(\Delta t^2)$.

Applying a force f_e to the hard bound particle results in a wrench $(f_c, \tau) = W(q)^T f_e$ where $f_c = f_e$ is applied to the center of mass and $\tau = r(q) \times f_e$ is the torque. Next consider a velocity-dependent damping force $f_e = G_e v_e$ on a hard bound particle, which results in $(f_c, \tau) = W(q)^T G_e W(q)(v_c, \omega) = G(v_c, \omega)$ where $G = W(q)^T G_e W(q)$ maps twists to wrenches directly and retains the symmetry and definiteness of the damping matrix G_e. Similarly elastic wrench differentials are given by $(\Delta f_c, \Delta \tau) = W(q)^T K_e W(q)(\Delta x_c, \Delta q) = K(\Delta x_c, \Delta q)$ where $K = W(q)^T K_e W(q)$

is the stiffness matrix expressed in terms of the rigid body particle and $K_e = \partial f_e / \partial x_e$ is the stiffness matrix expressed in terms of the hard bound particle. We use these results to treat rigid body particles in the same fashion as other parent particles in our time integration scheme. Note that the definition of the global damping matrix $\mathbf{G} = \mathbf{W}^T \bar{\mathbf{G}} \mathbf{W}$ and global stiffness matrix $\mathbf{K} = \mathbf{W}^T \bar{\mathbf{K}} \mathbf{W}$ trivially extends to the case of rigid body bindings, by incorporating the interpolation matrix $W(q)$ into the global matrix $\mathbf{W}$ used in these equations. Finally, the global mass matrix $\mathbf{M}$ is augmented with the diagonal entry $\text{diag}(m\mathbb{I}, I)$ for each rigid body particle.

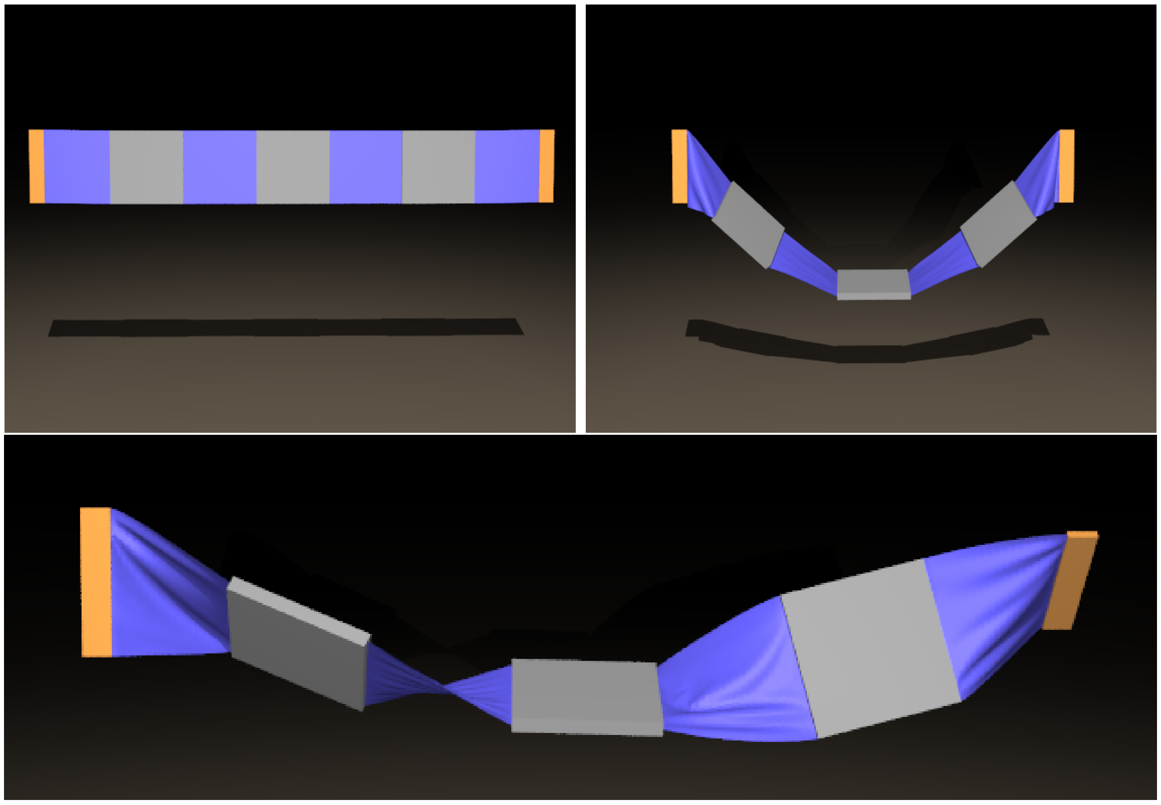
Figure 6.14: Rigid plates coupled with cloth sheets.

We preliminarily couple our hybrid simulation of rigid body particles to a state-of-the-art rigid body simulation scheme as follows:

1. Copy the state of the rigid bodies to the rigid body particles

2. Compute $\tilde{\mathbf{v}}^{n+\frac{1}{2}}$ as in section 6.3.3
3. Compute $\tilde{\mathbf{x}}^{n+1}$ using $\tilde{\mathbf{x}}_{\mathbf{c}}^{n+1} = \mathbf{x}_{\mathbf{c}}^{n} + \Delta t \tilde{\mathbf{v}}_{\mathbf{c}}^{n+\frac{1}{2}}$ and $\tilde{\mathbf{q}}^{n+1} = \mathbf{q}(\Delta t \omega^{n+\frac{1}{2}})\mathbf{q}^{n}$
4. Process collisions for deformable objects only
5. Compute $\mathbf{v}^{n+1}$ using steps 4 and 5 from section 6.3.3
6. Copy *momentum only* from the rigid body particles to the rigid bodies
7. Separately, time integrate and collide the rigid bodies

This last step evolves the rigid bodies forward in time using a standard scheme as in [75], and the two-way coupling is integrated into the standard rigid body solver using steps 1 through 6 where only the effects of momentum are preserved. Although it may be possible to extend the rigid body particles to include more advanced contact, collision, friction, stacking and complex articulation, it is not yet clear how this could be done and we defer it to future work. Figure 6.13 shows the use of implicitly integrated binding springs to enforce a simple point joint articulation between rigid body hard bound particles. Figures 6.14 and 6.15 illustrate two-way coupling between cloth and rigid bodies.

6.6 Summary

We proposed a novel method for augmenting a mesh-based approach to deformable solids simulation with point-based simulation technology. Although we have only shown a few illustrative examples, we believe that this framework will have widespread use, especially on problems where the mesh-based framework is too restrictive. As future work, we would like to improve the formulation of soft bound particles by deriving the assignment of masses from an underlying physical principle and extending the force mapping algorithm to include damping forces. We will also focus on extending our preliminary algorithm for coupling deformable and rigid bodies to more complex rigid body scenarios, including contact, collision, stacking, friction and more complex articulation.

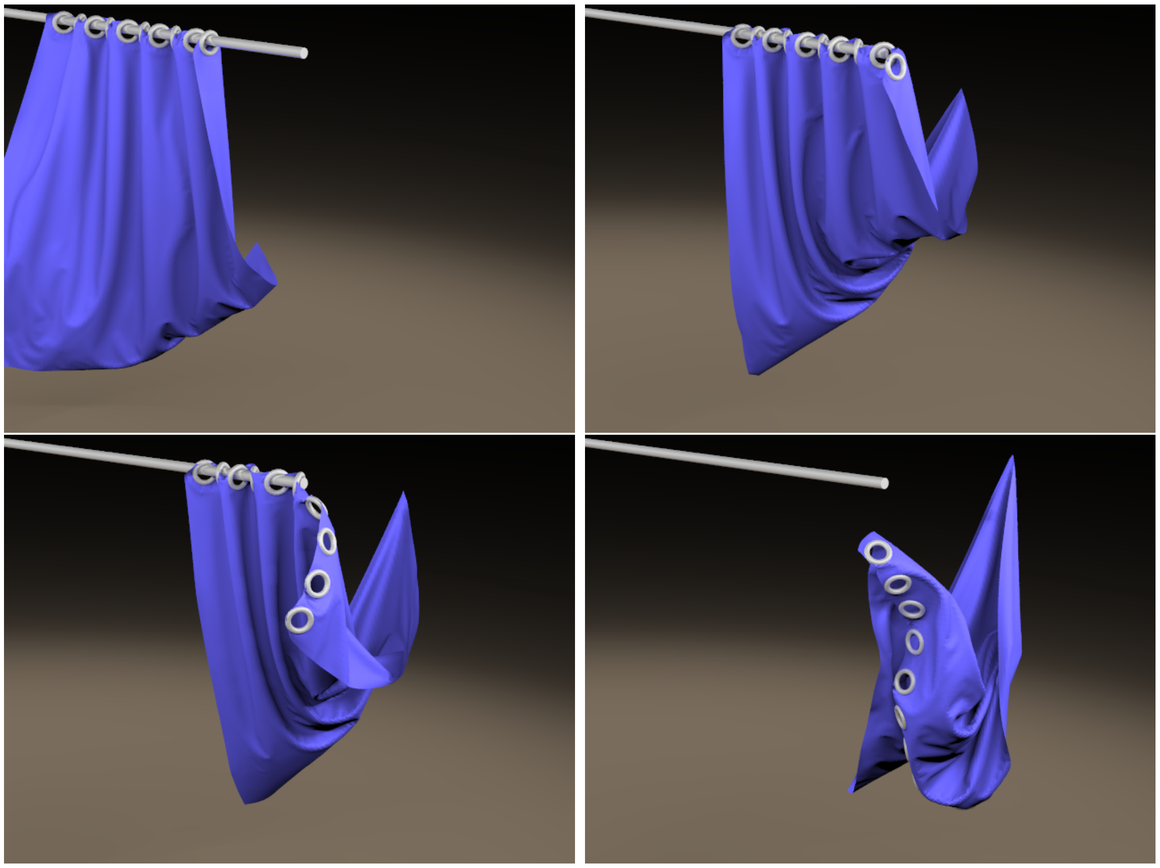

Figure 6.15: Simulation of a cloth curtain bound to rigid rings.

Chapter 7

Cutting and virtual surgery

We propose a flexible geometric algorithm for placing arbitrary cracks and incisions on tetrahedralized deformable objects. Although techniques based on remeshing can also accommodate arbitrary fracture patterns, this flexibility comes at the risk of creating sliver elements leading to models that are inappropriate for subsequent simulation. Furthermore, interactive applications such as virtual surgery simulation require both a relatively low resolution mesh for efficient simulation of elastic deformation and highly detailed surface geometry to facilitate accurate manipulation and cut placement. Thus, we embed a high resolution material boundary mesh into a coarser tetrahedral mesh using our cutting algorithm as a meshing tool, obtaining meshes that can be efficiently simulated while preserving surface detail. Our algorithm is similar to the virtual node algorithm in that we avoid sliver elements and their associated stringent timestep restrictions, but it is significantly more general allowing for the arbitrary cutting of existing cuts, sub-tetrahedron resolution (e.g. we cut a single tetrahedron into over a thousand pieces), progressive introduction of cuts while the object is deforming, and moreover the ability to accurately cut the high resolution embedded mesh.

7.1 Introduction

Methods for modeling and animating cracks and incisions on rigid or deformable solids have received significant attention in computer graphics, some dating back to the early days of deformable models [168, 169]. One class of applications dealt with modeling the phenomena of material failure and fracture (see e.g. [133, 131]) while other methods focused on incisions that are explicitly placed on a deformable model, for example in the context of a surgical manipulation in a virtual simulation environment.

Virtual surgery in particular has spawned a very active thread of research due to the high visibility and impact of such applications as well as the unique algorithmic requirements it entails. Interactivity in a virtual surgery application mandates the use of a relatively coarse simulation mesh to obtain high frame rates. At the same time, a substantially higher resolution is required on the surface geometry to prevent distorting features that are essential in planning incisions and manipulating the flesh. These requirements place severe limitations on the design of a cutting algorithm. For example, methods that decompose tetrahedra into subelements in order to cut the mesh (e.g. [16, 14]) create many more tetrahedra (essentially making a fine resolution mesh near the cuts) and create ill-conditioned tetrahedra that impose severe time step restrictions (see e.g. [114, 15]) making the simulation impractical even on the low resolution base mesh. Thus, some authors have avoided element decomposition resorting instead to other techniques such as deleting any element touched by the blade [62], or restricting cutting to mesh element boundaries [127]. Whereas the second option can produce a jagged surface, subsequent snapping of nodes can smooth the cut surface [128, 152]. Of course, this also tends to produce sliver elements, and some authors have proposed the use of meshing algorithms such as edge collapse to remove degenerate tetrahedra [64, 160].

Although first proposed in the context of fracture, the virtual node algorithm [112] alleviates many of the aforementioned difficulties. By duplicating elements to obtain new degrees of freedom necessary for modeling the topological change, it preserves the conditioning of the initial mesh while creating a minimal number of new elements. Thus one obtains the goal of a low number of well-conditioned elements. A limitation of the virtual node algorithm is that if the mesh were completely fractured, the smallest possible units would be individual nodes surrounded by their incident material. A tetrahedron can be cut at most once along each edge and can only be fractured into four pieces. Thus, the resolution of the simulation mesh effectively limits the resolution of the cutting surface. For a virtual surgery application, where a coarser mesh and precise cuts are required, such restrictions could be detrimental. For example, even a simple V-shaped cut would have to be heavily distorted so that the cutting surface only “turns” along element boundaries. More elaborate cuts, such as T-shaped incisions, would incur even more serious distortion or could even lead to spurious material connections. For fracture on the other hand, where finer meshes are typically used and jagged cracks are acceptable, these restrictions are not as severe. An improved version of the virtual node algorithm was proposed in [167] for embedding a high resolution tetrahedral muscle geometry into a coarser resolution tetrahedral simulation mesh. They diagrammatically showed that the virtual node concept could be extended to split a tetrahedron into as many pieces as required to embed arbitrary geometry, and used the high resolution tetrahedral muscle geometry mesh to prescribe the connectivity of the coarse embedding mesh. An important limitation of their method, however, is the requirement for a full tetrahedralization of the embedded geometry (rather than a description of its boundary surface), rendering it inapplicable to virtual surgery applications which define a cutting surface without any predetermined notion of volumetric connectivity.

[161] proposed a meshless method for simulating a finite element constitutive model similar to the fracture work in [140] (see also [187]). However, they stress that the crack itself is better modeled with explicit triangulated geometry rather than with the meshless method. This decoupling of the cutting surface from the simulation geometry (i.e. an explicit triangulated cutting surface combined with a meshless discretization of the simulation geometry) is similar in spirit to the fracture work of [123, 112] both of which embedded a triangulated surface into a background simulation mesh. Stressing that this explicit triangulation of the cutting surface permits geometric operations such as subdivision and self-intersection resolution, we devise

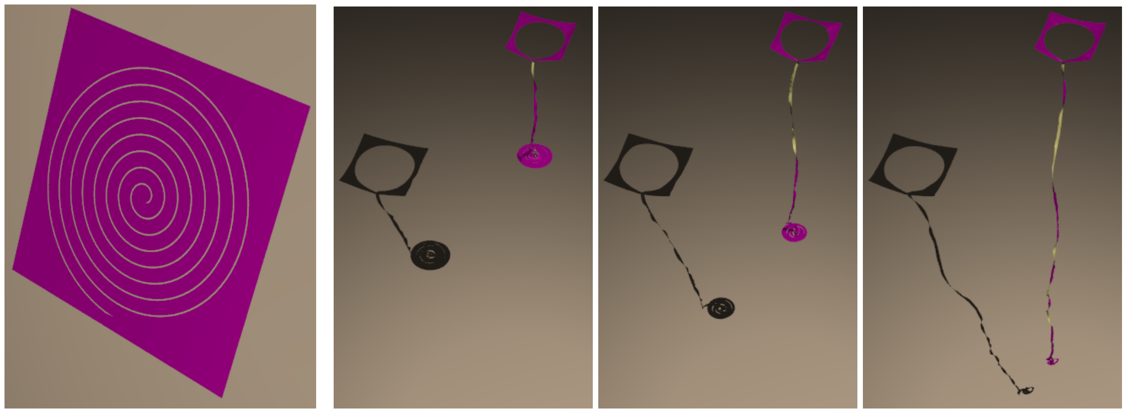

Figure 7.1: A spiral cut applied to a piece of cloth creates a long ribbon.

a novel algorithm that allows for a direct implementation of the conceptual diagram from [167] without requiring any predetermined notion of volumetric connectivity (as they do). Thus, our algorithm can also be used as an embedded meshing tool which, using only the boundary surface of an input object, will embed it in an arbitrary simulation mesh duplicating the degrees of freedom of the embedding mesh as needed to resolve the topology and connectivity of the original embedded object. This feature is demonstrated in figure 7.11 where a high-resolution triangulated surface of a face is used to carve an embedded flesh volume out of a coarser embedding mesh. In this example, embedding tetrahedra touching both the upper and the lower lip are automatically duplicated to allow opening of the mouth. Finally, such embedded models may be further cut if desired, as shown in figure 7.11.

Our main contribution is a cutting algorithm that yields efficient simulation models while imposing no restriction on the geometry of the cuts. In this aspect it combines the favorable traits of other methods in the literature while alleviating most of their limitations. For example, it provides the versatility of remeshing-based or point-based methods in accommodating arbitrary cuts, while maintaining the simplicity and efficiency of an underlying tetrahedral representation (unlike point-based methods) and preventing stringent timestep restrictions due to ill-conditioned sliver elements (unlike methods based on remeshing). Similar to the virtual node algorithm it creates new degrees of freedom to reflect the topological changes incurred by the cuts while preserving the good conditioning of the initial mesh, but avoids the restrictions on the geometry of the cutting surface imposed by the formulation of [112]. When used as an embedded meshing tool, it provides the freedom to embed an arbitrary high resolution geometry on a coarser simulation mesh in the fashion of [167] but only requires a surface representation of the embedded geometry as contrasted to the full volumetric representation required by their approach. Finally, our method supports incremental introduction of cuts as well as cutting an object while it deforms.

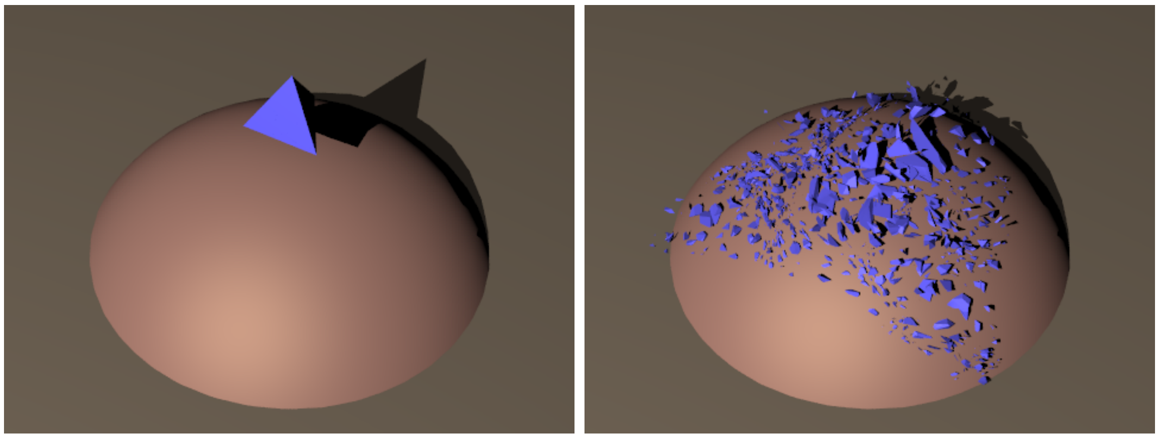

Figure 7.2: A single tetrahedron sliced into over a thousand pieces

7.2 Previous Work

Early fracture work includes [168, 169]. See also [130, 81, 110, 159] where connections between elements are broken under high stress, [125] who addressed blast waves, [119, 123] who considered fracture along element boundaries in an FFD framework, [133, 191, 131] who handled fracture events using continuous remeshing, and [122] who treated fragments as rigid bodies in between collisions. The virtual node algorithm has also been used for melting and burning solids [104]. [69, 78] treated the fracture of thin shells, and [8] addressed the fracture of rigid materials. Modeling of cuts has also been addressed in the context of virtual surgery, especially on facial models [95, 142, 93] (see also [116]) where the cut surface is the result of an explicit surgical

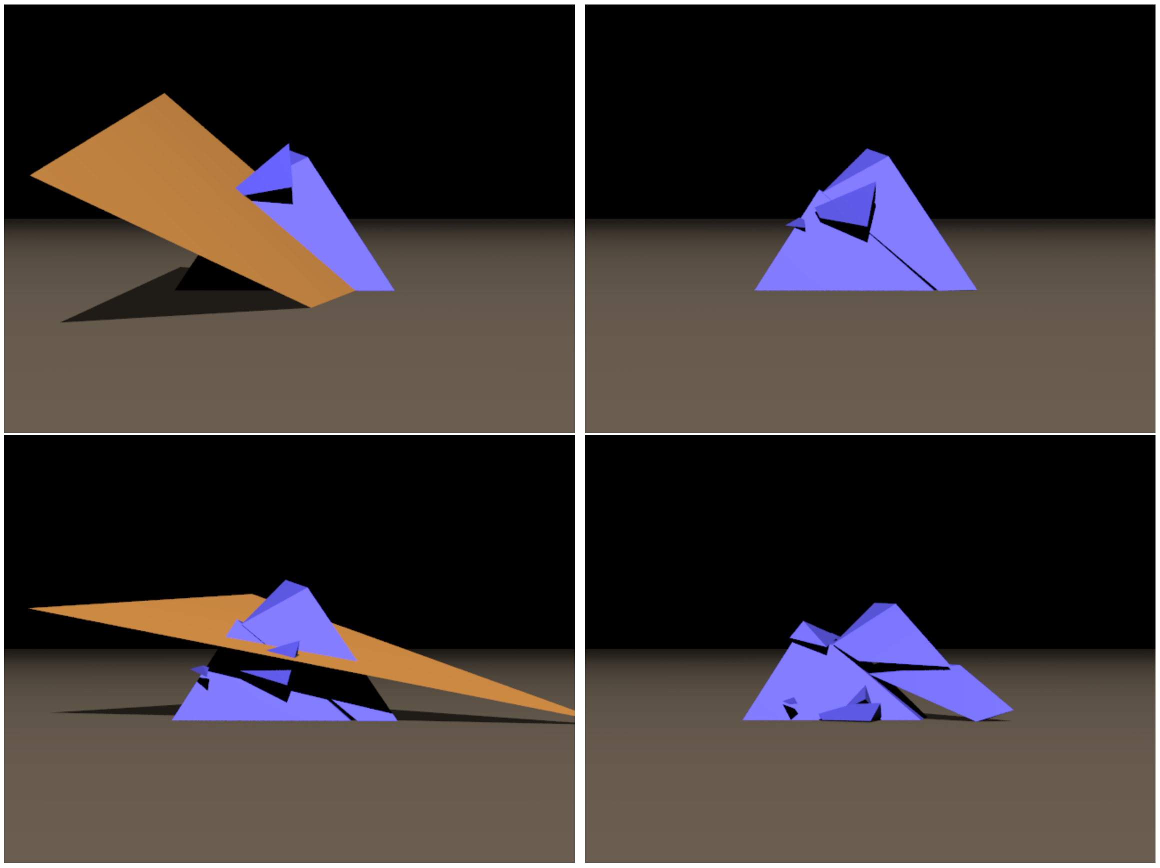

Figure 7.3: A single tetrahedron is progressively cut while being simulated.

manipulation, rather than a physical Rankine condition for fracture.

7.3 Cutting Algorithm

In our approach there are two simulation primitives, a background tetrahedral simulation mesh and a triangulated cutting surface. When embedding a high resolution material surface mesh into the background tetrahedral mesh, the high resolution surface is considered part of the cutting surface and in fact our cutting algorithm is used to generate this embedded geometry (see Figure 7.11). Virtual surgery applications progressively add triangles to the cutting surface along the blade's swept front, while fracture applications add triangles based on stress criteria.

7.3.1 The Cutting Surface

Triangles in the cutting surface may be initially intersecting, e.g. T-cuts are common to many surgical procedures. Leveraging our explicit description of the cutting surface, we resolve self-intersections through subdivision of the cutting triangles. One approach is to compute intersections among all triangles in the cutting surface and then to retriangulate into an intersection-free surface, but we have found that using a polygonal subdivision is more robust and efficient. In particular, one could triangulate our polygonal elements, but subsequent cuts that yield intersections would force further subdivision of these triangles unnecessarily creating more elements and possibly poorly conditioned elements. Given a soup of cutting triangles, the polygonal subdivision is unique and can be updated incrementally preserving uniqueness whereas the triangulation is not unique, based on heuristics, and may be found suboptimal when further cutting triangles are added. This means that one would be forced to undo previous triangulations in order to avoid degeneracies and poor conditioning due to newly introduced cuts, injuring the incremental nature of the algorithm.

Our approach is to resolve the cutting triangles into an intersection-free, nonmanifold *polygon mesh*. Each cutting triangle is subdivided into a number of polygons, whose edges are line segments that result from the pairwise intersection of this cutting triangle with all other cutting triangles. When adding new cuts incrementally, these new triangles are resolved with the existing polygons obviating the need to reexamine all past cuts. We will of course eventually triangulate these polygons for collisions and visualization, but the potentially-resulting sliver elements are not used by our cutting algorithm and thus do not impact its conditioning.

We compute the polygonization of any triangle in the triangle soup as follows. A node of the polygon mesh is defined in one of three possible ways: a vertex of a triangle in the soup, an intersection between a triangle edge and a face at a point interior to the face, or an intersection of three triangles that occurs at a point interior to all three (see Figure 7.6). Each triangle is then subdivided into polygons by connecting incident nodes with segments. We place segments between any two incident nodes that are also on a given second triangle. That is, if node 1 and node 2 are both incident on triangle 1 and triangle 2, then a segment is added between them. We also add

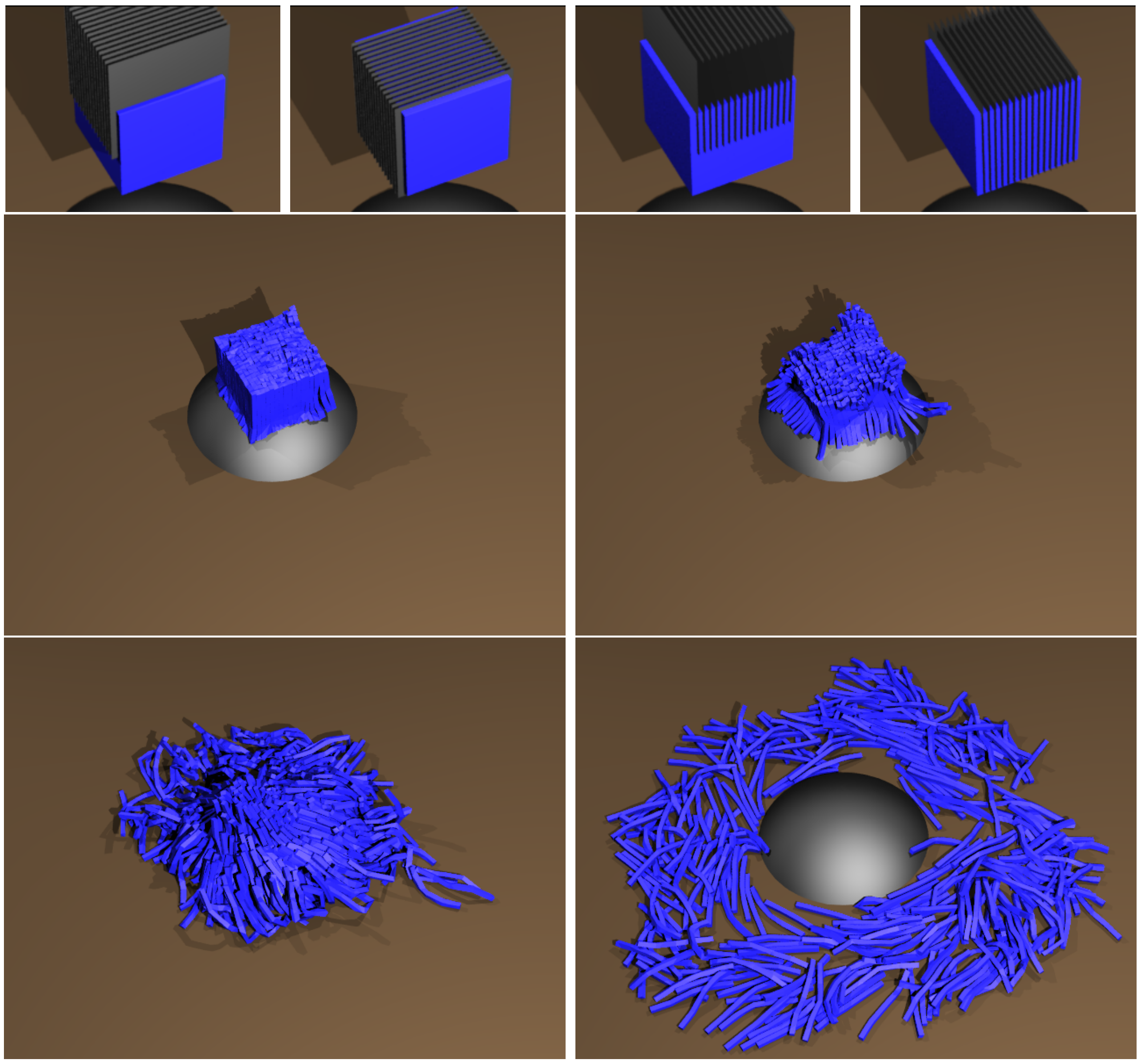

Figure 7.4: 32 intersecting planes slice a tetrahedral mesh into 289 sticks.

segments between any two nodes that lie on the same edge of a triangle from the soup. This definition results in overlapping segments when multiple nodes are collinear, and thus we only construct the minimal set of non-overlapping segments. Finally, the triangle is projected into two spatial dimensions where a boundary tracking algorithm is used to identify each closed polygon (also see Figure 7.6).

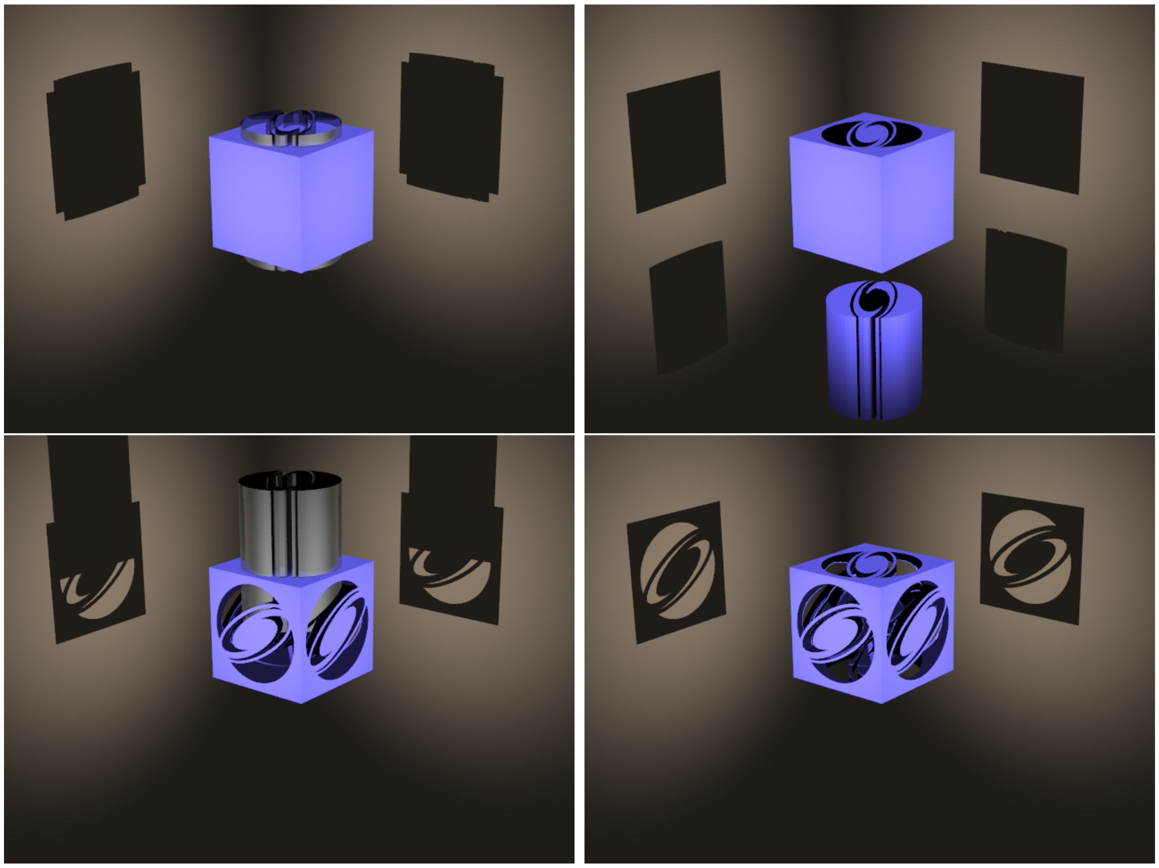

Figure 7.5: A cookie cutter surface is used to cut into a tetrahedral mesh.

7.3.2 Clipping to the Simulation Mesh

Besides resolving the cutting surface into non-intersecting polygons rather than triangles, a second key to our approach is to further subdivide the cutting surface in a manner that removes intersections with the faces of the tetrahedral simulation mesh. This allows for a tetrahedron-centric approach to the algorithm facilitating subsequent topological processing. This also preserves the incremental nature of the algorithm, since adding a new cut only affects the intersected tetrahedra. Thus, most importantly, we treat both cutting triangles and tetrahedron faces as elements of a large triangle soup and we resolve *all* of these triangles simultaneously into an intersection-free polygon mesh. As mentioned in the previous subsection, incrementally-added cutting triangles are resolved with the the existing polygon mesh where our notion of a polygon mesh now includes polygons on tetrahedron faces as well.

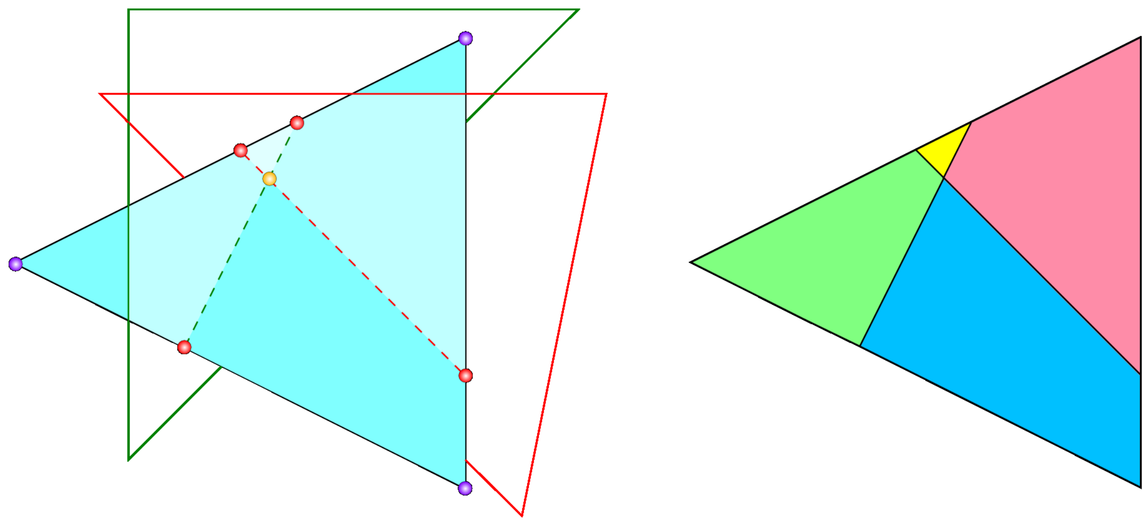
Figure 7.6: A triangle in the triangle soup intersecting other triangles

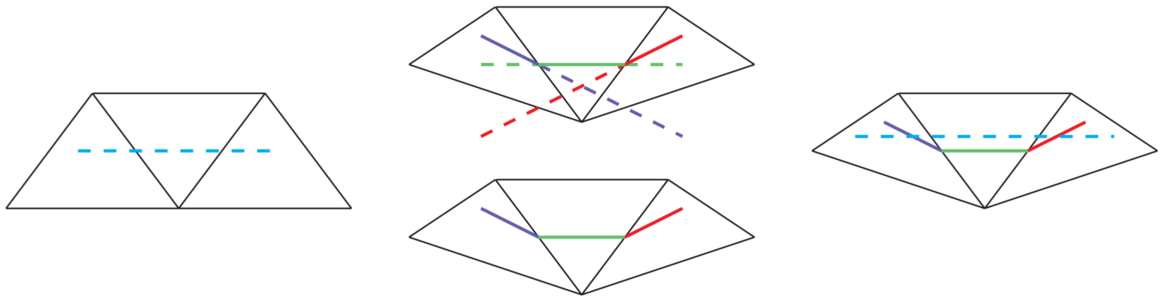
Figure 7.7: Clipping cutting elements in the 2D cutting algorithm.

In practice, our tetrahedron-centric algorithm is carried out as follows. First, a tetrahedron intersected by a cutting triangle is assigned a copy of that triangle for independent processing, where the triangle is projected into the tetrahedron's barycentric space. When polygonizing that triangle copy, we do so using a pool of segments collected only from the faces of this tetrahedron and any other cutting triangles that intersect this tetrahedron. The resulting polygons can be independently

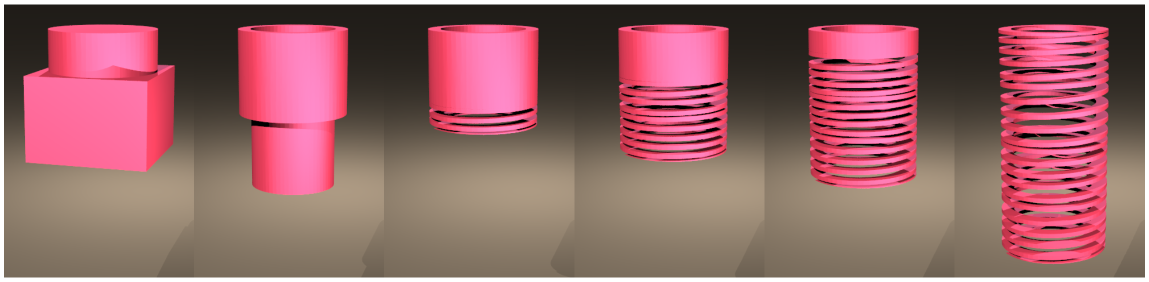

Figure 7.8: Progressive carving of a deforming slinky.

generated per tetrahedron. Since the faces of the tetrahedron are also in the triangle soup, no polygon segment crosses a tetrahedron boundary, allowing us to trivially prune away polygons that are not on or within the tetrahedron. Note that neighboring tetrahedra that share a face will find the exact same polygons on that common face. See Figure 7.7, which illustrates a two-dimensional version of our algorithm, where a cutting segment spans three triangles from the background mesh. Since each triangle owns a copy of this cutting segment stored in barycentric coordinates, subsequent deformation of the mesh leads to three non-coincident copies of this cutting segment that only agree at element boundaries.

7.3.3 Per-Tetrahedron Subdivision

After each tetrahedron has generated its subset of the polygon mesh, we determine the volumetric connected components of material within each tetrahedron. Each connected component is a polyhedron that we compute using boundary tracking on the polygons in the tetrahedron. Then each connected component is embedded in a duplicate copy of the original tetrahedron. Unlike the virtual node algorithm, connected components may be completely interior to a tetrahedron and need not be incident on a node or a face (see Figure 7.2). When constructing the polygon mesh, tetrahedra that are only partially filled with material require special consideration. They are handled by constructing only the subset of the polygon mesh that is on or within actual material. This is straightforward since the material boundary is already a subset of the previously generated polygon mesh.

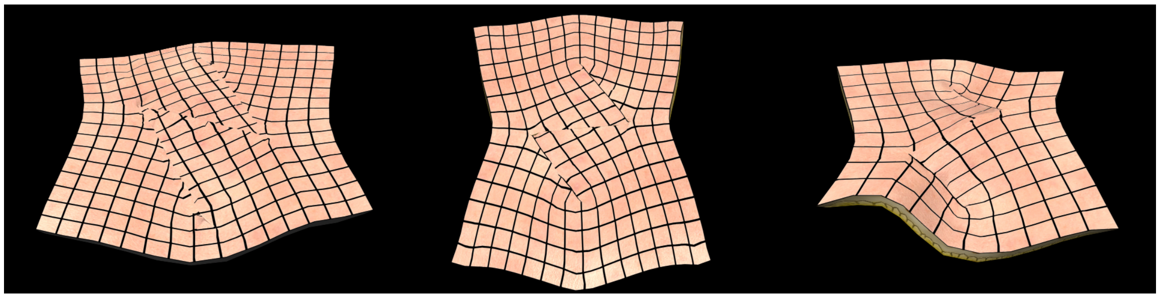

Figure 7.9: Comparison of Z-plasty procedures at different angles.

7.3.4 Determining Material Connectivity

Once each tetrahedron has been partitioned into its connected components, we identify pairs of connected components across adjacent tetrahedra as materially contiguous. We do so by examining each pair of tetrahedra that share a common face. Each tetrahedron has been duplicated into identical copies, each containing a single connected component of material embedded within it. The goal is to determine if a copy of tetrahedron A contains material contiguous to material in some copy of tetrahedron B, which we do by individually examining each possible pair of tetrahedron copies. We denote the copies as contiguous if their shared face contains a common polygon (see Figure 7.10). Finally, we collapse nodes on common faces of materially contiguous tetrahedra to obtain the final simulation mesh.

7.4 Examples

After cutting, we triangulate the resulting material surface for collisions, self-collisions, and rendering. We determine the material surface simply by keeping only the polygons incident on exactly one tetrahedron, removing those from the material's interior. The boundary polygons that remain are triangulated with a greedy ear-removal scheme. We employ the hybrid framework of [157] to simulate the embedded material geometry, allowing us to use an arbitrary finite element constitutive model while resolving object collisions and self-collisions on a soft constrained copy of the material surface.

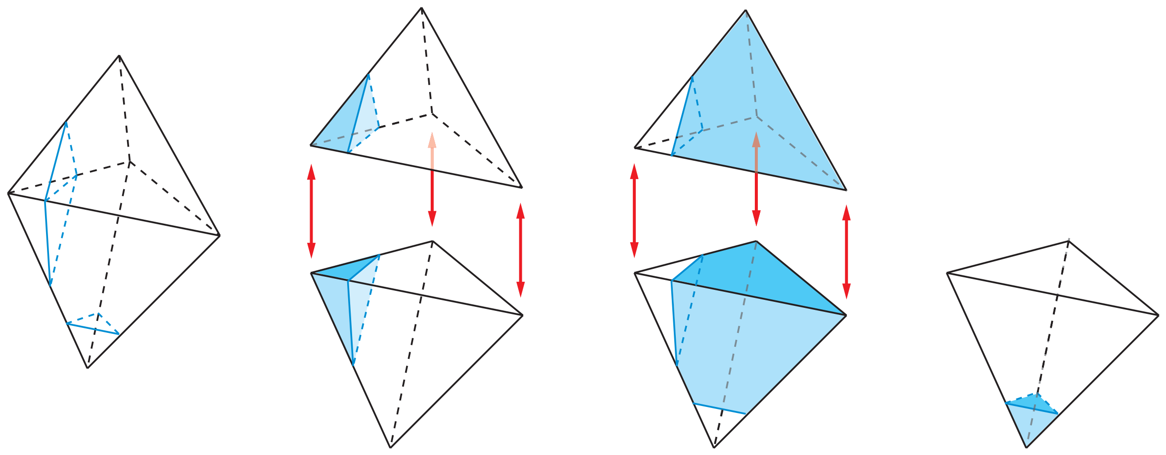

Figure 7.10: Topology and node duplication resulting from two cuts

Figure 7.1 shows a two-dimensional version of our algorithm used to make a spiral-shaped cut into a suspended piece of cloth. Figure 7.3 illustrates that we can progressively cut a single tetrahedron into multiple pieces as it deforms. Figure 7.2 shows a single tetrahedron cut into over a thousand pieces by fifty cutting triangles, many of which only partially intersect the tetrahedron rather than slicing completely through it. Figure 7.4 shows 32 intersecting planes cutting a cube into 289 sticks. In addition, our cutting algorithm can be used to sculpt objects with high surface detail embedded within coarse simulation geometry. In Figure 7.5, we apply the same cutting stencil normal to each face of a cube. In Figure 7.8, we progressively sculpt a slinky out of a block. Figure 7.11 (top) shows our algorithm used as a meshing tool to embed a 250k triangle scan of the human face into an 850k tetrahedral simulation mesh. The resulting face model itself can then be further cut, as shown in Figure 7.11 (middle).

7.5 Clinical Applications

Today we train surgeons on the backs of the poor through free clinics and low-income care. In the absence of computerized training tools for predicting surgical outcomes, practitioners often have to learn through their own mistakes limiting the quality of care they can provide to their patients. A virtual surgery environment helps to

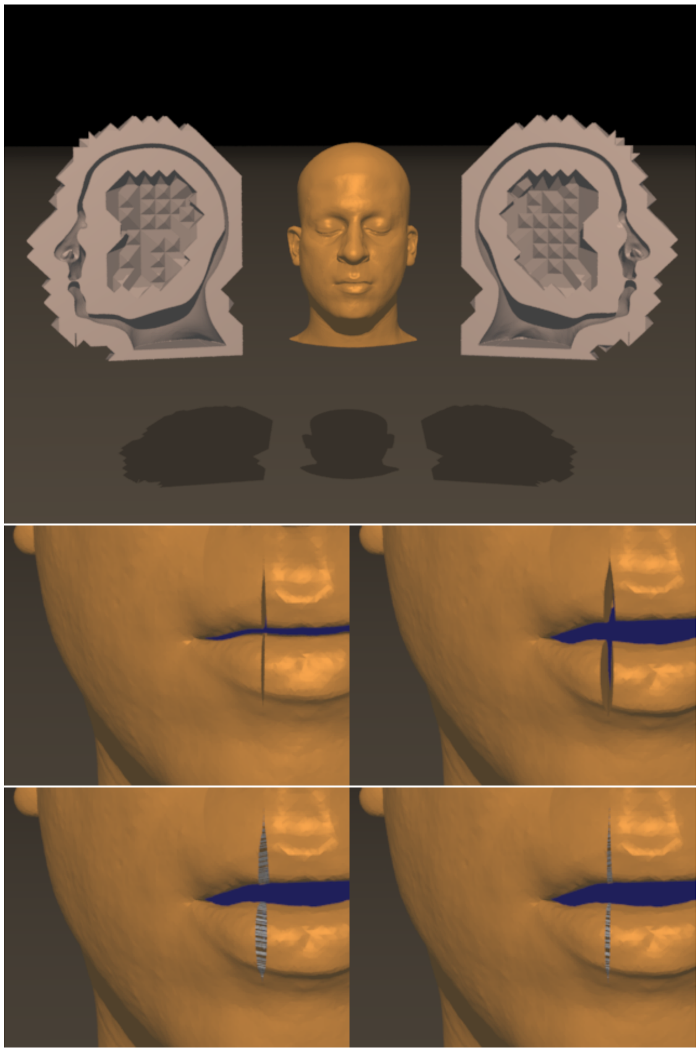

Figure 7.11: The cutting algorithm used as an embedded meshing tool

alleviate this by providing scenarios for most known accidents and complications that could arise during a surgery. Moreover, it allows one to confront never-before-seen situations as well. The essence of surgery is topological change and therefore one needs a very robust and flexible cutting algorithm in any surgical simulator.

Plastic surgery has wide-ranging techniques of various complexity, but there are certain fundamental maneuvers common to many procedures. Z-plasty is such a procedure and is used in at least 25% of reconstructions. A surgeon performs thousands

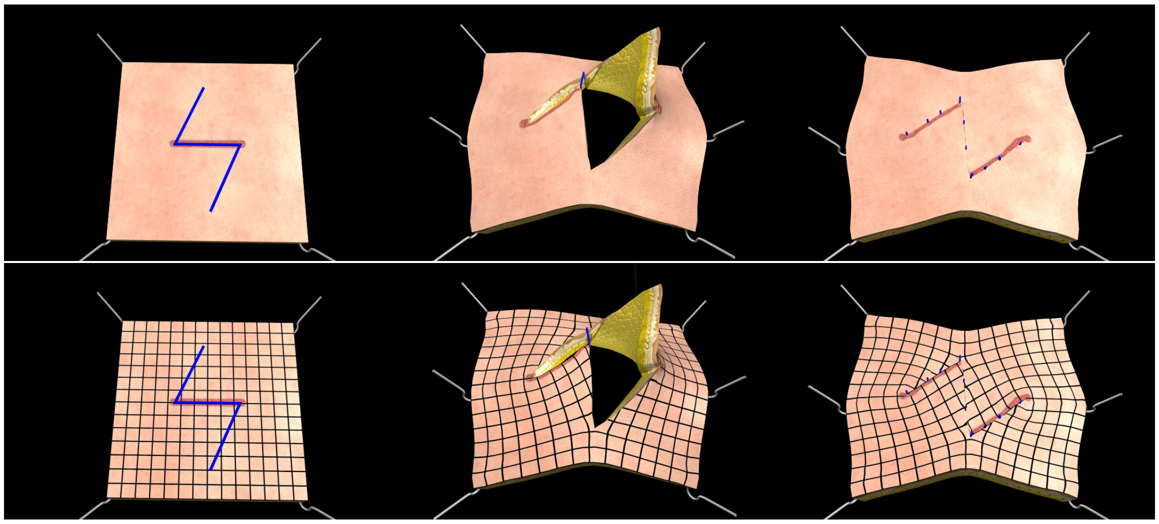

Figure 7.12: A Z-plasty procedure is used to alleviate a scar contracture.

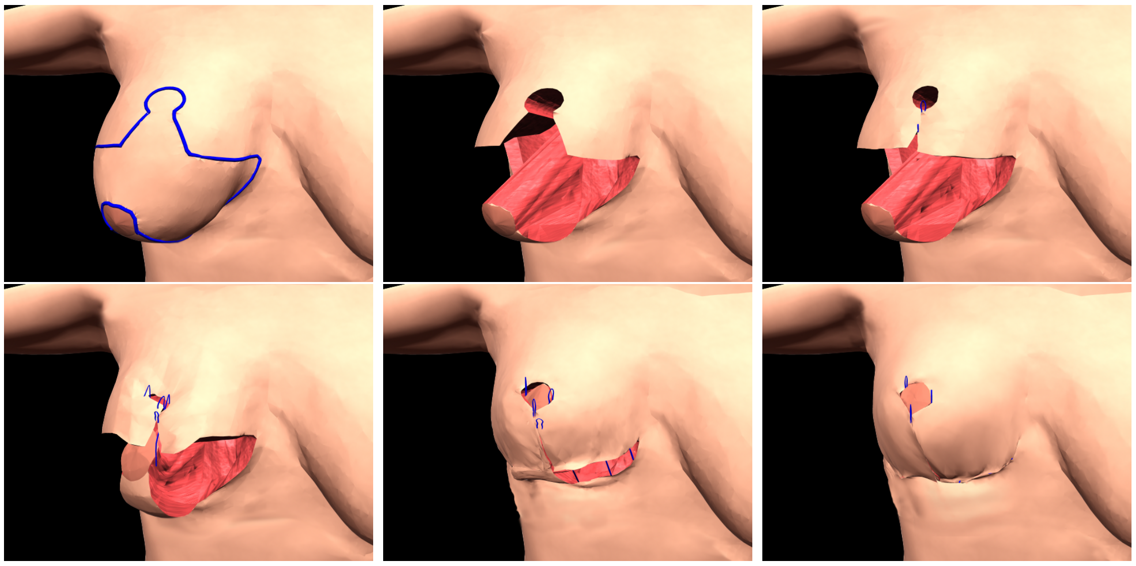

Figure 7.13: Simulation of a reduction mammoplasty.

of them over the course of a career. The Z-plasty is used to elongate scar contracture, to redirect existing scars into natural tissue creases, and for general scar revision. Often a wound or an old scar is contracted and a Z-plasty procedure can alleviate the contracture by elongating the tissue in a desired direction. For example a severe burn victim may experience scar contracture serious enough to constrain normal movement. Being able to elongate such a scar is essential for the rehabilitation of the patient. A Z-plasty elongates tissue in the desired direction by shortening the tissue in the perpendicular direction through transposition of wedge-like flaps created from a Z-shaped incision. Though seemingly simple, there are mistakes to be made in the Z-plasty. Angles other than 60 degrees between incisions makes the directions of elongation and contraction non-perpendicular, and yields lower length gains (see figure 7.9).

Figure 7.12 illustrates a typical Z-plasty, where the scar is shown in red and the Z-shaped incision is depicted in blue. Incisions at 60 degrees produce optimal elongation. The middle figure shows how the two flaps (each with two edges) resulting from the cut are manipulated so that previously adjacent edges (along the scar) are swapped with their non-adjacent edges. The hooks around the exterior of the skin patch are used to illustrate constraints applied to our finite element mesh used to make the behavior of the skin near the flaps more realistic without simulating the surrounding body. As shown in blue in the second figure, a stitch is added partway through the procedure in order to facilitate the swapping of the adjacent flap edges. Stitching is crucial for realism in the surgery, and we also give an indication of the final stitching in the third figure. Note (especially in the video) how important stretching is to the realism of the surgical procedure, highlighting our statements about an accurate finite element model's impact on the proper prediction of surgical outcomes. The bottom three images in the figure are the same except with a superimposed grid used to illustrate patterns of tissue strain after the procedure.

Another common surgical procedure is the removal of a defect such as a malignant melanoma. Treatment for this deadly disease requires the removal of a region of skin around the growth. This produces a consistent defect on the patient's body. Covering this with local flap tissue while minimizing strain on surrounding tissues then becomes a recurring challenge for the surgeon. A Rhomboid flap is a simple technique for such a removal and subsequent covering. Often the surgeon starts with an original approach such as a Rhomboid flap but realizes that the skin did not stretch as easily as intended. The surgeon might then decide to complement the procedure with a Z-plasty to relieve the tension. This common occurrence demonstrates the importance of the ability to add progressive incisions in the deformed configuration (as we previously illustrated in Figure 7.3).

Figure 7.14 illustrates the use of our cutting algorithm in a Rhomboid flap procedure. The malignant melanoma being removed is depicted in red and our cut is depicted in blue. Removing the patch of skin containing the defect and opening the flaps (top middle). The final result after transposing the flaps and stitching (top right). Grid lines illustrate patterns of tissue strain (bottom). Each of the six incisions is made in practice by a different sweep of the scalpel, and will thus cause intersecting cuts. Even in the Z-plasty, the impreciseness of the three separate scalpel cuts will cause self-intersections of the cutting triangle mesh at the two corners of the Z. Moreover even for the most accurate surgeon, numerical errors lead to self-intersection. After making the cuts the patch of skin containing the melanoma is completely removed, and the remaining three-sided flap is lifted and moved to fill the missing patch of skin. Four pairs of edges need to be sewn together. For three of these pairs, one of their edges comes from the set of four edges exposed when the piece of skin is removed. The fourth pair of edges was formed by the two cuts that freed the flap of skin that is transposed. The stitches are depicted in blue and one can see the importance of the ability to stitch in order to carry out the surgery. Moreover, the effect of skin stretching and its importance on the surgical outcome are obvious (also see the video). Finally, we once again show a grid to illustrate the patterns of tissue strain after the procedure.

Interactivity in virtual surgery is crucial, thus a low resolution simulation mesh is required. This is not a problem for a simple tissue slab, but many procedures such as cleft lip and palate repair, breast reductions, etc. require more complex anatomical geometry that can be difficult to represent with a low number of tetrahedron. Conversely, the results of the simulated surgery need to be as reliable as possible and such results can only be confirmed under mesh refinement. Fortunately, the proposed cutting algorithm, when used as a mesh generator, allows for such scalability of detail in a nearly automatic fashion. A surgeon can block out a surgery with a

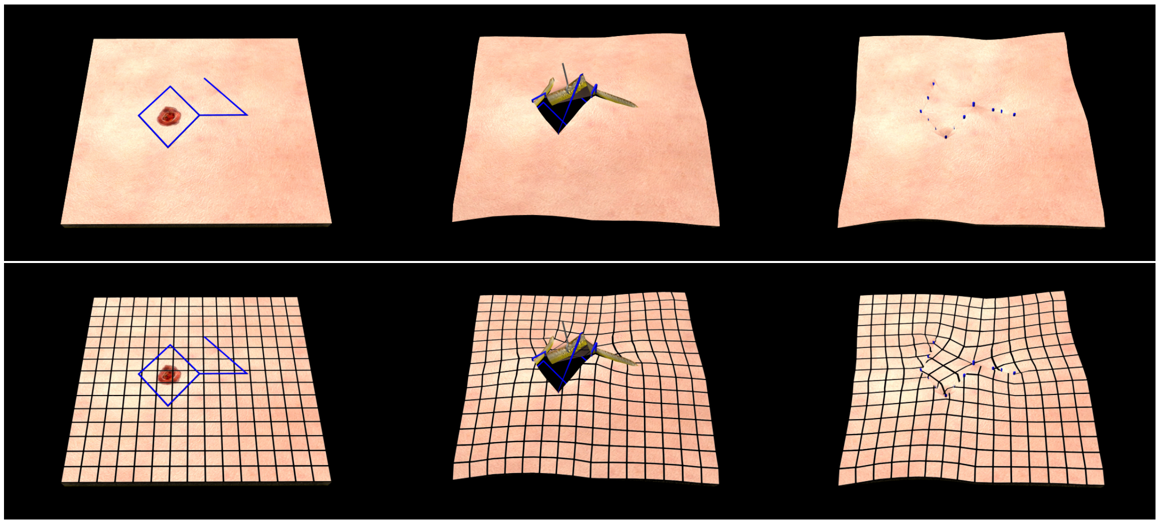

Figure 7.14: Rhomboid flap repair following the removal of a skin cancer.

coarse tetrahedral model (determined by the anatomical geometry's embedding in a low resolution background tetrahedron mesh) and then re-simulate the procedure with the same steps in an off-line high resolution simulation to confirm the results. The high and low resolution meshes are created by simply varying the resolution of the embedding tetrahedron mesh.

In Figure 7.13 our cutting algorithm is used to sketch out and simulate a breast reduction surgery. The patient's macromastia leads to problems such as neck, shoulder, and back pain and hinders physical activity and exercise. The reduction surgery removes excess breast tissue and skin resulting in breasts that are higher on the chest wall and have a more aesthetic appearance. The surgery is blocked out interactively on a low resolution model (without self-collision). The results are confirmed by repeating the simulation at higher resolution off-line with self-collision handling enabled. Accurate prediction of the surgical result requires modeling the nonlinear constitutive properties of breast skin, as well as the internal breast tissue which exhibits different material behavior, highlighting the need for a high resolution finite element model. In our simulated reduction operation a scanned triangulated surface of the patient's chest is used to carve an embedded material geometry out of a background BCC mesh using our cutting algorithm. Additional incisions subsequently remove the excess skin and breast tissue followed by the manipulation and closure of the retained tissue.

7.6 Discussion and Limitations

Despite the versatility of our cutting algorithm, a number of notable limitations exist. Our formulation assumes that cuts are introduced in the form of triangulated surfaces. Although this accommodates processes such as material fracture or explicit cutting (e.g. surgical incisions) by discretizing the cracks or incisions into a triangulated surface, there may be certain cases (e.g. elaborate scalpel models, embeddeding of geometries represented by spline or subdivision surfaces) where this additional triangulation is rather inconvenient. We note however that, with adequate refinement, a triangulated surface can represent any cut and even an increased triangle count on the cutting surface will not have a negative impact on the complexity or conditioning of the embedding simulation mesh.

Embedding the high resolution material surface in a coarser simulation mesh prevents CFL restrictions associated with small or sliver elements. Nevertheless, our framework may still create small or ill-conditioned triangles on the material surface which can still have a negative effect on collision handling. This is less of a problem for object collisions and more important for self-collisions, highlighting the need for robust collision handling schemes. Finally, although our algorithm can process any arbitrary cutting surface, a cut that partially intersects a tetrahedron without separating it into disconnected fragments will not allow the material to separate within that embedding tetrahedron. However, subsequent cuts can extend this cut to one that fully slices through the tetrahedron. Additionally, such partial cuts are geometrically resolved by our algorithm even if the embedding constraint does not allow separation of the material surface on either side of the cut. A subelement elasticity model (such as the soft binding formulation of [157]) could enable separation of these two surfaces, if such behavior were required.

7.7 Summary

We proposed a new cutting algorithm that allows for arbitrary cutting of tetrahedral meshes including those that possess a higher-resolution embedded material surface. We illustrated the efficacy of our algorithm on a number of examples ranging from cutting a single tetrahedron into over a thousand pieces to modeling and cutting a high-resolution face models embedded in a BCC simulation mesh.

Appendix A

Quasistatic treatment of muscle forces

Two muscle models are analyzed below. The first approach models the bulk behavior of flesh as an isotropic Mooney-Rivlin rubber, which is augmented with a directional component along the fiber direction. The second model is derived from [22], is inherently anisotropic and addresses along-fiber stretch, cross-fiber shear and volume change explicitly and independently. The following derivations yield the constitutive properties that will be needed in the definiteness-corrected quasistatic solver of [166].

A.1 Simplified muscle model

We only address the isotropic component of the strain energy below; the anisotropic component is addressed in section A.2.3. The energy for the isotropic response is given by a Mooney-Rivlin constitutive model for the deviatoric component of the deformation plus a pressure term as follows

$$\begin{aligned}\Psi(\mathbf{C}) &= \mu_{10}(\mathrm{tr}\hat{\mathbf{C}} - 3) + \frac{1}{2}\mu_{01}[(\mathrm{tr}\hat{\mathbf{C}})^2 - \hat{\mathbf{C}} : \hat{\mathbf{C}} - 6] + \frac{1}{2}\kappa \log^2 J \\ &= \mu_{10} I_C III_C^{-1/3} + \frac{\mu_{01}}{2}(I_C^2 - II_C) III_C^{-2/3} + \frac{\kappa}{8}\log^2 III_C + const\end{aligned}$$

The first derivatives of this energy with respect to the invariants are

$$\begin{aligned}
\Psi_I &= \mu_{10} III_C^{-1/3} + \mu_{01} I_C III_C^{-2/3} \\
\Psi_{II} &= -\frac{\mu_{01}}{2} III_C^{-2/3} \\
\Psi_{III} &= -\frac{\mu_{10}}{3} I_C III_C^{-4/3} - \frac{\mu_{01}}{3}(I_C^2 - II_C) III_C^{-5/3} + \frac{\kappa}{4}\frac{\log III_C}{III_C}
\end{aligned}$$

The resulting isotropic stress is therefore

$$\begin{aligned}
P &= 2\Psi_I F + 4\Psi_{II} F F^T F + 2J^2 \Psi_{III} F^{-T} \\
&= (2\mu_{10} III_C^{-1/3} + 2\mu_{01} I_C III_C^{-2/3}) F \\
&\quad -2\mu_{01} III_C^{-2/3} F F^T F \\
&\quad + \left(-\frac{2\mu_{10}}{3} I_C III_C^{-1/3} - \frac{2\mu_{01}}{3}(I_C^2 - II_C) III_C^{-2/3} + \frac{\kappa}{2} \log III_C \right) F^{-T} \\
&= (2\mu_{10} J^{-2/3} + 2\mu_{01} I_C J^{-4/3}) F - 2\mu_{01} J^{-4/3} F F^T F \\
&\quad + \left(-\frac{2\mu_{10}}{3} I_C J^{-2/3} - \frac{2\mu_{01}}{3}(I_C^2 - II_C) J^{-4/3} + \kappa \log J \right) F^{-T}
\end{aligned}$$

For the second derivatives of the energy we have

$$\begin{aligned}
\Psi_{I,I} &= \mu_{01} III_C^{-2/3} \\
\Psi_{I,II} &= 0 \\
\Psi_{I,III} &= -\frac{\mu_{10}}{3} III_C^{-4/3} - \frac{2\mu_{01}}{3} I_C III_C^{-5/3} \\
\Psi_{II,II} &= 0 \\
\Psi_{II,III} &= \frac{\mu_{01}}{3} III_C^{-5/3} \\
\Psi_{III,III} &= \frac{4\mu_{10}}{9} I_C III_C^{-7/3} + \frac{5\mu_{01}}{9}(I_C^2 - II_C) III_C^{-8/3} + \frac{\kappa}{4}\frac{1 - \log III_C}{III_C^2}
\end{aligned}$$

The elements of the diagonalized stress derivative $\mathcal{T}$ are given as functions of the parameters $\alpha_{ij}, \beta_{ij}, \gamma_{ij}$ where

$$\begin{aligned}
\alpha_{ij} &= 2\Psi_I + 4\Psi_{II}(\sigma_i^2 + \sigma_j^2) \\
&= 2\mu_{10} III_C^{-1/3} + 2\mu_{01} I_C III_C^{-2/3} - 2\mu_{01} III_C^{-2/3}(\sigma_i^2 + \sigma_j^2)
\end{aligned}$$

$$\begin{aligned}
\beta_{ij} &= 4\Psi_{II}\sigma_i\sigma_j - 2III_C\Psi_{III}\frac{1}{\sigma_i\sigma_j} \\
&= -2\mu_{01}III_C^{-2/3}\sigma_i\sigma_j \\
&\quad + \left(\frac{2\mu_{10}}{3}I_C III_C^{-1/3} + \frac{2\mu_{01}}{3}(I_C^2 - II_C)III_C^{-2/3} - \frac{\kappa}{2}\log III_C\right)\frac{1}{\sigma_i\sigma_j}
\end{aligned}$$

$$\begin{aligned}
\gamma_{ij} &= \begin{pmatrix} 2\sigma_i & 4\sigma_i^3 & 2III_C\frac{1}{\sigma_i} \end{pmatrix} \frac{\partial^2\Psi}{\partial\left(I_C, II_C, III_C\right)^2} \begin{pmatrix} 2\sigma_j \\ 4\sigma_j^3 \\ 2III_C\frac{1}{\sigma_j} \end{pmatrix} \\
&\quad +4III_C\Psi_{III}\frac{1}{\sigma_i\sigma_j} \\
&= \begin{pmatrix} \sigma_i & \sigma_i^3 & \sigma_i^{-1} \end{pmatrix} H \begin{pmatrix} \sigma_j \\ \sigma_j^3 \\ \sigma_j^{-1} \end{pmatrix}
\end{aligned}$$

where

$$H = \begin{pmatrix} 4\Psi_{I,I} & 8\Psi_{I,II} & 4III_C\Psi_{I,III} \\ 8\Psi_{I,II} & 16\Psi_{II,II} & 8III_C\Psi_{II,III} \\ 4III_C\Psi_{I,III} & 8III_C\Psi_{II,III} & 4III_C^2\Psi_{III,III} + 4III_C\Psi_{III} \end{pmatrix}$$

Thus for our constitutive model

$$\begin{aligned}
\eta_{11} &= 4\mu_{01}III_C^{-2/3} \\
\eta_{21} &= 0 \\
\eta_{31} &= -\frac{4\mu_{10}}{3}III_C^{-1/3} - \frac{8\mu_{01}}{3}I_C III_C^{-2/3} \\
\eta_{22} &= 0 \\
\eta_{32} &= \frac{8\mu_{01}}{3}III_C^{-2/3} \\
\eta_{33} &= \frac{16\mu_{10}}{9}I_C III_C^{-1/3} + \frac{20\mu_{01}}{9}(I_C^2 - II_C)III_C^{-2/3} + \kappa(1 - \log III_C) \\
&\quad -\frac{4\mu_{10}}{3}I_C III_C^{-1/3} - \frac{4\mu_{01}}{3}(I_C^2 - II_C)III_C^{-2/3} + \kappa\log III_C
\end{aligned}$$

$$= \frac{4\mu_{10}}{9} I_C III_C^{-1/3} + \frac{8\mu_{01}}{9}(I_C^2 - II_C) III_C^{-2/3} + \kappa$$

A.2 Full muscle model

We give the expressions for the required derivatives of the constitutive model in [23]. The strain energy is the sum of the four components W_1, W_2, W_3, Ψ^{vol} analyzed below.

A.2.1 Along-fiber shear

$$W_1 = G_1 \left(\frac{\tilde{I}_5}{\tilde{I}_4^2} - 1 \right) = G_1 \left(\frac{I_3^{-2/3} I_5}{I_3^{-2/3} I_4^2} - 1 \right) = G_1 \left(\frac{I_5}{I_4^2} - 1 \right)$$

$$\begin{aligned}
\frac{\partial W_1}{\partial I_4} &= -\frac{2G_1 I_5}{I_4^3} \\
\frac{\partial W_1}{\partial I_5} &= \frac{G_1}{I_4^2} \\
\frac{\partial^2 W_1}{\partial I_4^2} &= \frac{6G_1 I_5}{I_4^4} \\
\frac{\partial^2 W_1}{\partial I_5 \partial I_4} &= -\frac{2G_1}{I_4^3}
\end{aligned}$$

A.2.2 Cross-fiber shear

$$\begin{aligned}
W_2 &= W_2(\omega) \\
\omega &= \frac{\tilde{I}_1 \tilde{I}_4 - \tilde{I}_5}{2\sqrt{\tilde{I}_4}} = \frac{I_3^{-2/3}(I_1 I_4 - I_5)}{2 I_3^{-1/6} \sqrt{I_4}} = \frac{I_1 I_4 - I_5}{2\sqrt{I_3 I_4}} \\
\frac{\partial W_2}{\partial I_k} &= W_2'(\omega) \frac{\partial \omega}{\partial I_k} \\
\frac{\partial^2 W_2}{\partial I_k \partial I_l} &= W_2''(\omega) \frac{\partial \omega}{\partial I_k} \frac{\partial \omega}{\partial I_l} + W_2'(\omega) \frac{\partial^2 \omega}{\partial I_k \partial I_l}
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial\omega}{\partial I_1} &= \frac{1}{2}\sqrt{\frac{I_4}{I_3}} \\
\frac{\partial\omega}{\partial I_3} &= -\frac{I_1 I_4 - I_5}{4\sqrt{I_3^3 I_4}} = -\frac{\omega}{2I_3} \\
\frac{\partial\omega}{\partial I_4} &= \frac{I_1}{4\sqrt{I_3 I_4}} + \frac{I_5}{4\sqrt{I_3 I_4^3}} = \frac{I_1 I_4 + I_5}{4\sqrt{I_3 I_4^3}} \\
\frac{\partial\omega}{\partial I_5} &= -\frac{1}{2\sqrt{I_3 I_4}} \\
\frac{\partial^2\omega}{\partial I_3 \partial I_1} &= -\frac{1}{4}\sqrt{\frac{I_4}{I_3^3}} \\
\frac{\partial^2\omega}{\partial I_4 \partial I_1} &= \frac{1}{4\sqrt{I_3 I_4}} \\
\frac{\partial^2\omega}{\partial I_3 \partial I_3} &= \frac{3\left(I_1 I_4 - I_5\right)}{8\sqrt{I_3^5 I_4}} = \frac{3\omega}{4I_3^2} \\
\frac{\partial^2\omega}{\partial I_4 \partial I_3} &= -\frac{I_1 I_4 + I_5}{8\sqrt{I_3^3 I_4^3}} \\
\frac{\partial^2\omega}{\partial I_5 \partial I_3} &= \frac{1}{4\sqrt{I_3^3 I_4}} \\
\frac{\partial^2\omega}{\partial I_4 \partial I_4} &= -\frac{I_1}{8\sqrt{I_3 I_4^3}} - \frac{3I_5}{8\sqrt{I_3 I_4^5}} = -\frac{I_1 I_4 + 3I_5}{8\sqrt{I_3 I_4^5}} \\
\frac{\partial^2\omega}{\partial I_5 \partial I_4} &= \frac{1}{4\sqrt{I_3 I_4^3}}
\end{aligned}
$$

$$
\begin{aligned}
W_2(\omega) &= G_2\left(cosh^{-1}\omega\right)^2 \\
W_2'(\omega) &= 2G_2\frac{cosh^{-1}\omega}{\sqrt{\omega^2-1}} \\
\lim_{\omega\to 1} W_2'(\omega) &= 2G_2 \lim_{\omega\to 1}\frac{cosh^{-1}\omega}{\sqrt{\omega^2-1}} = 2G_2 \lim_{\omega\to 1}\frac{\frac{1}{\sqrt{\omega^2-1}}}{\frac{\omega}{\sqrt{\omega^2-1}}} = 2G_2 \lim_{\omega\to 1}\frac{1}{\omega} = 2G_2 \\
W_2''(\omega) &= 2G_2\frac{\frac{1}{\sqrt{\omega^2-1}}\sqrt{\omega^2-1} - cosh^{-1}\omega\frac{\omega}{\sqrt{\omega^2-1}}}{\omega^2-1} = \frac{2G_2 - \omega W_2'(\omega)}{\omega^2-1}
\end{aligned}
$$

$$\begin{aligned}
\lim_{\omega\to 1} W_2''(\omega) &= \lim_{\omega\to 1}\frac{2G_2-\omega W_2'(\omega)}{\omega^2-1} = -\lim_{\omega\to 1}\frac{W_2'(\omega)+\omega W_2''(\omega)}{2\omega} \Rightarrow \\
&\Rightarrow \lim_{\omega\to 1} W_2''(\omega) = -\frac{2G_2+\lim_{\omega\to 1} W_2''(\omega)}{2} \Rightarrow \lim_{\omega\to 1} W_2''(\omega) = -\frac{2G_2}{3}
\end{aligned}$$

A.2.3 Fiber stretch

$$\begin{aligned}
W_3 &= W_2(\lambda) \\
\lambda &= \sqrt{\tilde{I}_4} = I_3^{-1/6}\sqrt{I_4} \\
\frac{\partial W_3}{\partial I_k} &= W_3'(\lambda)\frac{\partial\lambda}{\partial I_k} = T(\lambda)\frac{\partial\lambda}{\partial I_k} \\
\frac{\partial^2 W_3}{\partial I_k\partial I_l} &= W_3''(\lambda)\frac{\partial\lambda}{\partial I_k}\frac{\partial\lambda}{\partial I_l} + W_3'(\lambda)\frac{\partial^2\lambda}{\partial I_k\partial I_l} = T'(\lambda)\frac{\partial\lambda}{\partial I_k}\frac{\partial\lambda}{\partial I_l} + T(\lambda)\frac{\partial^2\lambda}{\partial I_k\partial I_l} \\
T(\lambda) &= \frac{\sigma_{\max}}{\lambda_{ofl}} f_{total}
\end{aligned}$$

where f_{total} is the normalized force-length function for the passive and active component (when present).

$$\begin{aligned}
\frac{\partial\lambda}{\partial I_3} &= -\frac{1}{6}I_3^{-7/6}\sqrt{I_4} = -\frac{\lambda}{6I_3} \\
\frac{\partial\lambda}{\partial I_4} &= \frac{I_3^{-1/6}}{2\sqrt{I_4}} = \frac{\lambda}{2I_4} \\
\frac{\partial^2\lambda}{\partial I_3^2} &= \frac{7}{36}I_3^{-13/6}\sqrt{I_4} = \frac{7\lambda}{36I_3^2} \\
\frac{\partial^2\lambda}{\partial I_4\partial I_3} &= -\frac{I_3^{-7/6}}{12\sqrt{I_4}} = -\frac{\lambda}{12I_3I_4} \\
\frac{\partial^2\lambda}{\partial I_4^2} &= -\frac{I_3^{-1/6}}{4\sqrt{I_4^3}} = -\frac{\lambda}{4I_4^2}
\end{aligned}$$

A.2.4 Volume change

$$\begin{aligned}
\Psi^{vol} &= \frac{k}{2}(\log J)^2 = \frac{k}{8}(\log I_3)^2 \\
\frac{\partial \Psi^{vol}}{\partial I_3} &= \frac{k \log I_3}{4I_3} \\
\frac{\partial^2 \Psi^{vol}}{\partial I_3^2} &= \frac{k(1-\log I_3)}{4I_3^2}
\end{aligned}$$

A.3 Enforcing definiteness in general anisotropy

A note on differentiation Sometimes we need to take the partial derivative $\partial \mathbf{Y}(\mathbf{X})/\partial \mathbf{X}$ where the vector quantity $\mathbf{X}$ is restricted to a linear subspace of its ambient space. For example, the second Piola-Kirchhoff stress is defined as $\mathbf{S} = 2\partial\Psi(\mathbf{C})/\partial\mathbf{C}$, where $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ is restricted to the linear subspace of symmetric matrices. To prevent any inconsistency in the definition of the derivative, we define it in such a way that it has the proper action $\delta\mathbf{X} = \partial\mathbf{X}/\partial\mathbf{Y} : \delta\mathbf{Y}$ on any increment $\delta\mathbf{Y}$ that belongs in the proper subspace of $\mathbf{Y}$ and *annihilates* any matrix $\mathbf{M}$ in the complement of that linear subspace, that is $\partial\mathbf{X}/\partial\mathbf{Y} : M = 0$. Therefore, $\mathbf{S}$ is defined to annihilate any antisymmetric matrices ($\mathbf{S} : \mathbf{A} = 0$), which implies that it has to be symmetric itself. The same holds for higher order derivatives, i.e. $\partial^k\mathbf{X}/\partial\mathbf{Y}^k : \mathbf{Z} = \mathbf{0}$ for any argument $\mathbf{Z}$ in the orthogonal complement of the allowable space for $\mathbf{X}$.

Rotated stress We have

$$\begin{aligned}
\mathbf{P} &= \mathbf{FS} \\
&= 2\mathbf{F}\frac{\partial\Psi}{\partial\mathbf{C}} \\
&= 2\mathbf{F}\sum_m \frac{\partial\Psi}{\partial I_m}\frac{\partial I_m}{\partial C} \\
&= 2\mathbf{F}\sum_m \Psi_m \mathbf{J}_m \\
&= 2\mathbf{U}\left[\mathbf{\Sigma}\sum_m \Psi_m \hat{\mathbf{J}}_m\right]\mathbf{V}^T
\end{aligned}$$

$$= \mathbf{U}\boldsymbol{\Sigma}\hat{\mathbf{S}}\mathbf{V}^T$$

where

$$\begin{aligned}
\mathbf{J}_m &= \partial I_m / \partial \mathbf{C} \\
\hat{\mathbf{J}}_m &= \mathbf{V}^T \mathbf{J}_m \mathbf{V} \\
\Psi_m &= \partial \Psi / \partial I_m \\
\mathbf{S} &= 2\partial \Psi / \partial \mathbf{C} = 2 \sum_m \Psi_m \mathbf{J}_m \\
\hat{\mathbf{S}} &= \mathbf{V}^T \mathbf{S} \mathbf{V} = 2 \sum_m \Psi_m \hat{\mathbf{J}}_m
\end{aligned}$$

Rotated stress differentials The linearized stress is given by

$$\begin{aligned}
\delta \mathbf{P} &= \delta(\mathbf{F}\mathbf{S}) \\
&= \delta \mathbf{F}\mathbf{S} + \mathbf{F}\delta\mathbf{S} \\
&= \mathbf{U}\left[\left(\mathbf{U}^T \delta \mathbf{F} \mathbf{V}\right)\left(\mathbf{V}^T \mathbf{S} \mathbf{V}\right) + \boldsymbol{\Sigma}\mathbf{V}^T \left(\delta \mathbf{S}\right)\mathbf{V}\right]\mathbf{V}^T \\
&= \mathbf{U}\left[\left(\delta \hat{\mathbf{F}}\right)\hat{\mathbf{S}} + \boldsymbol{\Sigma}\mathbf{V}^T\left(\frac{\partial \mathbf{S}}{\partial \mathbf{C}} : \delta \mathbf{C}\right)\mathbf{V}\right]\mathbf{V}^T \\
&= \mathbf{U}\left[\left(\delta \hat{\mathbf{F}}\right)\hat{\mathbf{S}} + 2\boldsymbol{\Sigma}\mathbf{V}^T\left\{\frac{\partial^2 \boldsymbol{\Psi}}{\partial \mathbf{C}^2} : \mathbf{V}\delta\mathbf{C}\mathbf{V}^T\right\}\mathbf{V}\right]\mathbf{V}^T \\
&= \mathbf{U}\left[\left(\delta \hat{\mathbf{F}}\right)\hat{\mathbf{S}} + \boldsymbol{\Sigma}\left\{\hat{\mathcal{T}} : \delta \mathbf{C}\right\}\right]\mathbf{V}^T
\end{aligned}$$

where

$$\begin{aligned}
\delta \hat{\mathbf{F}} &= \mathbf{U}^T \delta \mathbf{F} \mathbf{V} \\
\mathcal{H}_m &= \partial^2 I_m / \partial \mathbf{C}^2 \\
\hat{\mathcal{H}}_m : \delta \mathbf{C} &= \mathbf{V}^T\left(\mathcal{H}_m : \mathbf{V}\delta\mathbf{C}\mathbf{V}^T\right)\mathbf{V} \\
\Psi_{mn} &= \partial^2 \Psi / \partial I_m \partial I_n
\end{aligned}$$

$$\begin{aligned}
\mathcal{T} &= 2\partial^2\Psi/\partial\mathbf{C}^2 = 2\sum_{m,n}\Psi_{mn}\mathbf{J}_m \otimes \mathbf{J}_n + 2\sum_m \Psi_m \mathcal{H}_m \\
\hat{\mathcal{T}} : \delta\mathbf{C} &= \mathbf{V}^T\left(\mathcal{T} : \mathbf{V}\delta\mathbf{C}\mathbf{V}^T\right)\mathbf{V} \\
\hat{\mathcal{T}} &= 2\sum_{m,n}\Psi_{mn}\hat{\mathbf{J}}_m \otimes \hat{\mathbf{J}}_n + 2\sum_m \Psi_m \hat{\mathcal{H}}_m
\end{aligned}$$

A sufficient condition for positive definiteness of the stress differential is the positive definiteness of the tensors $\hat{\mathbf{S}}$ and $\hat{\mathcal{T}}$

Define the 5 transversely isotropic invariants by

$$I_1 = \mathrm{tr}\mathbf{C},\ I_2 = \mathbf{C} : \mathbf{C},\ I_3 = \det\mathbf{C},\ I_4 = \mathbf{v}^T\mathbf{C}\mathbf{v},\ I_5 = \mathbf{v}^T\mathbf{C}^2\mathbf{v}$$

The corresponding gradients and Hessians are

$$\begin{aligned}
\mathbf{J}_1 &= \frac{\partial I_1}{\partial\mathbf{C}} = \mathbf{I} \\
\mathbf{J}_2 &= \frac{\partial I_2}{\partial\mathbf{C}} = 2\mathbf{C} \\
\mathbf{J}_3 &= \frac{\partial I_3}{\partial\mathbf{C}} = I_3\mathbf{C}^{-1} \\
\mathbf{J}_4 &= \frac{\partial I_4}{\partial\mathbf{C}} = \mathbf{v}\mathbf{v}^T \\
\mathbf{J}_5 &= \frac{\partial I_5}{\partial\mathbf{C}} = \mathbf{C}\mathbf{v}\mathbf{v}^T + \mathbf{v}\mathbf{v}^T\mathbf{C}
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{H}_1 : \delta\mathbf{C} &= \delta\frac{\partial I_1}{\partial\mathbf{C}} = \mathbf{0} \\
\mathcal{H}_2 : \delta\mathbf{C} &= \delta\frac{\partial I_2}{\partial\mathbf{C}} = 2\delta\mathbf{C} \\
\mathcal{H}_3 : \delta\mathbf{C} &= \delta\frac{\partial I_3}{\partial\mathbf{C}} = -I_3\mathbf{C}^{-1}\left(\delta\mathbf{C}\right)\mathbf{C}^{-1} + I_3\left(\mathbf{C}^{-1} : \delta\mathbf{C}\right)\mathbf{C}^{-1} \\
\mathcal{H}_4 : \delta\mathbf{C} &= \delta\frac{\partial I_4}{\partial\mathbf{C}} = \mathbf{0} \\
\mathcal{H}_5 : \delta\mathbf{C} &= \delta\frac{\partial I_5}{\partial\mathbf{C}} = \delta\mathbf{C}\mathbf{v}\mathbf{v}^T + \mathbf{v}\mathbf{v}^T\delta\mathbf{C}
\end{aligned}$$

And their rotated versions are

$$\begin{aligned}
\hat{\mathbf{J}}_1 &= \mathbf{I} \\
\hat{\mathbf{J}}_2 &= 2\boldsymbol{\Sigma}^2 \\
\hat{\mathbf{J}}_3 &= I_3\boldsymbol{\Sigma}^{-2} \\
\hat{\mathbf{J}}_4 &= \mathbf{w}\mathbf{w}^T \\
\hat{\mathbf{J}}_5 &= \boldsymbol{\Sigma}^2\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\boldsymbol{\Sigma}^2
\end{aligned}$$

and

$$\begin{aligned}
\hat{\mathcal{H}}_1 : \delta\mathbf{C} &= \mathbf{0} \\
\hat{\mathcal{H}}_2 : \delta\mathbf{C} &= 2\delta\mathbf{C} \\
\hat{\mathcal{H}}_3 : \delta\mathbf{C} &= -I_3\boldsymbol{\Sigma}^{-2}(\delta\mathbf{C})\boldsymbol{\Sigma}^{-2} + I_3\left(\boldsymbol{\Sigma}^{-2} : \delta\mathbf{C}\right)\boldsymbol{\Sigma}^{-2} \\
\hat{\mathcal{H}}_4 : \delta\mathbf{C} &= \mathbf{0} \\
\hat{\mathcal{H}}_5 : \delta\mathbf{C} &= \delta\mathbf{C}\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\delta\mathbf{C}
\end{aligned}$$

where $\mathbf{w} = \mathbf{V}^T\mathbf{v}$.

Consider the following parameterization of $\delta\mathbf{C}$

$$\delta\mathbf{C} = \sum_i d_i\mathbf{e}_i\mathbf{e}_i^T + \sum_{i<j} s_{ij}\frac{1}{\sqrt{2}}\left(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T\right)$$

The mapping from $\delta\mathbf{C}$ to $\delta\mathbf{c} = \begin{pmatrix} d_1 & d_2 & d_3 & s_{12} & s_{13} & s_{23} \end{pmatrix}^T$ is linear and orthogonal. The action of $\hat{\mathcal{T}}$ on $\delta\mathbf{C}$ can be written in matrix form as

$$\hat{\mathcal{T}} : \delta\mathbf{C} \sim \hat{\mathbf{T}}\delta\mathbf{c} = \begin{bmatrix} \mathbf{M} & \mathbf{L} \\ \mathbf{L}^T & \mathbf{N} \end{bmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ s_{12} \\ s_{13} \\ s_{23} \end{pmatrix}$$

where $\mathbf{M}, \mathbf{N}, \mathbf{L}$ are 3×3 matrices.

We can also write

$$\hat{\mathbf{T}} = 4 \sum_{m,n} \Psi_{mn} \mathbf{j}_m \mathbf{j}_n^T + 4 \sum_m \Psi_m \mathbf{H}_m$$

where

$$\mathbf{j}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{j}_2 = \begin{pmatrix} 2\sigma_1^2 \\ 2\sigma_2^2 \\ 2\sigma_3^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{j}_3 = \begin{pmatrix} I_3\sigma_1^{-2} \\ I_3\sigma_2^{-2} \\ I_3\sigma_3^{-2} \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{j}_4 = \begin{pmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ \sqrt{2}w_1w_2 \\ \sqrt{2}w_1w_3 \\ \sqrt{2}w_2w_3 \end{pmatrix} \quad \mathbf{j}_5 = \begin{pmatrix} 2\sigma_1^2 w_1^2 \\ 2\sigma_1^2 w_2^2 \\ 2\sigma_3^2 w_3^2 \\ \sqrt{2}(\sigma_1^2+\sigma_2^2)w_1w_2 \\ \sqrt{2}(\sigma_1^2+\sigma_3^2)w_1w_3 \\ \sqrt{2}(\sigma_2^2+\sigma_3^2)w_2w_3 \end{pmatrix}$$

and

$$\mathbf{H}_1 = \mathbf{H}_4 = \mathbf{0}, \;\; \mathbf{H}_2 = 2\mathbf{I}$$

$$\mathbf{H}_3 = I_3 \begin{bmatrix} 0 & \sigma_1^{-2}\sigma_2^{-2} & \sigma_1^{-2}\sigma_3^{-2} & & & \\ \sigma_1^{-2}\sigma_2^{-2} & 0 & \sigma_2^{-2}\sigma_3^{-2} & & & \\ \sigma_1^{-2}\sigma_3^{-2} & \sigma_2^{-2}\sigma_3^{-2} & 0 & & & \\ & & & -\sigma_1^{-2}\sigma_2^{-2} & & \\ & & & & -\sigma_1^{-2}\sigma_3^{-2} & \\ & & & & & -\sigma_2^{-2}\sigma_3^{-2} \end{bmatrix}$$

$$\mathbf{H}_5 = \begin{bmatrix} 2w_1^2 & & & \sqrt{2}w_1w_2 & \sqrt{2}w_1w_3 & \\ & 2w_2^2 & & \sqrt{2}w_1w_2 & & \sqrt{2}w_2w_3 \\ & & 2w_3^2 & & \sqrt{2}w_1w_3 & \sqrt{2}w_2w_3 \\ \sqrt{2}w_1w_2 & \sqrt{2}w_1w_2 & & w_1^2+w_2^2 & w_2w_3 & w_1w_3 \\ \sqrt{2}w_1w_3 & & \sqrt{2}w_1w_3 & w_2w_3 & w_1^2+w_3^2 & w_1w_2 \\ & \sqrt{2}w_2w_3 & \sqrt{2}w_2w_3 & w_1w_3 & w_1w_2 & w_2^2+w_3^2 \end{bmatrix}$$

Appendix B

Implicit springs stability

B.1 Stability conditions for 2×2 ODE systems

Consider the integration scheme

$$\mathbf{A} \begin{pmatrix} x \\ v \end{pmatrix}^{n+1} = \mathbf{B} \begin{pmatrix} x \\ v \end{pmatrix}^{n}$$

where $\mathbf{A}, \mathbf{B}$ are real-valued 2×2 matrices. The eigenvalues of the iteration matrix $\mathbf{A}^{-1}\mathbf{B}$ are roots of the quadratic $f(\lambda) = \det(\lambda \mathbf{A} - \mathbf{B}) = a\lambda^2 + b\lambda + c$. Without loss of generality, we can assume that $a > 0$ (if that is not originally the case, we can just flip the signs in one of the equations of the original system). Explicitly forming the solution of $f(\lambda) = 0$ and requiring $\|\lambda\| < 1$ can be particularly difficult since, at first glance, it yields a high order constraint on the coefficients a, b, c. However, we can show that

$$\left\{ \begin{array}{c} |c| < a \\ |b| < a + c \end{array} \right\}$$

are the necessary and sufficient conditions for stability without explicitly solving $f(\lambda) = 0$.

Let λ_1, λ_2 be the roots of f and consider the 2 cases

A. $\lambda_1, \lambda_2 \notin R$. In this case $\|\lambda_1\| = \|\lambda_2\| = \lambda_1\lambda_2 = c/a$. If we have nonreal roots then we must have $c > 0$, therefore $|c| = c$ and stability is equivalent to $c/a < 1 \Leftrightarrow |c| < a$. This is the necessary and sufficient condition for this case. We will show that $|b| < a + c$ is also necessary. Since we have nonreal roots, we must have $b^2 < 4ac < (a+c)^2 \Rightarrow |b| < |a+c| = a+c$, where the last equality hold because of $|c| < a$.

A. $\lambda_1, \lambda_2 \in R$. Stability requires $|\lambda_1| < 1 \bigwedge |\lambda_2| < 1 \Rightarrow |\lambda_1\lambda_2| < 1 \Rightarrow |c/a| < 1 \Rightarrow |c| < a$, which is a necessary condition. In this case $|b| < a + c$ is the sufficient condition. For the case of real roots, we can restate the condition $-1 < \lambda_1, \lambda_2 < 1$ as $f(1) > 0 \bigwedge f(-1) > 0 \bigwedge f'(1) > 0 \bigwedge f'(-1) < 0$ (f is a convex quadratic with roots between -1 and 1). These inequalities are written

$$\left\{\begin{array}{c} a+b+c>0 \\ a-b+c>0 \\ 2a+b>0 \\ -2a+b><0 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} |b|<a+c \\ |b|<2a \end{array}\right\}. \tag{B.1}$$

Given the condition $|c| < a$ the second inequality $|b| < 2a$ becomes redundant. Therefore, in this case, too, $\left\{\begin{array}{c} |c|<a \\ |b|<a+c \end{array}\right\}$ are the necessary and sufficient conditions.

B.2 Stability of modified Newmark

Consider the ODE

$$\ddot{x} + b\dot{x} + kx = 0 \Leftrightarrow \begin{pmatrix} x \\ v \end{pmatrix}_t = \begin{pmatrix} 0 & 1 \\ -k & -b \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}$$

for the motion of a simple 1D oscillator (we incorporate the mass of the oscillator into the constants k and b). We wish to derive the stability restriction for the following modified Newmark integration scheme

$$\begin{aligned}
x_{n+1} &= x_n + \Delta t v_{n+1/2} && \text{(B.2)}\\
x_{n+1/2} &= x_n + \frac{\Delta t}{2} v_{n+1/2} && \text{(B.3)}\\
v_{n+1/2} &= v_n + \frac{\Delta t}{2}\left(-k x_{n+1/2} - b v_{n+1/2}\right) && \text{(B.4)}\\
\hat{v}_{n+1/2} &= v_n + \frac{\Delta t}{2}\left(-k \hat{x}_{n+1/2} - b \hat{v}_{n+1/2}\right) && \text{(B.5)}\\
v_{n+1} &= 2\hat{v}_{n+1/2} - v_n && \text{(B.6)}
\end{aligned}$$

In the previous scheme, $\hat{x}_{n+1/2}$ is a position variable used as a placeholder for different approaches, each of which will be subsequently analyzed.

B.2.1 Stability of the step $(x_n, v_n) \rightarrow \left(x_{n+1/2}, v_{n+1/2}\right)$

Using equations (B.3,B.4) we have

$$\begin{pmatrix} 1 & -\Delta t/2 \\ k\Delta t/2 & 1 + b\Delta t/2 \end{pmatrix} \begin{pmatrix} x_{n+1/2} \\ v_{n+1/2} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix}$$

The characteristic polynomial for this system is

$$f(\lambda) = \det(\lambda \mathbf{A} - \mathbf{B}) = \left(1 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{4}\right)\lambda^2 + \left(-2 - \frac{b\Delta t}{2}\right)\lambda + 1$$

Thus, the conditions (B.1) become

$$\begin{aligned}
1 &< 1 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{4}\\
2 + \frac{b\Delta t}{2} &< 2 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{4}
\end{aligned}$$

which unconditionally hold for any positive Δt.

B.2.2 Stability of the step $(x_n, v_n) \rightarrow \left(x_{n+1}, v_{n+1/2}\right)$

This is a somewhat particular formulation, since it timesteps position and velocity to different points in time. However, we consider it as an upper bound to evaluate the growth in x_n, after a full step Δt. The iteration system, using equations (B.2,B.3,B.4) becomes

$$\begin{pmatrix} 1 & -\Delta t/2 \\ k\Delta t/4 & 1 + b\Delta t/2 \end{pmatrix} \begin{pmatrix} x_{n+1} \\ v_{n+1/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -k\Delta t/4 & 1 \end{pmatrix} \begin{pmatrix} x_n \\ v_n \end{pmatrix}$$

The characteristic polynomial for this system is

$$f(\lambda) = \det(\lambda \mathbf{A} - \mathbf{B}) = \left(1 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{8}\right)\lambda^2 + \left(-2 - \frac{b\Delta t}{2} + \frac{k\Delta t^2}{8}\right)\lambda + 1$$

Thus, the conditions (B.1) become

$$\begin{aligned} 1 &< 1 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{8} \\ \left|2 + \frac{b\Delta t}{2} - \frac{k\Delta t^2}{8}\right| &< 2 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{8} \end{aligned}$$

which unconditionally hold for any positive Δt.

B.2.3 Stability of the step $(x_n, v_n) \rightarrow (x_{n+1}, v_{n+1})$

Here we investigate several different definitions for the position $\hat{x}_{n+1/2}$

Assumption: $\hat{x}_{n+1/2} = x_{n+1/2}$

The assumption, combined with equations (B.3,B.4,B.5,B.6) gives the system

$$\begin{pmatrix} 1 & -\Delta t/2 \\ k\Delta t/4 & 1/2 + b\Delta t/4 \end{pmatrix} \begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t/2 \\ -k\Delta t/4 & 1/2 - b\Delta t/4 \end{pmatrix} \begin{pmatrix} x_n \\ v_n \end{pmatrix}$$

The characteristic polynomial for this system is

$$f(\lambda) = \det(\lambda\mathbf{A} - \mathbf{B}) = \left(\frac{1}{2} + \frac{b\Delta t}{4} + \frac{k\Delta t^2}{8}\right)\lambda^2 + \left(-1 + \frac{k\Delta t^2}{4}\right)\lambda + \left(\frac{1}{2} - \frac{b\Delta t}{4} + \frac{k\Delta t^2}{8}\right)$$

Thus, the conditions (B.1) become

$$\begin{aligned} \left|\frac{1}{2} - \frac{b\Delta t}{4} + \frac{k\Delta t^2}{8}\right| &< \frac{1}{2} + \frac{b\Delta t}{4} + \frac{k\Delta t^2}{8} \\ \left|1 - \frac{k\Delta t^2}{4}\right| &< 1 + \frac{k\Delta t^2}{4} \end{aligned}$$

which unconditionally hold for any positive Δt.

Assumption: $\hat{x}_{n+1/2} = x_{n+1}$

After eliminating $v_{n+1/2}$ from equations (B.2,B.3,B.4) we have

$$\left(1 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{4}\right)x_{n+1} = \left(1 + \frac{b\Delta t}{2} - \frac{k\Delta t^2}{4}\right)x_n + \Delta t v_n$$

Also, equations (B.5,B.6) give

$$\frac{k\Delta t}{2}x_{n+1} + \left(\frac{1}{2} + \frac{b\Delta t}{4}\right)v_{n+1} = \left(\frac{1}{2} - \frac{b\Delta t}{4}\right)v_n$$

The combined system is

$$\begin{pmatrix} 1 + \frac{b\Delta t}{2} + \frac{k\Delta t^2}{4} & 0 \\ \frac{k\Delta t}{2} & \frac{1}{2} + \frac{b\Delta t}{4} \end{pmatrix}\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} 1 + \frac{b\Delta t}{2} - \frac{k\Delta t^2}{4} & \Delta t \\ 0 & \frac{1}{2} - \frac{b\Delta t}{4} \end{pmatrix}\begin{pmatrix} x_n \\ v_n \end{pmatrix}$$

The characteristic polynomial for this system is

$$\begin{aligned} f(\lambda) &= \det(\lambda\mathbf{A} - \mathbf{B}) \\ &= \left(\frac{1}{2} + \frac{b\Delta t}{2} + \frac{(b^2+k)\Delta t^2}{8} + \frac{kb\Delta t^3}{16}\right)\lambda^2 + \left(-1 - \frac{b\Delta t}{2} + \frac{kb\Delta t^3}{8} + \frac{k\Delta t^2}{2}\right)\lambda \\ &\quad + \left(\frac{1}{2} - \frac{(b^2+k)\Delta t^2}{8} + \frac{kb\Delta t^3}{16}\right) \end{aligned}$$

Thus, the conditions (B.1) become

$$\begin{aligned}\left|\frac{1}{2}-\frac{(b^2+k)\Delta t^2}{8}+\frac{kb\Delta t^3}{16}\right| &< \frac{1}{2}+\frac{b\Delta t}{2}+\frac{(b^2+k)\Delta t^2}{8}+\frac{kb\Delta t^3}{16}\\ \left|-1-\frac{b\Delta t}{2}+\frac{kb\Delta t^3}{8}+\frac{k\Delta t^2}{2}\right| &< 1+\frac{b\Delta t}{2}+\frac{kb\Delta t^3}{8}\end{aligned}$$

The first condition is always true, while the second one gives

$$1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{4}>0 \Rightarrow \Delta t<\frac{b+\sqrt{b^2+4k}}{k}$$

Assumption: $\hat{x}_{n+1/2}=x_{n+1}-\frac{\Delta t}{2}v_n$

After eliminating $v_{n+1/2}$ from equations (B.2,B.3,B.4) we have

$$\left(1+\frac{b\Delta t}{2}+\frac{k\Delta t^2}{4}\right)x_{n+1}=\left(1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{4}\right)x_n+\Delta t v_n$$

Also, equations (B.5,B.6) give

$$\frac{k\Delta t}{2}x_{n+1}+\left(\frac{1}{2}+\frac{b\Delta t}{4}\right)v_{n+1}=\left(\frac{1}{2}-\frac{b\Delta t}{4}+\frac{k\Delta t^2}{4}\right)v_n$$

The combined system is

$$\begin{pmatrix} 1+\frac{b\Delta t}{2}+\frac{k\Delta t^2}{4} & 0\\ \frac{k\Delta t}{2} & \frac{1}{2}+\frac{b\Delta t}{4}\end{pmatrix}\begin{pmatrix}x_{n+1}\\ v_{n+1}\end{pmatrix}=\begin{pmatrix}1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{4} & \Delta t\\ 0 & \frac{1}{2}-\frac{b\Delta t}{4}+\frac{k\Delta t^2}{4}\end{pmatrix}\begin{pmatrix}x_n\\ v_n\end{pmatrix}$$

The characteristic polynomial for this system is

$$\begin{aligned} f(\lambda) &= \det(\lambda\mathbf{A}-\mathbf{B})\\ &= \left(\frac{1}{2}+\frac{b\Delta t}{2}+\frac{b^2\Delta t^2}{8}+\frac{k\Delta t^2}{8}+\frac{kb\Delta t^3}{16}\right)\lambda^2\\ &\quad+\left(-1-\frac{b\Delta t}{2}+\frac{k\Delta t^2}{4}-\frac{k^2\Delta t^4}{16}\right)\lambda\\ &\quad+\left(\frac{1}{2}-\frac{b^2\Delta t^2}{8}+\frac{k\Delta t^2}{8}+\frac{3kb\Delta t^3}{16}-\frac{k^2\Delta t^4}{16}\right)\end{aligned}$$

Thus, the conditions (B.1) become

$$\begin{aligned}\left|\frac{1}{2}-\frac{b^2\Delta t^2}{8}+\frac{k\Delta t^2}{8}+\frac{3kb\Delta t^3}{16}-\frac{k^2\Delta t^4}{16}\right| &< \frac{1}{2}+\frac{b\Delta t}{2}+\frac{b^2\Delta t^2}{8}+\frac{k\Delta t^2}{8}+\frac{kb\Delta t^3}{16}\\ \left|-1-\frac{b\Delta t}{2}+\frac{k\Delta t^2}{4}-\frac{k^2\Delta t^4}{16}\right| &< 1+\frac{b\Delta t}{2}+\frac{k\Delta t^2}{4}+\frac{kb\Delta t^3}{4}-\frac{k^2\Delta t^4}{16}\end{aligned}$$

As $\Delta t \to \infty$ the Δt^4 term dominates in the first inequality, which eventually becomes false. Therefore there is a finite timestep restriction.

B.2.4 Stability of Newmark with explicit positions

We use the equations

$$\begin{aligned}v_{n+1/2} &= v_n+\frac{\Delta t}{2}\left(-kx_n-bv_{n+1/2}\right)\\ x_{n+1} &= x_n+\Delta t v_{n+1/2}\\ \hat{v}_{n+1/2} &= v_n+\frac{\Delta t}{2}\left(-k\hat{x}-b\hat{v}_{n+1/2}\right)\\ v_{n+1} &= 2\hat{v}_{n+1/2}-v_n\end{aligned}$$

The first two equations yield

$$\left(1+\frac{b\Delta t}{2}\right)x_{n+1}=\left(1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{2}\right)x_n+\Delta t v_n \tag{B.7}$$

Assumption: $\hat{x}=x_{n+1}$

The last two equations of the Newmark scheme yield

$$\frac{k\Delta t}{2}x_{n+1}+\left(\frac{1}{2}+\frac{b\Delta t}{4}\right)v_{n+1}=\left(\frac{1}{2}-\frac{b\Delta t}{4}\right)v_{n+1}$$

The combined system is

$$\begin{pmatrix}1+\frac{b\Delta t}{2} & 0\\ \frac{k\Delta t}{2} & \frac{1}{2}+\frac{b\Delta t}{4}\end{pmatrix}\begin{pmatrix}x_{n+1}\\ v_{n+1}\end{pmatrix}=\begin{pmatrix}1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{2} & \Delta t\\ 0 & \frac{1}{2}-\frac{b\Delta t}{4}\end{pmatrix}\begin{pmatrix}x_n\\ v_n\end{pmatrix}$$

The characteristic polynomial for this system is

$$\begin{aligned} f(\lambda) &= \det(\lambda \mathbf{A} - \mathbf{B}) \\ &= \left(\frac{1}{2} + \frac{b\Delta t}{2} + \frac{b^2\Delta t^2}{8}\right)\lambda^2 + \left(-1 - \frac{b\Delta t}{2} + \frac{3k\Delta t^2}{4} + \frac{kb\Delta t^3}{8}\right)\lambda \\ &\quad + \left(\frac{1}{2} - \frac{b^2\Delta t^2}{8} - \frac{k\Delta t^2}{4} + \frac{kb\Delta t^3}{8}\right) \end{aligned}$$

The conditions (B.1) become

$$\begin{aligned} \left|\frac{1}{2} - \frac{b^2\Delta t^2}{8} - \frac{k\Delta t^2}{4} + \frac{kb\Delta t^3}{8}\right| &< \frac{1}{2} + \frac{b\Delta t}{2} + \frac{b^2\Delta t^2}{8} \\ \left|-1 - \frac{b\Delta t}{2} + \frac{3k\Delta t^2}{4} + \frac{kb\Delta t^3}{8}\right| &< 1 + \frac{b\Delta t}{2} - \frac{k\Delta t^2}{4} + \frac{kb\Delta t^3}{8} \end{aligned}$$

For positive Δt these conditions reduce to

$$k\Delta t^2 - b\Delta t - 2 < 0 \Rightarrow \Delta t < \frac{b + \sqrt{b^2 + 8k}}{2k}$$

Assumption: $\hat{x} = x_{n+1/2} = \frac{x_n + x_{n+1}}{2}$

The last two equations of the Newmark scheme yield

$$\frac{k\Delta t}{4}x_{n+1} + \left(\frac{1}{2} + \frac{b\Delta t}{4}\right)v_{n+1} = -\frac{k\Delta t}{4}x_n + \left(\frac{1}{2} - \frac{b\Delta t}{4}\right)v_n$$

The combined system is

$$\begin{pmatrix} 1 + \frac{b\Delta t}{2} & 0 \\ \frac{k\Delta t}{4} & \frac{1}{2} + \frac{b\Delta t}{4} \end{pmatrix} \begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} 1 + \frac{b\Delta t}{2} - \frac{k\Delta t^2}{2} & \Delta t \\ -\frac{k\Delta t}{4} & \frac{1}{2} - \frac{b\Delta t}{4} \end{pmatrix} \begin{pmatrix} x_n \\ v_n \end{pmatrix}$$

The characteristic polynomial for this system is

$$\begin{aligned} f(\lambda) &= \det(\lambda \mathbf{A} - \mathbf{B}) \\ &= \left(\frac{1}{2} + \frac{b\Delta t}{2} + \frac{b^2\Delta t^2}{8}\right)\lambda^2 + \left(-1 - \frac{b\Delta t}{2} + \frac{k\Delta t^2}{2} + \frac{kb\Delta t^3}{8}\right)\lambda \end{aligned}$$

$$+\left(\frac{1}{2}-\frac{b^2\Delta t^2}{8}+\frac{kb\Delta t^3}{8}\right)$$

The conditions (B.1) become

$$\begin{aligned}\left|\frac{1}{2}-\frac{b^2\Delta t^2}{8}+\frac{kb\Delta t^3}{8}\right| &< \frac{1}{2}+\frac{b\Delta t}{2}+\frac{b^2\Delta t^2}{8}\\ \left|-1-\frac{b\Delta t}{2}+\frac{k\Delta t^2}{2}+\frac{kb\Delta t^3}{8}\right| &< 1+\frac{b\Delta t}{2}+\frac{kb\Delta t^3}{8}\end{aligned}$$

For positive Δt these conditions reduce to

$$k\Delta t^2-2b\Delta t-4<0\Rightarrow \Delta t<\frac{b+\sqrt{b^2+4k}}{k}$$

Modification: Use Backward Euler instead of Trapezoidal rule

We replace the last two equations of the Newmark scheme with

$$v_{n+1}=v_n+\Delta t\left(-kx_{n+1}-bv_{n+1}\right)$$

The combined system is

$$\begin{pmatrix}1+\frac{b\Delta t}{2} & 0\\ k\Delta t & 1+b\Delta t\end{pmatrix}\begin{pmatrix}x_{n+1}\\ v_{n+1}\end{pmatrix}=\begin{pmatrix}1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{2} & \Delta t\\ 0 & 1\end{pmatrix}\begin{pmatrix}x_n\\ v_n\end{pmatrix}$$

The characteristic polynomial for this system is

$$\begin{aligned}f(\lambda) &= \det(\lambda\mathbf{A}-\mathbf{B})\\ &= \left(1+\frac{3b\Delta t}{2}+\frac{b^2\Delta t^2}{2}\right)\lambda^2+\left(-2-2b\Delta t-\frac{b^2\Delta t^2}{2}+\frac{3k\Delta t^2}{2}+\frac{kb\Delta t^3}{2}\right)\lambda\\ &\quad+\left(1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{2}\right)\end{aligned}$$

The conditions (B.1) become

$$\begin{aligned}\left|1+\frac{b\Delta t}{2}-\frac{k\Delta t^2}{2}\right| &< 1+\frac{3b\Delta t}{2}+\frac{b^2\Delta t^2}{2}\\ \left|-2-2b\Delta t-\frac{b^2\Delta t^2}{2}+\frac{3k\Delta t^2}{2}+\frac{kb\Delta t^3}{2}\right| &< 2+2b\Delta t-\frac{k\Delta t^2}{2}+\frac{b^2\Delta t^2}{2}\end{aligned}$$

For positive Δt these conditions reduce to

$$\frac{kb\Delta t^3}{2}+\left(2k-b^2\right)\Delta t^2-4b\Delta t-4<0$$

For critical damping ($b^2 = 4k$) this gives $\Delta t < 2.08b/k$ (compare with $\Delta t < 2.41b/k$ for the formulation in B.2.4 and $\Delta t < 1.37b/k$ for the formulation in B.2.4)

Appendix C

Implementing the cutting algorithm

In section 7.3 we defined our geometrical algorithm without descending into the implementation details of each geometric operation. This appendix provides specific details on the data structures and code organization in our implementation.

Our principle data structures are:

Per-tetrahedron Triangle List Each embedding tetrahedron stores all triangles that border the material contained inside it. Prior to any cuts, each tetrahedron's triangle list contains just its four faces. After subsequent cutting, the triangles list will also contain triangles of the cutting surface. Barycentric coordinates of the triangle vertices with respect to the tetrahedron are also stored.

Per-tetrahedron Polygon Mesh Each tetrahedron stores the polygonized boundary of its embedded material.

Intersection Registry As mentioned in section 7.3.1 the points referenced by our algorithm are either vertices of the polygon soup, intersection of an edge and a triangle, or intersection of three triangles. In fact, we represent all such points as the intersection of three triangles, using the following process: If a segment pq is common to triangles pqr and pqs, then the intersection between pq and a different triangle xyz

is encoded as the intersection of the three triangles pqr,pqs, and xyz. If pqr is the only triangle incident on pq, we introduce a virtual triangle $pq\o$ (where ø denotes an unspecified auxiliary point) such that the same point is encoded as the intersection of pqr, $pq\o$, and xyz. Similarly, a vertex p of the triangle soup is represented as the common point of three triangles pqr,pst, and puv. If no such three triangles exist, e.g. if p only appears on pqr, the point may be described as the intersection of pqr,$pq\o$, and $pr\o$, where the last two virtual triangles have been introduced accordingly. The intersection registry stores which triangles intersect at each polygon point (which may be more than three) and determines whether the intersection of three given triangles has already been registered (possibly as the intersection of three different triangles). Additionally, we store the barycentric coordinates of each intersection point with respect to each of the triangles defining the intersection. Note that virtual triangles pose no problem as the barycentric weight of the virtual vertex will be identically zero.

An uncut tetrahedralized volume is initialized with the triangle list containing the faces of each tetrahedron, the polygon mesh containing the same faces as the polygonal boundaries of each uncut tetrahedron, and the intersection registry containing only the vertices of the uncut volume, as the intersection of all embedding faces incident on each of them.

Each new cutting triangle T_{new} added to the triangle soup is processed according to the following steps:

- **Intersection with embedding volume.** T_{new} is assigned a unique index. We compute the set of embedding tetrahedra that intersect T_{new} (in world space, the embedding volume may have deformed) and we create a copy T^k_{new} for each of them, assigning each copy a unique index. Each T^k_{new} is appended to the triangle list for its corresponding tetrahedron, converted to barycentric coordinates with respect to it, and keeps a record of the unique index of its parent T_{new}.

- **Computation of new intersections.** Each new triangle T^k_{new} is intersected against all possible pairs of old triangles in the list of its respective tetrahedron.

If the intersection already exists, the intersection registry is updated to include T^k_{new} as a triangle incident on this intersection. Otherwise, a new intersection is registered and a unique index is assigned.

- **Update of old polygons.** Any old polygon edge in the polygon mesh for a tetrahedron intersected by T_{new} that contains a newly introduced intersection point is bisected into two collinear edges, and the polygon mesh is updated accordingly. Additionally, any newly created segments on the plane of an existing polygon are tested for intersection against that polygon. If they are found to be intersecting, the old polygons are split to include the new segment.

- **Creation of new polygons.** Each triangle T^k_{new} is clipped to the material contained in the embedding tetrahedron to yield a new polygon, which is added to the polygon mesh.

- **Computation of connected components.** The polygon mesh for each tetrahedron is used to compute the new connected components of material, as in section 7.3.3. Each component is defined as a polyhedral volume and the tetrahedron is split into as many copies as distinct material fragments.

- **Topology update based on connectivity.** The process of section 7.3.4 is used to rebuild the topology of the embedding volume. All of our data structures are updated to reflect the topological changes.

This process may be iterated for each new triangle added to the cutting surface. In practice, we have generalized the previous steps to accommodate the introduction of any number of new cutting triangles at once.

Appendix D

Finite Element Damping

D.1 Linear Damping

In our work we use a simplified Rayleigh damping model, in an effort to adapt the damping behavior as much as possible to the elastic constitutive model used. Using a true Rayleigh damping model (where $\partial\mathbf{f}_d/\partial\mathbf{v} = \gamma\partial\mathbf{f}_e/\partial\mathbf{x}$ and γ is the Rayleigh coefficient) would require continuous re-evaluation of the stiffness matrix $\partial\mathbf{f}_e/\partial\mathbf{x}$ during simulation. Instead, we use a constant coefficient linear damping model which is identical to Rayleigh damping around the rest state

$$\mathbf{f}_d = \gamma\left(\left.\frac{\partial\mathbf{f}_e}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}\right)\mathbf{v}$$

We can show that for isotropic constitutive models the damping stress $\mathbf{P}_d$ for each element reduces to

$$\mathbf{P}_d = \beta\left(\dot{\mathbf{F}} + \dot{\mathbf{F}}^T\right) + \alpha\mathrm{tr}\dot{\mathbf{F}}\cdot\mathbf{I} \tag{D.1}$$

for appropriate constants α and β. To prove equation (D.1) we need to show the general form of the rest state stiffness matrix $\partial\mathbf{f}_e/\partial\mathbf{x}|_{\mathbf{x}=\mathbf{x}_0}$.

Using equation (2.8) we have that in the rest state of the element

$$2\,\Psi_{I_1}|_{\mathbf{F}=\mathbf{I}} + 4\,\Psi_{I_2}|_{\mathbf{F}=\mathbf{I}} + 2\,\Psi_{I_3}|_{\mathbf{F}=\mathbf{I}} = 0 \tag{D.2}$$

since the rest state incurs no stress. Differentiating equation (2.8) around the rest state and with the help of equation (D.2) we get

$$\begin{aligned}\delta\mathbf{P}|_{\mathbf{F}=\mathbf{I}} &= \left(4\ \Psi_{I_2}|_{\mathbf{F}=\mathbf{I}} - 2\ \Psi_{I_3}|_{\mathbf{F}=\mathbf{I}}\right)\left(\delta\mathbf{F} + \delta\mathbf{F}^T\right) \\ &\quad + \left\{ \begin{pmatrix} 2 & 4 & 2 \end{pmatrix} \left.\frac{\partial^2\Psi}{\partial\left(I_1, I_2, I_3\right)^2}\right|_{\mathbf{F}=\mathbf{I}} \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} + 4\ \Psi_{I_3}|_{\mathbf{F}=\mathbf{I}} \right\} \mathrm{tr}\left(\delta\mathbf{F}\right)\mathbf{I}\end{aligned}$$

Consequently, the damping coefficients α and β in equation (D.1) are given as

$$\begin{aligned}\alpha &= \gamma\lambda = \gamma\left\{ \begin{pmatrix} 2 & 4 & 2 \end{pmatrix} \left.\frac{\partial^2\Psi}{\partial\left(I_1, I_2, I_3\right)^2}\right|_{\mathbf{F}=\mathbf{I}} \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} + 4\ \Psi_{I_3}|_{\mathbf{F}=\mathbf{I}} \right\} \\ \beta &= \gamma\mu = \gamma\left(4\ \Psi_{I_2}|_{\mathbf{F}=\mathbf{I}} - 2\ \Psi_{I_3}|_{\mathbf{F}=\mathbf{I}}\right)\end{aligned}$$

where γ is the Rayleigh coefficient and λ, μ are the *effective* Lamé coefficients of the material around its rest state.

For anisotropic material models, when the strain energy is clearly separated into an isotropic and a directional component, we use the isotropic part of the energy to determine the damping coefficients. We note that due to this simplification, our damping model need not coincide with Rayleigh damping even around the rest state. When no clear separation of the strain energy into isotropic and directional terms exists, the parameters of the linear damping model are specified experimentally.

D.2 Optimization for BCC meshes

The formulation of chapter 6 enables the use of adaptive, possibly non-manifold tetrahedral meshes consisting purely of BCC tetrahedra. We show here how this uniform shape of the simulation elements can be used to optimize the computation of finite element damping forces. Notably, the same result applies to any linear force (e.g.

linear elasticity) although the following discussion focuses on linear *damping* forces.

In the diagonalized framework of [82] equation (D.1) is reformulated as

$$\begin{aligned} \mathbf{P}_d &= \mathbf{U}\hat{\mathbf{P}}_d\mathbf{V}^T \\ \hat{\mathbf{P}}_d &= \beta\left(\hat{\dot{\mathbf{F}}} + \hat{\dot{\mathbf{F}}}^T\right) + \alpha\text{tr}\hat{\dot{\mathbf{F}}} \cdot \mathbf{I} \\ \hat{\dot{\mathbf{F}}} &= \mathbf{U}^T\dot{\mathbf{D}}_s\mathbf{D}_m^{-1}\mathbf{V} \end{aligned}$$

Any BCC tetrahedron results from the scaling and rotation of the canonical tetrahedron with vertices $X_0(0,0,0)$, $X_1(1,0,0)$, $X_2(.5,.5,-.5)$ and $X_3(.5,.5,.5)$. Thus, the rest shape matrix $\mathbf{D}_m$ is given as a function of the shape matrix $\hat{\mathbf{D}}_m$ as

$$\mathbf{D}_m = \gamma\mathbf{Q}\hat{\mathbf{D}}_m \tag{D.3}$$

where γ is a scaling factor, $\mathbf{Q}$ is a rotation and the following are true about $\hat{\mathbf{D}}_m$

$$\hat{\mathbf{D}}_m = \begin{pmatrix} 1 & .5 & .5 \\ 0 & .5 & .5 \\ 0 & -.5 & .5 \end{pmatrix}, \quad \hat{\mathbf{D}}_m^{-1} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \hat{\mathbf{D}}_m^{-T} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 1 \\ 0 & -1 & 1 \end{pmatrix}$$

Since $\hat{\mathbf{D}}_m^{-1}$ and $\hat{\mathbf{D}}_m^{-T}$ contain only the values $\{0,1,-1\}$, multiplication with these matrices can be performed only using addition and subtraction. Taking the determinants of the two sides of equation (D.3) yields $\gamma = \sqrt[3]{2\det\mathbf{D}_m}$.

The diagonalized Piola-Kirchhoff stress can be written as

$$\begin{aligned} \hat{\mathbf{P}}_d &= \beta\left(\hat{\dot{\mathbf{F}}} + \hat{\dot{\mathbf{F}}}^T\right) + \alpha\text{tr}\hat{\dot{\mathbf{F}}} \cdot \mathbf{I} \\ &= \beta\mathbf{U}^T\dot{\mathbf{D}}_s\mathbf{D}_m^{-1}\mathbf{V} + \beta\mathbf{V}^T\mathbf{D}_m^{-T}\dot{\mathbf{D}}_s^T\mathbf{U} + \alpha\text{tr}\left(\mathbf{U}^T\dot{\mathbf{D}}_s\mathbf{D}_m^{-1}\mathbf{V}\right)\mathbf{I} \\ &= \gamma^{-1}\beta\mathbf{U}^T\dot{\mathbf{D}}_s\hat{\mathbf{D}}_m^{-1}\mathbf{Q}^T\mathbf{V} + \gamma^{-1}\beta\mathbf{V}^T\mathbf{Q}\hat{\mathbf{D}}_m^{-T}\dot{\mathbf{D}}_s^T\mathbf{U} + \gamma^{-1}\alpha\text{tr}\left(\mathbf{U}^T\dot{\mathbf{D}}_s\hat{\mathbf{D}}_m^{-1}\mathbf{Q}^T\mathbf{V}\right)\mathbf{I} \\ &= \gamma^{-1}\mathbf{U}^T\left\{\beta\dot{\mathbf{D}}_s\hat{\mathbf{D}}_m^{-1} + \beta\mathbf{R}\hat{\mathbf{D}}_m^{-T}\dot{\mathbf{D}}_s^T\mathbf{R} + \alpha\text{tr}\left(\mathbf{R}\hat{\mathbf{D}}_m^{-T}\dot{\mathbf{D}}_s^T\right)\mathbf{R}\right\}\mathbf{Q}^T\mathbf{V} \end{aligned}$$

where $\mathbf{R} = \mathbf{U}\mathbf{V}^T\mathbf{Q}$ is a precomputed rotation matrix. Additionally the damping force matrix for this element is

$$\begin{aligned}
\mathbf{G}_d &= s\mathbf{P}_d\mathbf{D}_m^{-T} \\
&= s\mathbf{U}\hat{\mathbf{P}}_d\mathbf{V}^T\mathbf{D}_m^{-T} \\
&= \gamma^{-1}s\mathbf{U}\hat{\mathbf{P}}_d\mathbf{V}^T\mathbf{Q}\hat{\mathbf{D}}_m^{-T} \\
&= s\gamma^{-2}\left\{\beta\dot{\mathbf{D}}_s\hat{\mathbf{D}}_m^{-1} + \beta\mathbf{R}\hat{\mathbf{D}}_m^{-T}\dot{\mathbf{D}}_s^T\mathbf{R} + \alpha\mathrm{tr}\left(\mathbf{R}\hat{\mathbf{D}}_m^{-T}\dot{\mathbf{D}}_s^T\right)\mathbf{R}\right\}\hat{\mathbf{D}}_m^{-T}
\end{aligned}$$

where $s = (1/6)\det\mathbf{D}_m$ is the volume of the undeformed tetrahedron. Let $\hat{\alpha} = s\gamma^{-2}\alpha$, $\hat{\beta} = s\gamma^{-2}\beta$ and $\tilde{\dot{\mathbf{F}}} = \dot{\mathbf{D}}_s\hat{\mathbf{D}}_m^{-1} = (v_1 - v_0|v_2 + v_3 - v_0 - v_1|v_3 - v_2)$. We have

$$\begin{aligned}
\mathbf{G}_d &= \left\{\hat{\beta}\left(\tilde{\dot{\mathbf{F}}} + \mathbf{R}\tilde{\dot{\mathbf{F}}}^T\mathbf{R}\right) + \hat{\alpha}\mathrm{tr}\left(\mathbf{R}\tilde{\dot{\mathbf{F}}}^T\right)\mathbf{R}\right\}\hat{\mathbf{D}}_m^{-T} \\
&= \mathbf{R}\left\{2\hat{\beta}\cdot\mathrm{Sym}\left[\mathbf{R}^T\tilde{\dot{\mathbf{F}}}\right] + \hat{\alpha}\cdot\mathrm{tr}\ \mathrm{Sym}\left[\mathbf{R}^T\tilde{\dot{\mathbf{F}}}\right]\cdot\mathbf{I}\right\}\hat{\mathbf{D}}_m^{-T}
\end{aligned}$$

where Sym[$\cdot$] is the symmetric part operator. Once again, the final multiplication with $\hat{\mathbf{D}}_m^{-T}$ can be computed with additions and subtractions only.

Finally, we note that using the aforementioned optimizations, every element needs to store only the rotation $\mathbf{Q}$ and the values $\hat{\alpha}$, $\hat{\beta}$, a total of five scalars, for the computation of damping forces. If the damping coefficients are constant throughout the tetrahedralized object only the rotation $\mathbf{Q}$ and the factor $s\gamma^{-2}$ need to be stored per element.

Bibliography

[1] I. Albrecht, J. Haber, and H.-P. Seidel. Construction and animation of anatomically based human hand models. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 98–109, 2003.

[2] B. Allen, B. Curless, and Z. Popovic. Articulated body deformation from range scan data. In *Proc. of ACM SIGGRAPH 2002*, pages 612–619, 2002.

[3] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):617–625, 2005.

[4] L. Ambrosio and H. M. Soner. Level set approach to mean curvature flow in arbitrary codimension. *J. Differential Geometry*, 43:693–737, 1996.

[5] Annosoft, LLC. Basic phoneset, 2006. Available at http://www.annosoft.com/phoneset.htm.

[6] 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.

[7] 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.

[8] Z. Bao, J.M. Hong, J. Teran, and R. Fedkiw. Fracturing rigid materials. *IEEE Trans. Viz. Comput. Graph.*, 2006. (in press).

[9] D. Baraff and A. Witkin. Partitioned dynamics. Technical report, Carnegie Mellon University, 1997.

[10] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. SIGGRAPH 98*, pages 43–54, 1998.

[11] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 22:862–870, 2003.

[12] S. Basu, N. Oliver, and A. Pentland. 3D lip shapes from video: a combined physical-statistical model. *Speech Communication*, 26:131–148, october 1998.

[13] S. Basu, N. Oliver, and A. Pentland. 3D modeling and tracking of human lip motions. pages 337–343. IEEE Comput. Society, 1998.

[14] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A state machine for real-time cutting of tetrahedral meshes. In *Pacific Graph.*, pages 377–386, 2003.

[15] D. Bielser and M. Gross. Interactive simulation of surgical cuts. In *Pacific Graph.*, pages 116–125, 2000.

[16] D. Bielser, V. A. Maiwald, and M. H. Gross. Interactive cuts through 3-dimensional soft tissue. In *Eurographics*, 1999.

[17] A.W. Black, P. Taylor, and R. Caley. The festival speech synthesis system. 1999.

[18] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. In *Proc. of Eurographics*, volume 22, 2003.

[19] V. Blanz, K. Scherbaum, T. Vetter, and H.-P. Seidel. Exchanging faces in images. In *Proc. of Eurographics*, volume 23, 2004.

[20] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proc. of ACM SIGGRAPH*, pages 187–194. ACM Press, 1999.

[21] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(9):1063–1074, 2003.

[22] S. Blemker. *3D Modeling of Complex Muscle Architecture and Geometry.* PhD thesis, Stanford University, June 2004.

[23] 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.

[24] S. Blemker, J. Teran, E. Sifakis, R. Fedkiw, and S. Delp. Fast 3d muscle simulations using a new quasistatic invertible finite-element algorithm. In *10th International Symp. on Comput. Simulation in Biomechanics*, July 2005.

[25] J. Bonet and R. Wood. *Nonlinear continuum mechanics for finite element analysis.* Cambridge University Press, Cambridge, 1997.

[26] G. Borshukov, D. Piponi, O. Larsen, J. P. Lewis, and C. Tempelaar-Lietz. Universal Capture – image-based facial animation for "The Matrix Reloaded". In *ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1. ACM Press, 2003.

[27] M. Brand. Voice puppetry. In *Proc. of ACM SIGGRAPH*, pages 21–28, 1999.

[28] C. Bregler, M. Covell, and M. Slaney. Video Rewrite: driving visual speech with audio. In *Proc. of ACM SIGGRAPH*, pages 353–360, 1997.

[29] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:594–603, 2002.

[30] 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.

[31] M. Byun and N. I. Badler. FacEMOTE: Qualitative parametric modifiers for facial animations. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 65–71. ACM Press, 2002.

[32] Y. Cao, P. Faloutsos, E. Kohler, and F. Pighin. Real-time speech motion synthesis from recorded motions. In *Proc. of 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 347–355, 2004.

[33] Y. Cao, P. Faloutsos, and F. Pighin. Unsupervised learning for speech motion editing. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 225–231, 2003.

[34] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:586–593, 2002.

[35] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. A multiresolution framework for dynamic deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.*, pages 41–48. ACM Press, 2002.

[36] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Doubille, S. Prevost, and M. Stone. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Proc. of ACM SIGGRAPH*, pages 413–420. ACM Press, 1994.

[37] J. Cassell, H. H. Vilhjálmsson, and T. Bickmore. BEAT: the Behavior Expression Animation Toolkit. In *Proc. of ACM SIGGRAPH*, pages 477–486, 2001.

[38] J. Chai, J. Xiao, and J. Hodgins. Vision-based control of 3D facial animation. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 193–206, 2003.

[39] J. T. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *Proc. ACM SIGGRAPH Symp. on Comput. Anim.*, pages 77–80, 2002.

[40] Y. Chang and T. Ezzat. Transferable videorealistic speech animation. *Eurographics/ACM SIGGRAPH Symp. on Comput. Anim.*, 2005.

[41] 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.

[42] B. Choe and H.-S. Ko. Analysis and synthesis of facial expressions with hand-generated muscle actuation basis. In *Proc. of Comput. Anim.*, pages 12–19, 2001.

[43] B. Choe, H. Lee, and H.-S. Ko. Performance-driven muscle-based facial animation. *J. Vis. and Comput. Anim.*, 12:67–79, 2001.

[44] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:604–611, 2002.

[45] M. G. Choi and H.-S. Ko. Modal warping: Realtime simulation of large rotational deformation and manipulation. *IEEE Trans. Viz. Comput. Graph.*, 11:91–101, 2005.

[46] E. Chuang and C. Bregler. Mood swings: expressive speech animation. *ACM Trans. Graph.*, 24(2):331–347, 2005.

[47] M. Cohen and D. Massaro. Modeling coarticulation in synthetic visual speech. *Models and Techniques in Comput. Anim.*, 1993.

[48] 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.

[49] 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.

[50] 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.

[51] D. DeCarlo, D. Metaxas, and M. Stone. An anthropometric face model using variational techniques. In *Proc. of ACM SIGGRAPH*, pages 67–74. ACM Press, 1998.

[52] S. Delp and J. Loan. A computational framework for simulating and analyzing human and animal movement. *IEEE Computing In Science And Eng.*, 2(5):46–55, 2000.

[53] Z. Deng, J. Lewis, and U. Neumann. Synthesizing speech animation by learning compact speech co-articulation models. *Comput. Graph. Int.*, pages 19–25, 2005.

[54] 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.

[55] P. Ekman and W. V. Friesen. *Facial Action Coding System.* Consulting Psychologist Press, Palo Alto, 1978.

[56] I. Essa, S. Basu, T. Darrell, and A. Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proc. of Comput. Anim.*, pages 68–79. IEEE Comput. Society, 1996.

[57] I. Essa and A. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):757–763, july 1997.

[58] T. Ezzat, G. Geiger, and T. Poggio. Trainable videorealistic speech animation. In *ACM Trans. Graph.*, volume 21, pages 388–398. ACM Press, 2002.

[59] T. Ezzat and T. Poggio. Visual speech synthesis by morphing visemes. In *Int. J. Comp. Vision*, volume 38, pages 45–37, 2000.

[60] 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.

[61] S. Fisher and M. C. Lin. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Comput. Anim. and Sim. '01*, Proc. Eurographics Wrkshp., pages 99–111, 2001.

[62] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from a manifold mesh. In *Comput. Anim.*, pages 225–229, 2002.

[63] T. Fukunaga, Y. Kawakami, S. Kuno, K. Funato, and S. Fukashiro. Muscle architecture and function in humans. *J. Biomech.*, 30, 1997.

[64] F. Ganovelli and C. O'Sullivan. Animating cuts with on-the-fly re-meshing. In *Eurographics 2001, Short Presentations Programme*, 2001.

[65] 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.

[66] 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.

[67] A.J.W. Gielen, P.H.M. Bovendeerd, and J.D. Janssen. A three dimensional continuum model of skeletal muscle. *Comput. Meth. in Biomech. and Biomed. Eng.*, 3:231–244, 2000.

[68] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, San Diego, USA, 1981.

[69] Y. Gingold, A. Secord, J. Han, E. Grinspun, and D. Zorin. A discrete model for inelastic deformations of thin shells. In *Poster, Eurographics/ACM SIGGRAPH Symp. on Comput. Anima.*, 2005.

[70] M. Gissler, M. Becker, and M. Teschner. Local constraint methods for deformable objects. In *Proc. of the 3rd Workshop in VR Interactions and Physical Simulation (VRIPHYS)*, pages 1–8, 2006.

[71] M. Gissler, M. Becker, and M. Teschner. Local constraint methods for deformable objects. In *Poster, Eurographics/ACM SIGGRAPH Symp. on Comput. Animation*, 2006.

[72] 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.

[73] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 62–67, 2003.

[74] E. Grinspun, P. Krysl, and P. Schröder. CHARMS: A simple framework for adaptive simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:281–290, 2002.

[75] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 22(3):871–878, 2003.

[76] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proc. ACM SIGGRAPH*, pages 55–66. ACM Press, 1998.

[77] J. Guilkey and J. Weiss. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *Int. J. Num. Meth. Eng.*, 57:1323–1338, 2003.

[78] X. Guo, X. Li, Y. Bao, X. Gu, and H. Qin. Meshless thin-shell simulation based on global conformal parameterization. *IEEE Trans. on Vis. and Comput. Graph.*, 12(3):375–385, 2006.

[79] B. Heidelberger, M. Teschner, R. Keiser, M. Müller, and M. Gross. Consistent penetration depth estimation for deformable collision response. In *Proc. of Vision, Model., Vis. (VMV)*, pages 339–346, Stanford, USA, 2004.

[80] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In *Proc. of Comput. Anim.*, pages 136–146, 2001.

[81] K. Hirota, Y. Tanoue, and T. Kaneko. Generation of crack patterns with a physical model. *The Vis. Comput.*, 14:126–187, 1998.

[82] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 131–140, 2004.

[83] D. James and K. Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 22:879–887, 2003.

[84] D. James and D. Pai. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:582–585, 2002.

[85] J. Jansson and J.S.M. Vergeest. Combining deformable and rigid body mechanics simulation. *The Vis. Comput. J.*, 2003.

[86] 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 Conf., pages 129–141. Springer-Verlag, 1993.

[87] P. Joshi, W. C. Tien, M. Desbrun, and F. Pighin. Learning controls for blend shape based realistic facial animation. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 365–373, 2003.

[88] K. Kahler, J. Haber, and H.-P. Seidel. Geometry-based muscle modeling for facial animation. In *Proc. of Graph. Interface*, pages 37–46, 2001.

[89] K. Kahler, J. Haber, and H.-P. Seidel. Reanimating the dead: Reconstruction of expressive faces from skull data. In *ACM Trans. Graph.*, volume 22, pages 554–561, 2003.

[90] K. Kahler, J. Haber, H. Yamauchi, and H.-P. Seidel. Head shop: Generating animated head models with anatomical structure. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 55–63, 2002.

[91] P. Kalra, A. Mangili, N. Magnetat-Thalmann, and D. Thalmann. Simulation of facial muscle actions based on rational free form deformations. In *Proc. of Eurographics*, pages 59–69, 1992.

[92] R. Kautzman, A. Maiolo, D. Griffin, and A. Bueker. Jiggly bits and motion retargetting: Bringing the motion of Hyde to life in Van Helsing with dynamics. In *SIGGRAPH 2004 Sketches & Applications.* ACM Press, 2004.

[93] E. Keeve, S. Girod, P. Pfeifle, and B. Girod. Anatomy-based facial tissue modeling using the finite element method. In *Proc. of Visualization*, pages 21–28, 1996.

[94] S. King and R. Parent. Creating speech-synchronized animation. *IEEE Trans. on Vis. and Comput. Graph.*, 11(3):341–352, 2005.

[95] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Buren, G. Fankhauser, and Y. I. H. Parish. Simulating facial surgery using finite element models. *Comput. Graph.*, 30(Annual Conf. Series):421–428, 1996.

[96] Rolf Koch, Markus Gross, and Albert Bosshard. Emotion editing using finite elements. *Proc. of Eurographics 1998*, 17(3), 1998.

[97] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proc. of the ACM SIGGRAPH Symp. on Comput. Anim.*, pages 153–159. ACM Press, 2002.

[98] S. Kshirsagar and N. Magnenat-Thalmann. Visyllable based speech animation. In *Proc. of Eurographics*, volume 22, 2003.

[99] T. Kurihara and N. Miyata. Modeling deformable human hands from medical images. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 365–373, 2004.

[100] S. P. Lee, J. B. Badler, and N. I. Badler. Eyes alive. In *Proc. of ACM SIGGRAPH*, pages 637–644. ACM Press, 2002.

[101] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic modeling for facial animation. *Comput. Graph. (SIGGRAPH Proc.)*, pages 55–62, 1995.

[102] J. Lenoir and S. Fonteneau. Mixing deformable and rigid-body mechanics simulation. In *Comput. Graph. Int.*, pages 327–334, june 16-19 2004.

[103] J. Lewis, M. Cordner, and N. Fong. Pose space deformations: A unified approach to shape interpolation a nd skeleton-driven deformation. *Comput. Graph. (SIGGRAPH Proc.)*, pages 165–172, 2000.

[104] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. Melting and burning solids into liquids and gases. *IEEE Trans. on Vis. and Comput. Graph.*, 12(3):343–352, 2006.

[105] R. Loubère and E. J. Caramana. The force/work differencing of exceptional points in the discrete, compatible formulation of lagrangian hydrodynamics. *J. Comput. Phys.*, 216:1–18, 2006.

[106] J.C. Lucero and K.G. Munhall. A model of facial biomechanics for speech production. *J. of the Accoustical Society of America*, 106(5):2834–2842, 1999.

[107] J.C. Lucero, K.G. Munhall, E. Vatikiotis-Bateson, V.L. Gracco, and D. Terzopoulos. Muscle-based modeling of facial dynamics during speech production. *J. of the Acoustical Society of America*, 101(5):3175–3176, May 1997.

[108] N. Magnenat-Thalmann, E. Primeau, and D. Thalmann. Abstract muscle action procedures for human face animation. *The Vis. Comput.*, 3(5):290–297, 1988.

[109] D. Marchal, F. Aubert, and C. Chaillou. Collision between deformable objects using fast-marching on tetrahedral models. In *Proc. of the ACM SIGGRAPH Symp. on Comput. Anim.* ACM Press, 2004.

[110] O Mazarak, C. Martins, and J. Amanatides. Animating exploding objects. In *Proc. of Graph. Interface 1999*, pages 211–218, 1999.

[111] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, 22(3):562–568, 2003.

[112] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:385–392, 2004.

[113] 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.

[114] A. Mor and T. Kanade. Modifying soft tissue models: progressive cutting with minimal new element creation. In *MICCAI*, pages 598–607, 2000.

[115] S. Morishima, T. Ishikawa, and D. Terzopoulos. Facial muscle parameter decision from 2D frontal image. In *Proc. of the Int. Conf. on Pattern Recognition*, volume 1, pages 160–162, 1998.

[116] J. Mosegaard and T. S. Sørensen. Technical aspects of the gpu accelerated surgical simulator. In *SIGGRAPH 2006 Sketches & Applications.* ACM Press, 2006.

[117] Moving Picture Experts Group. Information technology - coding of audio-visual objects part 2: Visual. Final draft of international standard ISO/IEC JTC1/SC29/WG11 N2501 14496-2, 1998.

[118] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.*, pages 49–54, 2002.

[119] M. Müller and M. Gross. Interactive virtual materials. In *Graph. Interface*, pages 239–246, May 2004.

[120] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 471–478, 2005.

[121] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 141–151, 2004.

[122] M. Müller, 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 Wrkshp., pages 99–111. Eurographics Assoc., 2001.

[123] M. Müller, M. Teschner, and M. Gross. Physically-based simulation of objects represented by surface meshes. In *Proc. Comput. Graph. Int.*, pages 156–165, June 2004.

[124] K. Na and M. Jung. Hierarchical retargetting of fine facial motions. In *Proc. of Eurographics*, volume 23, 2004.

[125] M. Neff and E. Fiume. A visual model for blast waves and fracture. In *Proc. of Graph. Interface 1999*, pages 193–202, 1999.

[126] 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.

[127] H.-W. Nienhuys and A. F. van der Stappen. Combining finite element deformation with cutting for surgery simulations. In *Eurographics 2000, Short Presentations Programme*, 2000.

[128] H.-W. Nienhuys and A. F. van der Stappen. Supporting cuts and finite element deformation in interactive surgery simulation. Technical report, Utrecht University, Institute for Information and Computing Sciences, 2001.

[129] J. Noh and U. Neumann. Expression cloning. In Eugene Fiume, editor, *Proc. of ACM SIGGRAPH*, pages 277–288. ACM Press, 2001.

[130] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney. Animation of fracture by physical modeling. *Vis. Comput.*, 7(4):210–219, 1991.

[131] J. O'Brien, A. Bargteil, and J. Hodgins. Graphical modeling of ductile fracture. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21:291–294, 2002.

[132] J. O'Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH 99*, volume 18, pages 137–146, 1999.

[133] J. O'Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In *Proc. of SIGGRAPH 1999*, pages 137–146, 1999.

[134] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins. Combining active and passive simulations for secondary motion. *IEEE Comput. Graph. Appl.*, 20, 2000.

[135] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, 2002. New York, NY.

[136] M. Pandy. Moment arm of a muscle force. *Ex. Sprt. Sci. Rev.*, 27, 1999.

[137] 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.

[138] F. I. Parke. Computer generated animation of faces. In *Proc. of ACM Conf.*, pages 451–457. ACM Press, 1972.

[139] F. I. Parke and K. Waters. *Computer Facial Animation.* AK Peters, Ltd., 1996.

[140] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):957–964, 2005.

[141] G. Picinbono, H. Delingette, and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *IEEE Int. Conf. Robot. and Automation*, 2001.

[142] S. Pieper, J. Rosen, and D. Zeltzer. Interactive graphics for plastic surgery: A task-level analysis and implementation. In *Proc. of Symp. on Int. 3D Graph.*, pages 127–134. ACM Press, 1992.

[143] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *Proc. of ACM SIGGRAPH*, pages 75–84. ACM Press, 1998.

[144] F Pighin, J Lewis, G Borshukov, D Bennett, P Debevec, C Hery, S Sullivan, L Williams, and L Zhang. Digital face cloning. In *SIGGRAPH Course Notes*. ACM, 2005.

[145] F. Pighin, R. Szeliski, and D. Salesin. Resynthesizing facial animation through 3D model-based tracking. In *Proc. of Int. Conf. on Comput. Vision*, pages 143–150, 1999.

[146] Stephen M. Platt and Norman I. Badler. Animating facial expressions. *Comput. Graph. (SIGGRAPH Proc.)*, pages 245–252, 1981.

[147] H. Pyun, Y. Kim, W. Chae, H. W. Kang, and S. Y. Shin. An example-based approach for facial expression cloning. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 167–176, 2003.

[148] L. Reveret and I. Essa. Visual coding and tracking of speech related facial motion. In *Proc. of IEEE CVPR Int. Wrkshp. on Cues in Communication*, december 2001.

[149] S. H. Roth, M. Gross, M. H. Turello, and S. Carls. A Bernstein-Bézier based approach to soft tissue simulation. *Comput. Graph. Forum (Proc. Eurographics)*, 17(3):285–294, 1998.

[150] 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.

[151] F. Scheepers, R. Parent, W. Carlson, and S. May. Anatomy-based modeling of the human musculature. *Comput. Graph. (SIGGRAPH Proc.)*, pages 163–172, 1997.

[152] D. Serby, M. Harders, and G. Székely. A new approach to cutting into finite element models. In *MICCAI*, pages 425–433, 2001.

[153] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.*, 93:1591–1595, 1996.

[154] E. Sifakis, K. Der, and R. Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim. (in press)*, 2007.

[155] E. Sifakis, I. Neverov, and R. Fedkiw. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 2005.

[156] E. Sifakis, A. Selle, A. Robinson-Mosher, and R. Fedkiw. Simulating speech with a physics-based facial muscle model. *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 261–270, 2006.

[157] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim. (in press)*, 2007.

[158] P. Sloan, C. Rose, and M. Cohen. Shape by example. In *Proc. of 2001 Symp. Int. 3D Graph.*, pages 135–143, 2001.

[159] J. Smith, A. Witkin, and D. Baraff. Fast and controllable simulation of the shattering of brittle objects. In D. Duke and R. Scopigno, editors, *Comput. Graph. Forum*, volume 20(2), pages 81–91. Blackwell Publishing, 2001.

[160] D. Steinemann, M. Harders, M. Gross, and G. Szekely. Hybrid cutting of deformable solids. In *Proc. of the IEEE Virtual Reality Conference*, pages 35–42, 2006.

[161] D. Steinemann, M. A. Otaduy, and M. Gross. Fast arbitrary splitting of deforming objects. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 63–72, 2006.

[162] W. Stinson and P. Thuriot. Bulging muscle and sliding skin: Deformation systems for Hellboy. In *SIGGRAPH 2004 Sketches & Applications*. ACM Press, 2004.

[163] R. Sumner and J. Popović. Deformation transfer for triangle meshes. In *ACM Trans. on Graph. (Proc. ACM SIGGRAPH)*, volume 23, pages 399 – 405, 2004.

[164] R. Taylor, E. Wilson, and S. Sacket. Direct solution of equations by frontal and variable band active column methods. In *Europe-U.S. Wrkshp.: Nonlinear Finite Element Analysis in Structural Mechanics*. Springer-Verlag, 1980.

[165] 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.

[166] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2005.

[167] J. Teran, E. Sifakis, S. Salinas-Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Trans. on Vis. and Comput. Graph.*, 11(3):317–328, 2005.

[168] D. Terzopoulos and K. Fleischer. Deformable models. *The Vis. Comput.*, 4(6):306–331, 1988.

[169] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Comput. Graph. (SIGGRAPH Proc.)*, pages 269–278, 1988.

[170] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Comput. Graph. (Proc. SIGGRAPH 87)*, 21(4):205–214, 1987.

[171] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis, and animation. *J. Vis. and Comput. Anim.*, 1:73–80, december 1990.

[172] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6), june 1993.

[173] D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. In *Graph. Interface*, pages 146–154, 1988.

[174] M. Teschner, S. Girod, and B. Girod. Direct computation of nonlinear soft-tissue deformation. In *Proc. of Vision, Modeling, and Visualization*, pages 383–390, november 2000.

[175] M. Teschner, B. Heidelberger, M. Müller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *Proc. Comput. Graph. Int.*, pages 312–319, 2004.

[176] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. of Vision, Modeling, Visualization (VMV)*, pages 47–54, Munich, Germany, 2003.

[177] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*, 40:1528–1538, 1995.

[178] U.S. National Library of Medicine. The visible human project, 1994. http://www.nlm.nih.gov/research/visible/.

[179] E. Vatikiotis-Bateson, K.G. Munhall, M. Hirayama, Y. Lee, and D. Terzopoulos. Dynamics of facial motion in speech: Kinematic and electromyographic studies of orofacial structures. In *Speechreading by Humans and Machines*, volume 150 of *NATO ASI Series on Computer and System Sciences*, chapter 16, pages 231–232. Springer-Verlag, March 1996.

[180] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. In *ACM Trans. Graph. (Proc. ACM SIGGRAPH)*, volume 24, pages 426–433, 2005.

[181] P. Volino and N. Thalman. Implementing fast cloth simulation with collision response. In *Proc. of the Int. Conf. on Comput. Graph.*, page 257. IEEE Comput. Society, 2000.

[182] X. C. Wang and C. Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proc. ACM SIGGRAPH Symp. on Comput. Anim.*, pages 129–138, 2002.

[183] Y. Wang, X. Huang, C. S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang. High resolution acquisition, learning and transfer of dynamic 3-D facial expressions. In *Proc. of Eurographics*, pages 677–686, september 2004.

[184] K. Waters and J. Frisbie. A coordinated muscle model for speech animation. In *Proc. of Graph. Interface*, pages 163–170, may 1995.

[185] Keith Waters. A muscle model for animating three-dimensional facial expressions. *Comput. Graph. (SIGGRAPH Proc.)*, pages 17–24, 1987.

[186] 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.

[187] M. Wicke, D. Steinemann, and M. Gross. Efficient animation of point-sampled thin shells. In *Proc. of Eurographics*, volume 24, 2005.

[188] J. Wilhelms and A. Van Gelder. Anatomically based modeling. *Comput. Graph. (SIGGRAPH Proc.)*, pages 173–180, 1997.

[189] L. Williams. Performance-driven facial animation. In *Comput. Graph. (Proc. of Int. Conf. on Comput. Graph. and Int. Techniques)*, pages 235–242. ACM Press, 1990.

[190] Y. Wu, P. Kalra, N. Magnenat-Thalmann, and D. Thalmann. Simulating wrinkles and skin aging. *The Vis. Comput.*, 15(4):183–198, 1999.

[191] Gary D. Yngve, James F. O'Brien, and Jessica K. Hodgins. Animating explosions. In *Proc. of ACM SIGGRAPH 2000*, pages 29–36, 2000.

[192] 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. Biomech.*, 35:1253–1262, 2002.

[193] 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.

[194] L. Zhang, N. Snavely, B. Curless, and S. Seitz. Spacetime faces: High resolution capture for modeling and animation. In *ACM Trans. Graph. (Proc. ACM SIGGRAPH)*, volume 23, pages 548–558. ACM Press, 2004.

[195] Q. Zhang, Z. Liu, B. Guo, and H. Shum. Geometry-driven photorealistic facial expression synthesis. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 16–22. ACM Press, 2003.

[196] Q. Zhu, Y. Chen, and A. Kaufman. Real-time biomechanically-based muscle volume deformation using FEM. *Comput. Graph. Forum*, 190(3):275–284, 1998.