Physically Based Forehead Modelling and Animation including Wrinkles

Mark Anthony Warburton

Department of Computer Science
The University of Sheffield

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

September 2014

Supervisor: Dr Steve Maddock

Abstract

There has been a vast amount of research on the production of realistic facial models and animations, which is one of the most challenging areas of computer graphics. Recently, there has been an increased interest in the use of physically based approaches for facial animation, whereby the effects of muscle contractions are propagated through facial soft-tissue models to automatically deform them in a more realistic and anatomically accurate manner. Presented in this thesis is a fully physically based approach for efficiently producing realistic-looking animations of facial movement, including animation of expressive wrinkles, focussing on the forehead. This is done by modelling more physics-based behaviour than current computer graphics approaches.

The presented research has two major components. The first is a novel model creation process to automatically create animatable non-conforming hexahedral finite element (FE) simulation models of facial soft tissue from any surface mesh that contains hole-free volumes. The generated multi-layered voxel-based models are immediately ready for simulation, with skin layers and element material properties, muscle properties, and boundary conditions being automatically computed. The second major component is an advanced optimised GPU-based process to simulate and visualise these models over time using the total Lagrangian explicit dynamic (TLED) formulation of the FE method. An anatomical muscle contraction model computes active and transversely isotropic passive muscle stresses, while advanced boundary conditions enable the sliding effect between the superficial and deep soft-tissue layers to be simulated.

Soft-tissue models and animations with varying complexity are presented, from a simple soft-tissue-block model with uniform layers of skin and muscle, to a complex forehead model. These demonstrate the flexibility of the animation approach to produce detailed animations of realistic gross- and fine-scale soft-tissue movement, including wrinkles, with different muscle structures and material parameters, for example, to animate different-aged skin. Owing to the detail and accuracy of the models and simulations, the animation approach could also be used for applications outside of computer graphics, such as surgical applications. Furthermore, the animation approach can be used to animate any multi-layered soft body (not just soft tissue).

Acknowledgements

I wish to express my sincere appreciation to everyone who has contributed to and supported me during this PhD. Specifically, I would like to thank the Engineering and Physical Sciences Research Council (EPSRC), and the Department of Computer Science at the University of Sheffield for providing the main source of funding for this research. Very special thanks to the Rabin Ezra Scholarship Trust for a bursary providing me with additional funds to attend and publish this research at conferences, and also to NVIDIA Corporation, a sponsor of the TPCG 2013 conference, for providing an NVIDIA Tesla K20 GPU Accelerator upon receipt of the Best Student Paper Award at TPCG 2013, which was extremely useful for running GPU-based simulations.

I would like to express my sincere gratitude to my supervisor, Dr Steve Maddock, for his continuous guidance throughout the PhD. I am extremely grateful for all of the valuable support and time that he has dedicated. I would like to thank Dr Mike Stannett for his involvement on the supervisory panel, and his helpful comments during panel meetings. Thanks also to Dr Richard Clayton (Department of Computer Science, The University of Sheffield) and Dr Xiaosong Yang (National Centre for Computer Animation, Bournemouth University) for a pleasant viva, and their much appreciated comments and suggestions on this thesis.

Special thanks to everyone who provided useful information and advice. Thanks to Dr Zeike Taylor (Department of Mechanical Engineering, The University of Sheffield) for information on the use of the TLED FE method for GPU-based soft-tissue simulation. Thanks to Professor Pat Lawford and Dr Andrew Narracott (Department of Cardiovascular Science, The University of Sheffield) for information on facial soft-tissue anatomy. Thanks to Dr Adrian Jowett (The School of Clinical Dentistry, The University of Sheffield) for information on the anatomy of forehead soft-tissue structures. Thanks to Dr Stephanie Davy-Jow (School of Natural Sciences and Psychology, Liverpool John Moores University) for providing a facial surface mesh, including skin, muscle and skull surfaces, that was experimented with during the early stages of the PhD. Thanks also to all members, past and present, of the Virtual Reality, Graphics & Simulation research group at the University of Sheffield for providing a pleasant and sociable working environment.

Finally, I would like to wholeheartedly acknowledge all of my family, especially my parents, Gary and Janette, and my brother and sister-in-law, Daniel and Anita, as well as all of my friends, those who have been there from the start, and those whom I have met along the PhD journey. I cannot thank my family and friends enough for the continuous support and encouragement they have provided, as well as the much needed relief from the stresses of PhD life. I greatly appreciate their support and encouragement that helped me through this PhD experience.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Facial modelling and animation is one of the most challenging areas of computer graphics. Facial surfaces are not only extremely complex, but any small inaccuracies of facial movement greatly reduce its realism due to our familiarity with how faces should appear and deform. Computational models of human or non-human faces, as well as being crucial in the films and games industries, are necessary for a range of other applications including low bandwidth teleconferencing, visual speech applications and facial surgery simulation. Since the pioneering work of Parke [Par72], there has been a vast amount of research with the aim of generating more realistic and efficient facial models and animations.

Today, surface-based or volumetric facial models can be created using various techniques, such as using data from laser scanners or photographs, using anthropometric data, or interpolating between existing faces in a face space. Some popular animation techniques include key framing, with the help of processes like facial rigging, and parameterised, performance-based and pseudo-muscle animation. However, by using a physically based approach, incorporating a biomechanical muscle and facial soft-tissue model, the effects of muscle contractions can be propagated through the facial soft tissue to automatically deform the model in a more realistic and anatomical manner. This research focusses on physically based animation.

As shown by Figure 1.1, the production of physically based animations involves three main stages:

1. Creation of a surface mesh for an object
2. Creation of a physically based simulation model
3. Simulation of the model over time

Physically based animations firstly require an appropriate physically based model to be created. Stages 1 and 2 form this modelling stage. Typically in computer graphics, a surface mesh is firstly created (Stage 1), which is a visual representation defining the boundaries of the object to be simulated. This is used to create a simulation model (Stage 2), which is a physics-based representation encapsulating the necessary data to simulate the object using the desired physics-based technique. The surface mesh is attached or bound to the simulation model, and, as this model is simulated using a physics-based technique (Stage 3), it moves and deforms the surface mesh to produce animation.

Physics-based soft-tissue simulation approaches often focus on either efficiently producing realistic-looking animations for computer graphics applications [TW90, KHS01], or simulating models with high physical accuracy for studying soft-tissue behaviour [FM08, KSY08b] or surgical simulation [KRG+02, ZHD06]. The former simulate large areas, such as the face, with just enough physics to efficiently produce the desired realism of animations, and normally only simulate gross movement using physics-based techniques, rather than fine details like wrinkles [ZST05b, SNF05]. While they typically consist of muscle and skin models, sometimes along with skull and wrinkle

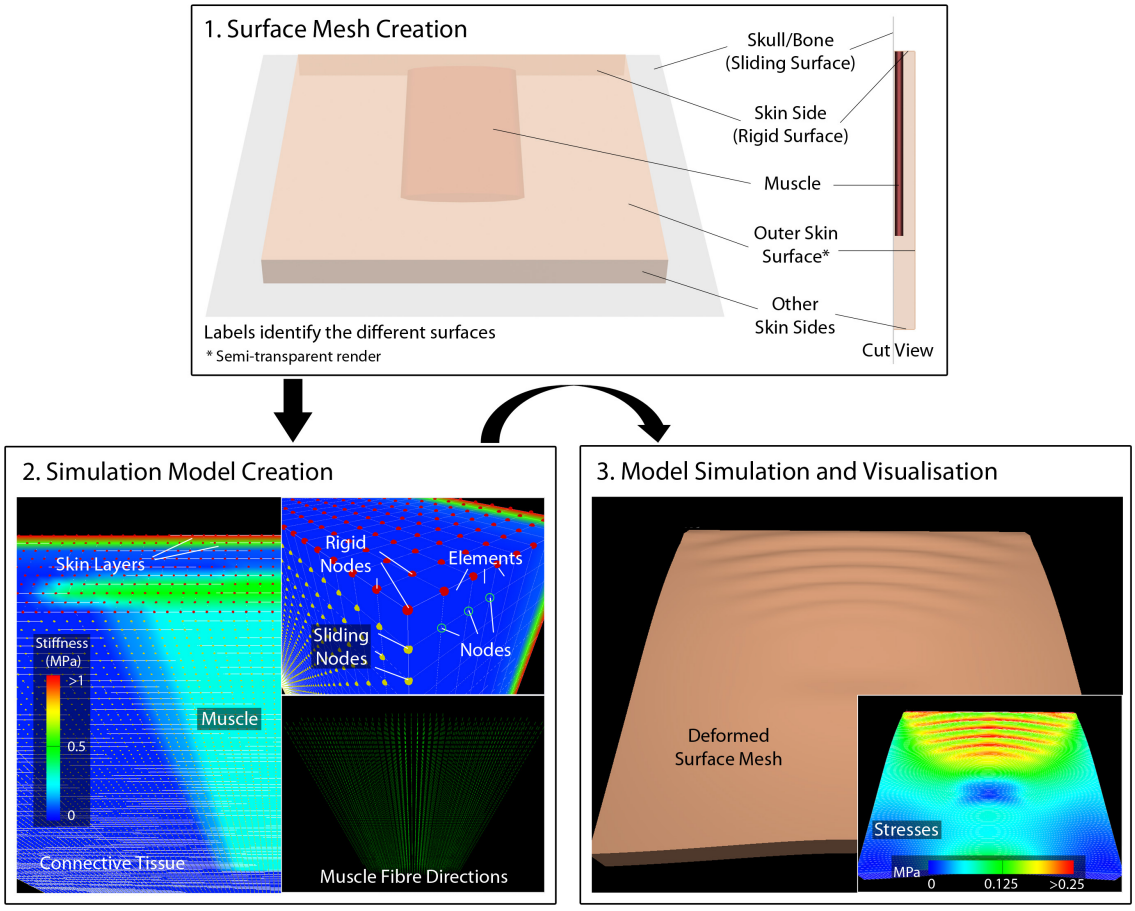

Figure 1.1: An overview of the physically based animation approach presented in this thesis, which contains the three main stages involved in the production of physically based animations.

models, normally only the skin and connective tissue are simulated using physics-based techniques. The skin and connective tissue are normally simulated using the efficient mass-spring (MS) method [KHS01, Fra05], or a physics engine that focusses on performance and stability [CP06, Fra12], such as NVIDIA PhysX[1]. Simpler and more efficient heuristic techniques are often used to animate muscles represented as vectors [Wat87, ZPS04] or geometric volumes [KHS01, Cou05], and layer skin wrinkles onto a surface mesh by modifying the mesh texture [RBM08, BKN02] or geometry [BKN02, MC10]. Such techniques have also been used to layer wrinkles onto MS facial models [WKMMT99, ZST05b].

On the other hand, applications with high accuracy requirements often use much more detailed physics-based models to accurately simulate small areas, like a block of skin for wrinkle simulation [KSY08b, HMSH09], or they involve small deformations [KRG+02, BJTM08]. These approaches are normally required to use the more accurate but computationally complex finite element (FE) method [ZHD06, FM08], or the FE-based but precomputation-heavy mass-tensor (MT) method [MSNS05, XLZH11]. Unlike the MS method, the FE method is based on continuum mechanics, making it more suitable for modelling continuous material like soft tissue, although with higher computational cost, and comparisons between MS and FE systems have shown these differences [HHR+03, MSN+07]. When modelled, these high-accuracy approaches often simulate muscles [HMSH09, BWL+10] and wrinkles [FM08, KSY08b] in a physics-based manner.

Given increases in computational power, and a drive for producing more realistic animations, physically based facial animation systems in computer graphics are simulating increasingly more physical phenomena with higher accuracy [SNF05]. GPU computing architectures like CUDA[2], OpenCL[3] and Microsoft DirectCompute (part of the DirectX APIs[4]) that allow exploitation of powerful GPUs have also allowed for huge improvements in computational speed of FE simulations, which are now possible in real time [TCO08, LGK11]. Such advances are expanding the use of FE simulations to more common applications like films and games [PO09].

Regarding the creation of a physically based model, the surface mesh can be created using various techniques, for example, from simply using 3D modelling tools to sculpt a mesh, to segmenting medical data [ZHD06, BJTM08]. The simulation model can then either conform to a surface mesh [BJTM08, Fra12], or a non-conforming model [SNF05, DGW11], such as a voxel representation, can be used. High-quality conforming models are often difficult and time-consuming to create, and require considerable manual work, although such models may be required for high-accuracy applications. In contrast, non-conforming models can enable more efficient production of stable, realistic-looking animations for computer graphics applications. With non-conforming models, the surface mesh is typically bound to and animated with the simulation model. Once a physically based model has been created, techniques have been used to adapt such models to fit a new surface mesh [KHYS02, LCC+12], for example, to create a new facial model from a reference facial model, simplifying the creation of successive models.

1.1 Project Overview

By modelling more physics-based behaviour than current computer graphics approaches, the aim of this research is to develop a fully physically based approach for efficiently producing realistic-looking animations of facial movement, including animation of expressive wrinkles, focussing on the forehead region of the face. This involves animation of both gross movement, such as raising the eyebrows, and complex finer details, such as forehead wrinkles, in a physically based manner using the accurate FE method. Such animations are produced by creating and simulating detailed multi-layered voxel-based FE models of facial soft tissue (the soft tissue between the skull and outer skin surface, as shown by Figure 1.2). While this research focuses on producing soft-tissue animations, the presented animation approach can also be used to create any multi-layered FE

[1] http://www.nvidia.com/object/physx_new.html
[2] http://www.nvidia.com/object/cuda_home_new.html
[3] http://www.khronos.org/opencl/
[4] http://msdn.microsoft.com/en-gb/directx/

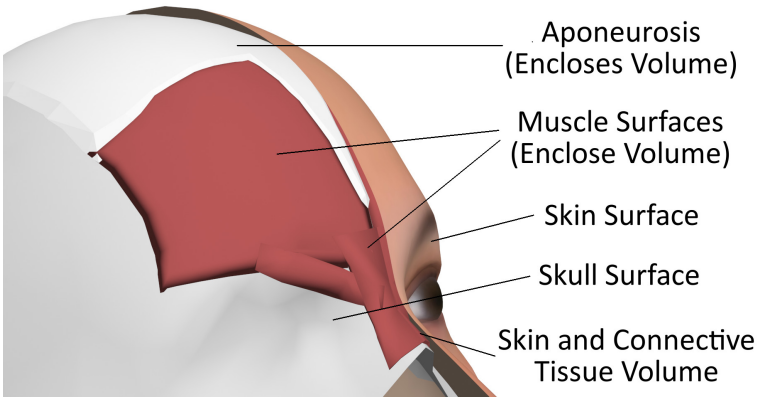

Figure 1.2: Surfaces and volumes of a facial soft-tissue model. The whole volume between the skull and skin surfaces (i.e. the skin and muscle volumes) is discretised to create an FE facial soft-tissue model.

model from any surface mesh, and animate any multi-layered soft body.

This research has two major components:

- A novel model creation process for the automatic creation of complete physically based simulation models (Stage 2 of physically based animations)
- An advanced optimised GPU-based simulation process to simulate and visualise the models over time (Stage 3)

Creation of a surface mesh (Stage 1), required as input to the model creation process, can be done using a variety of techniques from manual sculpting using 3D modelling tools, to using medical data. The model creation process automatically computes all simulation model properties, making the models immediately ready for simulation. It generates more detailed models than current approaches [SNF05, WHHM13]; for example, skin layers are included, enabling complex gross- and fine-scale behaviour to be simulated, including wrinkling. There are currently no known approaches that animate large structures, such as the forehead region of the face, including wrinkles, using entirely physics-based techniques. While the focus is on producing realistic-looking animations for computer graphics applications, this research is therefore further bridging the gap between the two categories of physically based soft-tissue simulation approaches. Detailed and accurate soft-tissue models and techniques, typically used for modelling small areas of soft tissue to study its behaviour [FM08, HMSH09], are used on large areas like the forehead, current models of which don't include enough detail to simulate wrinkles [SNF05, BJTM08].

The model creation process automatically generates animatable non-conforming hexahedral (voxel-based) FE simulation models of facial soft tissue from a surface mesh. This includes automatic computation of skin layers and material properties for these multi-layered models, as well as muscle properties (e.g. fibre directions) and boundary conditions (constraints). The voxel-based models offer various simulation advantages, particularly regarding performance and stability.

The simulation process is then used to simulate and visualise these models using the total Lagrangian explicit dynamic (TLED) formulation of the FE method, which is well suited for simulating large deformations with the highly nonlinear material behaviour of soft tissue. An anatomical muscle contraction model is used that considers both active and transversely isotropic passive muscle stresses. The processing of advanced boundary conditions enables the sliding effect between superficial soft-tissue layers, and the deep layers and skull to be modelled [WMSH10], which is often neglected [SNF05, BJTM08]. The simulation and visualisation system has been implemented on the GPU using CUDA, and it has been optimised to exploit the computational advantages offered when simulating voxel-based models on the GPU.

The research detailed in this thesis has also been presented in various international computer graphics conference and journal publications. Two conference publications focus on the model creation process, including a comparison between conforming and non-conforming simulation models [WM12, WM13a]. Further conference and journal publications focus on the simulation process [WM13c, WM14], and the GPU-based implementation and optimisation of the simulation and visualisation system [WM13b]. Additionally, the model creation and simulation procedures have been presented as a poster at an international physiological sciences conference, demonstrating the applicability of the research to a wider area, such as computational biomechanics [WM13d].

1.2 Research Applications and Limitations

This research is mainly targeted at computer graphics applications like films and games. Realistic and more accurate wrinkling behaviour can be automatically produced as a result of the simulations by simulating more physics-based behaviour than current computer graphics facial animation approaches, using highly detailed models with the accurate FE method. The feasibility and efficiency of the approach is also enhanced by using voxel-based models rather than conforming models, which offer various performance and stability advantages, particularly when simulated on the GPU. By using different muscle structures and material parameters, for example, to animate different-aged skin, a wide range of detailed fully physically based animations can be easily produced. Since the animation approach is not limited to animating human soft tissue, it could be used to create and simulate detailed models of other arbitrarily shaped and structured faces, such as fictional faces, which will deform automatically according to the physical laws implemented. Furthermore, as well as soft tissue, it can be used to create and simulate models of arbitrary soft bodies.

In addition to computer graphics applications, owing to the detail and accuracy of the models and simulations, the animation approach could also be useful in other fields where high accuracy is necessary. For example, it could be used in biomechanics, where detailed models are often used to study fine soft-tissue behaviour [FM08, KSY08b], but current such studies have been limited to models of small simple blocks of soft tissue. In the surgical field, the research could be incorporated into a system either to conduct virtual surgery for training or before a complex procedure is carried out on a real patient, or to produce post-operative surgical predictions, including detailed animations of the final face post operation. While there are already various physically based facial simulation systems that have been built for surgical applications [KRG+02, CLP03], most of these have limited animation capabilities of the final face, and don't simulate fine details like wrinkles. The research could also be similarly used, for example, in the field of forensic anthropology where detailed facial surfaces, including muscles, are reconstructed using a skull surface. Complex simulation models could be automatically created and simulated using these surfaces to enable detailed animations of these faces to be produced, rather than just static models.

Since the main aim of this research is to reproduce realistic-looking forehead movement during expressions, the animation approach is purely physics-based, modelling attributes that have a large impact of forehead soft-tissue deformation. For example, the general material properties of soft-tissue structures are modelled, but not less significant physical attributes like thermal effects, fluid dynamics, and abnormalities such as swelling, and biological and chemical attributes of soft tissue. Furthermore, while highly detailed models are simulated, such models do not consider the complete anatomy of soft tissue and the forehead. For example, small structures such as vessels, glands and hair follicles are not modelled (the effect of hair on the simulations is therefore not considered in this research). All of the soft-tissue models represent the skin and connective tissue using three soft-tissue layers, which is necessary and sufficient to accurately simulate most wrinkling properties [MTKL+02, FM08]. Each skin layer also has constant thickness, an assumption that has been validated [BJM11]. Deep soft-tissue structures are not modelled, and only the most significant superficial muscles that have a large active or passive effect on forehead soft-tissue movement during expressions are included in the forehead models. However, more muscles and soft-tissue layers could be created and simulated using the animation approach.

1.3 Thesis Outline

Chapters 2 to 5 present background information related to physically based soft-tissue and facial animation. Specifically, Chapter 2 details the relevant anatomy of the human head, including the skin and connective tissue, muscles and skull, an understanding of which is essential when producing physically based facial soft-tissue models. Chapter 3 introduces various aspects of physically based soft-body (e.g. soft tissue) animation, including physics-based simulation techniques, and the different types and components of physics-based models. Chapter 4 reviews current approaches for modelling and simulating the different structures of physics-based facial models, including full facial animation systems that incorporate physically based techniques, and approaches for simulating wrinkles. Both the efficient approaches used in computer graphics, and the higher-accuracy approaches used in other areas of science and engineering, and surgical applications are explored. Chapter 5 reviews current approaches of creating different types of physically based models, including the creation of a surface meshes and volumetric simulation models, some of which have been used to create various models described in Chapter 4.

Chapters 6 to 9 present the physically based modelling animation approach, starting with an overview of the approach in Chapter 6. This chapter also introduces the forehead model that is used to illustrate various stages detailed in Chapters 7 to 9, and describes the steps taken to create the surface mesh for this model. Following this, Chapter 7 presents the model creation process to automatically create animatable multi-layered voxel-based FE simulation models of facial soft tissue from a surface mesh. Chapter 8 then explains the simulation process used to simulate such models, and the GPU-based implementation and optimisation of the simulation and visualisation system. Chapter 9 demonstrates the animation approach by presenting example models and animations of varying complexity. Some qualitative comparison to other soft-tissue and wrinkle animation approaches is also presented, along with a technique for quantitative evaluation of wrinkle simulations. Finally, further discussion of the animation approach and results is presented, along with possible future work, and experimental work that has been completed towards the most significant improvements. Chapter 10 presents the conclusions.

Chapter 2

Anatomy of the Human Head

When producing a realistic physically based facial model, it is important to accurately model the biological structures of the head by which its appearance is determined; therefore, understanding the anatomy of the head is essential. The human head is a complex structure of bones, cartilage, ligaments, tendons, muscles, nerves, blood vessels, glands, fatty tissue, connective tissue, skin and hair; however, there are currently no known models that simulate anatomy to this level of detail, and details that are believed to have less effect on the deformations produced are often excluded. This chapter examines the major biological structures of the head that have a significant impact on the appearance and deformation of the face, and should therefore be represented in a realistic physically based facial model. These are (refer to Figure 2.1):

- Skin and connective tissue - The outermost soft tissue, the structure of which leads to the formation of wrinkles
- The skull - The main supportive structure of the head that defines an inner boundary and attachment surface for facial soft tissue
- Facial muscles - The structures situated between the skin and the skull that drive facial movement to produce expressions

Skin and connective tissue is composed of multiple layers (see Figure 2.1), from the outermost skin layers to the tough deep layers that surround bone. Soft-tissue structures, such as muscles, tendons and ligaments, may also be enclosed within a specific layer, or pass through multiple layers, connecting superficial layers to deeper layers and bone. The following sections present the general mechanics of soft-tissue layers and structures, before exploring the skin and connective tissue layers, skull and facial muscles in further detail. Several relevant resources have been continuously referred to while completing the following sections[1] [Sta09, Far10, PW08, MTKL+02, Bar05], although, due to the complexity, there are disagreements and uncertainties in the literature considering various areas of anatomy, such as the structure of different skin layers.

2.1 General Mechanics of Soft Tissue

While the composition of each soft-tissue layer and structure differs, and can be described by different material properties, each such layer and structure has nonlinear, anisotropic, inhomogeneous and viscoelastic properties. The elastic nature of soft tissue refers to its tendency to return to its rest shape when an applied load is removed. The viscous nature of soft tissue refers to the fact that the internal forces that are generated due to deformation are dependent on both the amount and rate of deformation. Soft tissue, particularly skin, also has plasticity properties, preventing it from returning to its exact rest shape after a particular load has been reached, leading to the

[1] Information on facial soft-tissue anatomy was also obtained from meetings with Professor Pat Lawford and Dr Andrew Narracott (Department of Cardiovascular Science, The University of Sheffield)

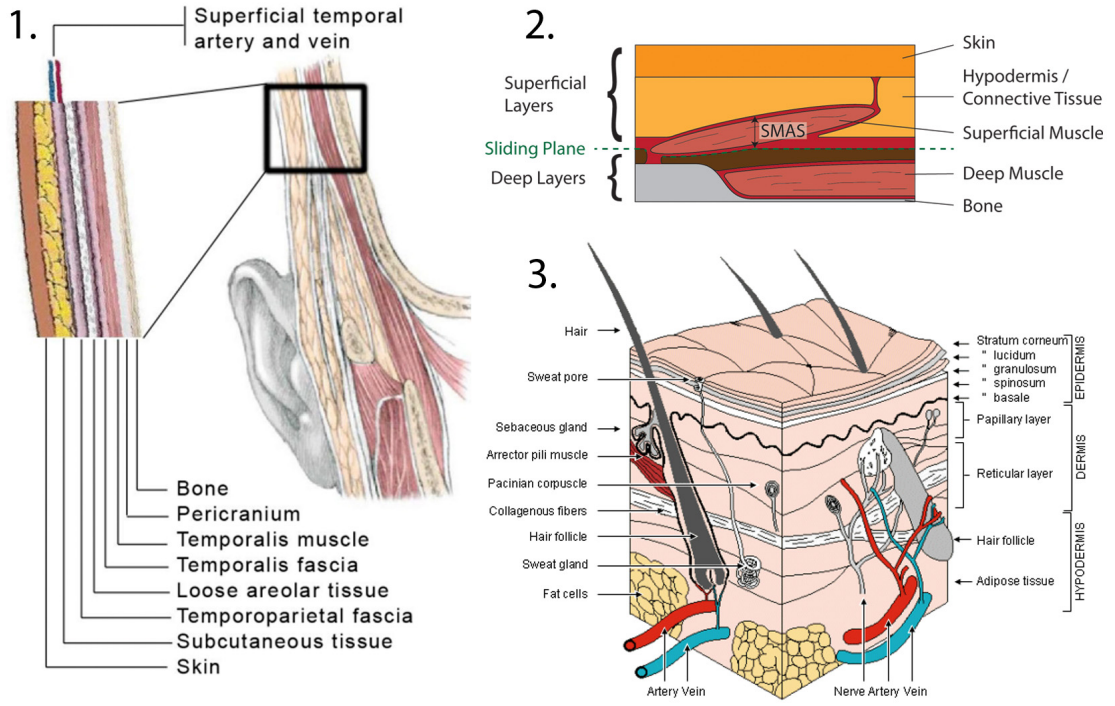

Figure 2.1: **1.** The facial soft-tissue layers [DSVO11] (used with permission). **2.** The superficial and deep soft-tissue layers. **3.** A more detailed diagram of skin, showing the general structure and layers of skin [MP95] (used with permission).

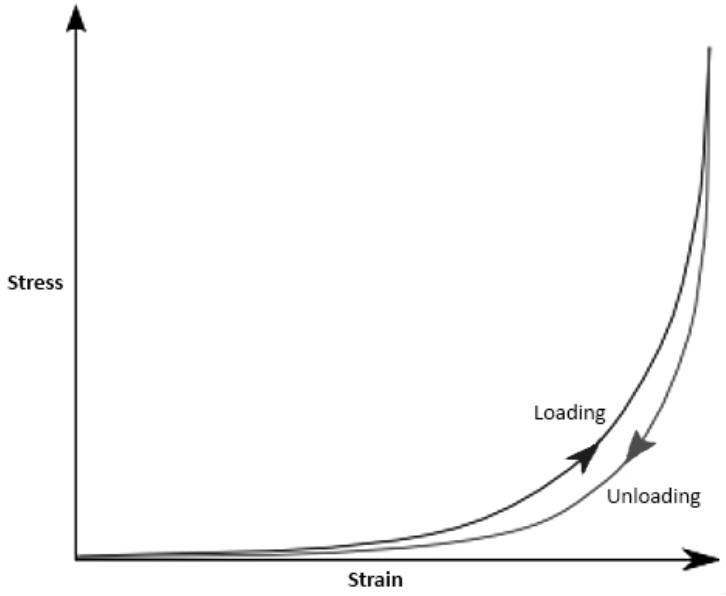

Figure 2.2: The stress to strain relationship of soft tissue under application of a load, also illustrating hysteresis.

formation of permanent ageing wrinkles. Other properties of soft tissue due to its viscoelastic nature include:

- Hysteresis - The stress-strain curve during loading is different to that during unloading.
- Stress relaxation - The force opposing a deformation that is held constant reduces over time.
- Creep - Strain increases over time when under a constant load.
- Preconditioning - Repeated applications of the same load can result in different deformations.

Soft tissue has an almost biphasic linear stress-strain relationship (see Figure 2.2), which is how many physically based computer graphics models represent soft tissue.

Various studies have been conducted to explore and find numerical values for the mechanical properties of soft tissue [Hen05]. Barbarino et al. focussed on facial soft tissue, using MRI scans and the inverse FE method to determine the mechanical properties [BJM11]. This study also validates the common assumption of modelling soft-tissue layers with a uniform mechanical response and thickness through different facial regions. Such studies are extremely useful, and values found can be used directly with an FE soft-tissue simulation.

Table 2.1 provides some numerical values for the mechanical properties of superficial soft-tissue layers, as well as muscle, including Young's moduli[2] and Poisson ratios[3], collected from various literature. For nonlinear material models, the Young's moduli represent initial Young's moduli in the material rest states, which change during simulations depending on the stretches of the materials, and other parameters of the specific material models used. The difficulty of accurately measuring and simulating the response of soft tissue can be seen in the wide range of Young's moduli, material models and soft-tissue thicknesses reported in Table 2.1. We qualitatively experimented with these values to determine those used for our animation examples, presented in Chapter 9.

2.2 Skin and Connective Tissue

In humans, skin is the largest and heaviest of the organs, covering the entire external surface of the body. As shown by Figure 2.1, two primary layers can be identified - the epidermis and the dermis, each of which are divided into several sublayers. The dermis rests on the hypodermis (subcutaneous layer), which is not a part of the skin but consists of connective tissue that attaches the skin to the underlying superficial musculo aponeurotic system (SMAS). Posterior to these superficial soft-tissue layers are the tough deep layers, including the deep fascia.

While the composition of soft-tissue layers and structures is similar all over the body, their thickness varies, and various factors such as age, gender and hydration have an effect on the composition, texture and appearance of soft tissue. Although often modelled in computer graphics with smooth boundaries, the interfaces between the layers of soft tissue in reality, such as at the dermo-epidermal junction, appear 'bumpy', as modelled by Magnenat-Thalmann et al. [MTKL+02]. The density of skin is roughly 1100kg/m^3, while the density of other soft-tissue layers and structures is approximately 1000kg/m^3 [Fly07]. The following subsections describe the structure and properties of the different soft-tissue layers identified in Figure 2.1.

2.2.1 Epidermis

The epidermis is the outer layer of the skin. It varies in thickness over different areas of the body, from between around 50μm on the eyelids to around 100μm on the palms and soles. The epidermis can be divided into five layers, the thickest and most significant being the stratum corneum (the outer layer). The stratum corneum is a layer, around 10 to 20μm thick, of dead cells that consists

[2] A measure of material stiffness often specified in pascals (Pa or N/m^2) or megapascals (MPa).

[3] The ratio of transverse contraction strain to longitudinal extension strain of a material (a ratio of 0.5 means the material is incompressible).

Resource	Soft-Tissue Location	Material Models	Young's Modulus (MPa)	Poisson Ratio	Thickness (mm)
Kuwazuru et al. 2008 [KSY08b]	Face	Linear Buckling	Young Skin: 6 (SC), 0.136 (VE), 0.04 (PD), 0.08 (RD), 0.034 (H) Aged Skin: 12 (SC), 0.272 (VE), 0.04 (PD), 0.16 (RD), 0.034 (H)	0.49	Young Skin: 0.02 (SC), 0.18 (VE), 0.2 (PD), 1.1 (RD), 2 (H) Aged Skin: 0.02 (SC), 0.08 (VE), 0.5 (PD), 1.1 (RD), 2 (H)
Hung et al. 2009* [HMSH09]	Forehead	Mooney-Rivlin	56 (SC), 0.056 (VE+D), 0.018 (H), 0.06 (M)	0.49	0.02 (SC), 3.96 (VE+D), 0.96 (H), 2.56 (M)
Barbarino et al. 2008, 2011* [BJTM08, BJM11]	Forehead	Mooney-Rivlin	0.025 (S), 0.0038 (H), 0.024 (M)	0.49	1.7 (S), 2.3 (H+M)
Flynn and McCormack, 2008* [FM08]	Forearm	Neo-Hookean (SC), Orthotropic Viscoelastic [BAG04] (D), Yeoh (H)	5 - 4000, ∼240 at 75% relative humidity (SC), 0.003 (D), 0.0098 (H)	0.49	0.02 (SC), 1.2 (D), 1.5 (H)
Magnenat-Thalmann et al. 2002 [MTKL+02]	General	Hookean	Young Skin: 6 (SC), 0.05 (D), 0.6 (H) Aged Skin: 12 (Dry SC), 0.05 (D), 1 (H)	0.49	Young Skin: 0.015 (SC), 0.05 (D), 1.235 (H) Aged Skin: 0.015 (SC), 0.2 (D), 1.085 (H)
Yin et al. 2010 [YGC10]	Fingertip	Hookean	3 (SC), 0.136 (VE), 0.08 (D), 0.034 (H), 17,000 (B)	0.48	0.15 (SC), 0.12 (VE), 1.16 (D), 3.86 (H), 4.2 (B)
Kim et al. 2010 [KJW+10]	Face	Hookean	0.003 (S+H), 0.5 (M)	0.46 (S+H), 0.43 (M)	
Beldie et al. 2010 [BWL+10]	Face	Hookean (S+M), Neo-Hookean (H)	0.015 (S), 0.00042 (H), 0.0062 (M)	0.49 (S), 0.495 (H), 0.49 (M)	-
Chabanas et al. 2003 [CLP03]	Face	Hookean	0.015 (S+H), 0.0062 (M)	0.49	
Gladilin et al. 2004 [GZDH04]	Face	St Venant-Kirchhoff	0.0022 (S+H)	0.45	
Wu et al. 1999** [WKMMT99]	Face	Hookean	10 (low strain), 100 (large strain), 0.008 (H)	0.333	3

Table 2.1: A summary of mechanical properties of superficial soft-tissue layers and structures collected from various experiments and models of soft tissue.
Key: SC: Stratum Corneum, VE: Viable Epidermis, PD: Papillary Dermis, RD: Reticular Dermis, D: Entire Dermis (PD+RD), S: Entire Skin (SC+VE+PD+RD), H: Hypodermis, M: Muscle, B: Bone
* Young's moduli estimated from parameters provided for specific material models
** Average general skin elasticity and connective tissue parameters

mainly of the strong protein keratin, making it tough and flexible. It acts as a barrier to bacteria and to retain water in the skin, while helping to hold the skin firmly together. It is composed of roughly 60% proteins and 20% water, which contrasts to roughly 70% water composition in the rest of the epidermis. Its surface is covered by a network of small grooves and changes frequently as new epidermal cells are continuously pushed up from the stratum basale during differentiation, with each cell having a life cycle of around 45 to 75 days before it falls off. It has been found that modelling the stratum corneum has a large effect on the deformations produced by skin models, whereas the effects of the other epidermal layers are much smaller [Fly07].

2.2.2 Dermis

The dermis is mainly composed of collagen and elastin fibres, and it ranges in thickness across the body from around 1mm to 3mm. The upper, papillary layer is composed mainly of thin collagen fibres, whereas the thicker, denser reticular dermis is composed of thicker fibres. The interwoven collagen fibres are arranged curled up in bundles parallel to the skin surface. They are tough and contribute the majority of the mass of the skin, whereas elastin fibres provide elasticity and resilience. The sudden increase in stress when skin is being deformed is due to the stretching of collagen once it has been uncurled and aligned with the deformation. The dermis also contains various structures, such as hair follicles, sweat glands, nerve endings and blood vessels. Various dense fibrous bundles called retaining ligaments originate on the skull and insert into the dermis, fixing deeper facial structures to the skin layers, and binding the soft-tissue layers and structures together.

2.2.3 Hypodermis, Superficial Musculo Aponeurotic System (SMAS) and Deep Fascia

The hypodermis consists of loose connective tissue composed of fat cells and loosely arranged fibres. This viscoelastic composition allows skin to freely move around, and the elasticity enables the skin to be then brought back to place again. Until recently, there has been little reported data about its mechanical properties [Gee09]. The hypodermis is quite variable in thickness depending on many factors such as gender, age, nutrition and body position. Some parts of the face, such as eyelids and external ear, are fat-free and therefore contain no hypodermis. While the hypodermis generally exhibits lower stiffness than the dermis, and also contains larger blood vessels and nerve fibres than those found in the dermis, no clear boundary exists between the dermis and hypodermis.

Posterior to the hypodermis is the superficial fascia, also called the SMAS, which is a 'fibro-fatty' layer consisting of collagen, elastin fibres and fat cells. The SMAS surrounds and interlinks the superficial facial muscles and structures, and fibrous septae extend through the hypodermis to connect the SMAS to the dermis, for example, at the locations of retaining ligaments and muscle insertions into the skin. Although the SMAS has no bony attachment, most such muscles run deep to the SMAS and have extensive origins in the facial skeleton. Other superficial muscles, like the frontalis, only attach to the SMAS. Deep muscles and structures, such as the temporalis muscle, insert into tougher deeper layers posterior to the SMAS. Thin but strong and dense fibrous tissue containing closely packed bundles of collagen fibres, known as the deep fascia, surrounds such deep structures. Deep fascia is also present inside superficial and deep muscles, dividing them into bundles of fibres. Upon muscle contraction, the SMAS is able to slide over the tougher deeper layers, such as the deep fascia and bone [WMSH10], although such sliding movement is restricted by retaining ligaments.

2.2.4 Wrinkles

Wrinkles are folds or creases in the skin. They can occur due to compression or distortion of the soft-tissue layers (e.g. during muscle contraction), and changes in material properties of these layers (e.g. due to increased hydration). The different material properties of the soft-tissue layers,

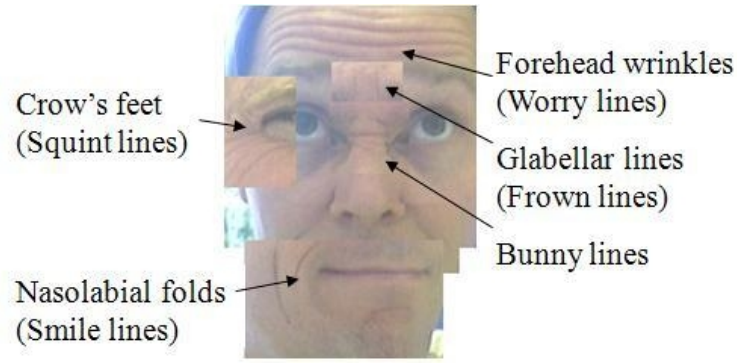

Figure 2.3: Diagram showing different types of facial wrinkles, courtesy of Dr Steve Maddock[4].

in particular the contrast between the tough stratum corneum and the looser posterior superficial layers, lead to the production of wrinkles under such conditions [MTKL+02, FM08].

Two groups of wrinkles can be identified, which are expression and permanent wrinkles. Expression wrinkles are temporary and caused by the contraction of muscles. The direction of such wrinkles are approximately perpendicular to the muscle's line of action. As humans age, their skin thins, dries, wrinkles and becomes unevenly pigmented. Permanent ageing wrinkles start to form, particularly at points affected by a lot of stress. This happens as, with age, the elastic fibres of the skin become stretched, reducing the elasticity of the skin, and the change of the collagen fibres make the skin less homogeneous. Also, a loss in subcutaneous fatty tissue results in sagging skin and fallen nasal tips, and the weakening of elasticity in the retaining ligaments cause the surrounding soft tissue to droop [Bli98]. Ageing wrinkles can be grouped according to Glogau's classification of mild wrinkles (typically occurring between ages 28 and 35), moderate (ages 35 to 50), advanced (ages 50 to 60) and severe wrinkles (ages 60 to 75) [Bra06]. The Fitzpatrick classification of wrinkle lines can be used to classify the degree of wrinkling around the mouth and eyes, from class I (fine wrinkles) to class III (fine to deep wrinkles with numerous lines) [Der10].

As well as age, other factors can also determine the properties of skin, which in turn determines the appearance of wrinkles; for example, females usually have softer and thinner skin than males, making wrinkles generally more pronounced in males [Bar05]. However, due to lower initial skin collagen content, women's skin appears to age earlier than men's, and it is suggested that women's skin appears around 15 years older than a man's skin of the same age throughout their adult life [Bar05]. Research also shows that wrinkling often appears in different areas according to gender; for example, women tend to develop more wrinkles in the perioral region around the mouth than men [PTKK09]. Factors such as sun exposure and smoking can damage the skin, causing permanent wrinkles to form, and factors including weight (particularly after losing subcutaneous fat, which happens naturally with age), skin moisture content and stress can also affect the structure of the skin, affecting the formation of wrinkles [Ken06].

Figure 2.3 illustrates some facial wrinkles. Significant facial wrinkles include horizontal forehead wrinkles, that occur due to contraction of the frontalis when raising the eyebrows. Glabellar lines appear as vertical forehead wrinkles between the eyebrows, and are typically caused by contraction of the procerus and corrugator supercilii muscles, for example, when frowning. Smiling can lead to the formation of crow's feet (fine wrinkles at the outside corners of the eyes), bunny lines (fine wrinkles on the sides of the nose), and deeper nasolabial folds (significant folds that extend from the corners of the mouth to the sides of the nose). Each of these wrinkles can become more permanent over time due to skin ageing, and the change of skin properties, which reduce the elasticity of skin.

[4] http://staffwww.dcs.shef.ac.uk/people/S.Maddock/

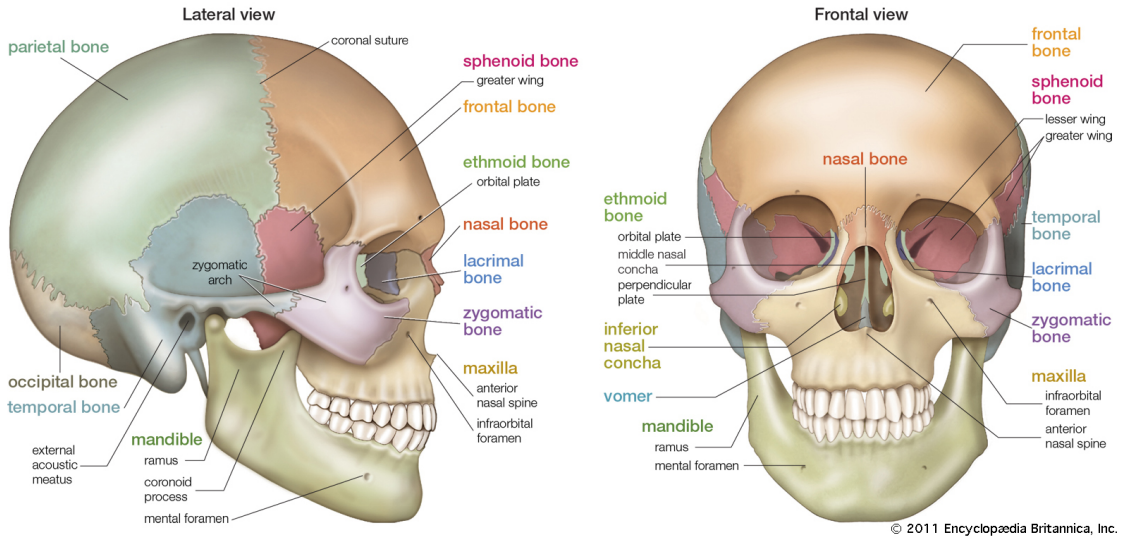

Figure 2.4: Diagram showing the bones of the skull [Enc11a] (by courtesy of Encyclopaedia Britannica, Inc., copyright 2011, used with permission).

2.3 The Skull

As shown by Figure 2.4, the skull is a series of flattened or irregular bones that define the overall shape of the face. The skull is divided into two parts - the cranium, consisting of 8 bones that enclose the brain, and the facial skeleton, which has 14 bones. The facial skeleton can be further divided into three sections - the upper third, consisting of the nasal bones and orbits, the central third, containing the maxillae, nasal cavities and nose, and the lower third, containing the mandible (jawbone). Although it is the contraction of muscles that move the mandible, many physically based animation approaches simply use geometric transformations to move the mandible, which moves and deforms the muscles that influence it [KHS01, ZPS04]. Most bones of the skull are either paired, or relatively symmetrical and cross the median plane (the plane which divides the face in an almost symmetrical manner), providing symmetry to the head. The average shapes and sizes of regions of the skull, including the proportion of the three facial skeleton regions, vary with gender and age [TMSP80].

Apart from the mandible, the bones are immovably jointed together. The mandible is hinged to the rest of the skull at the temporomandibular joints (TMJs) on both sides of the skull. A TMJ is actually composed of two joints, with the lower joint enabling rotational movement, and the upper joint enabling translational movement. Both TMJs can rotate along almost transverse (horizontal) axes together to open the jaw slightly, whereas larger movements where the mouth is opened more than 15 degrees involve translational movement [BSSS10]. Grinding movements to a particular side involve the rotation of a single TMJ around an almost vertical axis, while the other TMJ swings forward and inward in a translational movement.

2.4 Facial Muscles

2.4.1 Facial Muscle Anatomy

Muscles are contractile fibrous tissues responsible for the movement of body parts. Most facial muscles originate in the skull (muscle origin), extend through the SMAS, and attach to the dermis or connective tissue (muscle insertion), although some attach to just SMAS (e.g. the frontalis, responsible for raising the eyebrows). Deeper muscles originate in the skull, and insert into the SMAS or deeper layers, although some attach to just bone (e.g. the masseter, the primary muscle

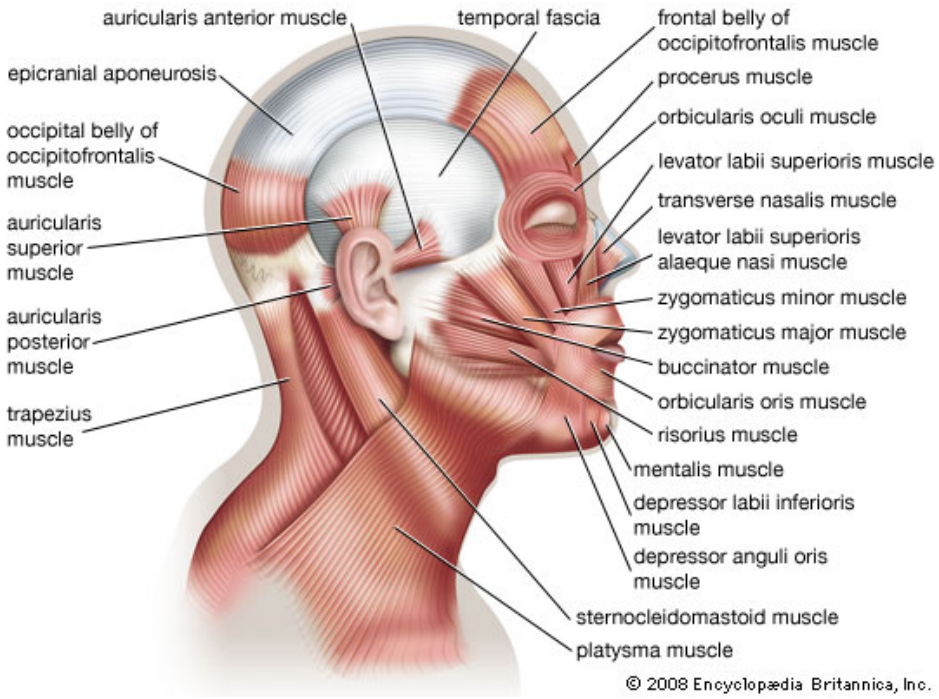

Figure 2.5: Diagram showing some of the most significant muscles of the head [Enc11b] (by courtesy of Encyclopaedia Britannica, Inc., copyright 2008, used with permission).

involved in elevation and protraction of the mandible). Most facial muscles also have one origin and one insertion. When a muscle contracts, the insertion is pulled towards the origin. Although there are more than 50 muscles on the head (some of which can be considered groups of smaller muscles themselves), some deeper muscles are not anatomically considered as facial muscles, and some have little influence on facial deformation. These are therefore often excluded from models. Figure 2.5 shows the positions and shapes of the most significant muscles on the head.

The muscles of the face belong to the group of skeletal muscles. Figure 2.6 shows the structure of a skeletal muscle. They contain bundles of cylindrical fibres (cells), usually around 10 to 100 microns in diameter, that can span the entire length of a muscle. The endomysium surrounds fibres within a bundle, while the perimysium surrounds these bundles, and each muscle is surrounded by the epimysium. Both of these are sheets of thin connective tissue that are part of the deep fascia. Thick, tough tissues called tendons, which are composed of around 86% collagen, are continuations of the epimysia, and these attach muscles to bones. Muscle fibres are composed of bundles of even smaller elements called myofibrils, normally around 1 to 2 microns in diameter, that run the length of the fibre. These bundles are surrounded by endomysium, and the myofibrils are surrounded by sarcoplasm. Each myofibril consists of many short filaments called sarcomeres, which are the contractile units of muscle fibres. They consist of cross-bridges of overlapping myosin and actin protein filaments, enabling muscle fibres to contract (see Figure 2.7).

Although muscle fibres aren't composed of elastin and collagen, they also have a biphasic passive stress-strain response (i.e. without stimulation) as they are stretched beyond equilibrium length. The increased stress at higher strain levels is due to the stretching of muscle fibres once they have been fully aligned with the deformation. This passive response can be illustrated by the passive curve on a muscle fibre tension-length graph, as shown by Figure 2.8. The combination of the passive response from muscle fibres and the surrounding fascia leads to significant transversely isotropic behaviour of muscles. Loose connective tissue attaches facial muscles to skin.

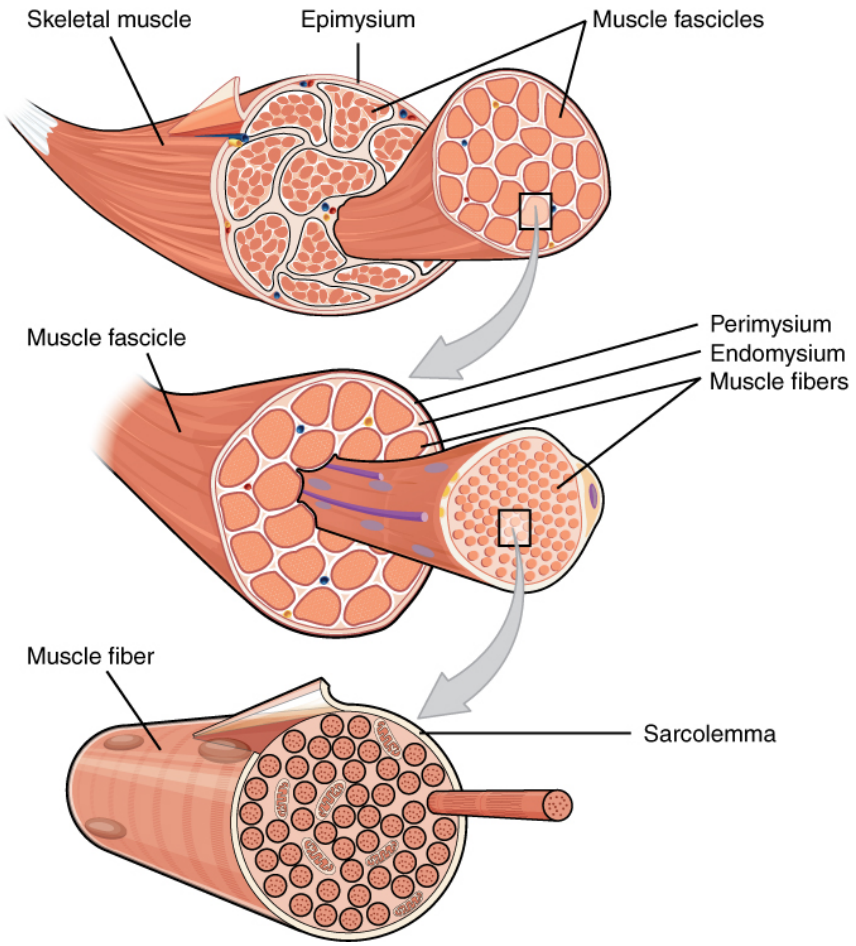

Figure 2.6: The structure of a skeletal muscle [Ope13] (OpenStax College, copyright 2013, licensed under a Creative Commons Attribution License 3.0).

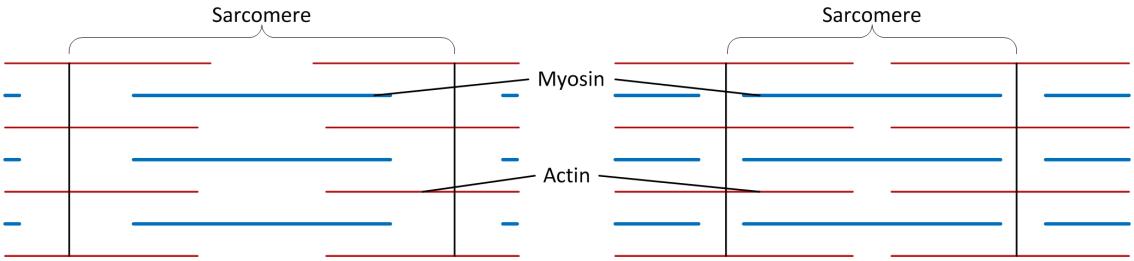

Figure 2.7: An illustration of a contracting sarcomere. **Left:** Fibre at optimal length (optimal cross-bridge formation). **Right:** Fibre with maximum possible contraction (maximal cross-bridge formation).

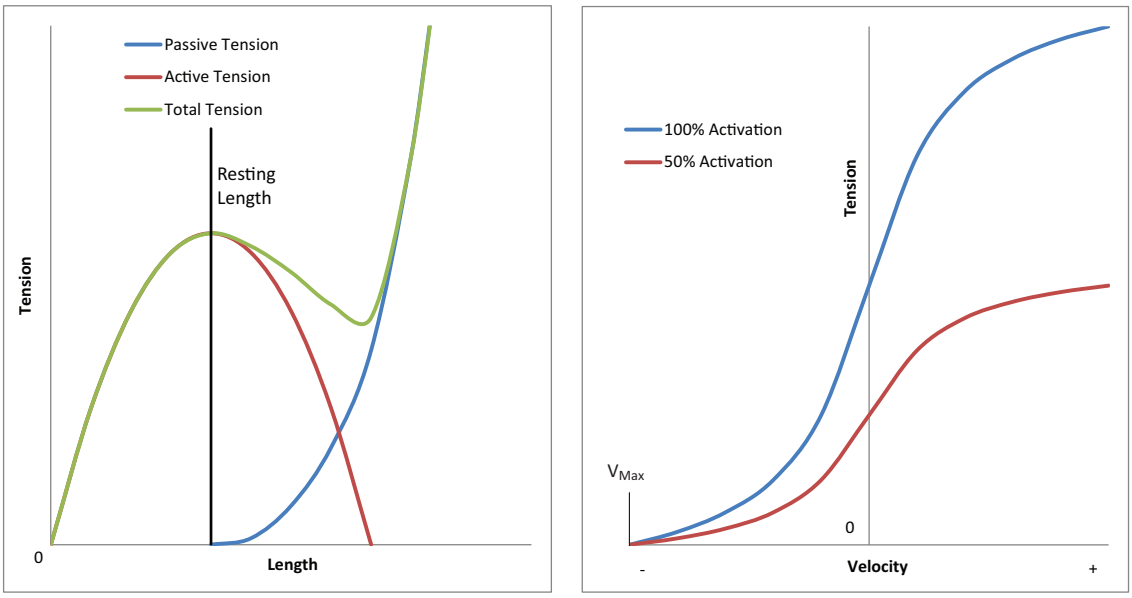

Figure 2.8: Typical muscle fibre tension-length (left) and tension-velocity (right) curves, illustrating the general patterns of forces that are produced by muscles. In this specific example, the muscle fibre equilibrium length (after which transversely isotropic passive tension increases) is equal to the optimal muscle fibre length (the length at which maximum active tension is generated). Functions that describe such curves can be found in literature [RP07].

2.4.2 Muscle Contraction

There are two major modes of muscle contraction. Under isotonic contraction, muscle fibres shorten, whereas under isometric contraction, there is tension on the muscle fibres but the fibre lengths stay the same (e.g. if a load is too heavy for the muscle to lift). During muscle contraction, when the actin and myosin filaments overlap optimally (refer to Figure 2.7), known as the optimal muscle fibre length, maximum active force is generated, and this force decreases with more or less actin and myosin overlap. This active response can be illustrated by the active curve on a muscle fibre tension-length graph (refer to Figure 2.8). Note that the optimal muscle fibre length is not necessarily be equal to the equilibrium fibre length (the length after which passive tension increases) for all muscles. Hill also proposed that the force exerted by a muscle fibre decreases as the speed of shortening increases [Hil38], as illustrated by the tension-velocity graph in Figure

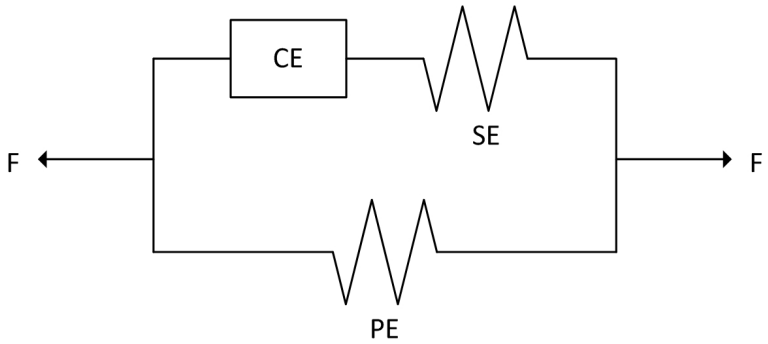

Figure 2.9: Hill's three-element muscle model. CE is the contractile element, SE is the serial element, PE is the parallel element, and F represents the total muscle force.

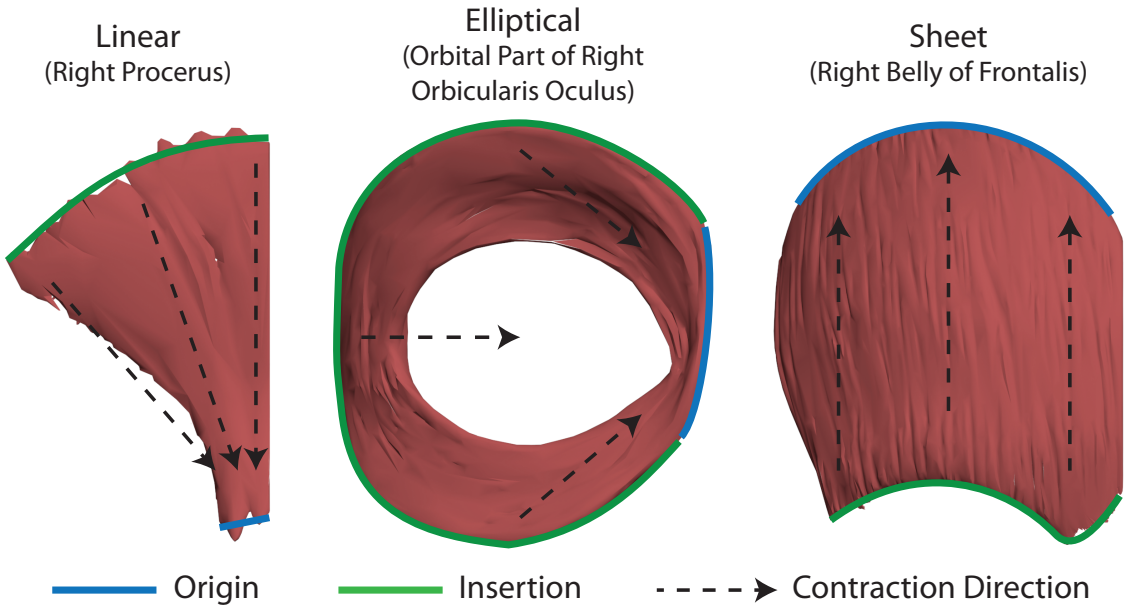

Figure 2.10: An example of a linear, elliptical and sheet facial muscle. The arrows show the directions in which the muscles pull due to the shortening of muscle fibres during muscle contraction. The muscle images were created using BodyParts3D/Anatomography [Dat14] (BodyParts3D, copyright 2008 Database Center for Life Science, licensed under CC Attribution - Share Alike 2.1 Japan).

2.8.

Hill proposed a three-element muscle model, shown by Figure 2.9, that takes into account both the active and passive muscle fibre forces, and the passive forces of the surrounding fascia [Hil38]. The contractile element represents the active fibre forces. The series element represents a mixture of passive forces, such as due to the elasticity from the actin and myosin attachments, and cross-bridge stiffness, whereas the parallel element represents the passive force due to fascia tissue surrounding muscles and muscle fibres. The series and parallel elements are lightly damped spring-like elements that take into account the damping effect of muscles. From this model, due to the rules for spring-like elements in series and parallel, the following equations for the total muscle force, f, and total muscle length, l, can be derived:

$$f = f_{PE} + f_{SE} \quad \text{and} \quad f_{CE} = f_{SE} \tag{2.1}$$

$$l = l_{PE} \quad \text{and} \quad l = l_{CE} + l_{SE} \tag{2.2}$$

2.4.3 Types of Facial Muscles

Contradictory to how most current physically based facial models function, facial muscles interweave with one another and operate as a coordinated team, rather than contracting independently. Individual facial muscles can be grouped into two main groups - muscles of expression and muscles of mastication, with some muscles falling into both groups. There are also three different types of facial muscle that can be grouped according to the orientation of the muscle fibres. As illustrated by Figure 2.10, these muscle types are:

- Linear/Parallel muscles - Bundles of fibres sharing a common origin and pulling in a common direction
- Elliptical/Circular sphincter-type muscles - Fibres that loop around orifices and squeeze towards some virtual centre

- Sheet muscles - A broad, flat sheet of almost parallel fibres without a common origin, behaving like a series of linear muscles spread over an area

As with the skull, most muscles that don't cross the median plane are paired. Most facial muscles are linear muscles. Some of these can also be categorised as convergent muscles, where the muscle origin is wider than the muscle insertion, such as the two temporalis muscles. The three elliptical muscles of the face are the two orbicularis oculi around the eyes and the orbicularis oris around the mouth, each of which attach only to skin. The orbicularis oris is not a simple sphincter muscle, however, and it is composed of four quadrants of fibres running in different directions that form a circular shape around the mouth and interlace [Sta09]. In computer graphics, it is often approximated using a simple standard sphincter muscle model [LTW95, EM01, KHS01], although it has also been represented using four linear muscles [UGÖ98]. A significant sheet muscle that has a large effect on facial wrinkles is the frontalis muscle on the forehead that lifts the eyebrows. This muscle also has no bony attachment. The complex muscle anatomy, such as the shape and size of the risorius muscle involved in smiling, also varies with different people [Sta09].

2.5 Summary

Major structures of the human head include the multiple layers of skin and connective tissue, muscles, and the skull, which are interleaved with complex structures of cartilage, ligaments, tendons, nerves, blood vessels, glands and hair. While the composition of each soft-tissue layer and structure differs, each such layer and structure has complex nonlinear, anisotropic, inhomogeneous, viscoelastic and plasticity properties. Due to its viscoelastic nature, soft tissue exhibits hysteresis, stress relaxation, creep and preconditioning properties, producing extremely complex stress-strain patterns and material behaviour. Such complex behaviour makes accurately measuring and simulating the response of soft tissue complex tasks, and a wide range of material properties and material models (summarised in Table 2.1) have been previously reported and used for soft-tissue simulation.

The outermost soft tissue, which leads to the formation of wrinkles, is the multiple layers of skin and connective tissue. The two primary skin layers are the epidermis and the dermis. The thin epidermis is the outermost skin layer, the thickest and most significant sublayer of which is the tough but flexible stratum corneum (the outer layer). The dermis is a thicker, more flexible fibrous layer, and is mainly composed of interwoven curled up bundles of tough collagen fibres that uncurl when the skin is stretched, and elastin fibres that provide elasticity and resilience. Posterior to the dermis is the hypodermis, loose connective tissue that is highly variable in thickness. On the face, the hypodermis is connected to the SMAS, a 'fibro-fatty' layer that surrounds the superficial facial muscles, and contains fibrous septae that extend through to the dermis. Deeper muscles are surrounded by the deep fascia, which is strong and dense fibrous tissue located posterior to the SMAS. Upon muscle contraction, the SMAS is able to slide over the tougher deep fascia.

Wrinkles (folds or creases in skin) occur due to compression or distortion of the soft-tissue layers, and changes in material properties of these layers. The different material properties of the soft-tissue layers lead to the production of wrinkles under such conditions. Two groups of wrinkles are temporary expression wrinkles that occur due to muscle contraction, and permanent ageing wrinkles that start to form over time as skin properties change, particularly at points affected by a lot of stress. Expression wrinkles therefore typically become more permanent over time.

The skull is another major structure of the head, and is composed of a series of flattened or irregular bones that define the overall shape of the face. Most such bones are immovably jointed together, apart from the mandible (jawbone), which is hinged to the rest of the skull at the TMJs, enabling rotational and translational jaw movement. Finally, muscles between the skin and bone are the contractile fibrous tissues responsible for the movement of body parts. The majority of facial muscles are linear muscles, although there are some sphincter (e.g. the orbicularic oculi) and sheet muscles (e.g. the frontalis). Most such muscles originate in the skull and insert into the dermis or connective tissue. Since muscle fibres have a similar biphasic passive stress-strain response to other soft tissue, and the combination of the passive response from muscle

fibres and the surrounding fascia leads to significant transversely isotropic passive behaviour of muscles. On the other hand, active muscle behaviour has a different stress-strain response, whereby maximum active force is generated at optimal fibre length, and this force decreases at longer or shorter lengths. Hill proposed a three-element muscle model that describes this active and passive behaviour of muscle fibres and surrounding fascia.

Chapter 3

Introduction to Physically Based Soft-Body Animation

The production of physically based animations typically requires three main stages:

1. Creation of a surface mesh, which is an infinitely thin discretised surface defining the boundaries of an object

2. Creation of a physically based simulation model, which is a discretised physics-based representation of an object encapsulating the necessary data to simulate the object using the desired physics-based technique

3. Simulation of the model over time

Typically in computer graphics, a surface mesh is created for an object, which is then used to create a simulation model. The surface mesh is attached or bound to the simulation model, and, as this model is simulated using a physics-based technique, it moves and deforms the surface mesh to produce animation. Chapter 4 reviews existing physically based facial models and the simulation of these to create animation (Stage 3), while Chapter 5 reviews approaches of creating such models (Stages 1 and 2). This chapter provides an introduction to various aspects of physically based soft-body animation, including physics-based simulation techniques, time-integration schemes, element types, conforming and non-conforming simulation models, and boundary conditions. Some existing physics-based simulation software packages are also discussed.

Popular physics-based soft-body simulation techniques for producing efficient, realistic-looking animations for computer graphics applications include physics engines (constraint-based techniques) [MHHR06, CP06, Fra12], like NVIDIA PhysX[1], and the mass-spring (MS) method [KHS01, ZPS04, Fra05]. High-accuracy applications in science and engineering, and surgical applications typically require a more accurate and computationally complex physics-based technique for simulations, such as the finite element (FE) [ZHD06, BJTM08, FM08] or mass tensor (MT) method [SC99, MSNS05, XLZH11]. With advances in computational power, and GPU Computing techniques, such advanced physics-based methods are becoming increasingly popular with computer graphics applications [TCO08, LGK11, MS13]. Physics-based simulations can be performed in a static, quasistatic or dynamic manner, and, to advance a simulation through time, a time-integration scheme is required.

To produce physically based animations, a simulation model is required. This is typically a discretisation of a domain into smaller parts called elements that are interconnected at points called nodes. This is analogous to the polygons and vertices that define a surface mesh; however, the domain of a simulation model can be more complex (e.g. volumetric) with stricter requirements, for example, regarding the types, sizes and shapes of elements, to produce accurate and stable

[1] http://www.geforce.co.uk/hardware/technology/physx

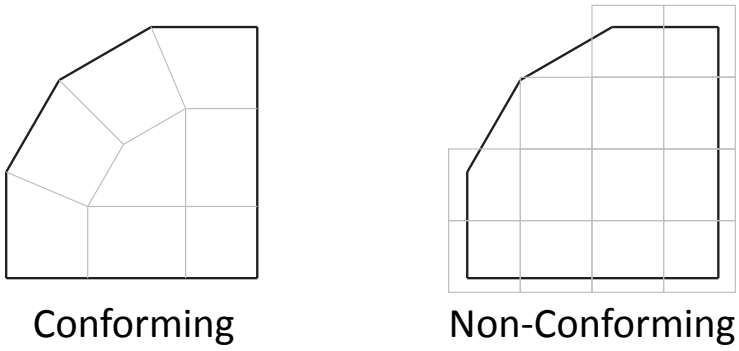

Figure 3.1: A conforming and non-conforming simulation model.

simulations. The element types that can be used for a model depends on the simulation technique to be used.

Domain discretisations can be grouped into two categories: conforming and non-conforming. As shown by Figure 3.1, in conforming simulation models, elements share faces with polygons of the surface mesh, and don't penetrate surface boundaries. The surface mesh is therefore attached to the simulation model, with nodes and vertices sharing the same positions. Such a requirement isn't imposed on non-conforming simulation models, such as a voxel-based models, where elements approximate the surface mesh. With such models, vertices of the surface mesh are normally bound to elements of the simulation models.

Simulation models also require properties to be set for simulation. All models require element material properties to be set, whereas, for example, soft-tissue models may contain additional information, such as fibre directions to be used for muscle contraction. Finally, boundary conditions must also be defined for physics-based simulations, which define the necessary behaviour of the environment surrounding the simulation domain. As it is infeasible and unnecessary to include every detail of a region of interest (e.g. the face) and its close environment in the domain of the simulation technique, it is necessary to consider the physics at the boundary of the domain that affect the simulation. For example, with physically based facial models, the skull is a rigid surface that isn't normally included in the domain of the soft-tissue simulations. Boundary conditions can therefore be set, for example, to fix muscle attachments, and prevent the simulated soft-tissue from penetrating the skull. These boundary conditions represent the physics of the close environment outside of the domain (facial soft-tissue). Section 4.1 discusses techniques for using the skull as a boundary during soft-tissue simulations.

3.1 Physics-Based Simulation Techniques

3.1.1 Mass-Spring Method

The most widely researched physics-based techniques for facial animation are the MS and FE methods. With the MS method, a continuous body is approximated by a system of mass points connected by massless springs (see Figure 3.2) that are simulated using some variant of Hooke's law. Nodal forces, $\mathbf{f}^{(\mathbf{p})}$, can be calculated as follows:

$$\mathbf{f}^{(\mathbf{p})} = \sum_{s=1}^{n} k^{(s)} \frac{(l_0^{(s)} - l^{(s)})}{l^{(s)}} (\mathbf{u}^{(\mathbf{p})} - \mathbf{u}^{(\mathbf{q})}) \tag{3.1}$$

where s is one of n springs connected to node p, p and q are the nodes of spring s, $k^{(s)}$ is the spring stiffness, $l_0^{(s)}$ is the spring rest length, $l^{(s)} = \|\mathbf{u}^{(\mathbf{p})} - \mathbf{u}^{(\mathbf{q})}\|$, and $\mathbf{u}^{(\mathbf{p})}$ are the nodal displacements of node p. The element type for the MS method is always a spring (line), and these can be connected

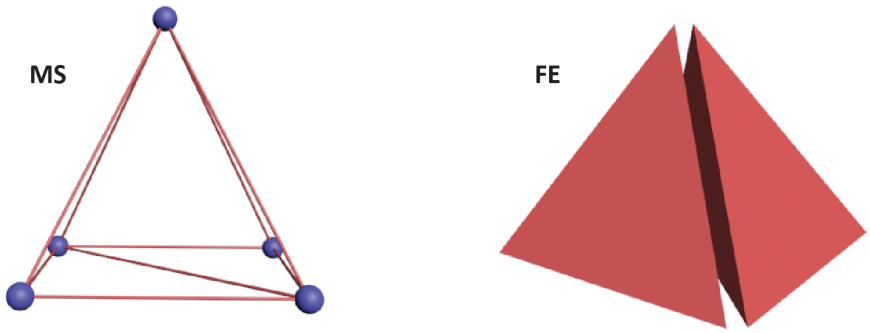

Figure 3.2: Two connected tetrahedra represented as an MS system and an FE system. The FE system elements have been slightly separated to illustrate that these enclose a volume, whereas the MS system contains no representation of a volume.

in any way desired, like with the tetrahedra in Figure 3.2. The simulated MS elements and system therefore have no volume, meaning properties such as volume preservation are often simulated using additional springs or heuristic functions. MS systems are usually quite efficient and suitable for real-time animation; however, they are unable to model the exact properties of soft tissue [KGKG98], and it is difficult to predict and generate the desired behaviour using such models.

3.1.2 Finite Element Method

The FE method involves finding an approximate solution to differential equations that define the dynamics of a continuum. The displacement-based FE method can be used for soft-body dynamics, such as soft-tissue simulation. This involves discretising the continuous body into a finite number of elements, such as linear or quadratic tetrahedra or hexahedra (see Figure 3.2), and the internal forces can be found by integrating the relationship between linear and nonlinear strains and stresses at element integration points over the volume of the element. Element nodal force contributions, $\mathbf{f^{(m)}}$, can be calculated as follows:

$$\mathbf{f^{(m)}} = \int_{V^{(m)}} \mathbf{B^{(m)}}^T \tau \, \mathrm{d}V^{(m)} \tag{3.2}$$

where $V^{(m)}$ is the volume of element m, $\mathbf{B^{(m)}}$ is the element strain-displacement matrix, and τ is the stress vector. The calculation of stresses is done at each integration point in the element, and they are combined, for example, using Gaussian quadrature. The FE method generally produces more accurate animations at the expense of increased computation and complexity. Unlike the MS method, the FE method is based on continuum mechanics, making it more suitable for modelling continuous materials like soft tissue. Figure 3.2 illustrates how two connected tetrahedra can be represented using the MS and FE methods. Some reviews are available that explain in further detail methods like MS and FE for simulating deformable materials in computer graphics [GM97, NMK+06].

There are various different formulations of the FE method, and extensive descriptions of these can be found in literature [Bat96, Bat86]. While many of these share similarities, for example, in terms of accuracy and computational efficiency when compared with the other physics-based methods like MS, only a few of these formulations are particularly suitable for this research project, which will involve computation of large nonlinear deformations. The updated Lagrangian (UL) approach computes updated values with respect to the current configuration. It is commonly used in commercial FE systems, and was originally preferred because of memory limitations [MJLW07]. However, the total Lagrangian (TL) approach computes updated values with respect to the initial configuration, enabling precomputation of various values to speed up solution computation;

therefore, the TL approach is used for this research project.

More recently, meshless FE formulations, including meshless TL FE formulations for simulating soft-tissue deformations [HWJM10], have also been proposed. These perform calculations using a cloud of points that aren't part of an element mesh. The formulations of these methods are largely the same as the formulations for traditional FE methods apart from the definitions of the shape functions. Meshless techniques support very large deformations with increased stability as badly shaped elements are less of a problem, although they are generally more complex and much more computationally expensive than traditional FE formulations.

3.1.3 Mass-Tensor Method

Cotin et al. introduced MT (mass-tensor) models, which have also been used for facial animation systems [SC99]. The MT method is based on the FE method, and therefore based on continuum mechanics elasticity theory. Because of this, results comparable to those from the FE method have been achieved using the linear MT method for modelling small displacements at much faster simulation speeds similar to those that can be achieved with MS systems [MSN+07]. Using the linear MT method, the force at a node, $\mathbf{f^{(p)}}$, can be calculated as follows:

$$\mathbf{f^{(p)}} = \mathbf{K_{pp}u^{(p)}} + \sum_{q=1}^{n} \mathbf{K_{pq}u^{(q)}} \tag{3.3}$$

where $\mathbf{K_{pp}}$ and $\mathbf{K_{pq}}$ are the sums of the stiffness tensors associated with the tetrahedra adjacent to vertex p and edge p, q respectively, $\mathbf{u^{(p)}}$ is the vector of displacements of node p, and q is one of n nodes connected to node p. The MT method is often described as a compromise between the speed of MS simulations and the accuracy of the FE method, although all of the few known implementations of the MT method are used with surgical systems involving small deformations. It also involves heavy precomputation and requires a lot of memory for storage of precomputed values. Furthermore, reported solution computation times using the faster TL FE approach and the nonlinear MT method are relatively similar [MSN+07, XLZH11], although no direct comparisons between the methods have been found using the same hardware. Compared with the MS and FE approaches, the MT method is also by far the least researched physics-based method, and only linear tetrahedral elements have been used with all known MT systems.

3.1.4 Physics Engines

There are also physics engines available that use different algorithms for simulating soft-body deformations. NVIDIA PhysX is an example of a popular physics engine, and various comparisons and reviews have rated it better than several free, open-source alternatives with respect to the overall optimum accuracy, performance and stability [BB07, SR06], although popular commercial physics engines like Havok[2] are not included in these reviews. PhysX soft-body simulation is based on a constrained particle dynamics simulation algorithm, called position-based dynamics, that focuses on speed, robustness and stability [MHHR06]. Unlike traditional physics-based approaches, the algorithm manipulates vertex positions directly using constraints, rather than calculating internal object forces first, requiring fewer integration steps for each timestep. As positions are directly manipulated, this allows collision constraints to be easily formulated. By simply using these constraints, the algorithm makes many simple approximations of physics to make it fast to compute. However, as physics engines are mainly aimed at the games industry, focussing on performance and stability rather than accuracy, they haven't been widely used in research for producing realistic and accurate physics-based soft-body models. Table 3.1 summarises the similarities and differences between the MS method, the FE method, and the PhysX physics engine.

[2] http://www.havok.com/

Criterion	Mass-Spring	Finite Element	PhysX Physics Engine
Accuracy	**(2)** Although formulae from classical physics are used, such as Hooke's law for spring deformations (elasticity), the biomechanics of skin cannot be accurately represented by a structure of point masses connected by springs, as skin is a continuous structure.	**(1)** Based on continuum mechanics, this method can accurately simulate deformations of continuous surfaces and volumes. Provided the correct material properties are simulated, the accuracy is limited only by the mesh resolution and element interpolation functions.	**(3)** Efficient algorithms based on physics are used, rather than implementations of computational physics procedures, resulting in less accurate estimates of deformations.
Computational cost	**(2)** Although techniques like volume preservation and prevention of edge collapse make the method more complex, MS systems are usually capable of producing real-time animation.	**(3)** Even simpler models of isotropic materials are often slower than real time (unless GPU computing is used).	**(1)** PhysX uses a time-integration algorithm designed for performance and stability which uses constraints to directly modify positions without having to calculate forces or velocities first.
Stability	**(3)** High spring stiffness has a large effect on stability, despite the time-integration algorithm used.	**(2)** FE systems tend to be more stable than MS systems, although this also depends on parameters such as material stiffness.	**(1)** As the time-integration algorithm directly modifies positions, it is stable for even large timesteps.
Ease of material definition/parameter specification	**(2)** Stiffness parameters, especially for complex materials like skin, can be difficult to get right to correctly simulate the desired material, although algorithms are available for automatically determining anisotropic MS model parameters according to reference models [BHS03].	**(1)** Parameters for creating models are usually known as they have physical relevance (such as Young's modulus) and can be calculated from laboratory experiments.	**(3)** Like with MS, parameters can be difficult to get right. Parameters like stiffness and volume preservation are based on values between 0 and 1 rather than real units.
Range of possible model materials	**(2)** Although nonlinear anisotropic materials can be modelled, materials must be approximated by distributing the mass over point nodes rather than over a continuum. MS material properties are a subset of those that can be modelled using the FE method.	**(1)** A large range of material properties can be modelled accurately provided the correct material matrix is used.	**(3)** Only homogeneous isotropic soft-bodies can be created, although a workaround is to attach multiple soft bodies together.
Estimated amount of work required	**(2)** MS systems are relatively simple to implement, although various techniques, such as volume preservation and spring incompressibility, are required for more accurate simulations.	**(3)** FE systems can become complex, particularly when including properties such as viscoelasticity, anisotropy and nonlinearity. A lot of optimisation is also required to improve computational efficiency.	**(1)** A soft-body mesh can be simply created using the PhysX framework. The simulation and features like collision detection are handled by the framework. A tetrahedral soft-body mesh can also be automatically created from a triangular surface mesh using PhysX Viewer (included with the PhysX SDK).

Table 3.1: A summary comparison of the MS method, FE method and NVIDIA PhysX soft-body deformations for modelling soft tissue, including a ranking (where 1 is best and 3 is worst) that ranks each method according to the specified criteria.

3.2 Time Integration

Physics-based simulations can be static, quasistatic or dynamic. Static simulations involve solving for the static equilibrium of a body based on given boundary conditions. Inertial forces are not considered, and the equilibrium equation in static analyses can be defined as follows:

$$\mathbf{Ku} = \mathbf{r} \tag{3.4}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ is the vector of displacements for each system degree of freedom, and $\mathbf{r}$ is the vector of external forces for each degree of freedom. Note that, with nonlinear analyses, the stiffness matrix is not constant, and varies as a function of the displacements. The static equilibrium equation can be solved directly provided there are no path-dependent nonlinear conditions or time-dependent phenomena; however, for accuracy and stability reasons, it is often solved quasistatically. This involves using a multi-step time-integration scheme to solve the static equilibrium equations a number of times with gradually increasing loads until the desired loads have been reached. Each timestep is the difference between such load increments, and computation of a timestep advances the system from the current to the next load. A multi-step process is also necessary for producing animations of a body, where multiple body configurations between the initial and final static equilibrium are required, rather than just a single final static equilibrium configuration.

Static and quasistatic analyses are best suited to simulations where inertial forces aren't important, such as those with slow deformations where loads are applied slowly over time. Dynamic simulations, on the other hand, take into account inertial forces, and the equation of motion of a dynamic simulation (the dynamic equilibrium) can be represented by the following second order ordinary differential equation:

$$\mathbf{M\ddot{u}} + \mathbf{C\dot{u}} + \mathbf{Ku} = \mathbf{r} \tag{3.5}$$

where $\mathbf{M}$ is the mass matrix, and $\mathbf{C}$ is the damping matrix. MS systems often include additional terms, for example, for volume preservation of the system. A time-integration scheme can be used to solve the dynamic equilibrium equation over time, where a timestep is the difference between incremental points in time, and computation of a timestep advances the system from the current to the next point in time. To simulate a system, initial displacements $\mathbf{u}$ and velocities $\dot{\mathbf{u}}$ for each node are required, as well as any constraints (e.g. to prevent movement of nodes that are attached to the skull). The simulation can then be advanced forward in time.

Inertial effects have a strong presence in highly elastic materials when forces are applied quickly, producing oscillations before the static equilibrium state is reached. In such cases, a quasistatic solution produces a highly viscous and damped response without oscillations, whereas a dynamic solution with inertial effects simulates the expected oscillations as the material oscillates past its equilibrium state with velocity. Such behaviour doesn't usually occur during soft-tissue deformations without extreme or sudden movements (e.g. as body fat wobbles during jogging), which have a highly viscous and damped response, and quastistaic approaches have been suitably used for facial soft-tissue simulations [SNF05]. Both quasistatic and dynamic approaches (with high damping) are therefore useful for simulating highly viscous soft-tissue behaviour; however, a dynamic analysis with inertial effects must be performed if an explicit time-integration scheme is used.

There are two main types of time-integration schemes: explicit and implicit [NMK+06]. Explicit schemes directly compute values at the next timestep using the known values for the current timestep, whereas implicit schemes use both the values for the current and next timestep and require an iterative solver. As explicit schemes project blindly into the future, they are conditionally stable and require a satisfactorily small timestep, whereas implicit schemes are unconditionally stable but are more computationally expensive. Explicit methods that use many relatively inexpensive timesteps are usually sufficient and are popular for dynamic simulations.

The simplest and least computationally complex scheme is the forward Euler method; however, this method usually requires a very small timestep to converge to a solution. A simple improvement with no additional cost is the forward-backward Euler method, which uses forward Euler to

calculate velocities, but then uses the updated velocities to calculate the displacements. A popular compromise between stability and complexity is the explicit Runge-Kutta method [PTVF07]. Alternatively, the central difference method can be used, rather than a forward difference integration scheme like the forward Euler or Runge-Kutta methods. Central difference time-integration schemes use values from the current and previous timesteps to calculate the next timestep.

Shinya et al. described the cost and stability of MS systems using various time-integration methods, and proposed the linearised explicit Euler method, which is a more stable version of the explicit Euler method for linear systems [Shi05]. PhysX uses the semi-implicit Verlet time-integration method, and the stability of the solver depends on the shapes of the constraint functions, rather than the timestep. Verlet integration schemes are popular with systems that require fast computational speed. The explicit Verlet integration scheme has a similar efficiency to the explicit Euler for computing a timestep with increased stability [THMG04], allowing longer timesteps to be used. More information on numerical schemes for solving ordinary differential equations can be found in literature [Asc08].

3.3 Element Types

With MS models, all elements are springs connected by two mass points (nodes); therefore, as discussed in Section 3.1.1, while MS elements can be connected to form more complex shapes, such as triangles and tetrahedra, the surface area or volume of these shapes isn't considered by the MS system, meaning additional techniques are required to prevent the collapse of such shapes. Contrary to this, a range of 1D, 2D and 3D elements are available to create FE and MT models. As we are modelling the volume of soft tissue between the outer skin surface and the skull, we focus on using 3D elements that can capture detail of complex soft-tissue volumes, such as skin layers and volumetric muscles. Physics engines typically also use 2D and 3D elements; for example, soft bodies in PhysX can be modelled using triangles or tetrahedra.

For FE simulations, the type of elements used for simulation models can have a significant impact particularly on the accuracy and efficiency of the simulations. Elements have a number of integration points at which element strains and stresses are evaluated, and interpolation functions to interpolate values across elements, which can be Lagrange or Hermite interpolation functions of different orders. While Lagrange elements are suitable with second order differential equations, whereby only C^0 continuity is required, as in the displacement-based FE method, Hermite elements maintain C^1 continuity, and therefore better satisfy the physical conditions for soft-tissue deformations, such as large bending and torsion. Fewer, although more computationally complex, Hermite elements are required to model complex geometries, although, as each element consists of a single material throughout the element, such large elements may not be able to accurately capture the heterogeneity of some materials, such as skin.

Tetrahedral and hexahedral elements are popular with 3D FE models [TCO08]. The advantages of tetrahedra include their simplicity compared to hexahedra when meshing complicated geometries, and they have simpler and smaller shape functions. Some complex geometries are almost impossible to mesh using a fully conforming hexahedral approach, and a volume discretisation with only well-shaped nonlinear tetrahedra can lead to very accurate simulations [BTPB07, WNR09]. On the other hand, tetrahedra are susceptible to volume locking (when a soft-body mesh 'resists' deformation and appears overly rigid) when used to model almost incompressible materials like soft tissue, as tetrahedra have limited flexibility under the constraint that their volume must be preserved [ITF06]. On the other hand, some low-cost methods to reduce this problem have been proposed, such as by using more compressible material properties with additional volume constraints [PDA03], and by modifying the element stress calculations to consider average element pressure [BB98, JWM09], although, even with the improvements, simulations using tetrahedral meshes can still appear overly stiff, particularly in bending problems [BMH01, WNR09]. Volume locking is also less of a problem with higher order tetrahedral elements, although these have increased computational complexity [MTPS07].

Hexahedra generally allow the creation of meshes with fewer degrees of freedom, and a lower

timestep is often required for tetrahedral elements [BTPB07]. Hexahedra are also generally more robust and accurate than tetrahedral elements [BPMC95, WNR09], particularly with incompressible materials like soft tissue [BTPB07, TCC+09]. While hexahedra are more computationally complex than tetrahedra, the lower number of elements required usually still outweighs the efficiency of tetrahedra. Higher-order hexahedra are more advanced, and are usually very robust and accurate, although extremely computationally expensive [BPMC95, WNR09]. Other less common elements can also be used with physics-based models, such as pentahedral elements. However, there is no single element type that is most suited to every problem, and there is a lot of discussion over which element types are most suited to particular problems [BPMC95, BTPB07, WNR09].

3.3.1 Hourglass Control

Despite the advantages of hexahedra, like with tetrahedral elements, they can also produce an overly stiff response, particularly with incompressible materials like soft tissue [Bat96]. Such limitations can be overcome by using reduced-integration hexahedral elements, which also have performance advantages due to the reduced number of integration points [JWM08] (reduced-integration linear hexahedra have only 1 integration point instead of 8). However, the more approximate evaluation of deformations and stresses can lead to element deformations that produce zero stress (stressless deformations). Such physically impossible deformation modes are referred to as zero-energy modes, or hourglass modes because of the deformation patterns they produce in a quadrilateral or hexahedral FE mesh. Such modes can dominate and destroy a simulation.

A hourglass control technique is required to reduce the hourglass effects. A popular hourglass control technique that is implemented in various FE software packages, including ABAQUS[3] and LS-DYNA[4], is stiffness-based perturbation hourglass control developed by Flanagan and Belytschko [FB81], which has also been used with the TLED formulation of the FE method [JWM08]. This adds artificial stiffness to elements to constrain the different hourglass modes based on element stiffness. An alternate approach is viscous-based perturbation hourglass control [FB81], which adds artificial damping, although this method is usually less effective, and tends to simply delay rather than prevent the occurrence of hourglass modes. Liu et al. proposed the physical stabilisation method using stabilisation matrices [LOU85], although, this method is expensive memory-wise, and can lead to volume locking. Belytschko and Bindeman proposed the assumed-strain stabilisation method [BB93], which has also been combined with the physical stabilisation method [Pus00]. While these are more accurate and computationally efficient for bending-dominated problems, such methods can cause instabilities with the arbitrary deformations in soft-tissue deformations.

3.4 Physics-Based Simulation Software Packages

A lot of research on using the FE method for facial animation has involved implementing a version of the method tailored to the task. There are, however, numerous FE solvers available that are suitable for simulating highly deformable materials like skin, some of which are free to use. Most such solvers perform FE analysis in several stages, rather than in real time. A preprocessing stage requires an FE mesh to be created or imported, and for parameters such as any boundary conditions, material properties (either a built-in or user-programmed material model) and forces to be set for a single or several stages of the FE analysis. The next stage then performs the whole FE analysis and outputs data that can later be read and analysed in the postprocessing stage. Therefore, these FE packages aren't suitable for use in an interactive scenario like games where the output is desirable at frequent intervals during the FE computation. On the other hand, they might be more suitable for scripted scenarios, for example, for creation of film scenes, and they can be useful for comparison and evaluation of FE solver implementations.

One popular FE solver for soft-tissue simulation is ABAQUS. This has better features than

[3] http://www.3ds.com/products-services/simulia/portfolio/abaqus/overview/
[4] http://www.lstc.com/products/ls-dyna

other commercial packages like ANSYS[5] for developing custom materials and elements, making it popular within academia where complex material models are often required. ABAQUS also has more capabilities with regard to nonlinear problems, although it seems that ANSYS is preferred in industry due to its intuitiveness and ease of use, and because many problems in engineering can also be approximated by the simpler material models already available with the package[6]. As mentioned in Section 3.2, an important factor to also consider is whether implicit or explicit simulation should be performed. As facial animation involves large deformations with self-contact, explicit simulation is preferable for this research project, whereas implicit simulations are preferable for low speed simulations where more accurate stress solutions are critical. ANSYS is an implicit solver, although it can be used in conjunction with LS-DYNA, an explicit solver, whereas ABAQUS contains both types of solvers. Both of these solutions allow switching between solver type within a simulation.

Some freely available packages have been developed within academia for simulation of complex materials like skin, including CMISS[7]. FEBio[8] was also developed specifically for FE analysis of nonlinear large-deformation problems in solid biomechanics. These packages, although lacking features of the commercial packages, are less complex, and can also be useful for simply evaluating the accuracy of FE solver implementations.

3.5 Summary

The production of physically based animations requires three main stages: creation of a surface mesh (defining the boundaries of an object), creation of a physically based simulation model (encapsulating the necessary data to simulate the object) to which the surface mesh is attached or bound, and simulation of the model over time. Popular physics-based soft-body simulation techniques for producing efficient, realistic-looking animations for computer graphics applications include physics engines [MHHR06, Fra12] like PhysX, which approximates physics using a constraint-based technique, and the MS method [KHS01, Fra05], which approximates a body as a volumeless system of mass points connected by massless springs. For high-accuracy applications, including surgical applications, the more computationally expensive FE [ZHD06, BJTM08] and MT methods [MSNS05, XLZH11] are more frequently used. Derived from continuum mechanics theory, these methods approximate a body as a system of connected elements, with 2D and 3D elements representing a surface and volume respectively.

Physically based simulations can be static, quasistatic or dynamic. Static simulations involve solving the static equilibrium equation for the final static equilibrium configuration of a body directly, while quasistatic simulations involve solving for this by applying gradually increasing loads using a time-integration scheme. Static and quasistatic simulations therefore neglect inertial forces, unlike dynamic simulations, which involve using a time-integration scheme to solve the dynamic equilibrium equation over time. Time-integration schemes can be explicit, requiring small but inexpensive integration timesteps to remain stable, or implicit, being unconditionally stable but with computationally expensive timesteps.

A simulation model is required to produce physically based animations, which is a discretisation of a domain into interconnected elements. A simulation domain discretisation can be conforming, whereby elements conform to the surface mesh boundaries without penetrating them, or non-conforming (e.g. voxel-based models), whereby elements approximate the surface mesh. With MS models, elements are simply springs connected by two mass points. With FE models, a range of 1D, 2D and 3D elements can be used with Lagrange or Hermite interpolation functions of different orders.

Tetrahedra and hexahedra are common elements used with volumetric FE models. Conforming

[5] http://www.ansys.com/

[6] Some user comments comparing ABAQUS and ANSYS can be found on various posts on the iMechanica website, such as http://imechanica.org/node/6711.

[7] http://www.cmiss.org/

[8] http://mrl.sci.utah.edu/software

tetrahedral models are relatively simple to create, although linear tetrahedra are susceptible to high amounts of volume locking, particularly when modelling incompressible material like soft tissue [BMH01, ITF06]. On the other hand, while conforming hexahedral models are more complex to create, compared to linear tetrahedra, linear hexahedra are often more accurate, robust and efficient [BPMC95, WNR09]. Reduced-integration hexahedra can be used to improve the computational performance of hexahedral elements, and reduce volume locking with incompressible materials, although hourglass techniques are required to counter zero-energy modes [FB81, Pus00].

Simulation models also require properties to be set for simulation, including element material properties, and fibre directions for muscles in soft-tissue models. Boundary conditions must also be set, which define the behaviour of the environment at the boundary of the simulation domain, such as the rigid skull surface on the boundary of a facial models. Various physics-based simulation software packages are available, including commercial FE software such as ABAQUS and LS-DYNA, and FE solvers developed within academia specifically for soft-tissue simulations, such as CMISS and FEBio. Such solvers typically compute an entire FE analysis in a computation stage, the results of which can be examined in a postprocessing stage, and therefore aren't suitable for interactive scenarios.

Chapter 4

Physically Based Facial and Soft-Tissue Animation Approaches

A physically based facial model typically contains a muscle and a skin model. Low level inputs for animating the face are usually muscle contraction values for individual muscles that are used to deform the muscles and the skin they are attached to. Some previously developed systems have also provided higher level controls, for example, to specify facial expressions, which automatically adjust the lower level muscle controls [TW90]. Although the skin model often produces some larger wrinkles as it deforms, a wrinkle model may also be used to produce additional wrinkling effects that aren't produced directly by the skin model. It is usually the case that the skin model is based on physics to some extent, for example, using an MS or FE model, whereas muscles and wrinkles are often simpler physical or geometric models. A skull model is also an important part of physically based facial models. The skull greatly influences the shape of the face, and is therefore an important boundary of physically based facial models, for which geometric skull models can be used to set boundary conditions during simulations. Figure 4.1 shows the different levels of detail to which physics-based skull, muscle, skin and wrinkle models can be implemented, along with examples of currently implemented models. As of yet, there are no known models that represent the whole of a structure, such as the face, including wrinkles, using physics-based techniques.

The following sections review current approaches for modelling and simulating the different structures of physically based facial models. Starting with the innermost structure, skull models are firstly reviewed, followed by muscle models, muscle contraction techniques and methods of controlling muscle contractions to create facial expressions. Full facial animation systems that incorporate such models with physically based models of facial skin and connective tissue are then explored. This includes efficient, low-accuracy systems often used for computer graphics applications, and more complex, high-accuracy systems often used in other areas of science and engineering, for example, to study the behaviour of soft tissue, and for surgical applications. Geometric and texture mapping wrinkling techniques developed for computer graphics applications are also reviewed, some of which have been used in conjunction with physically based facial animation systems to produce more detailed animations.

Detailed soft-tissue simulations are then explored, which typically focus on simulating smaller areas of soft tissue with higher accuracy, such as skin blocks for studying soft-tissue behaviour, like wrinkling, or organs for surgical applications. Some approaches for validating such soft-tissue simulations are also briefly explored. Finally, some physically based approaches for animating generic soft-bodies in computer graphics are reviewed, followed by applications of facial animation in films and games. While some of the techniques discussed in this chapter have not been used with facial animations, such as some approaches to model and simulate generic skeletal muscle, or generic blocks of skin and connective tissue, such techniques can be used for modelling and simulating the relevant facial structures.

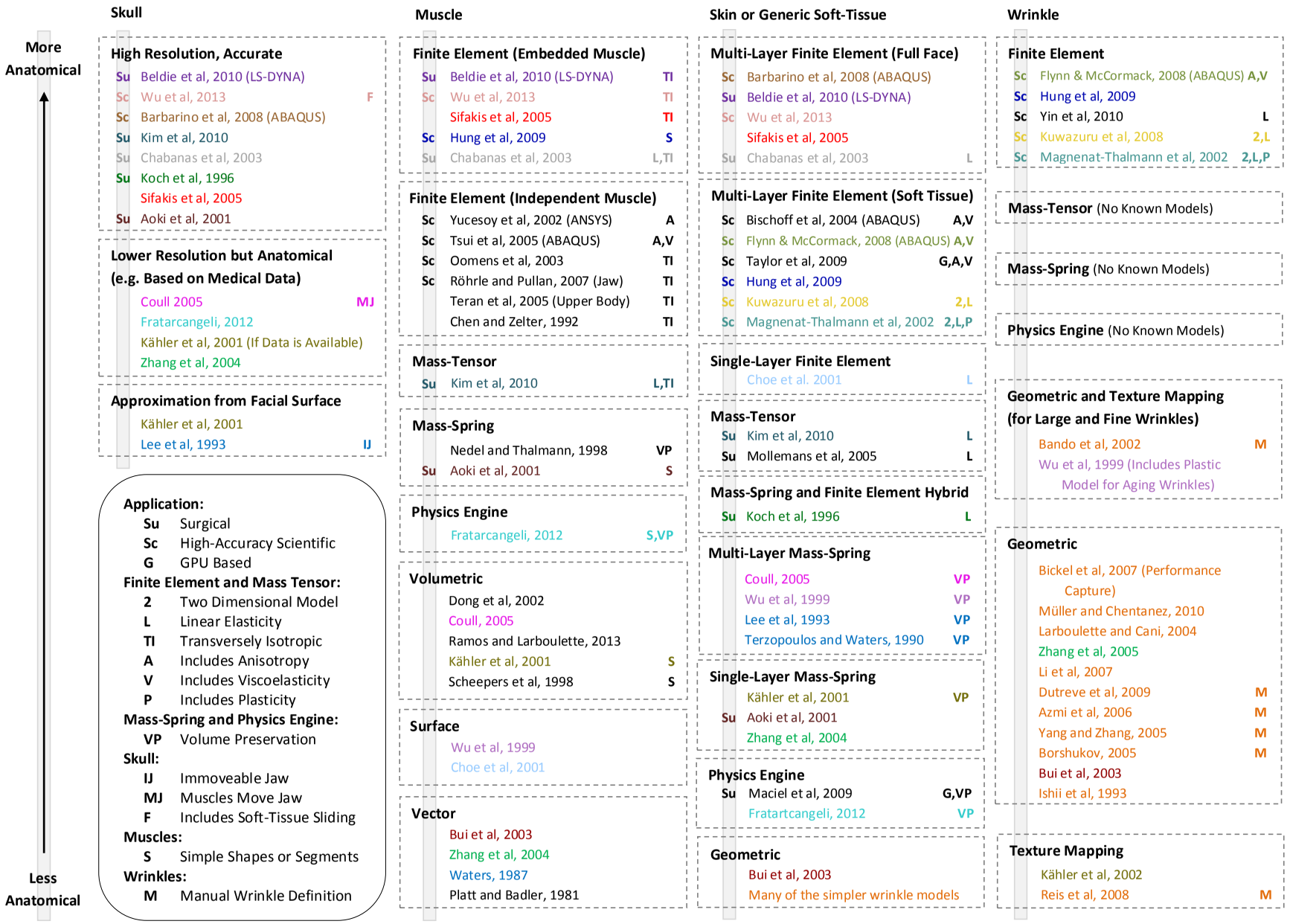

Figure 4.1: The different levels of detail to which aspects of physically based soft-tissue models can be implemented. References to some examples of physics-based soft-tissue animation systems are included and colour coded to show the techniques used to create the aspects (skull, muscles, skin and wrinkles) included in their models. For example, the system by Lee et al, 1993 [LTW93], used an approximation of the facial surface for a skull, Waters' vector muscle model [Wat87], and a multi-layer MS soft-tissue model. The examples within the subcategories are ordered approximately by their level of anatomical accuracy, although the level of difference within each subcategory varies.

4.1 Skull Animation

A simple low-resolution skull model can be created as an offset from a facial surface [LTW93, WD04]; however, the benefits of a more realistic high-resolution skull model include increased accuracy when considering skull penetration, and more realistic muscle origin attachments. Such accurate skull models are also necessary for many surgical systems which involve some form of bone displacement [KGC+96, CLP03, BWL+10]. Skull models created with simpler animation systems sometimes have a non-rotatable jaw [LTW93, WD04]; however, a rotatable mandible can greatly improve realism of actions involving jaw movement [ZPS04]. While many systems simplify the rotating of the mandible by using a simple geometric function, which in turn moves the soft tissue, in reality, muscles contract to move the jaw, which is simulated in Coull's animation system [Cou05]. However, the full complexity of the temporomandibular joint (TMJ) in Coull's system wasn't modelled, and a simple rotational model of the TMJ was used. Complex FE models have, on the other hand, been proposed that examine the TMJ and mandible movement [PSB+07].

The skull also acts as a boundary for the deformation of soft tissue during simulation (as the soft tissue cannot penetrate the skull), and can trigger the movement of soft tissue, for example, when the mandible rotates, or during surgical predictions involving bone displacement. Boundary conditions can therefore be set to restrict the movement of nodes on the skull boundary, and displace such nodes when the skull moves. A simple method of doing this is to simply fix the displacements of all nodes on the skull boundary, ensuring they remain at the same position on this boundary throughout simulations [SNF05, BJTM08]. However, such an approach doesn't consider the sliding of superficial soft-tissue layers over the deep layers and skull. More advanced models of muscle origins and ligaments enable nodes to move at such locations, but restrict the movement of these nodes [WHHM13]; for example, the penalty method can be used to prevent such nodes from penetrating the skull surface [HWHM11]. With the technique proposed by Wang et al., displacements of skull boundary nodes are imposed to move such nodes back to the closest position on the skull boundary, for example, after movement of the skull [WYX09]. The performance of such contact techniques is often improved by using collision detection techniques.

4.2 Muscle Animation

Modelling and simulating muscles is an extensive research topic, and survey papers are available that provide a comprehensive review of the area [LGK+12]. When developing a muscle model, two factors need to be considered:

- How to represent the muscle, which can be either:
  - A geometric representation
  - A physically based system
- How the muscle will contract, which includes computation of:
  - The magnitudes of contracting muscle displacements or forces
  - The directions of muscle contractions

These factors are discussed in the following sections. Once the musculature has been created, it is then necessary to activate the correct muscles to produce the desired action or expression, which can be assisted through the use of expression coding systems. This is discussed in Section 4.3.

4.2.1 Geometric Muscles

The first known work on physics-based facial animation integrated a simple muscle model into a single-layer tension net [PB81], which is a simple MS system where muscles are represented as forces applied to regions of nodes. Waters later improved upon this by developing new 2D geometric muscle models for linear and sphincter muscles [Wat87]. A linear muscle is represented

by a single vector that has an area of influence (AOI) defined by an angle and length, and a sphincter muscle is represented by a single centre point with an elliptical AOI. The models were originally used to impose fixed displacements on nodes within the AOI [TW90], although forces can be imposed on the nodes instead to produce more physically based contractions [LTW95]. This vector approach was later extended to include sheet muscles [ZPS04]. Other variations on this model include representing linear muscles as quadrilaterals where the origin and insertion are line segments [CLK01], and using a 3D sphincter muscle to allow effects such as 'pursing up' of the lips without considering collision detection [EM01, CLK01]. However, combining the influence of multiple muscles on an individual node can be difficult using these 2D geometric muscle models.

Although simple, vector muscle models are limited as the muscles have no volume within the soft-tissue model. As a real muscle deforms and its shape changes to preserve volume (e.g. when a muscle bulges due to contraction), this affects the shape of surrounding soft tissue. Muscles can be modelled with volume-enclosing surface meshes, and geometric functions used to deform muscles, for example, to preserve volume. Such muscle models have been used for animations that aren't physically based to produce more realistic skin deformation in character and facial animations [Wan10, RL13]. In such cases, an animator may, for example, move bones or muscle ends, which leads to deformation of muscles. Geometric constraints between the muscles and skin then update the skin surface mesh to produce the effects of muscle bulges on the skin. Geometric muscle models have also been used with contraction methods in physically based animations [KHS01, Cou05].

Scheepers et al. developed volumetric muscle models to model arm muscles [SPCM97]. Ellipsoids are used to model muscle bellies as these can simply be scaled along three axes to simulate bulging while preserving volume and height-to-width ratio. More complex muscles can also be modelled using multiple muscle bellies. Kähler et al. developed muscle models that also use ellipsoid shapes but in a piecewise fashion to represent muscle fibres [KHS01, HKA+01, Käh03]. Similar to the non-volumetric 3D sphincter muscle models, effects such as 'tucking in' the lips can be simulated by also allowing sphincter muscles to move along a central axis, rather than being defined by a centre point. Unlike with the previous models, the intertwining of muscles is handled using springs to connect muscles together.

Coull also used a similar muscle model with the main difference being that arbitrarily shaped volume preserving facial muscles can be defined [Cou05]. Linear muscles can also contain numerous origins and insertions, which is useful for modelling sheet muscles. Ramos and Larboulette created anatomically shaped muscles by sweeping a cylinder along a curve [RL13]. While relatively simple, this method is best suited to creating fusiform and parallel muscles, although muscle bellies can also be chained together to create more complex muscles. As well as volume preservation, muscle tension is also considered, which causes changes to the muscle cross section without altering its volume. Isometric contractions can therefore be modelled, whereby the length of a muscle isn't altered, but an increase in tension changes its shape.

More advanced geometric muscle models are available in commercial software; for example, Maya Muscle [Aut11] has been used with facial muscles for speech animation [Wan10]. Unlike the previous approaches, collision detection is available to handle the interaction of muscles with other muscles and structures, although such approaches are often difficult to set up, and computationally expensive. Another muscle modelling approach also creates complex, arbitrarily shaped geometric models, but these are based on anatomy and automatically created from anatomical data [DCKY02]. Muscles are grouped into categories with different volume preserving deformation models, and collision detection is also handled. Although complex, this approach is likely to be real time on modern computers.

4.2.2 Physically Based Muscles

Physics-based muscle models have also been developed. Fratarcangeli modelled muscles using hexahedral blocks that conform to the underlying skull surface, and these were deformed using the position-based dynamics technique (also implemented in PhysX) [Fra12]. Nonlinear geometric constraints were used to handle volume preservation, to bind muscles to other soft-tissue structures, and for collision detection. For contraction, samples along action lines of linear and sphincter

muscles are displaced using geometric functions, and constraints between the samples and muscle nodes, along with the other geometric constraints, lead to the displacement of these nodes. Aubel and Thalmann used piecewise vectors to form arbitrary muscle action lines, and these were deformed using a 1D MS system [AT01]. Another simple MS muscle approach has been developed to model muscle volumes using MS systems [NT98]. Arbitrarily shaped MS muscles can be created, and the concept of angular springs was introduced to preserve muscle volume. Additional behaviours like collision detection are handled using constraints. While real time, due to their lack of physical accuracy, and the number of constraints involved, such physics-based approaches usually contain many parameters that can be difficult to correctly set.

The FE method has been used for accurately simulating muscle contractions but with increased computational cost. The more advanced approaches often include structures such as tendons [OMvO+03, TTL+04]. Physically based facial systems that use the FE method for the simulation of skin usually also use FE models for muscles [SNF05, WHHM13]. Isotropic hyperelastic material models have been used for muscles in FE soft-tissue simulations involving small [BJTM08] and large displacements [HMSH09]. Surgical applications often also use isotropic constitutive models, but only consider linear elasticity which is suitable for small displacements [CLP03, BWL+10].

As well as the passive stresses produced by deformation of the muscle and surrounding soft tissue, which are normally modelled using an isotropic material model, an additional passive stress function can also be used with physically based muscles to model the additional stresses along the fibre directions that are produced when muscles are stretched [RP07, WHHM13]. Such functions approximate the muscle passive tension-length curve, and produce the transversely isotropic properties of muscles, which are particularly useful when muscles may undergo large deformations and stretches [SNF05].

Chen and Zelter modelled transversely isotropic muscles using 20-node hexahedral elements [CZ92, Che92]. When these elements deform, they also deform an underlying higher-resolution mesh representing the actual muscle shape using free-form deformation. However, their approach used linear homogeneous materials for simplicity. Teran et al. simulated transversely isotropic hyperelastic muscles of the arms and upper body using an embedded approach, whereby high-resolution volumetric muscles were embedded in a low-resolution tetrahedral mesh designed for efficient simulation [TSB+05]. The surrounding fascia was also modelled, although, as the work focussed on muscle simulation, this wasn't modelled and simulated using the physically based technique, but was instead represented using constraints.

Oomens et al. developed a hexahedral FE model of skeletal muscle using nonlinear transversely isotropic materials, and, although the results were decent qualitatively, quantitatively, the strains produced were too high [OMvO+03]. Other FE muscle models have been implemented using commercial FE packages [JMB00, TZT09]. Röhrle and Pullan created a detailed FE model of the muscles of the jaw to simulate actions such as opening and closing the jaw, and biting [RP07]. The complex cubic Hermite hexahedral elements that were used have the advantage of producing a smooth implicit fibre field when oriented correctly. Motion capture data was used as boundary conditions. The results showed the importance of modelling the muscles of mastication as complex 3D shapes, rather than simple actions lines, to achieve desirable animations.

More anatomically accurate muscle models have been proposed that, for example, simulate viscoelasticity, and use detailed models of connective structures such as tendons [TTL+04, TTC+05]. Much finer details have also been modelled, such as cellular physiology, and groupings of muscle fibres into motor units [RDP12]. Rather than modelling active and passive properties of muscles using a single FE model, Yucesoy et al. simulated skeletal muscle over two domains: the intracellular domain (cells of muscle fibres), and anisotropic extracellular matrix domain (tissue surrounding muscle fibres and fibre bundles, such as the endomysium) [YKHG02]. These domains are linked elastically, enabling interaction between the muscle fibres and extracellular matrix to be simulated, which isn't possible using a single FE model; however, this complex approach requires simulation of and interaction between two FE models to simulate a single muscle.

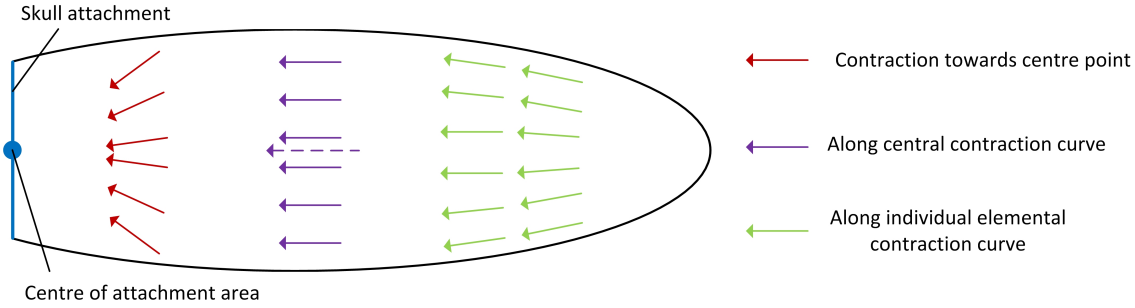

Figure 4.2: Possible directions of contraction that could be computed by different muscle contraction models.

4.2.3 Muscle Contraction

Various methods to contract muscles have been proposed. Geometric muscle models typically scale muscles according to an activation parameter. With Waters' muscle model, the displacement or force applied to a node within an AOI is also dependent on its location within the AOI and its distance from the muscle origin [Wat87]. Volumetric geometric muscle models often use a function to scale or displace a muscle insertion or control points, and geometric constraints lead to the deformation of the muscle volume [Cou05, Fra12, RL13]. The control points are typically scaled along paths representing action lines or curves.

More advanced contraction methods are required for use with physics-based muscle models, the majority of which are based on a Hill-type model, or a variant of such a model [CZ92, RP07]. To generate active stresses, a function that computes a stress or force magnitude based on a fibre length is typically scaled according to an activation parameter. Lee and Terzopoulos developed a biomechanical model of human neck muscles that focussed purely on muscle contractions to move the skull and neck bones, and no soft tissue or volumetric muscles were modelled [LT06]. A Hill-type muscle model was used, and the generated active forces took into account both muscle length and velocity, which is often neglected; however, a linearised function was used to generate the forces. As the soft-tissue wasn't modelled, a function to generate hyperelastic passive forces was also included. Such forces would normally be produced by the physics of the soft-tissue model. Hunter's steady state model also uses a linear function to generate active stresses [Hun95]. This uses a more advanced biologically inspired function, considering the relationship between fibre stress and calcium concentration, although it doesn't consider the time-dependent activation of myofibrils. The model has been used for static FE soft-tissue simulations [HMSH09], for which time dependency isn't necessary.

Compared with a linear active stress function, a nonlinear function can better approximate the muscle active tension-length curve, and produce more anatomically accurate stresses [RP07]. More advanced nonlinear Hill-type models that consider muscle contraction velocity have also been developed [OMvO+03, TZT09]. The post-operative surgical prediction technique proposed by Beldie et al. used such a model [BWL+10]. Extensions to Hill's model have also been proposed; for example, a multi-fibre Hill-type model with multiple contractile and serial elements has been used, which produces non-uniform stress distributions and considers the heterogeneous material structure of muscle fibres [SK07]. Rather than a Hill-type model, functions based on Feldman's lambda model have also been used to generate active stresses for facial muscles [NPP13]. With this model, force (or stress) is a consequence of the muscle threshold length, rather than a control variable, and active muscle stresses change as a nonlinear rather than linear function of activation level, enabling complex unloading behaviour to be simulated that would require complex activation functions with a Hill-type model.

4.2.4 Muscle Fibre Directions

Various approaches have been proposed to compute directions of muscle contraction. Figure 4.2 illustrates some possible contraction directions of a muscle. The simplest methods contract nodes towards a single origin point [Wat87]. Alternatively, muscle fibres can contract in a direction parallel to a central action line or curve [TZT09]. However, more anatomically accurate approaches consider the contraction directions of individual fibres [GZDH04].

By modelling muscles using B-spline patches or volumes, a continuous muscle fibre field can be produced using the partial derivative of the B-spline with respect to the parametric coordinate that is aligned with the length of the muscle [WKMMT99, TSB+05]. B-spline volumes have also been used to compute muscle fibre directions for facial animations [SNF05]. However, it can be difficult to create complex shapes using B-spline patches and volumes without a large number of control points, which are tricky to manipulate and increase the amount of computation required to compute the fibre directions. Also, while B-spline surfaces can be created using various 3D modelling tools, no tools that support B-spline volumes were found.

Due to these difficulties, and the usefulness of a parametric volume for generating a continuous fibre field, techniques have been proposed to simplify the construction of B-spline volumes. For example, they can be constructed automatically from closed B-spline surfaces by shrinking them to their central curve [MLC01]. Ng-Thow-Hing et al. also proposed several approaches for the construction of B-spline volumes [NTHF02]. One approach created such volumes from segmented MRI data. B-spline contour curves are extracted from the MRI data, and a central curve is used to create a continuous volume sampling function (CVSF), which is then sampled to compute the control points of the B-spline volume. An interactive editor was also implemented for automatic creation of a volume from user-defined B-spline contour curves representing the muscle origin and insertion, and a central action curve. This could potentially be extended by enabling the user to define more contours, providing further control over the shape of the muscle. As each of the described approaches use linear basis functions to interpolate between the volume boundaries and central curves, they work best with convex shapes. It would also be necessary to subdivide branching surfaces or volumes, although such muscle structures don't occur on the forehead (the focus of this research project).

Rather than requiring creation of B-spline surfaces or volumes, Aina proposed a technique to construct muscle fibres automatically by simply using painted muscle attachments on a texture map for a reference skull [AZ10, Ain11]. Using a facial and fitted skull surface, a new surface representing the SMAS is firstly semi-automatically constructed as a variational implicit or radial basis function (RBF) surface. The texture map is then used to automatically generate convex hulls of the muscle origins and insertions, both on the SMAS surface, connected by their mutual tangents. A number of muscle fibres are then created by constructing geodesics between subdivisions along the muscle origins and insertions. A continuous fibre field could be produced by interpolating between such fibres. While muscle fibres are very easy to define with this approach, the user has little control over the fibre directions between the attachments. Also, like with B-spline patches, generating fibres along a surface doesn't consider the fibre directions throughout the entire muscle volume.

Alternatively, if volumetric elements are used, a fibre field can be produced by creating a mutually orthogonal curvilinear coordinate system from the local coordinate axes of the elements, with one of the axes aligned with the fibre direction [RP07, MHSH10]. The partial derivative with respect to this axis can then be used as a fibre field. However, this approach requires the same local element axis to be aligned with the muscle fibre direction for all muscle elements. This imposes restrictions on the volumetric meshing process and the types of elements that can be used, which may be difficult to adhere to with more complex muscle shapes. Also, complex high order elements with C^1 continuity, such as cubic Hermite hexahedral elements, are required to produce a smooth fibre field.

4.3 Musculature Control Parameterisation

To produce physically based facial animations by contracting muscles, it is necessary to know which muscles to contract, how much to contract them by, and the contraction timings to produce the desired action or expression. Expression coding systems can help with this task. Such systems aim to taxonomise facial movements, decomposing them into fundamental actions. This can then assist with the synthesis of such facial movements, as muscles can be activated based on these fundamental actions. Ekman and Friesen introduced the Facial Action Coding System (FACS) [EF78, EFH02], whereby facial movements are broken down into Action Units (AU). Each AU describes a fundamental action produced by a muscle or group of muscles. The AUs are independent of a particular face, and, being based on muscle activations, they are independent of personal interpretation. Using the muscles corresponding to the AUs, facial expressions can be simulated by contracting the relevant muscles. FACS can also be used to validate synthesised facial expressions by comparing the AUs of the synthesised expressions to those specified by FACS.

Similar to FACS, abstract muscle action procedures (AMAs) can be used, each of which simulates the specific action of one or several facial muscles, and higher-level expressions can be constructed from these [MTPT88]; however, AMAs are not independent, and the order in which they are executed is important. Other similar multi-level facial parameterisations have also been introduced [KMMTT91]. MPEG-4 Facial and Body Animation (FBA) is a feature-based parameterisation of facial animation that defines facial definition parameters (FDPs), which define the shape of the face, and facial animation parameters (FAPs), which control animation [PF02]. Each FAP corresponds to the movement of FDPs to deform a facial model from its neutral state. FAPs can be low level, representing fundamental movements, or high level, representing expressions or visemes. By creating a mapping between FAPs and muscle activations, these can be used to synthesise facial expressions [Fra05]. Like with FACS-based systems, FAPs and the movement of FDPs could also be used for validation of synthesised facial expressions. Alternatively, rather than trying to manually define muscle activations for facial expressions, such activations could be estimated from performance-capture data [CLK01, SNF05].

4.4 Facial Animation

Various survey papers are available that provide comprehensive reviews of the wide range of facial animation techniques that have been proposed, including physically based approaches [NN99, DN08, LLZ12]. Physically based facial animation approaches typically incorporate skull and muscle models discussed in the previous sections with physically based models of facial skin and connective tissue. Such approaches are discussed in the following sections, starting with efficient, low-accuracy approaches, using a physics engine or the MS method, that typically trade accuracy for performance to be used with computer graphics applications. These are followed by complex, high-accuracy approaches, using the FE or MT method, that are often used to study the behaviour of facial soft tissue. Finally, approaches developed specifically for surgical applications are reviewed, ranging from high-performance approaches using the MS method, to high-accuracy approaches using the FE method.

While no known facial models produce wrinkles as a result of the simulated physically based behaviour, geometric and and texture mapping wrinkling techniques have been used in computer graphics to add further detail to such animations. These techniques are explored in Section 4.5. Furthermore, highly accurate physically based skin and wrinkle models that typically focus on small areas of soft tissue, rather than the full face, are discussed in Section 4.6.

4.4.1 Efficient, Low-Accuracy Approaches

Physically based facial animation approaches that focus on, for example, efficiently producing realistic-looking animations for computer graphics applications normally trade accuracy for performance. Such approaches simulate just enough physics to efficiently produce the desired realism

for animations, for example, by using simple skull and geometric muscle models with a lower-accuracy technique, such as the MS method or a physics engine, for skin and connective tissue simulation. Waters was the first to model the face using a surface mesh with his geometric muscle model [Wat87]. Terzopoulos and Waters then developed a multi-layered MS skin model using the same geometric muscle model and arranging the springs into tetrahedral and cross-strutted hexahedral elements [TW90]. The model consists of three 3D layers, and different nonlinear spring stiffness values within each layer represent the different properties of various sublayers. Five different spring types were used to represent the epidermis, dermis, subcutaneous fatty tissue, fascia and muscle layer properties of real soft tissue. Incompressibility and volume preservation were also modelled, although viscoelasticity wasn't. To deform the model, fixed displacements were applied to nodes upon muscle contraction using the geometric muscle model, propagating unbalanced forces through the remainder of the model.

Many other MS skin models are based on Terzopoulos and Waters' model, and various enhancements have been made to it; for example, Lee et al. simplified the model using two layers of triangular prism elements that modelled the same layers of skin as the original model, and forces were applied to fascia nodes upon muscle contraction, rather than geometric displacements, producing less predictable but more physically based contractions [LTW93]. A simple skull model (using an offset from the skin surface) was also introduced, with an approach that generated forces to prevent skull penetration [LTW95]. This model has been further improved for use with a behavioural animation system to create faces that act intelligently based on artificial intelligence techniques [TL04]. The improved facial model uses the same muscle model and prism elements but with viscoelastic springs, and 42 instead of 28 facial muscles are modelled. Real-time frame rates were achieved using explicit Euler time integration and models with roughly 7,500 springs, even when simulating several of the improved facial models with the behavioural techniques.

Zhang et al. developed a similar MS facial model that also uses Waters' muscle model [ZPS04]. Although the skin was modelled as a single layer, various enhancements to MS facial models were proposed. Edge repulsion springs were used to prevent edge collapse (when a node in a triangular element crosses to the vector opposite), helping to simulate skin incompressibility, and a real skull with a hinged jaw was modelled to allow more accurate muscle-skull attachments and improved skull penetration constraints. Adaptive refinement was used to adapt the local resolution depending on the level of detail in a particular area. Further improvements to this include an automatic muscle mapping approach for efficient creation of muscles from an image, and an adaptive simulation algorithm which simulates only nodes whose deformation is less than a specified distance (determined by an offline computation step) [ZPS03]. Adaptive simulation, however, can only be done for expressions for which the offline computation stage has been performed.

Kähler et al. and Coull also created facial models that used their previously discussed muscle models [KHS01, HKA+01, Käh03, Cou05]. Coull's facial model used a skin model with a similar two-layered prism structure to that proposed by Lee et al. [LTW93], and additional stiff structural springs were used for volume preservation instead of keeping track of the volumes of the elements as they deform. Also, rather than the common approach of using biphasic springs, a nonlinear function was used to vary the spring stiffness values as they compress. For simplicity, the facial model developed by Kähler et al. used a 2D skin mesh where nodes are attached to either muscle or the skull by springs.

Fratarcangeli et al. used manually picked MPEG-4 FDPs and FAPs to define and control morph targets for an MS facial mesh, combining key-framing and physics-based animation [Fra05, Fra09]. Starting with a skin surface mesh with associated FDPs, the anatomical MS soft-tissue mesh proposed by Lee et al. [LTW93, LTW95] with fully nonlinear springs and an adaptive simulation technique with predefined influence areas, Waters' geometric muscle model [Wat87], and the method of skull surface approximation introduced by Kähler et al. [KHS01] were used. Unlike many other approaches that use Waters' muscle model, the shapes of the muscles can be affected by the motion of the anatomical mesh, rather than by contraction only. Each FAP is mapped to the contractions of muscles to create morph targets, which can be interpolated between using MPEG-4 FBA; therefore, although the reference morph targets are produced in a physically based manner, the animation isn't.

Chen and Prakash proposed and designed a GPU-based system for facial animation using techniques similar to those often used by physics engines like PhysX, although only a CPU-based version was implemented [CP06]. The simulation technique was based on the weighted constraint-based algorithm by Teschner et al., which was later used when developing the technique upon which PhysX soft-body deformations are based [THMG04]. Teschner et al., like with PhysX, used a Verlet time-integration scheme for efficiency and stability. Chen and Prakash used a simple technique of creating the required tetrahedral mesh by subdividing a prism structure created from an offset of the facial surface mesh. Nonlinear constraint stiffness enabled nonlinear elasticity to be simulated, although the single volumetric skin layer was isotropic. Muscle fibres for the 14 muscles, defined by Bezier curves, are divided into ellipsoids, similar to the muscle modelling approach of Kähler et al. [KHS01], and convergence anchor points are used to control the activation of several muscles simultaneously at areas where their insertions converge, such as around the lips. On the CPU, real-time frame rates were not achieved, particularly with constraints such as volume preservation.

Rather than the MS method used in his earlier work [Fra05], Fratarcangeli used the position-based dynamics technique for the simulation of skin and connective tissue, as well as volumetric muscles [Fra12, Fra13]. While less physically accurate than the MS method, the geometric constraints enable features such as volume preservation to be modelled without the use of internal or external forces, simplifying the simulations while making them more stable. Multiple muscle layers were used with overlapping muscles, enabling the arrangement of superficial and deeper muscles to better reflect the anatomy of the face. His model contained 25 linear muscles and one sphincter muscle over four layers. Nonlinear geometric constraints were used for volume preservation of the skin, connective tissue and muscles, to bind superficial muscle layers to deep muscle layers and the skull, to bind the skin surface to the underlying muscles and skull (skin is therefore treated as a single layer), and for collision detection between the facial soft tissue and the skull. His research shows that complex facial animations can be produced in real time using the techniques of physics engines. However, the many constraints and parameters that have little physical relevance can be difficult to correctly set, and the large number of constraints required for such complex animations can sometimes lead to instability, requiring a lower time step.

4.4.2 Complex, High-Accuracy Approaches

High-accuracy physically based facial simulation approaches are becoming more common, and these are often necessary for high-accuracy applications, such as studying the behaviour of facial soft tissue. Such approaches normally use complex skull models, and simulate hyperelastic transversely isotropic muscles, and hyperelastic isotropic skin and connective tissue using an accurate technique, such as the FE or MT method. Dao et al. did some work on simulating subject-specific facial expressions using tetrahedral FE models constructed from MRI data [DDP+13], although this approach has currently only been used with a single facial muscle. Choe et al. developed a technique for estimating muscle activations from video recordings to drive their FE facial model, although this didn't use a skull model and considered only linear elasticity [CLK01].

Sifakis et al. also developed a technique for determining muscle activations from performance-capture data [SNF05]. These activations were used primarily for simulating speech using a high-resolution volumetric FE model of the face including a skull and 32 muscles [SSRMF06]. The FE model was constructed using MRI data. An implicit time-integration scheme was used to solve for the facial state based on muscle activations, and both simpler quasistatic and more computationally complex dynamic simulations were considered, with the quasistatic simulations offering up to 10x performance increases. Ignoring inertial forces also had little effect when ballistic (sudden) facial movements weren't simulated. Tetrahedral finite elements were used with Mooney-Rivlin material models, which included transverse isotropy in the muscles. Although the animations produced appear extremely detailed and realistic, the system has very high computation times, and the properties of individual skin layers, viscoelasticity and a physics-based mandible controlled by muscles were not considered, as the focus was on the determination of the muscle activations. As skin layers weren't considered, wrinkles were also unable to be simulated.

The data and algorithms used to develop the model by Sifakis et al. are available as part of the PhysBAM multi-physics simulation library[1]. This is an advanced physics library that, unlike many physics engines like PhysX, focusses on realism, and many of the algorithms aren't currently capable of real-time simulations. It is being developed by the Stanford University, and was publicly released in Q4 2010. Various components of this library, including the quasistatic FE facial simulation component, have been parallelised and tested on chip multi-processors (CMPs) [HGS+07]. A roughly 50x solution time speed-up was achieved for the facial simulations using a 64-core CMP compared with serial execution on a single CPU, although real-time frame rates were not achieved.

Similar to the FE model by Sifakis et al., Wu et al. also used an FE model to simulate facial expressions [WHHM13]. Instead of the commonly used Lagrange elements, cubic Hermite elements were used to create the model. This enabled much fewer, although more computationally complex elements to be used to capture the complex geometry, with only 560 cubic Hermite hexahedral elements being used to create an example facial model, while Sifakis et al. used 370,000 linear Lagrangian tetrahedral elements for theirs. The C^1 continuity of such elements also better satisfies the physical conditions for soft-tissue deformations, such as large bending and torsion, and enables the generation of a smooth muscle fibre field [RP07, MHSH10]. However, thin individual skin layers are unable to be modelled using so few elements. A much higher model resolution would be required to capture such detail, which would be infeasible to simulate due to the complexity of the elements, meaning it would be infeasible to simulate phenomena such as wrinkling.

Due to the large elements of the facial model, muscles would be difficult to capture; therefore, they were modelled separately, and embedded inside elements of the facial model. Active muscle stresses were computed independently over the muscle domains, and then transferred to the facial model as discrete point loads. As, in reality, superficial soft tissue layers slide over the stiff deep layers and bones [WMSH10], frictionless contact was simulated using the penalty method, enabling the superficial fascia to slide over, but not detach from the deep fascia, modelled as an immobile rigid surface, and the skull. Such detail has been neglected in other FE facial simulations [SNF05, BJTM08]. Retaining ligaments were also modelled as boundary conditions by constraining the geometric coordinates and certain derivatives of nodes on the skull to prevent excessive movement of the soft tissue.

Rather than the common hyperelastic neo-Hookean or Mooney-Rivlin material models, more advanced soft-tissue-based constitutive models have been used with FE facial simulations. For example, Mazza et al. used Rubin and Bodners' nonlinear anisotropic elastic-viscoplastic constitutive model [RB02] to simulate the behaviour of an ageing human face, simulating the deformations that occur after 30 years of exposure to gravity using the FE method [MPRB05]. A simplified facial shape was used with four soft-tissue layers, including muscles, although no muscle contractions were modelled. Their research was further developed using an anatomical facial model constructed from MRI scans [BJTM08], and the influence of gravity on the ageing of the facial tissue was again considered [BJT+09]. Various model assumptions, such as constant skin thickness in different facial regions, were validated through comparison with measurements of actual facial tissue response (e.g. due to gravity loads). Even though the viscoplasticity terms of the Rubin and Bodner model were not considered, the calculated responses of facial tissue correlated well with data from MRI images, although the displacements were relatively small (typically less than 5mm).

4.4.3 Surgical Applications

Physics-based modelling approaches have received particular attention for surgical planning and for use in virtual surgery systems due to their potential accuracy. Koch et al. proposed one of the first such systems for predicting facial outlook after surgery using a 2D FE skin model that is connected to the skull by springs [KGC+96]. This is a rare example of a combination of the MS and FE approaches being used. Although the MS method isn't as popular for such systems due to its accuracy limitations, there are some examples of MS surgical systems. Aoki et al. developed such

[1] http://physbam.stanford.edu/

a system for the simulation of post-operative facial morphology after the skull has been operated on [AHTN01]. A 2D skin layer of nonlinear springs with just 14 facial muscles, also represented by nonlinear springs, were modelled, and a common geometric model for jaw movement was used. Various post-operative expressions could also be predicted. Qualitative evaluations suggest the system works well for very small deformations, although relevant patient stiffness parameters were required to be determined by an expert. Duysak et al. used a nonlinear MS system for predicting the outcome of facial bone realignment, and a neural network identification method was used to identify the stiffness and damping parameters automatically [DZI03]. Relatively high accuracy was able to be achieved, although only small displacements were considered. Other techniques have also been proposed to determine MS model parameters [BHS03].

The surgical planning system by Chabanas et al. used the FE method, and this system was one of the first to allow the prediction of post-operative facial expressions using the FE method [CLP03]. Several other surgical systems also allow such facial expression predictions [GZDH04, BWL+10]. Various other facial surgical planning systems, similar to that developed by Chabanas et al., also use linear isotropic materials with simple tetrahedral elements [ZGHD00, Gla02]. Beldie et al. used LS-DYNA, a commercial FE package, to create a detailed, patient-specific facial model for simulation of maxillofacial surgery and facial expressions, which had high accuracy and error within a 2mm threshold [BWL+10]. This also shows that deformations of complex facial models can be accurately simulated using commercial FE packages. Subcutaneous tissue was considered, and was modelled using a hyperelastic neo-Hookean material; however, like with many surgical simulation systems, a linear isotropic constitutive model was used for the skin, which is suitable for such simulations due to the small deformations that are undergone. Several comparisons and reviews of physically based approaches for surgical planning on the face are available [SLG+07, MSN+07].

The MT method has also been used for facial simulations with applications in surgery. Mollemans et al. used the linear elastic MT method for fast predictions in maxillofacial facial surgery planning systems [MSNS05], which was validated using ten clinical cases [MSN+06]. The accuracy of the method was almost identical to the FE method, although only small linear deformations were considered. Kim et al. used transversely isotropic tetrahedral elements for their surgical simulation system [KJW+10]. An average displacement difference of just 0.023mm was observed compared with an FE simulation using a commercial FE package, although simulations only involved small deformations to start with.

4.5 Independent Wrinkle Animation

To add further detail to animations of skin, including facial animations, for computer graphics applications, various geometric and texture mapping wrinkling techniques have been proposed. Geometric techniques deform the geometry of a mesh, while texture mapping techniques use an approach like bump or normal mapping to render wrinkles. Although texture mapping techniques are generally less computationally complex, simple to use to define wrinkles, and don't require a high-resolution mesh, geometric techniques applied to a suitably high-resolution mesh are often able to produce more realistic looking wrinkles, particularly with close up views and shadows. The more complex, highly accurate and detailed physically based skin and wrinkle models that are typically used to simulate small areas of soft tissue are discussed in Section 4.6.

Reis et al. proposed a texture mapping technique that uses normal maps to define facial wrinkles, and a texture map to define various activation areas [RBM08]. Although simple and computationally efficient, the appearance of wrinkles is limited by the limitations of texture mapping techniques, and it is dependent on the quality of the texture maps. The geometric method by Ishii et al. uses Voronoi division and a shading algorithm to account for light absorption and scattering within layers to generate detailed skin textures including furrows and fine details [IYYT93]. Kähler et al. used a simple wrinkling approach that uses several bump maps [KHYS02]. One bump map is created directly from a synthetic human skin model and used to create a skin structure in a similar way to Ishii et al. but without geometric deformation, and another is created

from hand-drawn wrinkle images depending on the facial pose.

Various geometric techniques allow the user to simply define dynamic wrinkles using an image [YZ05], and the system by Bando et al. also automatically generates and renders fine wrinkles using bump mapping, greatly improving the wrinkling detail [BKN02]. Alternatively, the approach of Azmi et al. enables wrinkles to be drawn directly onto an uploaded 3D facial mesh [AWM06]. Dutreve et al. presented a facial wrinkling approach that blends reference wrinkles maps (each of which are associated with a skeleton pose) according to the current facial pose, meaning wrinkles for many poses can be created from a small set of wrinkle definitions [DMB09]. Each of these approaches also allow various different parameters, such as wrinkle width or influence area, to be specified; however, the appearance of the wrinkles will depend on the definition of the wrinkles by the artist.

The approach by Li et al. automatically generates dynamic wrinkles on a facial mesh according to a user specified number of wrinkles, user defined regions, and the movement of key nodes [LYKL07]. Although less work is required, the user also has less control over the positions and sizes of the wrinkles that lack anatomical accuracy. The method by Larboulette and Cani automatically generates wrinkles according to some user-defined locations and parameters, refining the resolution of the underlying mesh if necessary [LC04]. Alternatively, the approach by Müller and Chentanez automatically adds wrinkles to an existing mesh by loosely attaching a high-resolution wrinkle mesh to the base mesh [MC10]. Unlike previous approaches, the loose wrinkle mesh attachment can deform in the tangential direction, rather than just deforming the mesh in the normal direction. The geometric wrinkle solver is also independent of the main solver for the base mesh, enabling the method to be used with existing animations in a plug-and-play fashion. On a modern computer, it is expected that most of these wrinkling techniques that make use of some geometric deformation function should be capable of real-time animation on reasonably high-resolution meshes.

Bickel et al. proposed a system to create facial animations with wrinkles using performance-capture techniques [BBA+07]. Both fine and large wrinkles can be captured by tracking colouring applied to a subject's face, although the minimum size of wrinkles is limited by the colouring that can be applied to the face to guide wrinkling, and the captured data cannot be used in other scenes or on a different model than the one it was captured for. This has since been implemented on the GPU to achieve huge speed increases (by up to roughly 10x compared to on a quad-core CPU) when driving a model from the motion capture data [BL11]. Sanchez developed a real-time, model-independent wrinkling system that uses motion capture data to evaluate the strain of facial movements, and produces a normal map of wrinkles that is layered onto the face [MES05, San06]. The wrinkling model, however, must first be configured to the performer, relating the wrinkling effects observed on the performer to strain sustained by the facial tissue.

Some wrinkling techniques use a geometric muscle model to produce wrinkles. Bui et al. extended Waters' muscle model [Wat87] to produce bulges and wrinkles on the face [BHN03], although this still required the face to be manually divided into regions, and only a simple single-layer skin model was used. On the other hand, Zhang et al. used a layered skin mesh, similar to that proposed by Lee et al. [LTW93], with Waters' muscle model, and developed a geometric wrinkle model that produced wrinkles according to muscle contractions [ZST05b]. This approach therefore has increased anatomical accuracy to produce more accurate and realistic wrinkles, and it is still capable of real-time animation, although wrinkle production involves the use of a heuristic function and is therefore not fully physically based. A limitation of many wrinkling approaches is that a high-resolution mesh is required at areas that wrinkle, even if there is currently no wrinkling in such an area, whereas the approach of Zhang et al. uses adaptive refinement to dynamically refine mesh resolution when required.

Wu et al. developed a similar wrinkling method using a different layered MS model and muscle model to simulate both large and fine static and dynamic wrinkles [WKMMT99]. Unlike previous approaches, ageing wrinkles are simulated and a linear plastic model is used to change the skin shape based on previous deformations, like with the skin model by Magnenat-Thalmann et al. [MTKL+02]. Both geometric and texture mapping approaches are used to render wrinkles. On the other hand, various simplifications to the wrinkle and skin models (e.g. incompressibility isn't modelled) mean that the wrinkles produced don't look particularly realistic, particularly in the

real-time rendering mode. The facial model also lacks a model of a skull.

4.6 Detailed Soft-Tissue Simulation

Various physically based approaches for producing detailed soft-tissue simulations have been proposed, which typically focus on simulating small areas of soft tissue with high accuracy, rather than the full face. Such approaches are reviewed in the following sections. Detailed skin-block simulations that focus on accurate simulation of skin, typically to simulate wrinkling, are firstly discussed. Some such detailed simulations use advanced bespoke skin and soft-tissue material models to better simulate the response of soft tissue, which are discussed in the section following. The final two sections review generic low-accuracy (using a physics engine or the MS method) and high-accuracy (using the FE or MT method) soft-tissue simulations that typically focus on simulating biological organs for surgical applications.

4.6.1 Skin-Block Models

Various detailed models of blocks of skin have been created that focus on accurate simulation of skin deformation using the FE method rather than computational cost. These are usually used to study the deformation and experiment with the mechanical properties of skin. Larrabee and Sutton created the first such model, which is a 2D FE model attached to an immovable surface by springs representing the subcutaneous tissue [LS86]. Boissieux et al. developed a two-layer linear isotropic elastic FE model for effects of cosmetics and ageing by simulating temporary and permanent wrinkles [BKTK00]. Each triangle in the skin mesh has a shape memory, and the rest shapes of the triangles are modified according to previous deformations, simulating the plasticity of skin. Due to the simplicity of the skin model, the wrinkles formed in unrealistic sinusoidal patterns and thus had to be postprocessed. This model has since been further developed into three-layer model to study the mechanical properties of skin with ageing [MTKL+02], and experiments show that the two-layer model leads to a contradiction between the prediction of the folding capacity of skin and clinical observations, whereas the three-layer model predicts quite well the clinical aspects of skin wrinkling with age.

Yin et al. experimented with a four-layer linear FE model of a wrinkling fingertip in water that also used isotropic layers [YGC10]. The model is based on the fact that the vasoconstriction of substrate tissue due to water immersion leads to mismatched deformation between the films (such as the epidermis) and the substrate. Although skin wrinkling due to water immersion is not directly applicable to our research project, wrinkles can be modelled by films that undergo compression due to other types of mismatched deformation, and, as an extended application, the developed mechanical model was also used to provide qualitative insights into general skin wrinkling and ageing.

Kuwazuru et al. used a more abstract approach by modelling different stages of facial skin wrinkling using multi-stage linear buckling theory with multiple 2D static beam models [KSY08b], each of which captured different skin layers. Three stages of buckling were considered, each of which were simulated independently using different multi-layered 2D beam models to evaluate buckling in a total of five skin layers. For example, the first stage considered extremely fine wrinkles due to the stratum corneum buckling under the support of the viable epidermis using models that captured only these two layers. As with the previously described models, linear isotropic materials were used, although the natural tension of skin was included, which is useful for simulating ageing. For each buckling stage, the critical strain and the wavelength, representing wrinkle size, at this strain was computed using a range of skin layer parameters for different-aged skin.

The results showed the increase in size of larger wrinkles with age, and also suggest that changes in the thicknesses and material properties of the epidermis and papillary dermis with age lead to the sudden appearance of small permanent wrinkles between the ages of around 25 and 30 years old, rather than a more gradual appearance from very young skin. This phenomenon could

not be captured using a three-layer model [MTKL+02]. The results were further demonstrated by simulating each buckling stage with 2D FE skin models [KSY08a]. It is important to note that experiments with the various soft-tissue models considered here have found that the epidermis, in particular the stratum corneum, and the dermis have a large effect on skin wrinkling compared to the hypodermis [MTKL+02, YGC10].

Flynn and McCormack created an inhomogeneous FE model to simulate forearm wrinkling that also modelled viscoelasticity and natural tension [Fly07, FM08, FM10]. Four different models were experimented with, each of which modelled different combinations of the stratum corneum, dermis and hypodermis, and results again found that only the three-layer model correctly followed the trends of real skin. The results of this model correlated well with an in vivo laboratory experiment and followed experimental trends, with the maximum range and average roughness of wrinkles being mostly within the range of values from the experiment. The commercial FE package ABAQUS was used for the FE simulation. Hung et al. used a similar model to this for simulating wrinkles due to contraction of underlying muscle [HMSH09], although the muscle was simply modelled as an extra layer in the block. This model was also experimented with by varying the layer thicknesses and properties to simulate, for example, mature skin. Like with previous similar experiments, these showed the importance of modelling the inhomogeneity of skin. The freely available CMISS FE package, developed by the same institution where the research took place, was used for the simulation.

4.6.2 Bespoke Skin and Soft-Tissue Material Models

Bischoff et al. performed FE simulations of skin using a nonlinear isotropic constitutive law, concluding that, although some deformations can be accurately simulated, an anisotropic viscoelastic material model would achieve better results [BAG00]. A nonlinear rheological constitutive model of soft tissue that includes both anisotropic and viscoelastic properties according to various material parameters was therefore developed [BAG04, Bis06]. The one-dimensional quasilinear viscoelasticity theory was used at the level of representing collagen fibres - a more detailed level than previous approaches that use the theory. ABAQUS was also used for this FE simulation. The results showed that complex anisotropic viscoelastic properties of skin, such as hysteresis and preconditioning, could be simulated following trends of real skin. Skin wrinkling was not explicitly considered.

Rubin and Bodner proposed a nonlinear anisotropic elastic-viscoplastic constitutive model using three-dimensional viscoelasticity theory that can also simulate complex properties such as preconditioning and hysteresis [RB02], and plasticity effects are also modelled, which would be useful for simulating skin ageing. However, the model requires the specification of 14 parameters, as opposed to 7 for Bischoff's reduced parameter model, which also models three different classes of fibres independently (such as collagen and elastin), as opposed to modelling them altogether as a single class. Material parameters for Rubin and Bodners' model have also been determined that agreed well with experiments on facial skin, and this model has been used by various projects involving simulations of facial skin (see Section 4.4.2) [MPRB05, BJTM08].

4.6.3 Generic Efficient, Low-Accuracy Soft-Tissue Models

The only known application of the PhysX physics engine in research has been to develop a real-time virtual surgery system [MHL+09]. Some limitations of PhysX were found through this; for example, soft bodies must be isotropic and homogeneous, although a workaround is to attach two soft bodies using a rigid body and a distance joint. It was also found that soft body to soft body collision detection can be relatively poor. One advantage of PhysX, however, is that it can be readily hardware accelerated on the GPU using CUDA, and some tests showed speed increases of over 10x when performed on the GPU.

The MS method has been used for surgical applications of soft tissues, although these are frequently used for surgical training purposes where visual realism and real-time simulations are

more important than exact patient-specific deformations. Brown et al. proposed such a non-linear MS surgical training system that is capable of both dynamic and quasistatic simulations of volumetric soft-tissue objects [BSB+01]. Although the quasistatic simulations make various assumptions, such as the positions of several control nodes are given, and the velocities of these nodes are slow, these are usually acceptable for such surgical systems. The quasistatic simulations use a 'wave-propagation' technique to calculate nodal displacements starting from the most displaced nodes, enabling a cut-off point to be defined after a particular low displacement level has been reached. Fast simulation speeds could be achieved, particularly with the quasistatic approach, although factors like tissue incompressibility were not modelled. Qiao et al. also proposed a simple 2D MS system for surgical training purposes, which compared the stability of various mesh topologies, finding that a topology with overlapping regular hexagons gave best stability [QCY09]. Three-dimensional topologies, however, were not considered.

Duysak and Zhang proposed a modified MS algorithm for simulation of deformable soft bodies, which they called the mass-spring-chain (MSC) method [DZ04, Duy06]. With this approach, deformation starts from moved mass points, and points connected to these are called semi-active points. Deformation propagates through the mesh, although, after a mass spring has been deformed, it is not deformed again, ending a cycle of deformation. The next cycle then starts from the semi-active points. Elasticity and rigidity vectors are also used, which define the possible deformation bounds, and constrain the movement and deformation of a spring. The MSC method doesn't require using a large number of timesteps, and similar results to the MS method can be achieved with faster simulation times, although more parameters than with MS systems that have little physical relevance must be specified, and it is essentially less physically based than the MS method, using various heuristic functions to control the deformation. The MSC method has also been applied to facial simulations for surgical purposes [DZ05]. Nesme et al. also proposed a physically based method similar to the MS method called Phymul [NMP+05]. Heuristic functions and constraints were used to improve simulations and stability, and shape memories were used to enable an object to recover to its exact original shape after deformation. Although this method was much more efficient than the FE method for simulating soft-tissue materials, the accuracy of the FE method was generally better, particularly for simulating large deformations.

Several GPU MS systems have been implemented for use with virtual surgery applications or arbitrarily shaped deformable objects [TE05, MHS05]. Georgii and Westermann proposed two GPU MS implementations [GW05], discovering that using edge-centric rather than point-centric data structures enables best performance, particularly with unstructured meshes, although their research used programmable shaders to utilise the GPU as it was done before the release of architectures like CUDA, which enable the GPU to be utilised more effectively for general purpose computation. By utilising the CUDA architecture, Leon et al. made use of shared memory with their GPU MS virtual surgery system, achieving nearly 80x speed-up compared to serial execution on the CPU [LEG10], although the use of shared memory placed limitations on the structure and ordering of mesh nodes.

4.6.4 Generic Complex, High-Accuracy Soft-Tissue Models

Complex physically based approaches have been used to model areas of the body other than the face, such as limbs [Coo98] and the neck [LT06]. Lee et al. extended their biomechanical model of the neck to an extremely detailed physics-based model of the human upper body, excluding the face [LST09]. A non-conforming tetrahedral FE mesh, coupled to a surface mesh, was used to simulate the soft tissue, while active muscle forces for the 814 muscles were determined using a piecewise linear Hill-type muscle model. The FE system used is similar to that used with a facial model by Sifakis et al. [SNF05].

Various implementations of the FE method on the GPU for biomechanics or surgical purposes have been proposed [WH04, SM06]. Liu et al. used a CUDA implementation of an implicit dynamic FE algorithm and applied this to a virtual surgery system, although only linear elasticity was considered [LJWD08]. For small models with around 5,000 nodes, the GPU version was slower than the quad-core CPU version, although speed-ups of up to 4x were observed with large models.

An explicit MS system was also implemented using CUDA for comparison, for which speed-ups of up to 20x were observed. It should be noted that, since this paper was written, various changes have been made to the CUDA architecture that, along with more powerful hardware, have enabled much further performance increases.

Lapeer et al. developed a GPU implementation of a nonlinear FE soft-tissue model for use with an interactive surgical simulator [LGK11]. The TLED formulation of the FE method was used to formulate and integrate the equations of motion, which has also been used with various other nonlinear soft-tissue simulations due to its efficiency [Gha08, VJMW11]. Three different isotropic hyperelastic FE models were considered, and the parameters of the models were determined from in vitro experiments on skin. Although the GPU uses lower floating point precision than the CPU, comparisons showed that this led to no significant differences in the accuracy between the CPU and GPU implementations, and the simulations closely matched the simulation of a skin model created using ABAQUS for validation. Huge performance increases were also recorded when using the GPU, with over 1,000 iterations per second (required for the haptic device) capable when simulating a model with 50,000 tetrahedral elements, whereas the CPU implementation could only handle 500 elements at that threshold.

Taylor et al. developed a slightly more advanced framework for modelling anisotropic viscoelastic soft tissue on the GPU that also uses the TLED algorithm[2] [TCO08]. A speed-up of up to 56.3x compared with a CPU implementation was observed, and transversely isotropic materials modelled with roughly 55,000 hexahedral elements can be simulated with a cost of around 2ms per timestep on a mid-range desktop PC. It was also found that modelling properties such as anisotropy and viscoelasticity increase computational cost by as little as 5.1% using the developed model. This framework has not yet, however, been used to create a model as complex as the human face. The developed software ('Nifty Sim') is freely available[3]. The CUDA TLED FE implementation has also been implemented into the SOFA open source framework[4] [CTA+08]. SOFA is an environment, primarily aimed at medical simulations, for creating and simulating complex deformable models.

The MT method has also been used for soft-tissue simulations with applications in surgery. Cotin et al. first used the linear elastic MT method for surgical simulation, which was capable of real-time simulation of liver surgery due to the precomputation of values derived from the FE method [SC99], and it was later extended to allow the cutting of tissue [CDA00]. For virtual surgery applications, large deformations can occur while the surgeon is performing the surgery, even if the final deformations post-surgery are small, meaning a more accurate soft-tissue model would be better suited. Picinbono et al. therefore proposed the nonlinear MT method with an anisotropic material model, improving biomechanical realism [PDA03]. Incompressibility constraints were also introduced, which could also be used with the FE method, enabling incompressible materials to be simulated with a lower Poisson ratio to prevent the instability problems associated with such materials. Real-time frame rates could still be achieved in the liver surgery scenario, and nonlinearity was only modelled for high deformation areas. Other similar nonlinear MT formulations have also been proposed [SLML01]. Xu et al. proposed a more advanced tetrahedral MT soft-tissue model by also including viscoelastic effects in the constitutive law, enabling effects such as creep and hysteresis to be simulated with as little as 5% additional computation compared with the nonlinear anisotropic model [XLZH11].

4.7 Validating Soft-Tissue Simulations

The high complexity of soft-tissue simulations makes them extremely difficult to validate. Many soft-tissue simulation techniques are validated by comparing a simple simulation to its analytical

[2] Information on this use of the TLED FE method for GPU-based soft-tissue simulation was also obtained from meetings with Dr Zeike Taylor (Department of Mechanical Engineering, The University of Sheffield), the lead author of this approach.

[3] http://sourceforge.net/projects/niftysim/

[4] http://www.sofa-framework.org/

result [LKH04, TCO08], or to an FE simulation using a well-known and validated FE software package [MJLW07, LGK11]. However, analytical results can only be found for very simple simulations, and FE software packages have their own intrinsic limitations regarding the modelling complexity and simulation accuracy of extremely complex biological models like soft tissue. Kerdok et al. suggested creating a gold standard of experimental results of soft-tissue deformations [KCO+03]. With their approach, fiducials are embedded into a volume in a pattern, and CT scans are taken of the undeformed and deformed object to find the displacements of the fiducials, providing deformation data at intervals throughout the entire volume of the object. However, while their approach could be used to obtain deformation data for soft tissue, it was only tested using a simple silicone cube. While no currently known techniques have been proposed specifically for the validation of skin wrinkling, such as the shapes and sizes of the simulated wrinkles, experimental results containing wrinkled skin surface deformation data, and FE simulations of skin wrinkling have been previously published, and could therefore be used to assist with validation [Fly07].

4.8 Generic Deformable Soft-Body Animation

As well as soft tissue, physically based techniques have been used to animate generic deformable soft bodies, particularly for computer graphics applications. Such approaches could potentially be extended to simulate more complex materials like soft tissue. Soft-body deformation techniques similar to those used by many physics engines are capable of deforming arbitrarily shaped simple elastic objects in real time [MHHR06], although the deformations of these objects are much simpler than the deformation of soft tissue. Various such techniques can also approximate effects such as plasticity [THMG04]. However, these approaches mainly focus on computational speed and stability, for example, when simulating many such objects in a game-like environment. On the other hand, various techniques have been proposed to simulate simple elastic objects using complex techniques like the FE method, which can now easily be used for simple real-time simulations. Yang et al. proposed a GPU-based FE system for simulating linear elastic objects using models that also contain rigid objects (e.g. bones), which was capable of real-time simulations of objects with over 33,000 tetrahedral elements [YRTG10].

Multigrid methods have been used for simulating deformations of linearly elastic FE models [GW06]. Such methods involve computing the physics-based equations on coarser simulation models, and propagating the necessary values to guide the solution of higher-resolution models. A model hierarchy with different model resolutions is therefore required. Multigrid methods offer performance advantages when using implicit time-integration schemes, where the solutions of coarser models can be used to significantly reduce the expensive timestep computation times of higher-resolution models. While they can increase the inexpensive timestep computation times of such models when using explicit time integration, stability is improved, which can enable longer timesteps to be used [WT04]. Dick et al. implemented a linearly elastic multigrid FE solver on the GPU, enabling 11 timesteps per second to be calculated for voxel-based models of 269,000 regular hexahedral elements, which is also almost 8x faster than a quad core CPU implementation [DGW11]. Furthermore, due to the simple voxel-based structures of the models, mesh hierarchies for the multigrid method could be computed easily and efficiently.

Sifakis et al. proposed an FE physically based simulation framework that uses a mesh-based FE model for simulating hyperelastic models, combined with a meshless technique for straightforward handling of collisions, fracture and plasticity [SSIF07]. Although the quality of simulations when handling such phenomena look promising, when used with an FE facial model previously constructed [SNF05], only slow computational speeds of just 18 minutes per frame were achieved. Other similar methods have also been proposed, but using just a mesh-based FE technique [NPF05, MTPS07, KMBG09]. These are able to simulate various phenomena, such as elasticity, viscosity and plasticity, often by using remeshing procedures as the soft-bodies deform; however, these usually use a simple isotropic linearly elastic constitutive law with the physics-based simulations, and other heuristic functions based on the deformations are used to simulate factors such as plasticity. Wojtan et al. used a fluid simulator to simulate viscosity effects of a

soft body that has deformed into a fluid-like object [WT08, WTGT09], although such extreme deformation isn't necessary for facial animations. These simulations are also extremely computationally complex and unsuitable for real-time simulations. Factors such as remeshing during simulation are also used with other types of simulation system, such as virtual surgery systems where the cutting of soft tissue is required. Although remeshing techniques could be used with this project, for example, to enable more realistic deformation of low-resolution areas affected by high stresses, remeshing complex volumetric meshes can require heavy computation [WRK+10].

4.9 Applications of Animation in Films and Games

The previous sections have reviewed a range of both low and high-accuracy physically based soft-tissue and soft-body animation approaches that have been proposed. Since the main target area of our research is computer graphics applications like films and games, this section reviews some applications of facial and physically based animation that have already been used in films and games. In the film industry, lifelike animation is often required, and it is often desirable for the uncanny valley to be traversed. The familiarity of a computer generated character increases with human likeness up to a certain point, after which likeness decreases hugely. With even more human likeness, however, familiarity is restored, creating the uncanny valley - a region where an entity is close to looking human but doesn't look realistic enough, triggering revulsion [Mor70].

Final Fantasy: The Spirits Within (2001) was one of the first computer animated films that attempted to animate realistic-looking humans, although the facial animations were considered to be in the uncanny valley [3D 10]. In The Curious Case of Benjamin Button (2008), an actor's expressions were mapped onto a photo-realistic computer generated aged face that includes details such as wrinkles [Syd09]. Benjamin Button is largely considered as the first animated character that has traversed the uncanny valley [3D 10]. A scene in the final instalment of The Matrix trilogy (2003) involving 'The Superpunch' also used computer graphics techniques to produce facial animation with a high amount of skin deformation and wrinkling [Bor05]. This short sequence was primarily created by an artist who added deformations to captured performance data, and, even though the animation looked lifelike, it required a lot of manual work.

Physically based animation has also been used in various films. A vector muscle approach similar to Waters' muscle model [Wat87] has been used by Pixar to animate the face of the baby, Billy, in the movie Tin Toy (1988) [PW08]. Skulls covered with muscles, and a fat and skin layers were created to produce facial animations for a range of different shaped human and non-human faces in the Shrek films (2001 - 2010) [Cul11], even though the films didn't use photo-realistic environments. The later Shrek films also include improvements that allow better wrinkle production on both faces and clothing. Avatar (2009) also used facial models represented by muscles, fat and skin but in photo-realistic environments, and complex muscles were able to intercollide and preserve their volume in an anatomically correct manner [Teo09]. Extremely high-resolution meshes enabled facial wrinkles to be layered onto the skin deformations. Facial muscle activations were captured from actors using performance-capture techniques. An earlier version of this system was also used in the making of King Kong (2005).

There are also various games where detailed and realistic facial animation is extremely important, including LA Noire (2011). For this game, a motion capture system involving 32 cameras was used to capture the detailed movement of the face including fine wrinkles [3D 11]. Although this technique isn't physics-based, extremely realistic facial animation was able to be produced, particularly for a real-time game; however, it would probably be difficult to transfer the detailed captured animation to a character with a differently proportioned face to that of the performance-capture subject. Systems and models used for films are normally too complex for use in current games, although a lot of current games simulate physics using simpler physics engines like PhysX or Havok that are designed for real-time performance. For Star Wars: The Force Unleashed (2008), on the other hand, the linear dynamic FE method was used to simulate rigid-body deformation and fracture [PO09]. Linear isotropic constitutive laws were used, and linear tetrahedral elements were used to create non-conforming meshes. Plasticity was simulated using a simple additive

function, and the simple FE formulation was implemented on the GPU for real-time performance with the game. Most current games use the rigid-body, fluid and cloth components of physics engines, although various games, such as those developed with Unreal Engine 3[5] that can be easily expanded with the Unreal Development Kit, can be user modified to include scenes with physics-based soft-body support.

4.10 Summary

A physically based facial model typically consists of a muscle and a skin model, normally along with a skull model to act as a boundary during simulations, and sometimes also a wrinkle model to produce additional wrinkling effects that aren't produced directly by the skin model. Starting with the innermost structure, a skull model can vary from a simple low-resolution offset of a facial surface [LTW93, WD04], to a more accurate higher-resolution model of a real skull [CLP03, BWL+10], possibly including a rotatable mandible [ZPS04, Cou05]. The skull is often used to set simulation boundary conditions, with the movement of nodes on the skull being fixed [SNF05, BJTM08] or restricted [WYX09, HWHM11].

A muscle model contains two components: a muscle representation (e.g. a geometric or physically based representation) and a muscle contraction method (the computation of the magnitudes and directions of muscle contractions). With geometric muscle models, muscles can be represented as vectors [Wat87, ZPS04], or more realistic volume-enclosing surfaces that preserve their volume when contracted [Wan10, RL13]. Volumetric muscles may be composed of simple components like ellipsoids [SPCM97, KHS01], or they may be arbitrarily shaped [DCKY02, Cou05]. More anatomically accurate muscle models have been developed that represent muscles as physically based volumes using physics-engined-based techniques [Fra12], the MS method [NT98] and the FE method [BJTM08, HMSH09], some of which model the transverse isotropy of muscles [RP07, TZT09], and connective structures such as tendons [OMvO+03, TTC+05].

Many muscle contraction methods are based on a linear [LT06] or nonlinear Hill-type model [RP07], some of which are biologically inspired [Hun95]. More advanced methods also consider contraction velocity [TZT09], multiple fibres [SK07] and complex activation behaviour [NPP13]. The direction of contraction can be approximated as parallel to a central action curve [TZT09], or, more anatomically, by using a fibre field [GZDH04], which can be created using the gradients of B-spline surfaces or volumes [WKMMT99, TSB+05], or elements with C^1 continuity [RP07, MHSH10]. Expression coding systems and facial parameterisations have been proposed to taxonomise facial movements [EFH02, PF02], which can assist with the synthesis of expressions from activations of the relevant facial muscles [Fra05]. Alternatively, muscle activations can be estimated from performance-capture data [CLK01, SNF05].

Physically based facial animation approaches have been proposed that incorporate skull and muscle models. Some such approaches model facial soft tissue using physics-engine-based [CP06, Fra12] and MS techniques [KHS01, ZPS04] that focus on efficiency and stability. Many such MS models are based on Terzopoulos and Waters' layered model [TW90], and capture multiple skin layers using biphasic [LTW95] or nonlinear springs [Fra05]. Additional functions are required to simulate, for example, incompressibility and volume preservation with MS systems [TW90, Cou05]. More complex facial models have been developed using the FE method [SNF05, WHHM13], which, due to its accuracy but computational cost, is mainly used for simulating small displacements with high-accuracy scientific [MPRB05, BJTM08] or surgical applications [CLP03, BWL+10]. The linear MT method has also been used to simulate small displacements with surgical applications [MSNS05, KJW+10]. Facial models composed of volumetric elements, such as many FE facial models, typically model skin as a single isotropic linear elastic [CLP03] or hyperelastic layer [WHHM13], and an extremely high resolution would be required to capture multiple skin layers using such elements.

To add further detail to animations of skin, including facial animations, for computer graphics applications, skin wrinkles can be layered onto a surface mesh using texture-mapping techniques

[5] http://www.unrealengine.com/

[KHYS02, RBM08], geometric techniques that deform the geometry of a mesh [YZ05, MC10], or a combination of texture-mapping techniques for fine wrinkles, and geometric techniques for large wrinkles [BKN02, WKMMT99]. Geometric techniques require a high-resolution surface mesh, although mesh resolution can be adaptively refined when required during animations [LC04, ZST05b]. Some approaches require wrinkles to be specified manually [YZ05, DMB09], while others compute them automatically [LYKL07, MC10] or using performance-capture techniques [MES05, BBA+07]. Geometric and texture-mapping wrinkling approaches have also been used to layer wrinkles onto physically based facial animations, rather than simulating them in a physically based manner [WKMMT99, ZST05b]. With such animations, the gross deformation produced in a physically based manner triggers the production of wrinkles using a simpler, more efficient technique.

Rather than simulating the full face, various physically based approaches have been proposed to simulate detailed models of small areas of soft tissue with high accuracy, normally for scientific or surgical applications. Various multi-layered FE skin-block models have been developed to accurately simulate expressive or ageing wrinkles for different-aged skin using the 2D [MTKL+02, KSY08a] or 3D FE method [FM08, HMSH09], some of which simulate complex anisotropic and viscoelastic behaviour [FM08], and plasticity can be modelled to simulate the formation of ageing wrinkles [MTKL+02]. Complex constitutive models have also been proposed specifically to simulate soft tissue [RB02, Bis06], which are able to simulate complex skin behaviour such as hysteresis and preconditioning. Physically based simulations of biological organs have also been produced, some of which use physics-engine-based [MHL+09] or MS systems [DZ05, MHS05] for efficiency, although more accurate approaches use the FE [WH04, TCC+09] or MT method [PDA03, XLZH11], some of which also simulate anisotropy and viscoelasticity [TCC+09, XLZH11]. Because of its efficiency, the TLED FE method has been used for various nonlinear FE soft-tissue simulations [MJLW07, VJMW11], including GPU-based implementations [TCO08, LGK11], resulting in large speed-ups. Soft-tissue simulations can be validated using results from laboratory experiments [Fly07], which may provide internal deformation behaviour rather than just the outer surface deformation [KCO+03].

Despite the various physically based muscle and skin models that have been developed, those that are able to simulate wrinkles in a physically based manner typically only model a small and simple area of soft tissue, such as a flat soft-tissue block [FM08, HMSH09]. There are no known models that represent the whole of a large structure, such as the forehead, including wrinkles, using physics-based techniques. Finally, physically based techniques including the FE method have also been used to animate simpler generic deformable soft bodies for computer graphics applications [SSIF07, DGW11], and many such approaches use isotropic linear elastic materials with lower resolution models, making them capable of real-time animation [GW06, YRTG10]. Physically based animations of both faces [Teo09, Cul11] and generic objects [PO09] have also been used in various films and games.

Chapter 5

Physically Based Model Creation Approaches

Physically based simulations require an appropriate physically based model to be created. In computer graphics, such models normally consist of a 3D surface mesh (the visual representation of the object to be simulated) linked to a simulation model (the physics-based representation of the object). The surface mesh is created first, and then used to produce the simulation model. Surface meshes (e.g. polygonal meshes) are widely used in computer graphics, and there are many techniques and modelling tools to assist with the creation of these. For volumetric models, the surface mesh will enclose a volume (e.g. the soft-tissue between the outer skin layer and the skull), and may also contain various volume-enclosing internal surfaces (e.g. muscles). The simulation model is then simulated using a physics-based technique, which moves and deforms the surface mesh to produce animation.

Creating a suitable simulation model is normally a difficult and time-consuming task. A simulation model requires:

- Creation of a simulation mesh, which is a subdivision of the domain into elements
- Simulation properties, such as material properties and boundary conditions, to be set

Creating a simulation mesh is analogous to discretising a surface into polygons and vertices when creating a surface mesh, although the domain and requirements of a simulation mesh can be more complex. There are three stages to creating a simulation mesh:

1. Choosing the element type to use
2. Choosing whether a conforming or non-conforming (e.g. voxel-based) simulation mesh will be used
3. Discretising the domain to create the simulation mesh

Regarding element type, either Lagrange or Hermite interpolation functions can be used. While fewer Hermite elements are required to model complex geometries, such large elements would not be able to capture the heterogeneity of skin, and thin skin layers with different material properties would be infeasible to model due to the complexity of the elements. As such detail is necessary for wrinkle simulation [MTKL+02, FM08], we only consider Lagrange elements. Also, as high-resolution models are required, we mainly consider linear elements with a single integration point for optimal computational performance.

Although the whole domain discretisation process could be done manually, for example, by just using a 3D modelling program, this can become tedious and difficult when creating anything more complex than simple simulation meshes, particularly when the accuracy of the model is important. Most modelling programs are also designed for creating surface meshes, meaning they

lack features to help create a volumetric model and export it to a suitable format. Complex and precise volumetric models are necessary for accurate physics-based simulation; therefore, it is desirable to use automated techniques to help with the creation of the simulation mesh, for example, to discretise the volume into a finite number of elements. Additional components such as eyes and hair can then be added to the model.

Figure 1.2 in Chapter 1 shows an example of a forehead surface mesh that is used with our research project, and discussed further in Chapter 6. The figure identifies the various surfaces and volumes, demonstrating the complexity of the domain (the volumes between the skull and skin surfaces) for which a simulation model must be created. The following sections firstly review approaches to create surface meshes. Techniques to create both conforming and non-conforming simulation meshes with different element types are then explored. This includes hybrid simulation meshes composed of more than one element type. Approaches to automatically compute and set simulation properties are then discussed, followed by approaches that deform an already created reference simulation model to fit a new target surface, rather than creating the simulation model from scratch.

5.1 Surface Mesh Creation

There are many techniques that can be used to create surface meshes. For example, a facial surface mesh can be created using a surface capture technique, using a 3D scanning technique (e.g. a laser scanner) or photographs, from existing models, such as by deforming existing meshes or using statistical techniques, or possibly from scratch in a 3D modelling program or using anthropometric measurements. Each of the many possible techniques is widely used in the computer graphics field. Surface capture techniques are useful for modelling real objects, and more expensive techniques like laser scanning are normally able to produce a more realistic model with finer detail, although these techniques require postprocessing. On the other hand, there are programs available that are able to automatically create simpler models, for example, FaceGen[1] uses statistical techniques to create facial surface meshes. A simple skull surface can be created as an offset of the facial surface [TW90]; however, such a surface would need to be cut to produce a separate jaw bone that can be animated [KHS01].

Muscle surfaces can be manually specified, for example, using 3D modelling tools. To simplify their creation, they can be defined using an interactive editor that, for example, uses origin and insertion points to define vector-based muscles [ZPS04], or processes grid points to create muscles using ellipsoidal segments [KHS01]. Some editors can also handle a complex interwoven and layered musculature [Fra12]. Cross-sectional profiles along the lengths of muscles can be used to create more advanced muscle shapes [Cou05], although, for complex shapes requiring definition of many contours, this may be tedious. While placing more limitations on muscle shapes, Ng-Thow-Hing et al. created B-spline volumes from just user-defined origin and insertion contours, which were quadratically interpolated along the lengths of the muscles [NTHAB+98, NTHF02]. Such parametric muscle volumes can also be used to generate complex fibre fields.

Aina proposed a technique to automatically construct muscle fibres from muscle attachments painted on a texture map of a skull, which involved generating muscle outlines on a SMAS surface (constructed semi-automatically from the skull surface) using convex hulls connected by their mutual tangents [AZ10, Ain11]. While only muscle fibres were produced, rather than surfaces, such surfaces enclosing muscle volumes could be constructed by extruding the areas defined by the muscle outlines on the SMAS surface. However, in reality, the majority of facial muscles originate on the skull, and are inserted into the dermis or subcutaneous tissue, rather than originating from and inserting into a SMAS layer. Also, the layered structure of facial musculature can't be modelled by aligning muscles along a single surface.

Techniques have been developed to create new physically based facial surface meshes by deforming all or part of a reference surface mesh to fit new facial surfaces [WD04]. It may be possible

[1] http://www.facegen.com/

to simply use affine transformations to deform a reference skull to create a very rough approximation of a skull for a new facial surface [KHS01]. Alternatively, Aina proposed an approach to automatically adapt a detailed reference skull model to any given face model using manually placed landmarks [Ain09, Ain11], although strategic landmark placement is required for desirable results. Similarly, muscle surfaces for a reference model can be defined and adapted to a new facial surface [HKA+01]. Fratarcangeli also used a landmark based procedure with radial basis functions, but to fit both reference skull and muscle surfaces to a new facial skin surface mesh [Fra12]. Coull used surface-oriented free-form deformations (SOFFDs) to fit reference skull and muscle surfaces [Cou05]. While this approach uses complexly shaped muscles that are difficult to manually create, once defined, a skull and complex muscle structure can be fitted to new facial surface in a single process, although the SOFFDs require some manual manipulation.

Medical data such as CT and MRI scans, or anthropometric data can be used to create more anatomically accurate surface meshes and physically based facial models [CLP03, ZHD06]. Software such as AMIRA[2] can be used to assist with the segmentation of such data [SNF05, BJTM08]. While this process still usually requires a lot of manual work, techniques are being proposed to automate the segmentation of facial muscles from medical data [RU12]; however, these are usually only tested on large, thick muscles like the masseter. A reference surface mesh or model can be created from such data, and then deformed to fit a new facial surface [AHTN01, KHYS02], although accurate surgical models benefit from constructing individual models using patient-specific medical data and material parameters [KRG+02, CLP03]. While highly accurate, the creation of models from medical data usually requires a substantial amount of manual work. The Visible Human Dataset[3] contains reference medical data and images that have been used by previous projects when creating reference facial surfaces or models [KGC+96, SNF05, WHHM13].

5.2 Simulation Mesh Creation

Discretising a complex volumetric domain enclosed by a surface mesh can be a complex task. Approaches to create the meshes are also dependent on the required mesh structure (e.g. element shapes). With MS models, simple automatic approaches have been used that just create a layered mesh from a surface mesh [WT93]. However, producing a mesh using more complex volumetric elements that are well shaped for stable and accurate simulations can be much more challenging. Several resources have been continuously referred to while completing this section [Fle00, FG08, RG11].

Mesh generation is an extensive and actively researched topic, and a wide range of techniques have been proposed to generate tetrahedral and hexahedral meshes. Hexahedra are often preferred for FE simulations due to accuracy, performance and stability advantages compared to tetrahedra, although hexahedral mesh generation is a lot more difficult, especially for complex objects. Survey papers are available that discuss in detail the issues with hexehedral mesh generation, and compare different tetrahedral and hexahedral mesh generation approaches [BPMC95, Owe98, Sch00, RS06, BTPB07, LBPH07, WNR09]. As well as being either conforming or non-conforming, a hexahedral mesh can also be either structured or unstructured. Structured meshes have a regular grid structure, whereby all internal nodes connect an equal number of adjacent elements (each internal node connects 8 adjacent elements in the case of a structured hexahedral mesh). On the other hand, unstructured meshes can have any topology.

5.2.1 Tetrahedral Mesh Creation

The advancing-front technique starts at a surface, called the front, and progressively adds elements [PPF+88]. The front is updated by expanding faces of the front into the volume to create elements until the front contains no more faces, in which case the volume has been fully meshed. New points are created inside the mesh according to quality constraints and to generate decent elements,

[2] http://www.vsg3d.com/amira/overview
[3] Part of The Visible Human Project (http://www.nlm.nih.gov/research/visible/visible_human.html).

although points on the fronts are preferred to merge fronts. By generating elements inwards from the boundary surface, the advancing-front algorithm is motivated by the need to conform to the boundary, and generate high-quality elements near the boundary.

As no additional points are inserted on the surface, high-quality surface meshes are required as input, although surface refinement and smoothing processes can be used to help achieve this requirement [ZBX05, BM06, BHP06]. It is also difficult to control local mesh resolution of areas within a volume, which could be required for high stress areas within a mesh, whereas other areas don't usually require as high resolution. On the other hand, elements near the boundary can be easily controlled, as these depend on the surface mesh topology. The order in which the fronts advance often also influences the final mesh. With respect to a physics-based facial model, without using a modified version that handles internal boundaries, it is likely that the advancing-front algorithm would have to be applied several times, once for each muscle and skin layer, and then each of the meshes combined.

Various improvements have been made to the advancing-front technique. Choi et al. introduced a technique to locally refine a tetrahedral mesh based on the advancing-front technique during simulation [CSI04], increasing the number of elements in regions of interest undergoing high strains. Surface refinement regions are determined based on the effective strain rates and minimum interior angles of surface triangles. Such surface regions are then bisected and smoothed before the advancing from algorithm is applied to generate tetrahedra from the locally refined surfaces. This approach therefore only considers local mesh resolution near the surface, and not areas further inside the volume, which is difficult to control using the advancing-front technique.

The constrained Delaunay tetrahedralisation (CDT) algorithm generates a tetrahedral mesh that covers a volume bound by a surface mesh while preserving this boundary. The surface mesh can be an arbitrarily complicated shape and it may contain internal boundaries, which can enclose different regions or holes in the volume. A Delaunay tetrahedralisation (DT) of a set of points is a tetrahedralisation where, for each tetrahedron, there exists a circumsphere such that no point lies inside it. Optimal properties of such tetrahedralisations include that, among all tetrahedralisations of a set of points, they maximise the minimum angle and minimise the maximum circumradii, which therefore helps prevent highly irregular tetrahedra being generated, although the set of points to which the DTs are applied has a large effect on the possible tetrahedralisations that can be generated. To generate a CDT from a surface mesh, points will usually need to be inserted on the surface mesh, or inside the volume, or both.

One program that implements the CDT technique is TetGen[4], which can be run as a stand-alone program or a library for integrating into other applications. It uses the approach proposed by Si and Gärtner to implement CDT [SG05]. TetGen as a stand-alone program supports limited input file formats to define the surface mesh, although the .POLY and .SMESH formats developed for use with TetGen are able to describe complex input domains; for example, they allow faces that belong to a particular boundary to be labelled, node attributes to be specified, and regions to be defined. TetGen then assigns these region, boundary and node attribute values to the relevant elements, faces and nodes in the output files, which, for FE simulations, can be used to help determine where boundary conditions should be applied, and the material properties that should be assigned to elements. Three output files, a .NODE, .ELE and .FACE file, are normally produced as output, defining the nodes, elements and surface polygons respectively.

Various quality control measures can be specified in TetGen, such as a maximum tetrahedral radius-edge ratio (the ratio of the circumsphere to the length of the shortest edge), maximum volume constraints, and local mesh resolutions. A volume can be tetrahedralised using first- or second-order tetrahedral elements. A previously generated mesh can also be refined, and, as TetGen can be used as a library by another program, this feature could potentially be used to adaptively refine a mesh before or even during an FE simulation to simulate a facial model at different levels of detail.

CDTs generally contain few badly shaped tetrahedra, apart from those around small angles on the input surface, although they can contain slivers (very flat tetrahedra), which can also have

[4] http://tetgen.berlios.de/

a small radius-edge ratio, and are therefore difficult to detect and remove. TetGen uses local flip operations in a simplified sliver removal step, but this doesn't guarantee to remove all slivers. Other programs and libraries also exist for generating CDTs, such as the Computational Geometry Algorithms Library (CGAL)[5], a relatively large C++ library containing various functions for geometric computation, and Geompack[6] [Joe12].

Molino et al. proposed a meshing approach that produces simpler tetrahedral meshes than those produced by CDTs [MBTF03]. This makes use of body-centred cubic (BCC) tetrahedral lattices that are built from cubes and divided into tetrahedra. The lattice is then refined to desired resolutions using a signed distance function, which represents the object geometry, and a mesh subdivision algorithm. The refinement technique is able to produce meshes with coarse resolutions at locations where the accuracy of volumetric deformation generally isn't considered too important, such as in the middle of meshes. This mesh is then modified to produce a conforming mesh, and optimisation techniques can be used to relax the positions of the nodes, producing meshes with better tetrahedra. The quality of meshes are suitable for simulations involving large deformations. This meshing technique is available as part of the PhysBAM library, and similar meshing techniques have been used with some detailed FE models of facial and body soft tissue [SNF05, LST09], although most of these have used a BCC lattice as a non-conforming simulation mesh. Lee et al. describe how using a conforming meshing technique can be impractical for physics-based simulations, particularly for complex objects, due to the large number of small elements they generate [LST09].

5.2.2 Unstructured Hexahedral Mesh Creation

A technique called plastering is available for automatic hexahedral mesh generation [Can92], which works in a similar way to the advancing-front technique used for tetrahedral mesh generation. Fronts are defined as collections of faces of an input quadrilateral surface mesh. These fronts are then advanced inward to generate layers of hexahedral elements, and colliding fronts are merged; however, unless the quadrilateral discretisation of opposing fronts is well matched, there are often problems when the fronts collide, and in some cases it is impossible to fill the volume with only hexahedra. Due to the strict constraints that the boundary surface mesh places on the generation of the hexahedral mesh, the unconstrained plastering algorithm has been proposed [SKO+10]. Like plastering, this technique advances fronts inwards from the boundary, but, rather than generating layers of hexahedra, the advanced fronts create voids. A hexahedron is created where three fronts cross, and, when all voids are small and simple enough, they are all meshed with hexahedra. The final void is therefore not overconstrained, although it can be difficult to mesh the voids with high-quality hexahedra.

The whisker weaving algorithm is also based on the advancing-front technique [TBM96]. Whisker weaving attempts to generate the dual of a hexahedral mesh based on the concept of the spatial twist continuum (STC). The STC representation of a hexahedral mesh is the combinatorial dual of a hexahedral mesh, just as the Voronoi diagram is the combinatorial dual of a Delaunay tetrahedralisation. It is represented as a collection of sheet structures (surfaces) between faces on different sides of a quadrilateral surface mesh. The STC is first generated using a surface mesh as input, and this is then used to generate the hexahedral elements. While this approach can successfully generate a hexahedral mesh for most inputs, the resulting hexahedra are often poorly shaped. Various extensions to the whisker-weaving technique have been proposed to improve the quality of the hexahedra, for example, by preprocessing the input surface mesh, and introducing new rules to enhance the construction of the STC [FM99, LW08], although these still have limitations regarding the quality of the generated meshes, and they haven't been tested on a wide range of complex input surfaces.

Indirect hexahedral meshing techniques firstly discretise the domain into tetrahedra, and then convert this into a hexahedral mesh [Mit98, GSZ11]. Tetrahedral meshes can be easily generated

[5] http://www.cgal.org/
[6] http://members.shaw.ca/bjoe/

using the techniques described in Section 5.2.1. Each tetrahedron can then be split into four hexahedra; however, such hexahedra are likely to be badly shaped, even if the tetrahedron isn't too distorted. Alternatively, the H-Morph algorithm combines the tetrahedra to create hexahedra [OS00, NRP11].

Grid-based methods generate a fitted grid of regularly shaped hexahedra inside a volume, and then create distorted hexahedra to fill the gaps at the boundaries [Sch96, She07, ZZM07]. An octree decomposition of the domain can be used to create the hexahedral grid [Mar09, ZHB10], whereby octants are subdivided until a stopping condition is reached (e.g. maximum number of iterations or condition of surface approximation), with octants fully outside the volume being discarded. Hexahedra on the outside of the mesh should then be further processed to create a better approximation of the boundary surface mesh. One method of doing this would be to use the marching cubes algorithm to crop the cubes that cross the surface, which may not necessarily produce hexahedral elements around the surface. Another method would be to use a surface projection algorithm, distorting the cubes that cross the surface. While grid-based methods are robust, they often produce poor-quality hexahedra around the surface, and the generated mesh is dependent on the orientation of the interior grid. The same volumetric mesh resolution is also used throughout the mesh with no option to control the local resolution. This can lead to unnecessary computation during FE simulations around low stress areas of models that could be meshed using fewer elements.

Rather than processing an entire complex domain, decomposition methods can be used to split such a domain into simpler subdomains, meshes for which can be generated separately and then merged [SERB98, SSLS10]. Meshes for the subdomains can be generated using unstructured or structured hexahedral mesh generation approaches, and the merged submeshes typically form an unstructured hexahedral mesh. However, decomposition approaches often require a manual decomposition of the complex domain such that high-quality elements are able to be produced during the merging process, and generating well-shaped elements on the boundaries when merging these submeshes is a complex task. Similarly, an FE analysis can be performed on a model decomposed into several independent subdomains, which are combined to produce the full, complex model [Lam09].

Various programs have been developed for generating hexahedral meshes for FE simulations,[7] such as CUBIT[8], which can discretise surfaces and volumes using various 2D and 3D elements. While many hexahedral mesh generation algorithms suffer problems regarding element quality and robustness, particularly with complex geometries like soft tissue, techniques have been proposed to improve the quality of such meshes [ISS09, Isl11], although these can produce models with an increased number of elements. Quality-improvement techniques focus on the sections of meshes that contain poor-quality hexahedra, such as towards the boundaries of meshes generated using grid-based methods, and they typically involve node smoothing, and optimising the topological connections between hexahedra [SZM12].

5.2.3 Structured Hexahedral Mesh Creation

Rather than unstructured hexahedral meshes, approaches have been developed to generate structured hexahedral meshes that have a regular grid structure. A structured hexahedral mesh is essentially a cuboid divided into regular hexahedra that is warped to fit a desired boundary surface. Mapping techniques create such meshes by finding a mapping function to transform a cuboid grid of hexahedra into the desired geometry [CO82]; however, such techniques are typically only suitable with cube-like geometries. Sweeping techniques can also be used, whereby a quadrilateral surface mesh is swept between topologically equivalent source and target surfaces, producing topologically equivalent intermediate surfaces, and these surfaces are linked to produce hexahedra [Knu98, RS10]. Such techniques have been extended, for example, to sweep from multiple source surfaces [LBW00, SBO06].

[7] A list of mesh generation software can be found at http://www.robertschneiders.de/meshgeneration/software.html

[8] http://cubit.sandia.gov/

Structured hexahedral mesh generation algorithms can usually produce well-shaped elements for simple domains, but the strict boundary surface mesh constraints, such as the required surface mesh geometry and topology, limits their use to a relatively small number of simple geometries. Due to these restrictions, decomposition methods are often required for meshing complex domains with structured mesh generation techniques. For example, the submapping method decomposes a domain into extremely simple cube-like volumes that can be easily meshed using mapping techniques [WLBS95, RGS10], and the multi-sweeping method sweeps multiple quadrilateral surface meshes to multiple target surfaces by applying sweeping techniques to subdomains [WSO04, RGRS09]. However, such approaches also suffer the limitations of decomposition methods regarding the difficult domain decomposition and merging processes.

5.2.4 Hybrid Mesh Creation

Some mesh creation approaches create hybrid meshes using multiple element types, such as tetrahedral and hexahedral elements [YS03, BRM+14]. With the plastering advancing-front algorithm, voids can be meshed with tetrahedra if well-shaped hexahedral elements cannot be generated [MTT98, SKO+10]. Using local subdivision of elements with the regular octree method can also generate hybrid meshes [ISS09]. This can be done within an area bounded by an arbitrary internal surface by reapplying the regular octree method to this surface, subdividing the cubes inside or on the internal surface. However, before applying an algorithm to further process the elements around the internal boundary after the divisions, it is important to convert the mesh to a 1-irregular mesh, which is a mesh where each element face has at most one additional node that isn't part of the element inserted on each edge and one within the face. Meshes with such nodes that aren't part of each element that they connect are not valid for FE simulations. The 1-irregular hexahedra can then be split into other types of elements, such as tetrahedra, to make the mesh suitable for FE simulation.

Other approaches create hybrid meshes during the merging process when using decomposition methods. For example, decomposition methods can be used to generate simple subdomains that can be meshed independently, and tetrahedra can be used where necessary in the merging process to prevent the generation of badly-shaped hexahedra [Lo12]. The Geompack software package uses a decomposition method with sweeping and advancing-front techniques to generate hexahedral-dominant meshes [Joe12].

5.2.5 Non-Conforming Mesh Creation

Rather than creating a mesh whereby the elements conform to a surface mesh, a non-conforming model can be created, to which the surface mesh can be bound. Dick et al. used the regular octree method without processing the boundaries to generate arbitrarily shaped volumetric meshes composed of regular hexahedral elements [DGW11]. A high-resolution surface is then bound to the volumetric representation, so that, as an FE solver deforms the volumetric mesh, the high-resolution surface mesh with a visually continuous appearance is deformed with it.

As well as having a mesh consisting of only regular elements, with this approach, the shape function derivatives are the same for each element, meaning they only have to be stored for one instead of every element. For GPU simulations, this is not only useful due to the more limited memory available on the GPU, but these values could be stored in fast constant memory, or loaded once from slow global memory and cached (on Fermi architectures), rather than requiring many slow global memory loads. The FE simulations by Dick et al. had further speed increases as only the same linear elastic elements were used for the whole model, so the entire stiffness matrix could be precomputed and stored. Kumar et al. performed linear elastic FE simulations using structured non-conforming hexahedral grids, and compared these with conforming hexahedral simulation meshes [KPB08], which produced similar stresses, although only relatively simple models were examined.

A similar meshing approach was used by Cooper to develop a physically based leg model, although the hexahedra were divided up into tetrahedra [Coo98]. As the model included different

components, such as muscle and skin, each component was discretised using the same regular voxel grid, and overlapping hexahedra between two or more components were then assigned to a single component using an erosion process. The hexahedra were then tetrahedralised. Such meshes have similar advantages to those generated with the approach by Dick et al., although different shape functions would be required for each tetrahedron with a different orientation; therefore, using this approach, the hexahedral elements before subdivision would probably be more efficient for FE simulations.

5.3 Computation of Simulation Model Properties

Once a simulation mesh has been created, model properties, such as element material properties, must be set. With non-conforming models, elements may overlap different volumes (e.g. an element of a soft-tissue model might partially overlap a muscle, and partially overlap connective tissue surrounding the muscle), and different model properties may be associated with each such volume. Lee et al. approximated such properties for non-conforming tetrahedral FE soft-tissue models using a sampling procedure [LST09]. As discussed in Section 4.2.4, a variety of techniques have been used to compute muscle fibre directions, including using B-spline surfaces and volumes [WKMMT99, TSB+05], and using elements of simulation meshes with C^1 continuity [RP07, MHSH10].

Boundary conditions must also be set. With soft-tissue models, the skull surface is often used to set such constraints. The skull provides a surface to which muscle origins and ligaments can be attached, and boundary conditions are often set to restrict the movement of nodes on this surface, as discussed in Section 4.1. These nodes may be set as rigid [BJTM08], or their movement may be constrained to simulate the sliding effect between the superficial and deep soft tissue layers, for example, using the penalty method [HWHM11]. However, with non-conforming models, the elements of which don't conform to the skull surface, it is more difficult to determine such nodes. Restricting the movement of such nodes is also more complex; for example, using the penalty method, penalty forces must be computed for nodes that may not initially lie on the surfaces, and may therefore be initially penetrating the skull surface.

With non-conforming models, since nodes of the simulation model and vertices of the surface mesh don't share the same positions, the final step of creating a simulation model is to bind the surface mesh to the simulation model, enabling the surface mesh to be animated during simulation. Weights can be computed to bind each vertex of the surface mesh to an element of the simulation model such that their positions can be updated as the nodal positions change during simulation. Various techniques can be used to compute the weights and update the vertex positions, including trilinear interpolation and extrapolation with hexahedral elements [DGW11], and by using barycentric coordinates of tetrahedral elements.

5.4 Deformation of a Reference Physically Based Model

Some model creation approaches adapt an entire physically based reference model to fit a target facial surface. This includes adapting all surfaces, such as the skull and muscle surfaces, as well as the domain discretisation of the physically based model. Methods have been proposed to deform an MS head model, including a skull, muscles and skin, using sets of manually defined anthropometric landmarks [KHYS02, ZST05a]. Using such techniques, a reference head model can be fitted to even poor scan range data. Kähler et al. also proposed a technique for simulating geometry changes of the head due to growth and ageing according to anthropometric landmarks for different-aged heads [KHYS02], although such changes are mainly useful for creating young head models where such geometry changes are observed.

Alternatively, more accurate patient-specific head models can be created (e.g. for surgical applications) by adapting a reference head model using landmark-based procedures with the aid of CT data of patients [LCC+12]. Hung et al. used cranial and facial landmarks to adapt a

reference head model to fit MRI and surface data of a subject [HWHM14], and a reference head model transformation can also be approximated using only facial landmarks and surface data if MRI data isn't available. Kim et al. used CT data to extract extra-cranial soft tissue, and adapted only a reference muscle structure to fit patient-specific head models using thin plate splines [KJW+10]. While approaches that deform a physically based model are dependent on a specific physically based technique, unlike approaches that only deform the surfaces, generated target models are appropriately discretised and have boundary conditions set ready for simulation.

To assist with the creation of target models from a reference model, techniques have been proposed to adapt reference volumetric meshes to target surface meshes [LLT11]. Couteau et al. proposed such a technique called the mesh-matching algorithm [CPL00]. This approach is therefore useful for generating customised meshes of complex structures, particularly if other mesh generation techniques fail to generate a decent discretisation, and the technique has been used to create patient-specific FE maxillofacial models [LCSP05]. Various types of elements, or even a combination of elements, can be used in the reference mesh, although the same elements and local mesh resolutions are used for each customisation.

5.5 Summary

Physically based simulations require an appropriate physically based model to be created, which consists of a surface mesh (the visual representation of the object to be simulated) linked to a simulation model (the physics-based representation of the object). A facial surface mesh can be created using a variety of techniques, including surface capture techniques, statistical methods, which are used by FaceGen, or by deforming an existing facial surface. A simple skull surface can be created as an offset of the facial surface [TW90, KHS01], and interactive editors have been developed to assist with the creation of muscle surfaces [KHS01, Ain11]. Alternatively, skull and muscle surfaces can be deformed to fit a new facial surface [Cou05, Fra12], or more anatomically accurate surface meshes can be created using medical data such as CT and MRI scans [ZHD06, BJTM08].

To create a volumetric facial simulation model, it is necessary to discretise the volumes enclosed by the facial surface mesh into elements to create a simulation mesh, and then set simulation properties, such as material properties and boundary conditions. Complex tetrahedral simulation meshes with high-quality well-shaped tetrahedral elements can be automatically generated using a variety of techniques [PPF+88, MBTF03], including the constrained Delaunay tetrahedralisation (CDT) algorithm [SG05], which has been implemented by various programs including TetGen. However, hexahedral elements are often preferred due to accuracy, performance and stability advantages. While a variety of techniques have been proposed to create unstructured hexahedral meshes [SKO+10, ZHB10, NRP11], which can have any topology, such techniques often require considerable manual work, or produce poor-quality hexahedral elements, particularly with complex domains. Techniques have been proposed to improve the quality of such hexahedral meshes [ISS09, Isl11], but these typically produce meshes with an increased number of elements. Structured hexahedral meshes, which have a regular grid topology, can be automatically generated with well-shaped elements [CO82, SBO06, RS10], but only for a relatively small number of simple domains.

Rather than processing an entire complex domain, decomposition methods can be used to split a complex domain into simpler subdomains that can be meshed separately, using unstructured or structured meshing techniques, and then merged [SERB98, WSO04]; however, dividing a complex domain, and generating well-shaped elements when merging the submeshes are complex tasks. Some mesh creation approaches produce hybrid meshes, generating well-shaped hexahedra where possible, and meshing the remainder of the domain with tetrahedra [YS03, Lo12]. Alternatively, rather than creating a mesh whereby the elements conform to a surface mesh, a non-conforming tetrahedral [Coo98, LST09] or hexahedral (e.g. voxel-based) mesh [DGW11] can be created, to which the surface mesh can be bound for visualisation purposes. Non-conforming meshes with well-shaped elements can be automatically created for complex domains, and can have various performance advantages during simulations [DGW11].

Once a simulation mesh has been created, simulation model properties must be set. With non-conforming models, a sampling procedure may be used to approximate element material properties [LST09]. Muscle fibre directions can be computed using a variety of techniques described in Section 4.2.4, and boundary conditions can be set such that the movement of nodes on the skull surface may be fixed or restricted. Barycentric coordinates of tetrahedral elements, or trilinear interpolation and extrapolation with hexahedral elements [DGW11] may be used to bind and animate the surface mesh of a non-conforming simulation model. Alternatively, rather than creating a new simulation model, techniques have been developed to adapt an entire physically based reference model to fit a new facial surface [KHYS02, HWHM14], which adapt the domain discretisation of the physically based model (the simulation mesh), and the simulation model properties, as well as the surface mesh.

Chapter 6

Animation Process Overview

Our physically based animation process involves simulating models of facial soft tissue using the FE method to produce animations of gross- and fine-scale movements, such as wrinkles. As shown by Figure 6.1, we define facial soft tissue as the volume of soft tissue between the outer skin surface and the skull. This figure also shows an overview of our entire animation process, which involves three major stages:

1. Creation of a surface mesh for an object
2. Creation of a suitable FE simulation model
3. Simulation and visualisation of the model over time

The surface mesh can be created using various techniques, for example, from simply using 3D modelling tools to sculpt a mesh, to segmenting medical data.

For Stage 2, our model creation process is used to automatically discretise the volumes enclosed by the surface mesh into a collection of nodes that are connected to form elements, and compute FE model parameters to produce a simulation model [WM13a, WM13d]. We use non-conforming hexahedral (voxel-based) models, and the surface meshes are bound to and animated using these models as they are simulated using the FE method. The final stage of the animation process involves simulating and visualising the models using our GPU-based FE simulation and visualisation system [WM13c, WM13d, WM13b, WM14].

This chapter introduces our animation process, and discusses the creation of surface meshes, while the following two chapters present the details of our model creation process, and the theory and GPU-based implementation of our simulation and visualisation process. The forehead model and animation shown in Figure 6.1 will be used as a detailed example while describing each of these three stages. While the outer skin surface covers the entire face, only the forehead region that we are focussing on is modelled and simulated.

It should be noted that our model creation and simulation processes can be used independently, and therefore have much wider applications than for use within our animation process. For example, provided the data for the simulation model is stored in the necessary format, our model creation process could be used to create a model for use with another FE solver (e.g. that uses a different formulation of the FE method), whereas models created using an alternative creation technique (e.g. that computes element properties differently) could be used with our simulation process. However, if an alternative model creation process was used, this may impose different requirements on the input surface mesh, which is highly coupled to the model creation process.

6.1 Overview of the Models and Simulations

Our animation process involves using the FE method to simulate facial soft tissue. The FE method is more accurate and physically sound than other commonly used physically based techniques like

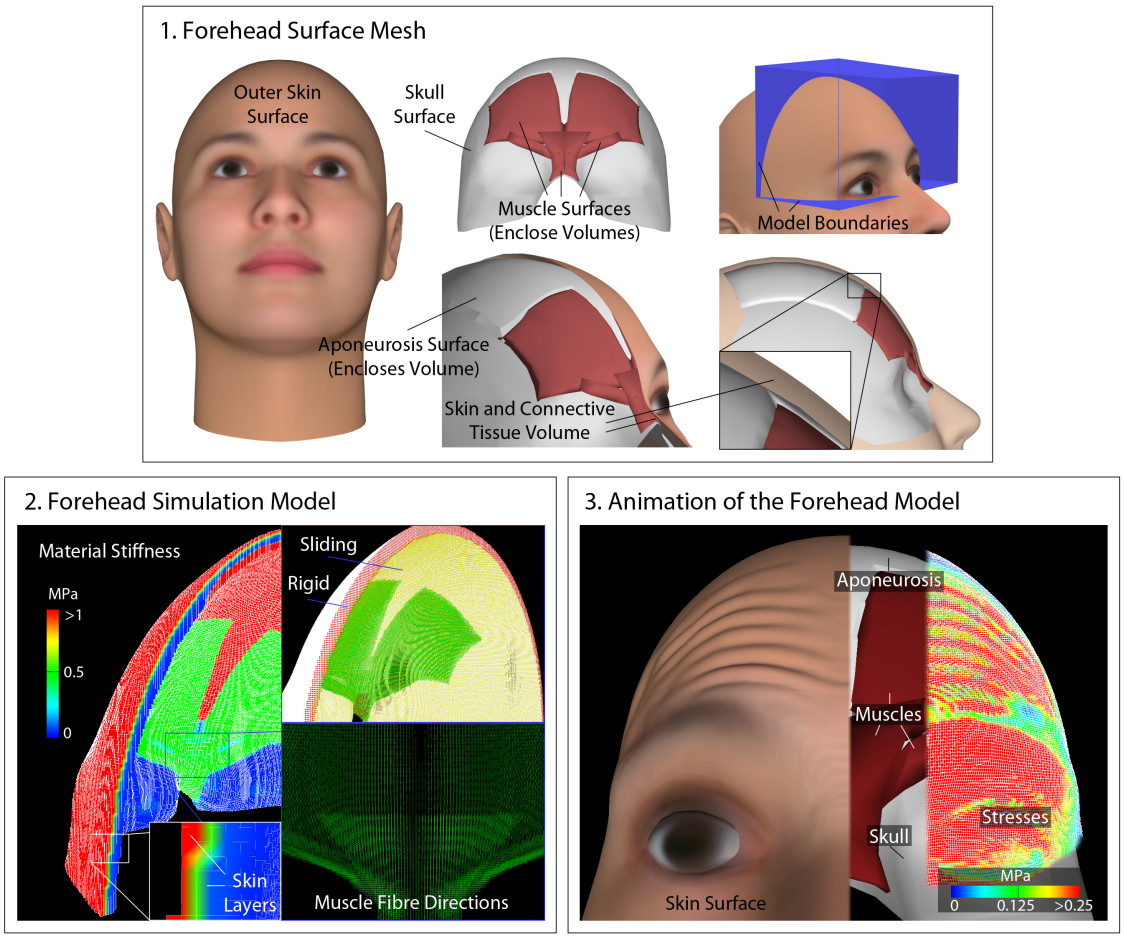

Figure 6.1: A forehead model and animation that will be used as a detailed example while explaining the stages of our animation process.

the MS method and physics engines, and is therefore suited to producing realistic-looking animations of complex soft-tissue behaviour, including wrinkling, in a fully physically based manner. Specifically, the nonlinear TLED formulation of the FE method is used. The TL formulation enables some simulation values to be precomputed, increasing the computational performance of simulations. Explicit time-integration schemes require small but efficient timesteps, and are inherently parallel, making them suited to simulating the complex nonlinear behaviour of soft tissue under large deformations on the GPU. Being dynamic, inertial and damping effects are also considered.

The elements used with our FE models are reduced-integration linear hexahedral Lagrange elements. Compared with tetrahedral elements, hexahedral elements are generally more robust and accurate [WNR09, TCC+09], suffering much less from volume locking when simulating incompressible material, such as soft tissue. They can also be more efficient since fewer elements are often required to create a model. Linear Lagrange elements are suitable for high-resolution models that capture details such as thin skin layers with different material properties, which would be infeasible to model using more complex Hermite or nonlinear Lagrange elements. Using reduced-integration elements overcomes the volume-locking limitations that can occur when simulating incompressible material, while greatly improving computational efficiency compared with fully integrated elements [TCO08].

Our FE models are non-conforming hexahedral (voxel-based) models. Compared with conforming hexahedral simulation meshes, non-conforming voxel-based meshes can be easily and automatically created for arbitrarily complex closed surface meshes using a voxelisation algorithm, such as that described in Section 7.2. Furthermore, voxel-based models have performance and stability advantages, and memory advantages when used with the TL FE formulation, over conforming hexahedral models due to the identical shape and orientation of the well-shaped cube or cuboid undeformed elements [WM12].

With our soft-tissue models, skull or bone is modelled using rigid surfaces, while skin, connective tissue and muscles are modelled using elements and simulated using the FE method, with wrinkles being produced automatically as a result of the simulated physically based behaviour. While our model creation process can create models containing any number of constant-thickness skin layers with overlapping interfaces, most of the soft-tissue models presented in this thesis contain three skin layers (the stratum corneum, dermis and hypodermis) with distinct non-overlapping interfaces, which is necessary and sufficient to accurately simulate most skin wrinkling properties [MTKL+02, FM08]. The assumption of using constant-thickness skin layers has been validated [BJM11].

Our forehead models contain the frontalis, procerus and corrugator supercilii muscles, which are the main muscles involved in producing expressions on the forehead. Muscles are represented using popular and well-understood Hill-type muscle models, considering active and transversely isotropic passive muscle stresses, and active muscles are contracted along fibre fields. Since, in reality, superficial soft-tissue layers can slide over the deeper layers and skull [WMSH10], we simulate this effect by applying boundary conditions to nodes that approximate the skull surface. All of our models use hyperelastic neo-Hookean materials due to their simplicity for predicting materially nonlinear behaviour under large deformations.

6.2 Animation System Structure

Figure 6.2 shows the main components of our animation process implementation, displaying the input and output of, and the relationship between the different components. Two main components have been implemented: the model creation system, and the simulation and visualisation system, corresponding to Stages 2 and 3 of the animation process respectively, while the surface mesh created in Stage 1 is used as input to the model creation system. Other input to the model creation system includes NURBS surfaces for muscles, which are required for computing muscle fibre directions, and a user-defined XML file containing links to the surface meshes, and all of the model creation parameters, such as the voxel size to use. The model creation system outputs

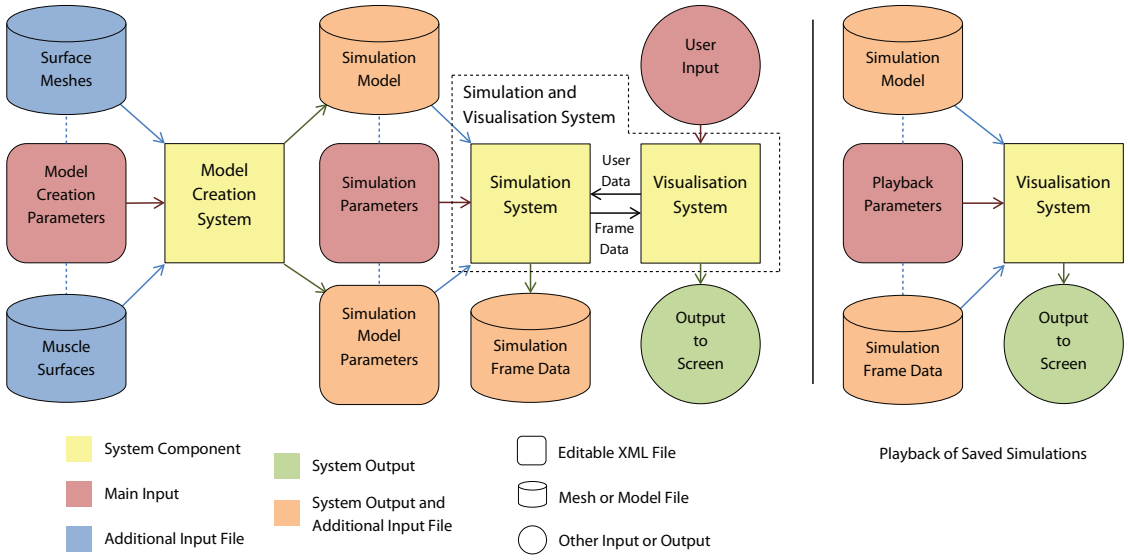

Figure 6.2: The main components of our animation process implementation, showing an overview of the input and output for these components.

two files: a binary file defining fixed simulation model data, such as its geometry, and an XML file defining variable simulation model parameters. This makes the overall system more flexible, enabling various properties to be altered, such as material properties for animating different-aged skin, without recreating the simulation model.

The user-defined input XML file for the simulation system contains links to the two simulation model files, as well as additional simulation parameters, including a simulation script. The script defines changes to muscle contraction parameters, and the timings of these changes. While such low-level control is sufficient for this project, whereby the majority of the simulations involve contracting only a few muscles to raise the eyebrows and producing horizontal forehead wrinkles, the script would be difficult and tedious to define for complex expressions requiring many contracting muscles, such as expressions involving the mouth area. However, this could be easily extended to include higher-level controls, for example, to produce an entire expression, which would automatically contract the relevant muscles at the correct times. Such an approach is used to produce expressions with various other physically based facial animation systems [TW90, ZPS04, Fra12].

The simulation system outputs frames of animation to screen as they are simulated, enabling user interaction with the animations, although this is typically only suitable with simple simulations that can be performed in real time. Simulation frame data can also be stored to a file, and simple playback component has been implemented to read this data, and playback the simulation. Splitting FE analyses into separate preprocessing, computation, and postprocessing stages is typically how they are are performed by current FE systems, such as the popular ABAQUS and CMISS packages, and with solvers developed in science and engineering research for complex simulations of detailed soft-tissue models [ZHD06, TCO08, LGK11]. This way, complex simulations with high-resolution models (which are unable to be simulated in real time) can be viewed and analysed in real time.

6.3 Surface Mesh Definition

Many physically based soft-tissue animation approaches use multiple disjoint surfaces, such as a facial surface, and a surface for each muscle [SNF05, BJTM08, WHHM13]. This is also the case with our approach, whereby the surface mesh can contain various surfaces, including internal surfaces that are located inside a volume enclosed by other surfaces. For the mesh creation process,

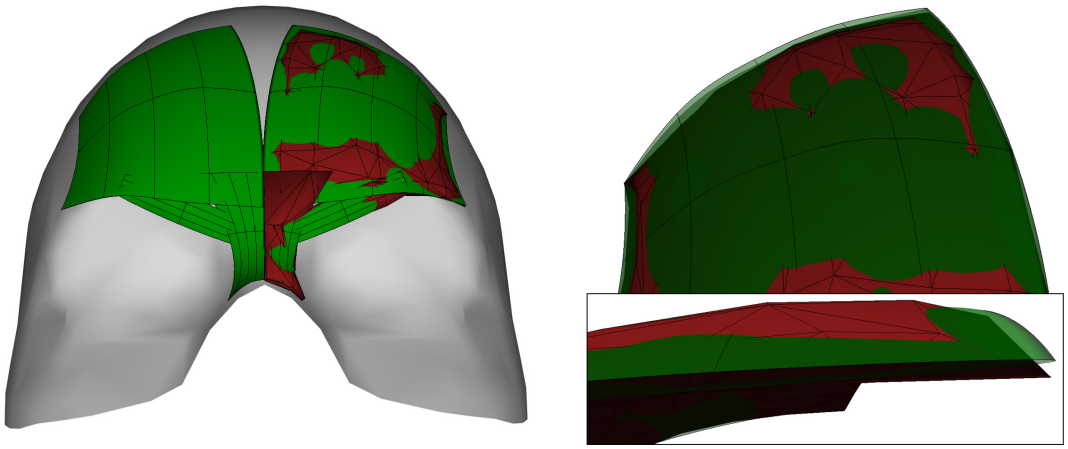

Figure 6.3: NURBS surface approximations (green) of muscles (red) for the forehead model. A close up of the right frontalis muscle and its approximating NURBS surface is displayed on the right, with the insertion end shown in the inset, where an open end of the thin tube-like structure modelled by the NURBS surface can be seen.

volumes must be defined by organising these surfaces into closed collections of surfaces without any holes. With a facial surface mesh (see Figure 6.1), the following surfaces may be defined:

- A facial surface (the outer skin surface)
- A skull surface
- A volume-enclosing surface for each muscle
- A volume-enclosing surface for every additional passive structure that is to be included in the model (e.g. tendons)

Surfaces for individual skin layers are not required, as these are automatically created from the outer skin surface during simulation model creation.

As shown by Figure 6.1, the following volumes can be defined from the facial surfaces:

- A skin and connective tissue volume (between the outer skin and skull surfaces)
- A volume for each muscle
- A volume for each additional structure

Internal volumes, such as muscles and additional structures inside the skin volume, overlap the volume they are contained within (i.e. the skin volume doesn't contain holes for the muscles), simplifying surface mesh creation. Volumes can also overlap, for example, to represent the blend of fibres between connecting muscles. For a standard soft-tissue model, the volumes are known, and can be automatically defined from appropriately labelled surfaces.

In reality, the skull contains gaps and holes, such as between the mandible and maxilla around the teeth, and foramina, which enable structures such as arteries and nerves to pass through, as well as large openings for the eyes and nose. With our model creation process, the skull surface actually represents the inner boundary of the watertight volume of skin and connective tissue that is to be modelled, and all such holes must therefore be closed. While this constraint doesn't have a large effect on the relatively hole-free forehead region of the skull, for a more complex model (e.g. a full facial model), a simple watertight surface could be used to define the soft-tissue boundary for mesh creation, while a more complex realistic skull surface could be used during simulation and visualisation.

A NURBS surface approximation of each muscle is also required for the computation of muscle fibre directions. Figure 6.3 shows an example of such surfaces for forehead muscles. These surfaces must be closed along one parametric dimension, while the other dimension is aligned with the muscle length, producing tube-like structures with open ends. The NURBS surfaces aren't required to exactly match the shapes of the muscles, meaning simpler approximations with fewer control points can be used, which also reduces the amount of computation required with the model creation process.

6.4 Surface Mesh Creation

6.4.1 Creation of the Surface Mesh for the Forehead Model

As discussed in Section 5.1, various techniques can be used to create the surface mesh for a physically based facial model, and many techniques can also be used to create a surface mesh for our animation approach provided suitable surfaces are produced (i.e. surfaces that together define hole-free volumes). For our example model in Figure 6.1, we used FaceGen and 3D modelling tools to create the surfaces. FaceGen was used to automatically generate the facial surface, and this was then edited using Autodesk 3ds Max[1] to smooth over openings around the eyes, producing a watertight surface. The forehead region was also tessellated to a sufficiently high resolution, which is necessary for the animation of smooth forehead wrinkles.

Using 3ds Max, a skull surface was then modelled for the forehead region as an offset of the facial surface. While techniques are available to fit a more realistic skull surface to a facial surface [KHS03, ZST05a, Ain09], a simple facial offset reasonably approximates the skull for this region. Constant soft-tissue thickness across the forehead region was therefore assumed, which has been shown to have little effect on simulations of this region [BJM11]. As the skull is simply an offset of the facial surface, this could have been implicitly computed during the model creation process, rather than being explicitly modelled. However, as the skull in the forehead region is a simple rigid surface that only has an indirect influence on the elements towards the outer skin surface, the skull surface was optimised to produce a much lower-resolution surface for model creation and simulation efficiency. This surface contains only 735 polygons, compared to over 21,000 polygons on the forehead region of the facial surface. A skull surface can also greatly assist with the modelling of muscle surfaces. A skull for the full facial surface wasn't required, as we only model and simulate the forehead region, defined by a bounding box.

Surfaces for the muscles were manually modelled using 3ds Max based on anatomy texts and diagrams [Kni01, Sta09, WN12]. The frontalis, procerus and corrugator supercilii muscles were modelled, which are the main muscles involved in producing expressions on the forehead. The surfaces were created by cutting out shapes from duplicated skull surfaces, extruding these to create a watertight volume, and then smoothing the sharp edges to create a smoother blend between surrounding structures and connective tissue. Using this approach, the exact thickness of the structures could be controlled by simply extruding the surfaces by a fixed amount. The regions of overlap between structures in our surface mesh, such as between the frontalis and procerus muscles, represents the smooth interweaving of fibres.

The galea aponeurotica was also modelled using the same approach, and was included to model the anchoring effect on the frontalis[2]. In reality, as this muscle has no skull attachment, it pulls the scalp forward slightly when it contracts as it pulls on the galea aponeurotica [Gol92]. Therefore, rather than simply contracting the frontalis to a fixed attachment area like with many current physically based facial animation approaches [SNF05, Fra12, WHHM13], the inclusion of the galea aponeurotica in our model enables this effect to be simulated, with soft-tissue movement towards the top of the head being restricted, but not fixed, as the frontalis contracts. There is also no additional performance cost of simulating this effect with our models, as it only leads to altered

[1] http://www.autodesk.co.uk/products/3ds-max/overview

[2] Some information on the frontalis and galea aponeurotica was obtained from meetings with Dr Adrian Jowett (The School of Clinical Dentistry, The University of Sheffield)

material properties of elements that are already included in the model (otherwise, rather than modelling aponeurosis tissue, these elements would model connective tissue).

The muscle volumes conform exactly to the shape of the skull surface. Aina similarly created muscle volumes that conform to another surface [AZ10, Ain11], although an intermediate SMAS surface was used, rather than the actual skull surface. In reality, while some facial muscles conform closely to the shape of the skull, such as the frontalis, muscle fibres normally originate from the skull or deep soft-tissue layers, and insert into skin layers such as the dermis [Sta09]. Muscles also form a layered structure, for example, with the corrugator supercilii being located deep to the frontalis. Further anatomical accuracy could therefore be achieved by considering such details, which can't be captured by modelling muscle volumes that conform to a single surface, although this would require additional modelling skill.

Finally, the NURBS surface approximations of the muscles were manually created by manipulating the control points of surfaces using Rhino 3D[3]. Figure 6.3 shows these surfaces. The simple approximations could be created relatively quickly using few control points (e.g. for the frontalis, 8 control points were used along the u dimension to define the closed cross-sectional profiles, while 4 control points were used along the v dimension over the length of the muscle). It should be noted that symmetry was assumed along the midsagittal plane when creating our surface mesh; therefore, only half of the surface mesh was modelled, and this was then mirrored to produce the other half.

6.4.2 Alternative Surface Mesh Creation Approaches

For a complex facial model, manually creating and tweaking all the surfaces can be a time-consuming task. This could be made easier with a semi-automatic approach of creating muscles to simplify muscle definition, for example, by having an interactive editor. Various current facial animation approaches use such an approach [KHS01, ZPS04, Cou05], some of which can create a complex layered and interwoven musculature [Fra12] with complex fibre fields [NTHF02]. However, a semi-automatic muscle creation approach would probably impact on flexibility and user control; for example, current such approaches usually only handle simple muscle shapes (e.g. constructed from ellipsoidal or hexahedral segments).

Alternatively, rather than manually defining the surfaces, medical data (e.g. CT and MRI data) could be used to create more anatomically accurate surfaces. To create a model using medical data, it would first be necessary to segment the data to create the necessary surfaces for each structure (including the skull, skin, muscles and tendons). While this can be assisted using segmentation tools like AMIRA, the segmentation process still usually involves a lot of difficult manual work [SNF05, BJTM08, WHHM13]. The surfaces would also need to be postprocessed, for example, to ensure they don't contain any holes, which is a requirement of our model creation process. NURBS surface approximations of the muscles would still be required, although these are relatively simple to manually create.

Therefore, a variety of techniques can be used to create a surface mesh for use with our animation system, each of which requires a different amount of manual work. Provided the surfaces meet the requirements specified in Section 6.3, it doesn't matter whether these are simple surfaces manually created using a 3D modelling tool, or anatomically accurate surfaces created from medical data. Once a single reference surface mesh has been created, either manually or from medical data, existing approaches could be used to deform the surfaces of this, such as the muscles, tendons and skull, for easy creation of a new surface mesh for a different face (e.g. based on range scan data) [Cou05, Ain09, Fra12].

6.5 Summary

Our physically based animation process involves simulation of FE facial soft-tissue models. We simulate non-conforming hexahedral (voxel-based) models with reduced-integration linear hexa-

[3] http://www.rhino3d.com/

hedral Lagrange elements using the TLED FE method, and most of our soft-tissue models contain three constant-thickness skin layers. The animation process requires three main stages: creation of a surface mesh, creation of a simulation model from the surface mesh, and simulation and visualisation of this model over time. While the model creation and simulation processes form part of our overall animation technique, they can also potentially be used independently. The implementation of this process contains two main components: a model creation system, which generates an FE model that can be animated using the simulation and visualisation system. The surface mesh is required as input to the model creation process, and the requirements and creation of the surface mesh were presented in this chapter.

The surface mesh can contain various surfaces, including internal surfaces, that are organised into closed collections of surfaces to define volumes. For example, our forehead surface mesh contains a skin and connective tissue volume (between the outer skin and skull surfaces), and a volume enclosed by each muscle and aponeurosis surface. Various approaches could be used to create a surface mesh, from manually sculpting the surfaces, to creating them from accurate medical data. However, for our forehead model, we used an outer skin surface of a face that was automatically generated, and created the remainder of the surfaces manually based on anatomy using 3D modelling tools (Autodesk 3ds Max and Rhino 3D). The next chapter examines our model creation process, which uses such surface meshes to generate animatable FE models.

Chapter 7

Model Creation

Our model creation process automatically creates animatable non-conforming hexahedral (voxel-based) FE simulation models of facial soft tissue from a surface mesh. All model properties are automatically computed, including skin layers and material properties, muscle properties (e.g. fibre directions), and boundary conditions (e.g. constrained nodes), making them immediately ready for simulation. Compared to existing models that have been used to simulate soft tissue [SNF05, BJTM08, WHHM13], more detailed models are generated by our approach, which, for example, include skin layers, and are able to simulate complex gross- and fine-scale behaviour, including wrinkling. The models are optimised for GPU simulation, as discussed in Chapter 8. Our process can also create any multi-layered FE model from any surface mesh that contains hole-free volumes (not just soft-tissue models). Details of our creation process have been previously published [WM13a, WM13d].

7.1 Model Creation Overview

Our model creation process involves five main stages:

1. Voxelising the surface mesh
2. Computing skin layers and element material properties
3. Computing element muscle properties, such as fibre directions
4. Computing boundary conditions, such as constrained nodes
5. Binding the surface mesh to the simulation model

A surface mesh is required as input, which, as discussed in Section 6.3, may contain multiple surfaces enclosing multiple volumes. For example, with a facial mesh, there may be a volume for skin and connective tissue (between the skin and skull surfaces), and a volume enclosed by each muscle surface. For voxelisation, all mesh volumes are grouped into a number of user-defined levels, where level 0 is the highest level. Semantically, volumes in a lower level are contained within, and bound by, volumes in the level immediately above. For example, as shown by Figure 7.1, level 0 might consist solely of a skin and connective tissue volume, whereas level 1 might consist of the muscle volumes, which are contained within the skin volume. For a standard soft-tissue model, as well as the volumes, the levels are also known, and can therefore be automatically defined. Other input consists of properties (such as material and muscle properties) associated with each volume, and model properties (such as voxel size).

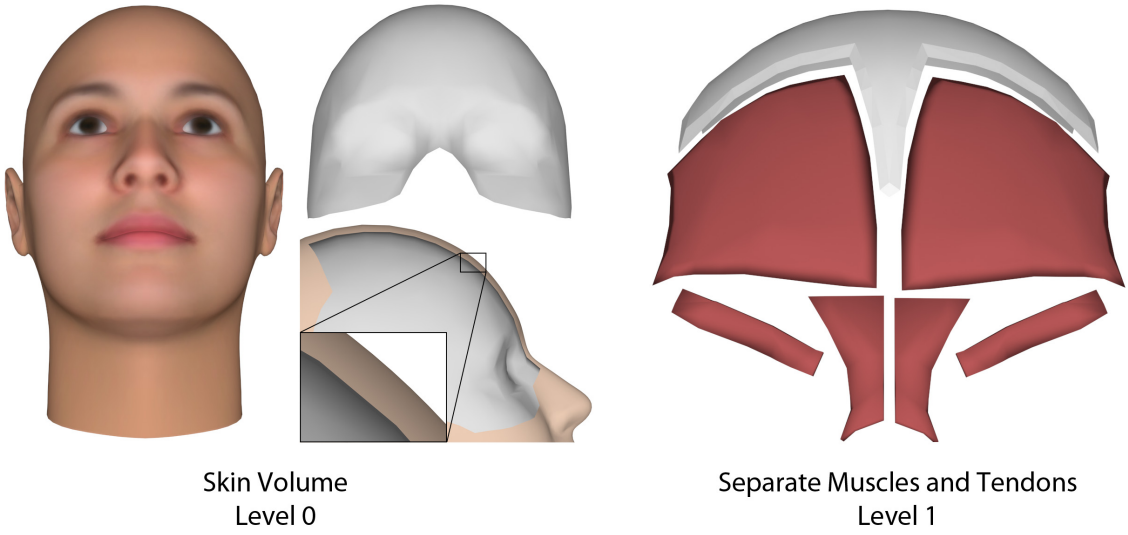

Figure 7.1: The mesh volumes of the forehead surface mesh, grouped into levels.

7.2 Voxelising the Surface Mesh

The voxelisation process starts with a grid of identically sized voxels that overlaps either the entire surface mesh, or a user-specified bounding box if a model is to be created for only part of the surface mesh, such as the forehead region of our facial surface mesh (refer to Figure 6.1). The voxels can be regularly (cubes) or irregularly shaped (cuboids), where irregularly shaped voxels can be useful performance-wise, for example, when modelling thin objects, although these more distorted elements can have an impact on simulation stability and accuracy. The aim of the voxelisation process is to compute the proportions of overlap between voxels and mesh volumes, which can then be used to compute the properties of voxels. Enclosed voxels with more than a user-defined proportion of overall overlap (with the union of all level 0 volumes) are used as hexahedral elements.

As described by Algorithm 7.1, the level-based procedure iterates over each level, considering each level mesh volume in turn during an iteration. A set of mesh volumes is associated with sections of voxels depending on the volumes that overlap them, and such associations with sections or subsections may then be overwritten or added to as further mesh volumes are considered. As shown by Figure 7.2, starting at level 0, mesh volumes in this level are associated with the sections of voxels they overlap. By iteratively considering the next level down, the associations with sections that are overlapped by a mesh volume in both the current and previous levels are overwritten; hence, the associations with sections overlapped by the skin volume that are also overlapped a muscle volume would be overwritten, and the overlapping muscle volume would be associated with such sections.

When multiple mesh volumes in the same level overlap a voxel section, the set of mesh volumes associated with this section contains each such volume. In such cases, the association of a mesh volume with the section is weighted depending on the number of volumes associated with the section, representing the blend between materials. Finally, after iterating over each level, the sizes of voxel sections, and the mesh volumes associated with them are used to compute the proportions of overlap, $o_{e,m}$, between a voxel, e, and overlapping mesh volumes, m:

$$o_{e,m} = \sum_{s \in S_m} \frac{V_s}{n_s V_e} \tag{7.1}$$

where S_m is the set of sections of e with which m is associated, n_s is the number of mesh volumes associated with s, V_s is the volume of s, and V_e is the volume of e overlapped by a level 0 volume.

```
1  Generate grid of voxels, G;
   // Identify mesh volumes associated with voxel sections
2  U_p ← U (universal set);
3  foreach level, L do
4  |  U_c ← ∅;
5  |  foreach mesh volume, M_l ∈ L do
6  |  |  foreach voxel, E ∈ G do
7  |  |  |  E_m ← E ∩ U_p ∩ M_l;
   |  |  |  // Overlapping previous level volume
8  |  |  |  E_1 ← E_m/(E ∩ U_c);
9  |  |  |  Clear associations with E_1;
10 |  |  |  Associate M_l with E_1;
   |  |  |  // Overlapping current level volume
11 |  |  |  foreach mesh volume, M_u ∈ U_c do
12 |  |  |  |  E_2 ← E_m ∩ M_u;
13 |  |  |  |  foreach subsection of E_2, E_s do
14 |  |  |  |  |  A_s ← mesh volumes associated with E_s;
15 |  |  |  |  |  Associate A_s ∪ M_l with E_s;
16 |  |  U_c ← U_c ∪ M_l;
17 |  U_p ← ⋃ M ∈ L;
   // Compute proportions of overlap
18 foreach voxel, E do
19 |  foreach voxel section, S ⊆ E do
20 |  |  Compute volume of S;
21 |  foreach mesh volume, M do
22 |  |  Compute proportion of overlap (see Equation 7.1);
```

Algorithm 7.1: A generic algorithm for the level-based voxelisation process to compute the proportions of overlap between voxels and mesh volumes. Note that multiple subsections can occur within E_2 when this section overlaps multiple existing sections that are associated with different mesh volumes.

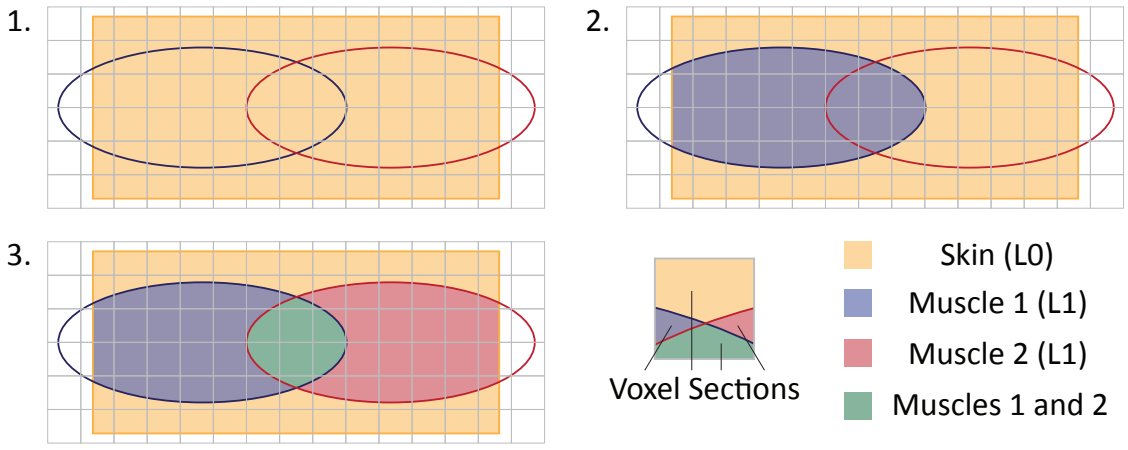

Figure 7.2: An example of the level-based voxelisation process for a soft-tissue block containing two muscles, showing the model state after each mesh volume has been considered in turn. Sections are also identified for a voxel that is overlapped by multiple mesh volumes. Note this is a 2D illustration of a 3D process.

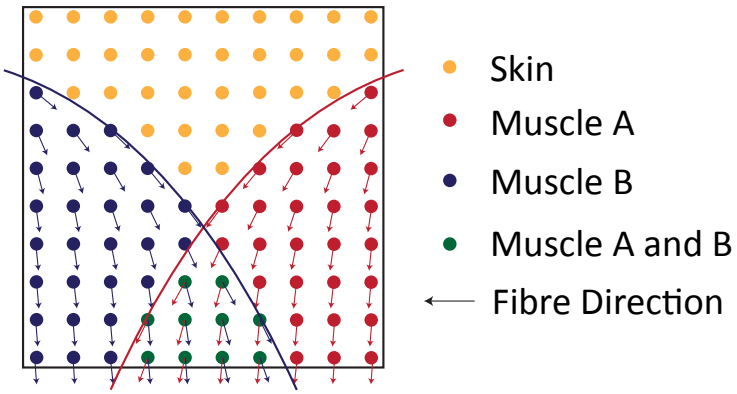

Figure 7.3: Element samples assigned material and muscle properties, which are used to calculate the overall element properties. Note this is a 2D illustration of a 3D process.

V_e is used rather than the entire volume of the voxel to normalise the proportions of overlap.

From Figure 7.2, it can be seen that only lenient requirements are imposed on the creation of the surface mesh; for example, as muscles should be contained within the skin volume, parts of muscle surfaces that cross the bounds of this volume are appropriately ignored. Muscle surfaces can therefore simply penetrate the skull, rather than having to attach and conform to the skull surface. This simplifies the modelling of surface meshes by enabling less accurate surfaces to be used without affecting the simulation model.

Similar to some existing approaches for approximating proportions of overlap between element and mesh volumes [LST09], we use a sampling procedure to sample voxels (see Figure 7.3). Our level-based voxelisation process is performed by testing whether point samples are inside mesh volumes to assign and overwrite the volumes associated with the samples, where groups of samples can be considered as approximating sections of a voxel. The generic voxelisation process defined by Algorithm 7.1 can be rewritten to take into account of this sampling procedure (see Algorithm 7.2). Like with the mesh volumes associated with voxel sections in the generic algorithm, the association of a mesh volume with a sample is weighted depending on the number of volumes associated with the sample. Equation 7.1 to compute the proportions of overlap, $o_{e,m}$, between a voxel, e, and overlapping mesh volumes, m, can be rewritten as:

$$o_{e,m} = \frac{1}{n_e} \sum_{s \in S_m} \frac{1}{n_s} \tag{7.2}$$

where S_m is the set of samples of e with which m is associated, n_s is the number of mesh volumes associated with s, and n_e is the number of samples in the element that are enclosed by a level 0 volume.

Ray-surface intersection tests determine whether a sample is inside a mesh volume. By uniformly sampling voxels (rather than, for example, randomly scattered samples [LST09]), a single ray can pass through many aligned samples. By knowing whether the sample at the start of the ray is inside the mesh volume, and the surface intersection points between the ray and mesh volume (i.e. the points where the ray enters and leaves the volume), it is possible to efficiently test each sample along an entire line using a single ray, rather than a ray for each sample, as shown by Figure 7.4. The result of the voxelisation stage for our forehead model is shown by Figures 7.5 and 7.6, where the coloured elements visualise the proportions of overlap between elements and mesh volumes.

In practice, to avoid using large amounts of memory due to the potentially huge number of samples when generating a high-resolution model, we perform the level-based voxelisation process in stages on subsections of the model. After computing the dimensions of the initial grid of voxels, the voxelisation process is performed on each row of voxels in the direction of a single axis, and

```
1  Generate grid of voxels, G;
   // Identify mesh volumes associated with samples
2  U_p ← U (universal set);
3  foreach level, L do
4  |  U_c ← ⋃ M ∈ L;
5  |  foreach mesh volume, M_l ∈ L do
6  |  |  foreach voxel, E ∈ G do
7  |  |  |  foreach voxel sample, s do
8  |  |  |  |  if s inside M_l then
9  |  |  |  |  |  A_s ← mesh volumes associated with s;
   |  |  |  |  |  // Inside previous level volume
10 |  |  |  |  |  if A_s ∩ U_p ≠ ∅ then
11 |  |  |  |  |  |  Clear associations with s;
12 |  |  |  |  |  |  Associate M_l with s;
   |  |  |  |  |  // Inside another current level volume
13 |  |  |  |  |  else if A_s ∩ U_c ≠ ∅ then
14 |  |  |  |  |  |  Associate A_s ∪ M_l with s;
15 |  |  U_c ← U_c ∪ M_l;
16 |  U_p ← ⋃ M ∈ L;
   // Compute proportions of overlap
17 foreach voxel, E do
18 |  foreach mesh volume, M do
19 |  |  Compute proportion of overlap (see Equation 7.2);
```

Algorithm 7.2: The level-based voxelisation process, using a sampling procedure to approximate the proportions of overlap between voxels and mesh volumes.

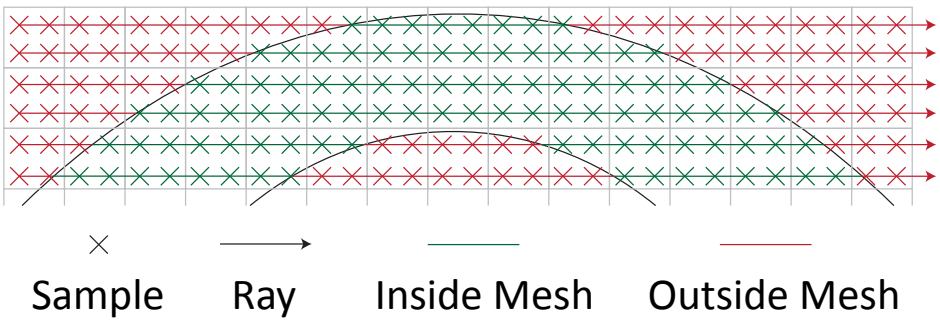

Figure 7.4: Efficient computation of whether samples are inside a mesh volume using few ray-surface intersections. Note this is a 2D illustration of a 3D process.

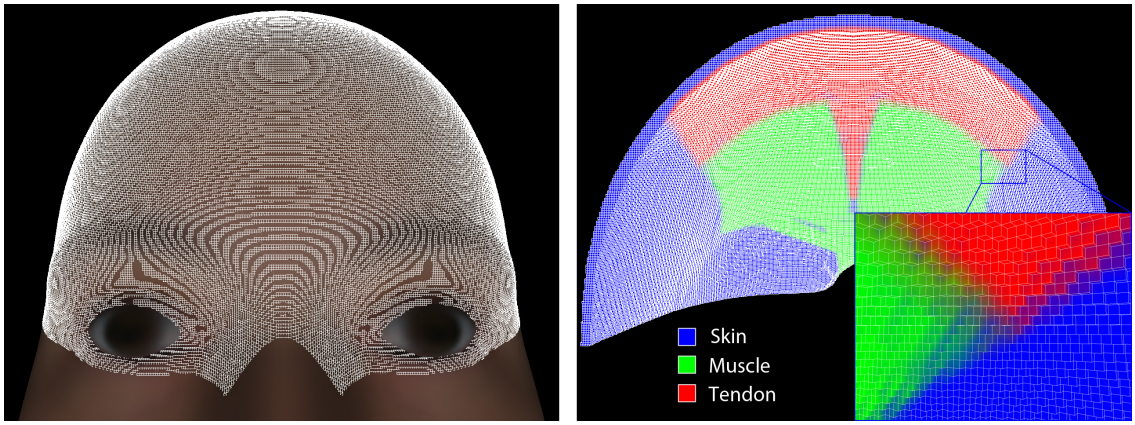

Figure 7.5: The result of the voxelisation stage for the forehead model, with elements colour coded to visualise the proportions of overlap between elements and mesh volumes.

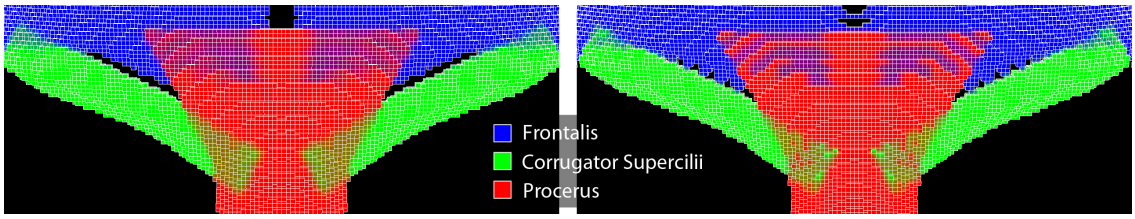

Figure 7.6: The proportions of overlap between elements and muscles of the forehead model are shown by the image on the left. As a comparison, more jagged boundaries between muscles are produced when only a single material is assigned to a voxel (right), rather than a smooth blend between muscles (left).

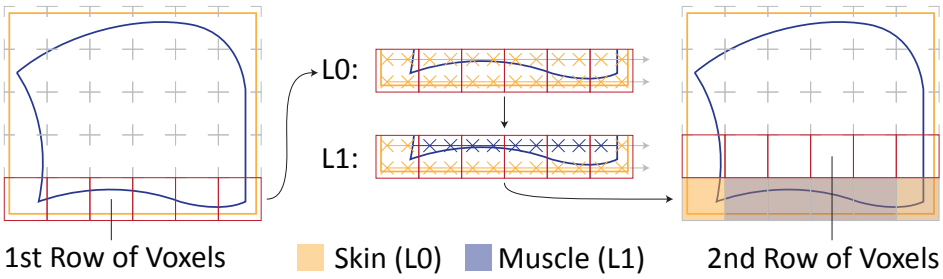

Figure 7.7: Voxelisation of a model in stages by executing the voxelisation process independently on each row of voxels (in the direction of the rays for the ray-surface intersection tests). Note this is a 2D illustration of a 3D process.

the rays for the ray-surface intersection tests are aligned with this axis, as shown by Figure 7.7. For example, rows of voxels in the direction of the x axis, and therefore perpendicular to the y-z plane, might be independently processed, in which case the rays would also be aligned with the x axis. The complete voxelisation process is then performed for all voxels in a row, and the sample data for such voxels is used to compute the properties of voxels in the row that are included in the model (sufficiently inside a level 0 volume), before the next row of voxels is processed. Sample data is therefore only required for the current row of voxels, rather than every voxel in the initial grid. The model could be subdivided further, although, if a row of voxels is subdivided, multiple overlapping rays would be required to test samples along a single line, reducing the efficiency of these tests.

7.3 Computing Skin Layers and Element Material Properties

Voxel element properties, including material properties, are calculated based on the proportions of overlap between the voxels and mesh volumes. For a particular element, the material properties associated with mesh volumes that overlap the element are weighted by their proportions of overlap, $o_{e,m}$ (calculated using Equation 7.2), to compute the overall element material properties, P_e:

$$P_e = \sum_{m \in M} o_{e,m} P_m \tag{7.3}$$

where M is the set of all mesh volumes, and P_m are the material properties of m. As shown by Figure 7.6, this blending of properties produces a smooth transition between the non-conforming volumes, rather than jagged boundaries, for example, to model the smooth interweaving of muscle fibres.

In practice, the simulation model file output from the creation process stores the proportions of overlap between elements and mesh volumes, and the combined properties are then computed during the precomputation stage of the simulation process, which takes the simulation model file and a model parameter file as input. This enables variable properties, such as material properties associated with mesh volumes, or stress references of muscles, to be easily changed without recreating the model.

Within the skin volume, constant-thickness layers with different material properties can be generated. Modelling skin layers is necessary to simulate fine details like wrinkles [FM08], and the assumption of using constant-thickness skin layers has been validated [BJM11]. By defining a start depth and thickness for each layer, these can also contain overlapping regions, for example, to help capture the non-uniform blend between real skin layers. The aim of this process is essentially to determine the proportions of overlap between voxels overlapped by the skin volume, and the generated skin layers, which is done using the sampling procedure during voxelisation. To determine which layers each sample inside the skin volume is contained within, the distances between the samples and the outer skin surface are tested. If a sample is inside a skin layer, it is no longer associated with the skin volume, but with the skin layer it is contained within.

However, the thin outer epidermal layers are unlikely to be captured using such an approach without using an extremely high element and sampling resolution, as shown by Figure 7.8, and these are therefore treated differently than the other thicker layers. For a single outer epidermal layer, the elements that approximate the outer skin surface are assigned a weight for the epidermal layer of t/e_{avg} (where t is the layer thickness, and e_{avg} is the average element dimension). As shown by Figure 7.9, such elements are identified as those that are intersected by the outer skin surface, or neighbour such a voxel that isn't included as part of the model (due to insufficient mesh volume overlap). Figure 7.10 visualises the material properties computed for the elements in our forehead model when using anatomically based properties.

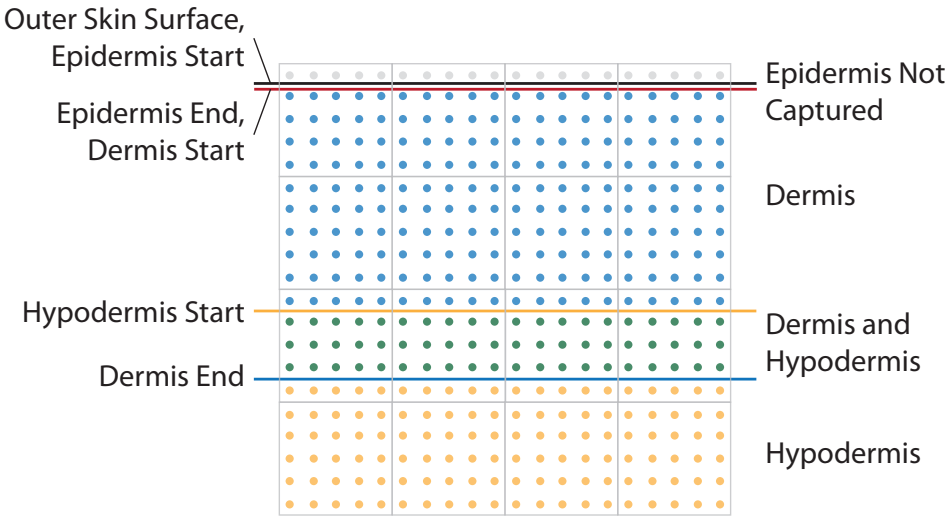

Figure 7.8: Assignment of skin layers to samples inside the skin volume. Note this is a 2D illustration of a 3D process.

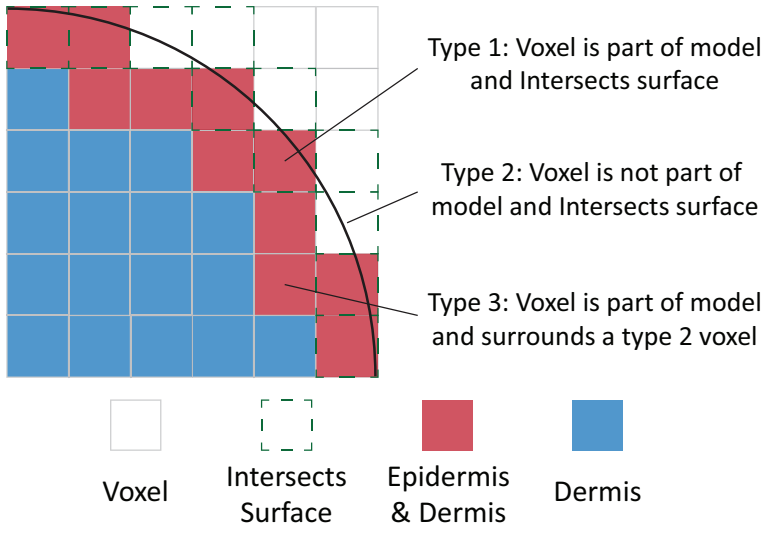

Figure 7.9: Identification of elements that approximate the outer skin surface and therefore contain epidermal properties. Note this is a 2D illustration of a 3D process.

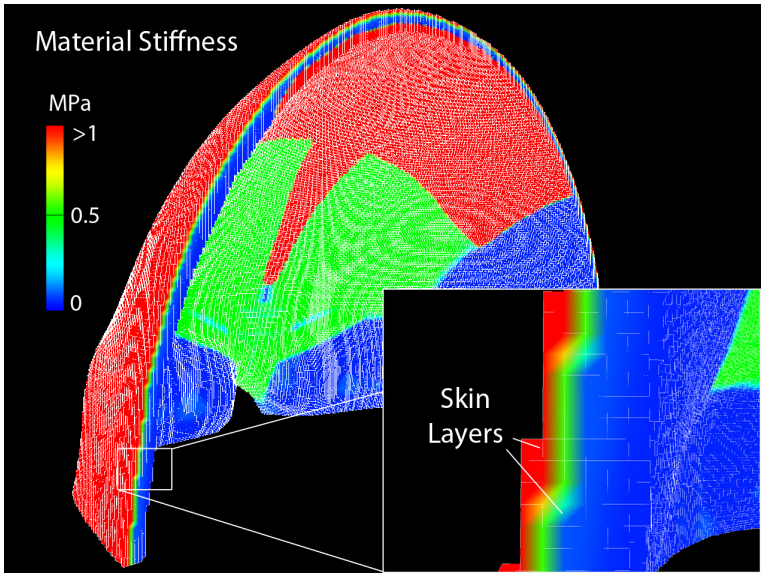

Figure 7.10: The material stiffness of elements in our forehead model when using anatomically based material properties.

7.4 Computing Element Muscle Properties

7.4.1 Computing Fibre Directions

Proportions of overlap between the voxel elements and muscles are used to weight muscle stresses, like with material properties (refer to Equation 7.3). For computation of muscle fibre directions for elements, we use our sampling procedure to combine fibre directions computed at element samples. As shown by Figure 7.3, at each element sample, s, a normalised fibre direction, $\mathbf{d_{s,m}}$, is also calculated for each muscle, m, that encloses the sample, and the sum of these are normalised to produce a fibre direction, $\mathbf{d_{e,m}}$, for each muscle that overlaps the element:

$$\mathbf{d_{e,m}} = \frac{\sum_{s=1}^{n_{e,m}} \mathbf{d_{s,m}}}{\left\| \sum_{s=1}^{n_{e,m}} \mathbf{d_{s,m}} \right\|} \tag{7.4}$$

where $n_{e,m}$ is the number of element samples inside the muscle. Note that, if a different procedure that didn't involve using samples was used to compute proportions of overlap between voxels and mesh volumes, samples could still be generated inside voxel sections overlapped by muscles for the sole purpose of computing element fibre directions. In such a case, there would be no efficiency advantage of using a uniform sampling procedure, which is beneficial for reducing the number of ray-surface intersection tests when computing proportions of overlap.

Sample fibre directions are calculated using NURBS volume approximations of the muscles. A NURBS volume, $V(\mathbf{u})$, is a parametric representation of a volume, whereby spatial coordinates,

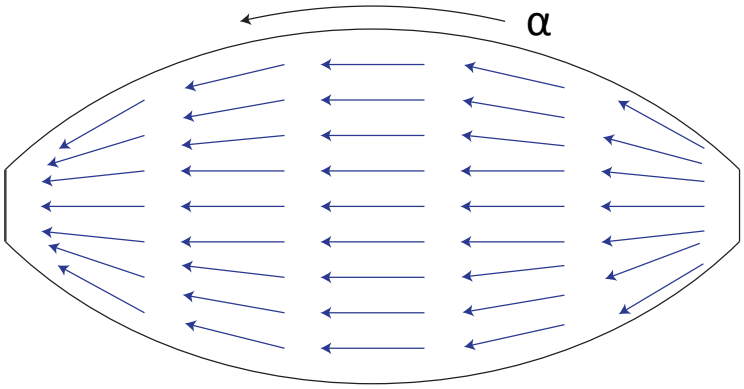

Figure 7.11: An illustration of the muscle fibre field, $\frac{\partial V}{\partial \alpha}$, produced at a cross-section of a NURBS volume, V, where α is the parametric coordinate along the length of the muscle.

$\mathbf{x}$, are computed from parametric coordinates, $\mathbf{u}$, using rational basis functions, $R_{i,j,k}(\mathbf{u})$:

$$V(\mathbf{u}) = \sum_{i}^{p}\sum_{j}^{q}\sum_{k}^{r} R_{i,j,k}(\mathbf{u})\mathbf{c_{i,j,k}} \tag{7.5}$$

$$R_{i,j,k}(\mathbf{u}) = \frac{N_{i,n}(u_1)N_{j,m}(u_2)N_{k,l}(u_3)w_{i,j,k}}{\sum_{a}^{p}\sum_{b}^{q}\sum_{c}^{r} N_{a,n}(u_1)N_{b,m}(u_2)N_{c,l}(u_3)w_{a,b,c}} \tag{7.6}$$

where $\mathbf{c_{i,j,k}}$ is a control point with weight $w_{i,j,k}$, and p, q and r are the number of control points along u_1, u_2 and u_3 respectively. Rather than the spatial coordinate, we require a fibre direction, which is obtained by taking the partial derivative of the NURBS volume with respect to the parametric coordinate along the length of the muscle, $\alpha \in \{u_1, u_2, u_3\}$, as shown by Figure 7.11. As can be seen from Equation 7.5, this requires $\frac{\partial R_{i,j,k}(\mathbf{u})}{\partial \alpha}$, which can be calculated using the quotient rule; for example, when $\alpha = u_1$:

$$\frac{\partial R_{i,j,k}(\mathbf{u})}{\partial u_1} = \frac{\frac{\partial f_{i,j,k}(\mathbf{u})}{\partial u_1} g_{i,j,k}(\mathbf{u}) - f_{i,j,k}(\mathbf{u})\frac{\partial g_{i,j,k}(\mathbf{u})}{\partial u_1}}{g^2_{i,j,k}(\mathbf{u})} \tag{7.7}$$

$$f_{i,j,k}(\mathbf{u}) = N_{i,n}(u_1)N_{j,m}(u_2)N_{k,l}(u_3)w_{i,j,k} \tag{7.8}$$

$$g_{i,j,k}(\mathbf{u}) = \sum_{a}^{p}\sum_{b}^{q}\sum_{c}^{r} N_{a,n}(u_1)N_{b,m}(u_2)N_{c,l}(u_3)w_{a,b,c} \tag{7.9}$$

$\frac{\partial f_{i,j,k}(\mathbf{u})}{\partial u_1}$ and $\frac{\partial g_{i,j,k}(\mathbf{u})}{\partial u_1}$ can be calculated using the product rule. Since $\frac{\partial N_{j,m}(u_2)}{\partial u_1} = 0$ and $\frac{\partial N_{k,l}(u_3)}{\partial u_1} = 0$, they can be simplified to:

$$\frac{\partial f_{i,j,k}(\mathbf{u})}{\partial u_1} = \frac{\partial N_{i,n}(u_1)}{\partial u_1} N_{j,m}(u_2)N_{k,l}(u_3)w_{i,j,k} \tag{7.10}$$

$$\frac{\partial g_{i,j,k}(\mathbf{u})}{\partial u_1} = \sum_{a}^{p}\sum_{b}^{q}\sum_{c}^{r} \frac{\partial N_{a,n}(u_1)}{\partial u_1} N_{b,m}(u_2)N_{c,l}(u_3)w_{a,b,c} \tag{7.11}$$

The gradient of a NURBS volume with respect to α can now be computed, and this can be used

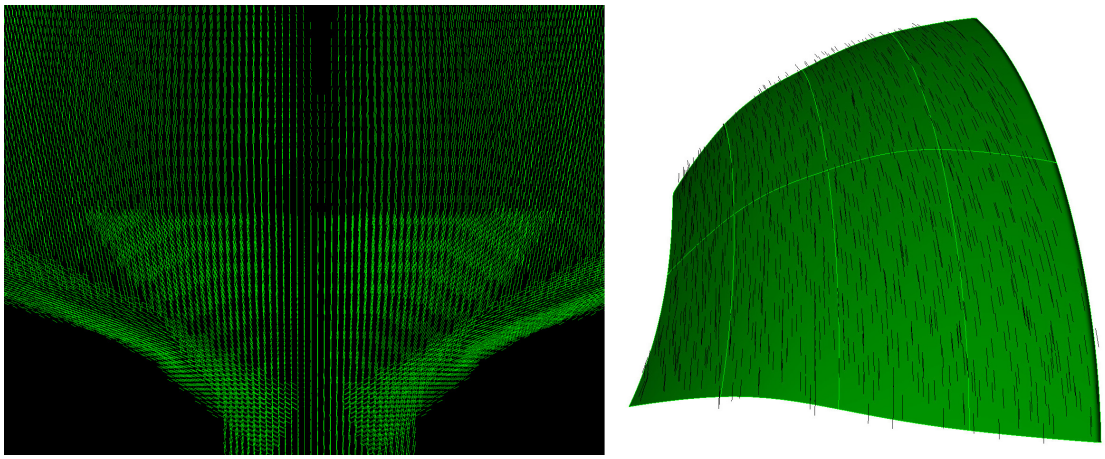

Figure 7.12: Element muscle fibre directions computed for our forehead model. The image on the right shows some element fibre directions of the right frontalis muscle overlaid onto the NURBS surface for this muscle, in which it can be seen that the fibre field follows the isocurves along the muscle length.

as an implicit fibre field for a muscle, $d(\mathbf{u})$ [TSB+05]:

$$\frac{\partial V(\mathbf{u})}{\partial \alpha} = \sum_{i}^{p} \sum_{j}^{q} \sum_{k}^{r} \frac{\partial R_{i,j,k}(\mathbf{u})}{\partial \alpha} \mathbf{c_{i,j,k}} \tag{7.12}$$

$$d(\mathbf{u}) = \frac{\dfrac{\partial V(\mathbf{u})}{\partial \alpha}}{\left\| \dfrac{\partial V(\mathbf{u})}{\partial \alpha} \right\|} \tag{7.13}$$

As Equation 7.13 requires parametric, $\mathbf{u}$, rather than the spatial coordinates, $\mathbf{x}$, of the samples, the parametric coordinates of sample points are estimated using the Newton-Raphson iterative root-finding method, which aims to successively find a better approximation of the parametric coordinates, $\mathbf{u_{n+1}}$, given an initial estimation, $\mathbf{u_0}$:

$$\mathbf{u_{n+1}} = \mathbf{u_n} - \left(\frac{\partial V(\mathbf{u_n})}{\partial \mathbf{u}} \right)^{-1} (V(\mathbf{u_n}) - \mathbf{x}) \tag{7.14}$$

where $\dfrac{\partial V(\mathbf{u_n})}{\partial \mathbf{u}}$ is the NURBS volume Jacobian matrix of parametric with respect to spatial coordinates. As $\mathbf{u_{n+1}}$ may fall outside the valid range of parametric coordinates for the NURBS volume, u_{n+1_i} is clamped for each parametric dimension i on each iteration to ensure they are within the valid range for that dimension. An error value, e, is also computed:

$$e = \|V(\mathbf{u_n}) - \mathbf{x}\| \tag{7.15}$$

Equation 7.14 is repeated with successive approximations, and, as the error can fluctuate, the minimum error and corresponding parametric coordinates are stored. This iterative procedure continues until either the error falls below a maximum value, or a maximum number of iterations have been performed.

The initial estimation can have a large effect on the convergence speed of the Newton-Raphson method, and, with some estimations, it may not be possible for the method to converge at all. To prevent large errors or failure, we compute a range of initial estimations that are uniformly spaced between the minimum and maximum parametric coordinates for the NURBS volume. The Newton-Raphson method is performed with each successive initial estimation, and the minimum

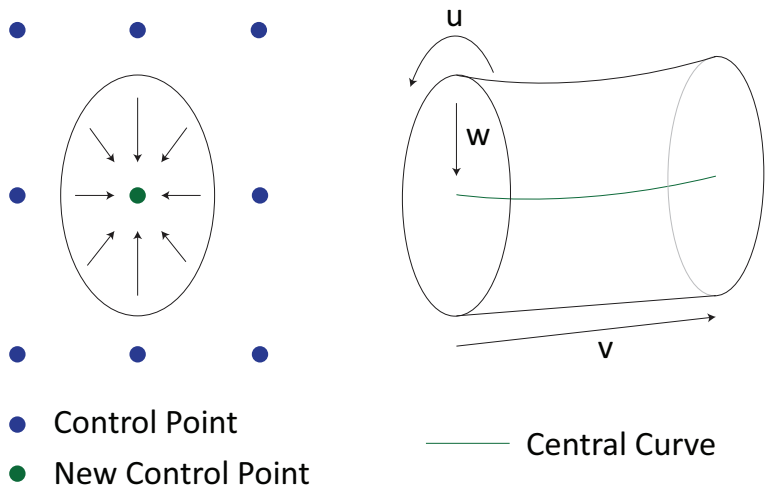

Figure 7.13: Creation of a NURBS volume by shrinking a NURBS surface, creating a 3rd dimension running from the NURBS surface to a central curve.

error and corresponding parametric coordinates are stored. If the error falls below a maximum value (the same maximum value used for the Newton-Raphson method), no further iterations are performed. By keeping track of the parametric coordinates with the lowest error, cases can also be handled whereby it may not be possible for the Newton-Raphson method to ever sufficiently converge, such as in the case that a sample lies inside a muscle but outside the NURBS volume approximation of that muscle. Using Equation 7.14 to estimate the parametric coordinates of sample points, $V^{-1}(\mathbf{x})$, the muscle fibre field in Equation 7.13 can now be represented in terms of spatial coordinates:

$$d(\mathbf{x}) = \frac{\dfrac{\partial V(V^{-1}(\mathbf{x}))}{\partial \alpha}}{\left\| \dfrac{\partial V(V^{-1}(\mathbf{x}))}{\partial \alpha} \right\|} \tag{7.16}$$

Figure 7.12 visualises the element muscle fibre directions computed for our forehead model.

7.4.2 Creating NURBS Volumes (NURBS Surface Shrinking)

NURBS volumes are difficult to create. While such a volume could potentially be manually created, no 3D modelling tools were found that support NURBS volumes, and editing such volumes would be tricky due to the number of control points. We therefore create NURBS volume approximations of muscles automatically from NURBS surfaces by shrinking them to their central curves [MLC01]. A NURBS surface approximation of each muscle is therefore required, which can be easily created using 3D modelling tools. For shrinking, such surfaces must be closed along one parametric dimension, and, for each row of control points forming an isocurve along this dimension, a new point is created at the centre of these points to create a central curve, as shown by Figure 7.13. Each central point is duplicated to create a corresponding central control point for each control point in the row. A NURBS volume can then be created, the 3rd parametric dimension of which runs linearly from the NURBS surface to this central curve. While more control points inside the volume would enable nonlinear fibre field variations throughout the volume, such detailed control seems unnecessary, as fibres are arranged in bundles, and generally follow similar directions to neighbouring fibres.

As an example of NURBS surface shrinking, the NURBS surface in Figure 7.14 is defined by parameters u_1 and u_2, and has control points $\mathbf{c_{i,j}}$ for $i = 1 \ldots p$ and $j = 1 \ldots q$, where p and q are the number of control points along u_1 and u_2 respectively. To create a NURBS volume, a new

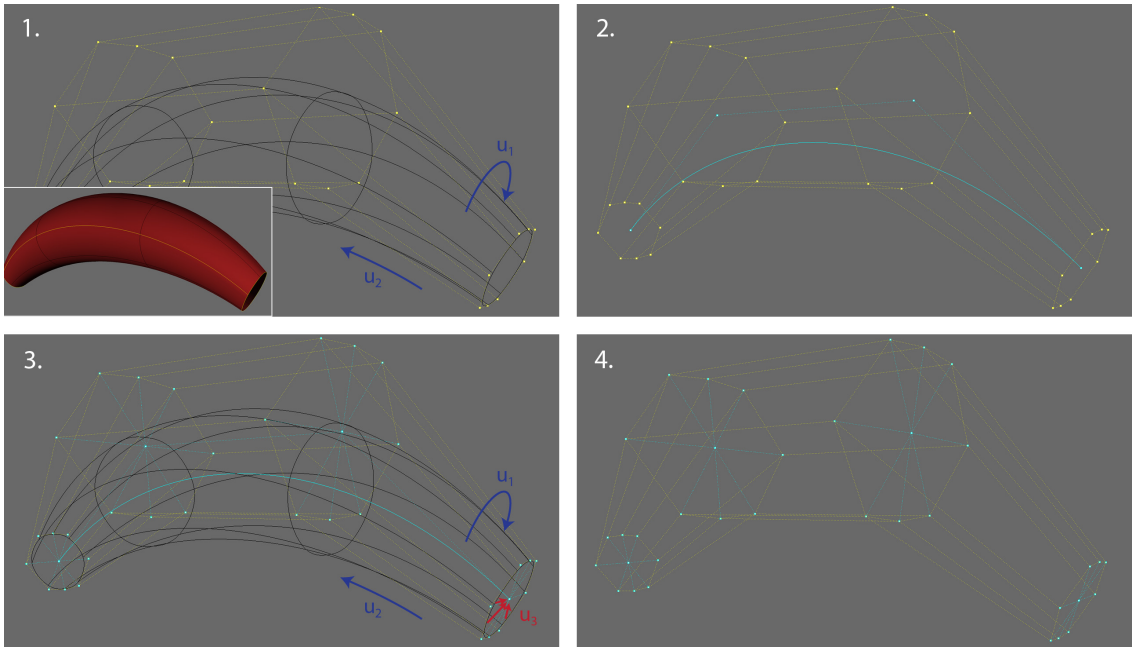

Figure 7.14: An example of creating a NURBS volume by shrinking a NURBS surface, closed along the u_1 dimension, to its central curve. The third parametric dimension, u_3, runs from the original NURBS surface boundary to the central curve.

parametric dimension, u_3, is introduced, and the control points are now represented as $\mathbf{c_{i,j,k}}$ with $k = 1$ for all current (surface) control points. The u_1 dimension is closed; therefore, the surface is shrunk to create a central curve along the u_2 dimension. For a particular value of j, a new point is created as $\frac{1}{|C_j|}\sum_{\mathbf{c}\in C_j}\mathbf{c}$, where C_j is the set of control points along u_1 with distinct positions. The new point is duplicated p times to create control points $\mathbf{c_{i,j,2}}$ for $i = 1 \ldots p$. This is done for each value of j (each row along u_1), resulting in the $p \times q$ set of points that define the original surface, $\mathbf{c_{i,j,1}}$, and another that define an inner surface (which, in our case, has an infinitely thin width, forming a curve), $\mathbf{c_{i,j,2}}$. The interpolation between these sets represents the 3rd parametric dimension.

7.5 Computing Boundary Conditions

To define model boundary conditions, using our simulation system, it is possible to restrict the movement of particular nodes by setting them as rigid or sliding (bound by a surface). Rigid nodes remain fixed with zero displacement throughout a simulation, and can therefore model fixed muscle attachments (muscle origins). Sliding nodes remain a fixed distance from the non-conforming surface they are restricted by, and can be used to model, for example, the sliding effect between the superficial and stiff deep facial soft-tissue layers [WMSH10] (a phenomenon often neglected with current physics-based facial animations [SNF05, BJTM08]). At the model creation stage, it is necessary to identify such restricted nodes. More information about the use of boundary conditions during simulations can be found in Section 8.5.

As shown by Figure 7.15, and described by Algorithm 7.3, restricted nodes are identified using a collection of non-conforming internal and external restricting (rigid and sliding) surfaces. An internal surface might represent an attachment area inside the main volume (a level 0 volume), whereas the skull is an obvious external surface that defines part of the boundary of such a volume. Restricted nodes are identified to approximate such non-conforming surfaces. For each identified

```
   // Internal restricting surfaces:
 1 foreach internal restricting surface, S_int do
 2 |   E_int ← {e ∈ elements : S_int intersects e};
 3 |   N_int ← {n ∈ nodes : n is connected to e ∈ E_int};
 4 |   foreach n ∈ N_int do
 5 |   |   p_n ← position of n;
 6 |   |   p_s ← closest point on S_int to n;
 7 |   |   d_n ← p_n − p_s;
 8 |   |   d_s ← normal of S_int at p_s;
 9 |   |   θ ← arccos( (d_n · d_s) / (‖d_n‖‖d_s‖) );
10 |   |   if θ ≤ π/2 then
11 |   |   |   Set n as restricted;
   // External restricting surfaces:
12 N_outer ← {n ∈ nodes : n is an outer node};
13 l ← voxel element x, y, z dimensions;
14 foreach external restricting surface, S_ext do
15 |   N_ext ← {n ∈ N_outer : S_ext intersects AABB (centre n, dimension 2l)};
16 |   foreach n ∈ N_ext do
17 |   |   d_n ← normal of n;
18 |   |   p_s ← closest point on S_ext to n;
19 |   |   d_s ← normal of S_ext at p_s;
20 |   |   θ ← arccos( (d_n · d_s) / (‖d_n‖‖d_s‖) );
   |   |   // θ_ref ← π/2 by default
21 |   |   if θ ≤ θ_ref then
22 |   |   |   Set n as restricted;
```

Algorithm 7.3: The processes to identify restricted (rigid or sliding) nodes from internal and external restricting surfaces.

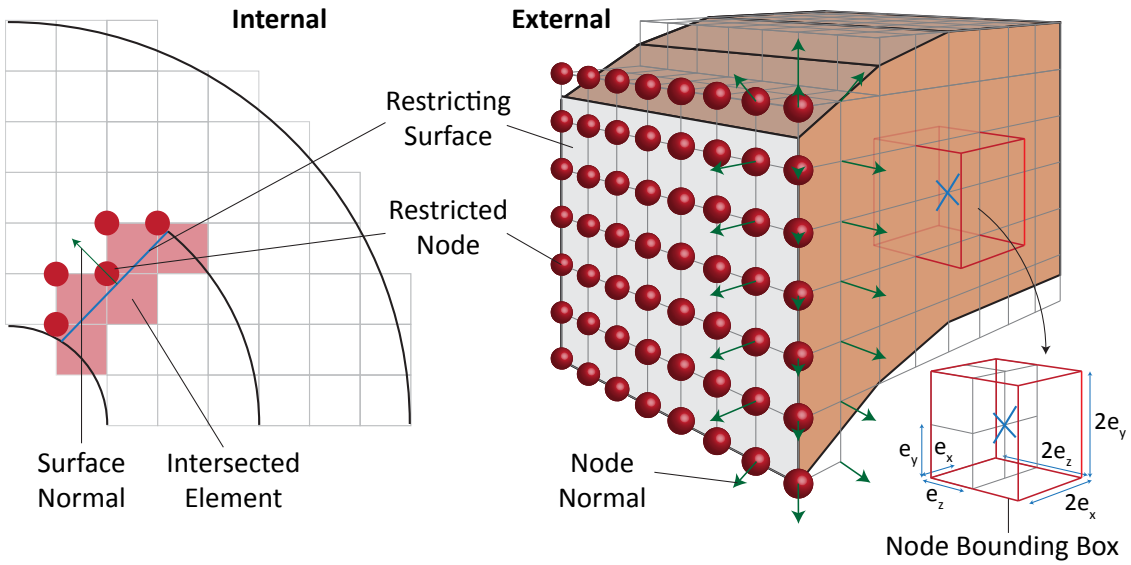

Figure 7.15: Identification of restricted (rigid or sliding) nodes from internal (left, in 2D) and external restricting surfaces (right). In this case, the angle between the external surface and outer node normal must be less than $\pi/2$ for the node to be set as restricted.

sliding node, the signed distance to the surface is also required, which is calculated using the closest point on the surface to the node, and the surface normal at this point.

As shown by Figure 7.15, an internal restricting surface should be located entirely inside the main volume. In this case, the surface will therefore intersect a series of connected voxel elements, enabling a connected set of nodes that approximate the surface to be easily identified. Firstly, elements that are intersected by the surface are determined. Then, for each node of each such element, if it is located in front of the surface (tested using the closest point on the surface and its normal), it is set as restricted (rigid or sliding).

A different approach is required with an external restricting surface, which may pass both inside and outside of the voxel structure. Also, as such a surface defines part of the main volume boundary, only the outermost nodes are considered (i.e. nodes that are connected to fewer than 8 voxel elements). For each outer node, an axis-aligned bounding box (AABB), aligned with the

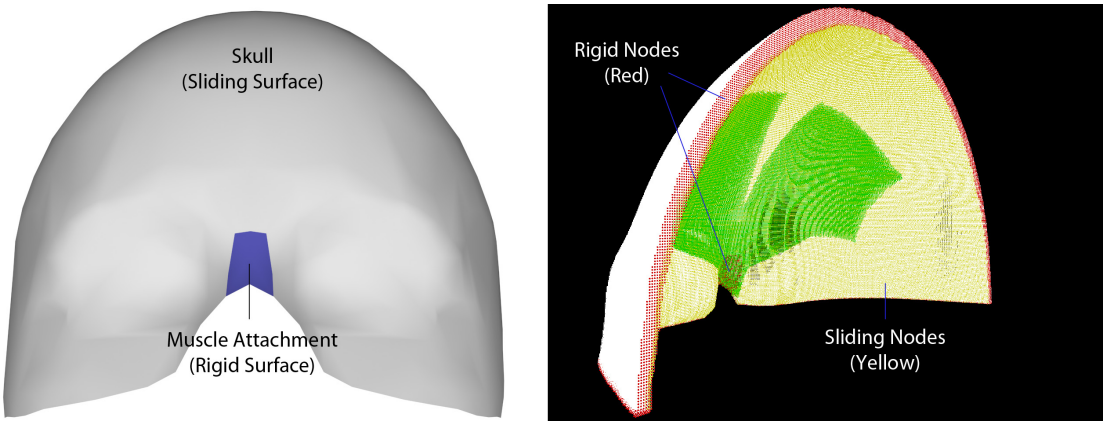

Figure 7.16: Restricting surfaces, and the rigid and sliding nodes for identified for our forehead model. Nodes along boundaries to the remainder of the head have been set as rigid.

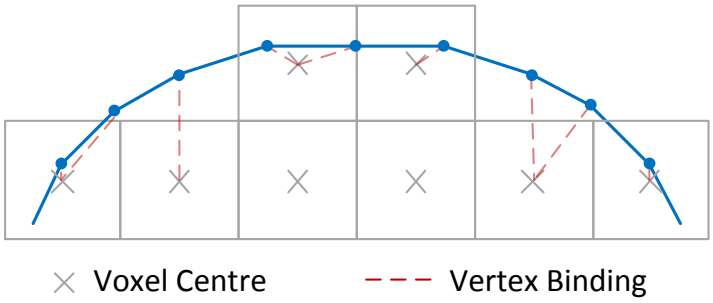

Figure 7.17: Vertices of a surface mesh bound to the closest elements of a simulation model. Note this is a 2D illustration of a 3D process.

voxel elements, is created, centred at the node, and with dimensions $2 \times \mathbf{l}$ (where $\mathbf{l}$ is a vector of the voxel element x, y, z dimensions). Nodes are recorded if the surface intersects their bounding box. The dimensions of the bounding box allow the surface to pass within one voxel from the node. This is the maximum distance a surface can reside from an outer node without the voxel being included as an element in the voxel structure if the surface is also used as part of the voxelisation process, like the skull surface.

For each such node, if the angle between the node normal (computed using the normals of each connected element face) and the normal of the closest surface point is less than a specified size (i.e. they are pointing in a similar direction), it is set as restricted. This test is necessary to prevent false positives being detected, for example, if a node lies within range of the surface but is part of an adjoining surface approximation (see Figure 7.15). Figure 7.16 shows the restricting surfaces, and the rigid and sliding nodes identified for our forehead model. Restricting surfaces include the skull (sliding surface), and rigid attachment surfaces for the procerus and corrugator supercilii muscles, while nodes on the boundaries to the remainder of the head have been set as rigid.

7.6 Binding the Surface Mesh to the Simulation Model

For the final stage of the model creation process, as with Dick et al.'s simulation approach [DGW11], the vertices of the surface mesh are bound to and animated with elements of the simulation mesh using trilinear interpolation and extrapolation (see Figure 7.17). A position, $\mathbf{p}$, is bound to the closest element (determined by distance tests from the element centres), using three weights, w_i, one along each local axis, $1 \leq i \leq 3$, from the first node of the element, $\mathbf{x}$. For an undeformed voxel element aligned with the global axes, these can be easily calculated:

$$w_i = 1 - \frac{p_i - x_i}{l_i} \tag{7.17}$$

where $\mathbf{l}$ is a vector of the voxel dimensions, and the first node, $\mathbf{x}$, is the node at position $(\frac{-l_1}{2}, \frac{-l_2}{2}, \frac{-l_3}{2})$ from the voxel centre. The weights are values between 0 and 1 for interpolation, where $w_i = 1$ when $\mathbf{p}$ shares the same position in the i^{th} dimension as $\mathbf{x}$, and $w_i = 0$ when $\mathbf{p}$ shares the same position in the i^{th} dimension as the element node connected to $\mathbf{x}$ along this same dimension. Other values of weights outside this range signify extrapolation, for the case when a voxel that overlaps a boundary of the main volume is not included in the FE model due to an insufficient proportion of overlap. Using the weights, trilinear interpolation and extrapolation can be applied before rendering the surface mesh.

If a model has been created for only part of the surface mesh, like the forehead region of our facial surface mesh, only vertices inside the model bounding box are bound to the model elements. This therefore requires a sufficiently large region to be modelled such that forces will

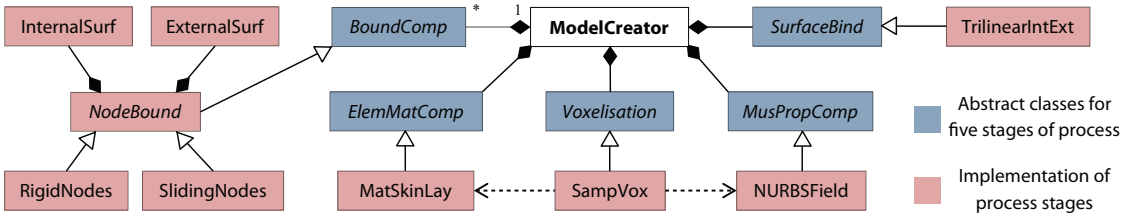

Figure 7.18: The relationships between the main classes used with our model creation system.

propagate to almost zero along the boundary nodes during simulation, in which case vertices outside the bounding box would remain almost fixed anyway, otherwise severe polygon stretching effects could occur with polygons overlapping the boundaries. Also, restricting (rigid and sliding) surfaces, such as the skull, remain fixed with respect to soft-body deformations, and vertices of these surfaces are therefore not bound.

7.7 Implementation Details

The input to the model creation system is an XML file that contains links to the surface mesh, and the NURBS surface approximation for each muscle, as well as model creation parameters, such as voxel dimensions and the voxel sampling resolution. Such parameters differ depending on whether a soft-tissue or generic soft-body model is to be generated. With a soft-tissue model, it is necessary to define the skin layers, although mesh volumes and levels can be automatically defined provided the surfaces of the surface mesh are appropriately labelled. With a generic soft-body model, named volumes are required to be manually defined as collections of labelled surfaces, and the volumes then manually grouped into levels, although skin layers may not be defined. Note that, like with grid-based hexahedral mesh generation approaches (discussed in Section 5.2.2), the generated models are dependent on the orientation of the voxel grid with respect to the input surface meshes; therefore, the surface meshes should be optimally aligned with the voxel grid.

Figure 7.18 shows the main classes used by our C++ implementation of the model creation system. The main `ModelCreator` class uses five independent abstract classes: `Voxelisation`, `ElemMatComp`, `MusPropComp`, `BoundComp` and `SurfaceBind`. These correspond to the five main stages of the model creation process respectively, while their subclasses contain the implementations of these stages described in Sections 7.2 to 7.6 respectively. Since multiple types of boundary conditions may be defined for models, `ModelCreator` uses a collection of `BoundComp`, each of which computes a particular type of boundary condition. We define two types of nodal boundary conditions, enabling nodes to be set as rigid or sliding. The `NodeBound` abstract class that extends `BoundComp` identifies such nodes from non-conforming restricting internal and external surfaces using algorithms implemented in `InternalSurf` and `ExternalSurf` respectively, while `RigidNodes` and `SlidingNodes`, which both extend `NodeBound`, compute and set the different data that is required to process rigid and sliding nodes respectively during simulation.

The independence of the abstract classes corresponding to the five main stages of the creation process enables an implementation whereby each of these stages are independent, and can therefore be easily altered, extended or replaced without affecting the rest of the process. With our implementation, for convenience and efficiency, the computation of skin layers and muscle fibre directions is performed during voxelisation, as these two processes use the sampling procedure. However, if the voxelisation stage used a different technique to compute the proportions of overlap between voxels and mesh volumes, samples could be generated independently for these two processes.

Due to the implementation flexibility that enables the stages to be independently altered, it would be straightforward to, for example, use a different technique to compute the proportions of overlap between the voxel elements and mesh volumes. If such a technique is more robust to

noisy data or volumes with holes, less strict requirements would be imposed on the input surface mesh. This would be especially useful, for example, if working with medical data, which would then require less surface mesh clean-up. It may also be desirable to use a different procedure to compute muscle fibre directions, for example, that doesn't require the creation of NURBS surface approximations, or add an additional technique to compute boundary conditions from different input data. Currently, nodal boundary conditions are computed using a collection of non-conforming restricting internal and external surfaces, although one might desire to compute these using restricting points, splines or volumes, or introduce a new type of boundary condition.

7.8 Summary

This chapter has presented our automatic model creation process to easily create animatable multi-layered non-conforming hexahedral FE simulation models to which surface meshes are bound, focussing on facial soft-tissue models. Starting with any closed surface mesh, this involves discretising the enclosed volumes into voxels, and calculating model properties (such as skin layers, and element material and muscle properties) based on the proportions of overlap between the voxels and volumes enclosed by surfaces of the surface mesh. We use an efficient uniform sampling procedure to approximate such proportions of overlap. Muscle fibre fields are generated using NURBS surfaces that are closed along one parametric dimension, enabling these to be converted to NURBS volumes. Boundary conditions are also computed, enabling nodes to be set as rigid (fixed) or sliding (bound by a surface) based on a collection of non-conforming surfaces. While we focussed on generating soft-tissue models, our creation process can be used to generate any multi-layered FE model from any surface mesh.

The flexible design and implementation of our model creation process enables each of the stages to be altered or extended independently without affecting the rest of the process, which would be useful if the input surface meshes or generated simulation models have different requirements to ours. Our forehead model has demonstrated the complexity of models that can be created. Additional soft-tissue and arbitrary soft-body models, as well as some limitations, are presented in detail with potential enhancements in Chapter 9. The simulation and visualisation of the models is explained in the next chapter, along with model optimisations by using resourceful data storage and organisation for efficient GPU simulation.

Chapter 8

Model Simulation

Our simulation process involves simulating multi-layered FE models of facial soft tissue using the nonlinear TLED FE method [MJLW07]. An anatomical muscle contraction model is used to generate active stresses for muscles under contraction, as well as transversely isotropic passive stresses in the directions of muscle fibres as they are stretched. Advanced boundary conditions constrain nodes, and can model the sliding effect between superficial soft-tissue layers, and the deep layers and skull [WMSH10], which is often neglected [SNF05, BJTM08].

We use the voxel-based models generated by our model creation process (discussed in the previous chapter) with the simulation process. By capturing detail such as skin layers, our models and simulation process are capable of simulating more complex gross- and fine-scale behaviour than existing models [SNF05, BJTM08, WHHM13], including wrinkling. The GPU-based implementation of our simulation and visualisation system, implemented using CUDA, has been optimised to exploit computational advantages offered with our voxel-based models. Our process can also be used to simulate any inhomogeneous soft body (not just soft tissue). Details of our simulation process, and our GPU-based implementation and optimisation thereof have been previously published [WM12, WM13c, WM13d, WM13b, WM14].

The following sections start by explaining the theory behind our simulation process, including the TLED FE solver with hourglass control, muscle contraction model, and the processing of boundary conditions. An introduction to CUDA is then presented, before the implementation details of our CUDA-based system, including memory organisation and processing of model data on the GPU, visualisation and interaction, as well as the optimisations for use with our voxel-based models.

8.1 Simulation Overview

Physically based techniques for soft-body deformations, such as the displacement-based FE method, typically involve formulating equations of motion that approximate the governing equations defining the equilibrium of a physical body to produce a physically based model. Using a numerical time-integration technique, approximate solutions to these time-dependent equations of motion can then be computed over time. The approximate configuration of the model is calculated at discrete points in time, simulating the time-varying deformation (and convergence to equilibrium) of the model.

At the core of our simulation process is the displacement-based dynamic TL formulation of the FE method with an explicit time-integration scheme for approximating and simulating soft-body deformations to produce animation. Solutions (displacements) of the equations of motion are computed at nodal points, and can be interpolated through elements using the element shape functions. Using an explicit time-integration scheme, each solution makes use of the current and possibly previous body configurations (known body configurations, at times t_0 to t_n) to compute the next body configuration (at time t_{n+1}), advancing a simulation from the current, $t_n = t$,

to the next point in time, $t_{n+1} = t + \Delta t$ (computation of a timestep, Δt). The number of timesteps required to be computed per frame of animation depends on the model and simulation properties, which affect the accuracy and stability of simulations that use a conditionally stable time-integration scheme. For example, a typical timestep size used with our forehead simulations is 5μs, whereas an animation at 30 frames per second would output a frame every 33.3ms.

The procedure to compute a timestep using our simulation process is summarised by Algorithm 8.1. A major process when solving the equations of motion is to compute the internal nodal forces from the deformations and stresses of the elements they connect. Element material stresses are calculated using the relevant constitutive equations that describe the material behaviour of the elements. As our models are non-conforming, elements may overlap multiple mesh volumes, such as the skin volume and a muscle volume of a soft-tissue model. As discussed in Chapter 7, the material properties associated with mesh volumes, and proportions of overlap between elements and these mesh volumes define the material behaviour of the elements. These are used during simulation to compute and combine weighted stresses for each material associated with an element.

```
1  foreach element, m do
2  |  foreach integration point do
3  |  |  foreach muscle overlapping m do
4  |  |  |  Calculate active muscle stresses (Section 8.4);
5  |  |  foreach material associated with m do
6  |  |  |  Calculate material stresses (Section 8.2.10);
7  |  |  Calculate internal nodal forces (Section 8.2.11);
8  |  Calculate hourglass forces (Section 8.3);
9  foreach node, n do
10 |  if n not rigid then
11 |  |  Calculate nodal displacements (Section 8.2.13);
12 foreach sliding node do
13 |  Update nodal displacements (Section 8.5);
```

Algorithm 8.1: The main stages of the process to compute a timestep for non-conforming elements. The most relevant sections that explain each such stage have been identified. Details on approximating element integrals by computing element values at integration points using Gaussian quadrature are presented in Section 8.2.11.

The hexahedral elements of our voxel-based simulation models are treated as reduced-integration 8-node hexahedra. While computationally efficient and accurate for soft-tissue simulations, a phenomenon called hourglassing can occur under large deformations when using reduced integration as opposed to fully integrated elements, and an additional process is required after the computation of internal nodal forces to reduce hourglass effects. As well as computation of internal nodal forces, a timestep also requires the computation of muscle stresses, and the solution of boundary conditions to constrain the simulation. Active muscle stresses introduce stresses into the simulations to trigger deformation.

8.2 The Total Lagrangian Explicit Dynamic Finite Element (TLED FE) Method

The FE method is a numerical technique that can be used to find approximate solutions to systems of partial differential equations for which analytical solutions would be almost impossible to find. Similar to other numerical physically based techniques, like the MS method, it can be divided into the following five stages:

1. Mathematical modelling (Sections 8.2.1 to 8.2.7) and discretisation (approximation) of the physical system (Chapter 7)

2. Formulation of the element equations (Sections 8.2.8 to 8.2.12, and Sections 8.3 and 8.4 for element hourglass and muscle forces respectively)

3. Assembly of the global equations (Section 8.2.13)

4. Incorporation of boundary conditions (Section 8.5)

5. Solution of the global equations (Section 8.2.13)

During the modelling stage, the governing equations of the developed mathematical model are approximated, which involves discretising the physical system into a number of elements that are connected at nodal points on their boundaries. As described in Chapter 7, our model creation process automatically discretises volumes enclosed by surface meshes to create non-conforming hexahedral FE models during Stage 1. For the simulation (Stages 2 to 5), we use the dynamic TL formulation of the displacement-based FE method, numerically approximating the global system equations to compute the nodal displacements over time using an explicit time-integration scheme. Detailed descriptions of the FE method, including the dynamic TL FE formulation with explicit time integration, can be found in the literature [Bat96, MJLW07, TCO08].

The TLED formulation of the FE method has various advantages for simulating soft tissue. Due to the requirement of a small but efficient simulation timestep, explicit time integration is highly suited for simulating the complex nonlinear material behaviour of soft tissue under large deformations, as well as contact and sliding. Such behaviour requires small timesteps to accurately simulate, which are computationally expensive with implicit time-integration schemes. Explicit methods are also inherently parallel, making them suitable for GPU implementation. To further increase simulation efficiency, the TL formulation enables some variables to be precomputed. Being dynamic, inertial and damping effects are also considered. Factors such as thermal effects and fluid dynamics are not considered necessary for the level of accuracy we desire.

8.2.1 Mathematical Modelling Considerations

During the modelling stage, it is necessary to decide upon a linear or nonlinear FE formulation, which is affected by the material models that will be used. Element dimensionality and structure (element type) must also be decided upon. The nonlinear TL formulation that we use enables various nonlinearities to be handled:

- Geometric nonlinearity - Nonlinear relationship between strains and displacements
- Material nonlinearity - Nonlinear relationship between stresses and strains

We take into account both geometric and material nonlinearities to handle the large deformations of complex soft-tissue material. Our simulation system currently only contains an implementation of the hyperelastic neo-Hookean material model, which has been used with the examples in this thesis due to its simplicity for predicting materially nonlinear behaviour under large deformations. Regarding element type, this can also be linear or nonlinear, where element linearity refers to the linearity of the variation of values, such as stress, strain and displacements, across an element (i.e. the linearity of the element shape functions). An appropriate numerical integration technique must also be chosen for the approximation of complex integrals over an element. The hexahedral elements of our voxel-based simulation models are treated as linear reduced-integration (Gaussian quadrature) 8-node hexahedra.

8.2.2 Coordinate Systems and Body Configurations

A coordinate system identifies points in space; for example, a Cartesian coordinate system is often used to describe Euclidean space. A coordinate system can be either:

- A global coordinate system, which describes the entire domain to be modelled

- A local coordinate system, which describes a subset of the global domain

We use a global Cartesian coordinate system to describe the entire space in which a body moves and deforms, while a local coordinate system is introduced for each element during the discretisation stage to enable simple definition and interpolation of element quantities.

When specifying coordinates in a coordinate system, these can refer to either:

- Spatial coordinates, defined with respect to a spatial basis, which identify spatial points (specific points in space)
- Material coordinates, defined with respect to a material basis, which identify material points of a body (specific points on the body)

A material point can also be assigned spatial coordinates and a spatial basis to describe the spatial configuration of that material point. The spatial coordinates (location) and basis of each spatial point remain fixed in space over time, and different material points may pass through a specific spatial point over time. On the other hand, the spatial coordinates of each material point change over time, and the spatial basis of each such point deforms according to the deformation of material fibres passing through that point. The material coordinates and basis of a material point remain fixed with respect to the body over time. In computer graphics, a vertex on a surface mesh can be considered as a material point, which can move through different spatial points over time. The material basis of the vertex might be aligned with some global Cartesian axes, which is rotated and deformed over time according to the rotation and deformation of the surface mesh to form the spatial basis for the point.

Lagrangian FE formulations are defined by considering quantities at material points using material coordinates, the spatial coordinates of which may change over time. To define material coordinates and compute quantities for material points, such as displacements and forces, we can define different configurations of a body:

- The reference configuration, with respect to which quantities are computed
- A deformed (e.g. current or next) configuration

The displacement of a material point, for example, is then the displacement from the spatial location of the material point in the reference configuration. With the TL FE approach, the reference configuration is the initial body configuration.

The material coordinates and basis of a material point can be defined as the corresponding spatial coordinates and basis of the spatial point while the body is in the reference configuration; as such, in the reference configuration, the material coordinates, X_i, and basis, E_i, of a material point equal the spatial coordinates, x_i, and basis, e_i, of the point. In the deformed configuration, the material coordinates and basis of the material point remain the same, and therefore don't necessarily equal the updated spatial coordinates, x'_i, and basis, e'_i, of the point. The displacement of the point, u_i, can be defined as $u_i = x'_i - X_i$ (i.e. it is measured with respect to the reference configuration). The reference configuration, and hence the material coordinates and basis can then be updated before computation of quantities for the next deformed configuration (the UL, updated Lagrangian, FE approach), or quantities can always be computed with respect to a non-changing reference configuration (the TL FE approach).

Continuing with the surface mesh example, to compute, for example, the displacement of a vertex (material point) between the current and next frame of an animation, the current surface mesh configuration can be defined as the reference configuration with respect to which the displacement is computed. The spatial coordinates and basis of the spatial point where the vertex is located in this reference configuration can be considered as the material coordinates and basis of the vertex, although material points in a discrete set, like vertices of a surface mesh, are often simply identified by a single integer, rather than their material coordinates. After the displacement has been applied, the spatial coordinates and basis of the vertex change, while the material coordinates and basis stay the same (unless the reference configuration is updated).

8.2.3 Mathematical Representation of a Deformable Body

The modelling stage firstly consists of converting a physical system into a mathematical model that describes the behaviour of the system. We are interested in the deformation of soft tissue under the influence of some action, such as a muscle contraction that introduces stress (forces) into the system. To produce such animation, the problem is therefore to find the time-varying deformation (e.g. displacements, stresses and strains) of a deformable physical body as it converges towards a static equilibrium state. To compute this deformation, we must first define the equilibrium state of the body in terms of its displacements and the forces acting on it.

For a body to be in equilibrium, the resultant forces over the entire body must equal zero. Forces acting on a deformable body can be classed as either:

- External forces that are influenced by external agents outside the system, including:
  - Point forces acting on individual particles
  - Surface forces (forces per unit area) that act on the boundary of the body, such as a contact force on the surface of the body
  - Body forces (forces per unit volume) that act on the entire body, such as gravity
- Internal forces that act between two parts of the system, such as the restoring force between two particles of a system

Internal forces are defined in terms of stresses within a body. These stresses depend on the material properties of the body, and the strains corresponding to the body displacements.

With a continuous surface or body, quantities such as displacement and force for the infinite number of material points are represented as distributions over the area of the surface, or volume of the body respectively. A displacement distribution, $\bar{u}_i$, of the material points inside a deformable body is given by:

$$\bar{\mathbf{u}}^{\mathrm{T}} = \begin{pmatrix} \bar{u}_1 & \bar{u}_2 & \bar{u}_3 \end{pmatrix} \tag{8.1}$$

where bar notation is used to denote a continuous function of material coordinates; as such, $\bar{u}_i$ is a differentiable function of material coordinates X_i. We use the bar notation to distinguish between a continuous function (distribution) of values for material locations (e.g. $\bar{x}_i = x_i(X_i)$ represents a function to compute spatial coordinates from material coordinates), as opposed to a specific value for a specific material location (e.g. x_i is a spatial coordinate that can denote the spatial location of material coordinate X_i). Displacement distributions for the surface of the body, and surface and body forces can be defined analogously to the distribution in Equation 8.1. Strain, $\bar{\epsilon}_{ij}$, and stress distributions, $\bar{\sigma}_{ij}$, are also similarly defined with a function for each strain and stress component:

$$\hat{\bar{\boldsymbol{\epsilon}}}^{\mathrm{T}} = \begin{pmatrix} \bar{\epsilon}_{11} & \bar{\epsilon}_{22} & \bar{\epsilon}_{33} & 2\bar{\epsilon}_{12} & 2\bar{\epsilon}_{23} & 2\bar{\epsilon}_{13} \end{pmatrix} \tag{8.2}$$

$$\hat{\bar{\boldsymbol{\sigma}}}^{\mathrm{T}} = \begin{pmatrix} \bar{\sigma}_{11} & \bar{\sigma}_{22} & \bar{\sigma}_{33} & \bar{\sigma}_{12} & \bar{\sigma}_{23} & \bar{\sigma}_{13} \end{pmatrix} \tag{8.3}$$

We use the hat notation to denote the vector form of a symmetric tensor.

Theoretically, the deformation of a continuous volumetric body under the influence of external forces could now be calculated using the material properties and constraints (such as fixed surfaces), and establishing and solving the governing differential equations of equilibrium. However, these consist of numerous coupled partial differential equations with numerous unknown values that would be almost impossible to solve analytically, particularly for complex objects and materials like soft tissue. Also, we are interested in the time-varying deformation of objects as they converge to equilibrium states to produce animation, not just the equilibrium states given by such closed-form analytical solutions. Alternatively, the equilibrium of a body can be expressed using the concept of virtual work.

8.2.4 Work and Virtual Work

Work is the energy associated with the action of a force during a displacement. When a force acts on a body, and there is displacement of the point, surface or volume of application in the direction of the force, work is done. This work can be classed as either external or internal depending on the force performing the work. For external work, let us consider the simple cases whereby points and material particles are subject to a constant forces, and displaced along straight (vector) trajectories, maintaining a constant area or volume for a surface or body respectively.[1] In such cases, the work done by concentrated force $f_i^{(p)}$ on point p, and surface, $\bar{f}_{\mathrm{S}i}$, and body forces, $\bar{f}_{\mathrm{B}i}$, denoted $W_{\mathrm{C}}^{(p)}$, W_{S} and W_{B} respectively, is given by:

$$W_{\mathrm{C}}^{(p)} = f_i^{(p)} u_i^{(p)} \tag{8.4}$$

$$W_{\mathrm{S}} = \int_A \bar{f}_{\mathrm{S}i} \bar{u}_{\mathrm{S}i} \, \mathrm{d}A \tag{8.5}$$

$$W_{\mathrm{B}} = \int_V \bar{f}_{\mathrm{B}i} \bar{u}_i \, \mathrm{d}V \tag{8.6}$$

where $u_i^{(p)}$ is the displacement of point p, and $\bar{u}_{\mathrm{S}i}$ and $\bar{u}_i$ are the displacements of the surface with domain A, and body with domain V respectively. The total external work, W_{Ext}, done on a deformable body during the displacements $u_i^{(p)}$, $\bar{u}_{\mathrm{S}i}$ and $\bar{u}_i$ can therefore be defined as:

$$W_{\mathrm{Ext}} = \int_V \bar{f}_{\mathrm{B}i} \bar{u}_i \, \mathrm{d}V + \int_A \bar{f}_{\mathrm{S}i} \bar{u}_{\mathrm{S}i} \, \mathrm{d}A + \sum_p f_i^{(p)} u_i^{(p)} \tag{8.7}$$

Internal work during displacements is defined using the concept of strain energy for a body, U:

$$U_{\mathrm{D}} = \int_0^{\bar{\epsilon}_{\mathrm{B}ij}} \bar{\sigma}_{ij}(\bar{\epsilon}_{ij}) \, \mathrm{d}\bar{\epsilon}_{ij} \tag{8.8}$$

$$U = \int_V U_{\mathrm{D}} \, \mathrm{d}V \tag{8.9}$$

where U_{D} is the strain energy density, and $\bar{\epsilon}_{ij}$ and $\bar{\sigma}_{ij}(\bar{\epsilon}_{ij})$ are the strain and stress tensors respectively, with $\bar{\epsilon}_{\mathrm{B}ij}$ denoting the actual strain values for the body. To compute internal work, we again consider the simple case whereby material particles are displaced along straight trajectories, and the body maintains a constant volume. In this case, the internal work, W_{Int}, due to stresses during strains corresponding to the displacements $\bar{u}_i$ can be computed as the difference between the current strain energy after the displacements, U, with strains $\bar{\epsilon}_{\mathrm{B}ij}$, and the initial strain energy before the displacements, U_{I}, with strains $\bar{\epsilon}_{\mathrm{I}ij}$:

$$W_{\mathrm{Int}} = U - U_{\mathrm{I}} = \int_V \int_{\bar{\epsilon}_{\mathrm{I}ij}}^{\bar{\epsilon}_{\mathrm{B}ij}} \bar{\sigma}_{ij}(\bar{\epsilon}_{ij}) \, \mathrm{d}\bar{\epsilon}_{ij} \, \mathrm{d}V \tag{8.10}$$

Using Equations 8.7 and 8.10, the total work, W, is defined as:

$$W = W_{\mathrm{Ext}} - W_{\mathrm{Int}} \tag{8.11}$$

where the internal work is negated since internal forces oppose external forces.

Analogous to work, virtual work δW is done by forces during virtual displacements $\delta u_i^{(p)}$, $\delta \bar{u}_{\mathrm{S}i}$, and $\delta \bar{u}_i$, corresponding to which are the virtual strains $\delta \bar{\epsilon}_{ij}$. On a body, a virtual displacement distribution $\delta \bar{u}_i$ is a possible (valid) displacement distribution at a particular instance of time, satisfying any constraints on the body at that time, that hasn't actually taken place (it is imaginary).

[1] In reality, points and material particles may move along curved trajectories while causing a change in area or volume of a surface or body. Additionally, the magnitudes and directions of forces may change during displacements, for example, as a function of time, particle positions or both. However, the consideration of such varied and complex cases isn't necessary for the explanation of virtual work.

Unless the body is completely fixed, or only an infinitesimal virtual displacement distribution satisfies the constraints[2], there are infinitely many possible virtual displacement distributions of the body, which will move along one such displacement path if it is not in equilibrium. An analogous description can be deduced for virtual displacements of a point and surface.

Virtual displacements occur at a particular instance of time without passage of time, meaning the configuration of the point, surface or body remains constant. As such, forces are held constant, as is the area of a surface, and volume of a body, and the virtual displacements must follow a straight trajectory (i.e. a virtual displacement of a point is a vector from the real to a virtual trajectory at the current time). As we also imposed these conditions with our definition of external real work in Equation 8.7, external virtual work, δW_{Ext}, during virtual displacements $\delta u_i^{(p)}$, $\delta \bar{u}_{\text{S}i}$, and $\delta \bar{u}_i$ can be calculated in an identical manner to how external real work is computed:

$$\delta W_{\text{Ext}} = \int_V \bar{f}_{\text{B}i} \delta \bar{u}_i \, \mathrm{d}V + \int_A \bar{f}_{\text{S}i} \delta \bar{u}_{\text{S}i} \, \mathrm{d}A + \sum_p f_i^{(p)} \delta \bar{u}_i^{(p)} \tag{8.12}$$

With internal real work as defined in Equation 8.10, body stresses will change during strains, and therefore cannot be held constant. However, as the configuration of a body, including stresses, is held constant during virtual displacements, internal virtual work, δW_{Int}, during virtual strains $\delta \bar{\epsilon}_{ij}$ can instead be computed analogously to virtual work of body forces:

$$\delta W_{\text{Int}} = \int_V \bar{\sigma}_{ij} \delta \bar{\epsilon}_{ij} \, \mathrm{d}V \tag{8.13}$$

Using Equations 8.12 and 8.13, the total virtual work, δW, is defined as:

$$\delta W = \delta W_{\text{Ext}} - \delta W_{\text{Int}} \tag{8.14}$$

8.2.5 The Principle of Virtual Work

The equilibrium of a body can be expressed using the principle of virtual work for deformable bodies, which is the basis of the displacement-based FE method. This states that, for a body to be in equilibrium, the total virtual work of all forces acting on the body must equal zero for every virtual displacement imposed on the body (i.e. the resultant forces over the entire body must equal zero). From Equation 8.14, this means that the total virtual work of internal forces acting on the body must equal the total virtual work of external forces for all such virtual displacements:

$$\int_V \bar{\sigma}_{ij} \delta \bar{\epsilon}_{ij} \, \mathrm{d}V = \int_V \bar{f}_{\text{B}i} \delta \bar{u}_i \, \mathrm{d}V + \int_A \bar{f}_{\text{S}i} \delta \bar{u}_{\text{S}i} \, \mathrm{d}A + \sum_p f_i^{(p)} \delta \bar{u}_i^{(p)} \tag{8.15}$$

Equation 8.15 can be used to solve for the static equilibrium of a body; however, we are interested in the time-varying deformation of the body to produce animation. This could be done in a quasistatic manner by solving Equation 8.15 various times with gradually increasing loads. While this approach doesn't consider inertial effects (e.g. oscillatory elastic behaviour), such effects aren't usually present with soft-tissue deformations, which have a highly viscous and damped response. However, for our FE simulations, we use an explicit time-integration scheme due to the small, efficient timesteps for accurate simulation of complex behaviour, and the inherently parallel nature of such techniques. With explicit time integration, it is necessary to include inertial effects, and solve for the dynamic equilibrium of a body over time.

The equations for dynamic equilibrium can be formulated using d'Alembert's principle by introducing inertial forces into Equation 8.15. Inertial forces are fictitious forces that oppose the directions of acceleration to balance all forces acting on the body, enabling it to be analysed as a

[2]Virtual displacements are often required to be infinitesimal to be compatible with the constraints of a system at an instance of time (e.g. a straight virtual displacement of a pendulum), although they can theoretically be of any magnitude, which is the case in our problem for an arbitrary deformable body.

static system. Therefore, by d'Alembert's principle, the total virtual work of all forces acting on a body (given by Equation 8.14) plus that of the inertial forces is zero for reversible displacements:

$$\delta W - \int_V (\bar{\rho}\ddot{\bar{u}}_j)\delta\bar{u}_i \, \mathrm{d}V = 0 \tag{8.16}$$

where $\bar{\rho}$ is the mass density distribution of the body.

In reality, energy is dissipated during dynamic responses, for example, due to the various types of friction acting on material particles as they slide against each other during deformation. While such damping is very difficult to accurately model, it is usually approximated using velocity-dependent damping forces that reduce the magnitude of the inertial forces while the body is in motion:

$$\delta W - \int_V (\bar{\rho}\ddot{\bar{u}}_j - \bar{\kappa}\dot{\bar{u}}_j)\delta\bar{u}_i \, \mathrm{d}V = 0 \tag{8.17}$$

where $\bar{\kappa}$ is the damping parameter distribution for the body. Over time, this causes the acceleration converge to a constant of zero at an equilibrium state. Without such energy dissipation, a dynamic system with unbalanced forces acting on it would oscillate indefinitely and never converge to a static equilibrium state. Using the principle of virtual work (Equation 8.15) and d'Alembert's principle with damping (Equation 8.17), the equations of dynamic equilibrium that form the basis of the dynamic displacement-based FE method can be written as:

$$\int_V (\bar{\rho}\ddot{\bar{u}}_j - \bar{\kappa}\dot{\bar{u}}_j)\delta\bar{u}_i + \bar{\sigma}_{ij}\delta\bar{\epsilon}_{ij} \, \mathrm{d}V = \int_V \bar{f}_{\mathrm{B}i}\delta\bar{u}_i \, \mathrm{d}V + \int_A \bar{f}_{\mathrm{S}i}\delta\bar{u}_{\mathrm{S}i} \, \mathrm{d}A + \sum_p f_i^{(p)}\delta\bar{u}_i^{(p)} \tag{8.18}$$

These equations hold for any internal stresses and external loads, whether the body is in a true (static or mechanical) equilibrium state or not.

8.2.6 Deformation, Stress and Strain

As the loads will change over time throughout our simulations, for example, as different muscles are contracted and relaxed, and we are interested in the time-varying deformation of a body to produce animation, Equation 8.18 needs to incorporate a time parameter, t, to describe the response of a body over time. An appropriate stress and strain measure must also be used for accurate results. Such quantities must be measured with respect to a particular reference configuration of the body. Using the TL formulation of the FE method, these quantities are measured with respect to the initial body configuration; therefore, the stress and strain measures must also be able to accurately handle the large deformations and rotations that can occur from the initial to the deformed body states.

The fundamental measure of the deformation of a body used when computing stresses and strains is the deformation gradient tensor, ${}^t_0\bar{X}_{ij}$, which describes the stretches and rotations of material fibres. In matrix form, it is a Jacobian matrix describing the transformation of tangent vectors (material fibres) from the reference configuration of the body to the current configuration. It can be formulated in terms of the displacement gradient tensor, ${}^t_0\bar{u}_{i,j}$, which describes the transformation of the displacement distribution:

$${}^t_0\bar{u}_{i,j} = \frac{\partial\, {}^t\bar{u}_i}{\partial\, {}^0x_j} \tag{8.19}$$

$$\begin{aligned} {}^t_0\bar{X}_{ij} &= {}^t_0\bar{x}_{i,j} = \frac{\partial\, {}^t\bar{x}_i}{\partial\, {}^0x_j} \\ &= \delta_{ij} + {}^t_0\bar{u}_{i,j} \end{aligned} \tag{8.20}$$

$${}^t\bar{J} = |\,{}^t_0\bar{X}_{ij}| = \frac{{}^0\bar{\rho}}{{}^t\bar{\rho}} \tag{8.21}$$

where ${}^t\bar{J}$ is the Jacobian (determinant) of the deformation gradient, which is a measure of volume change due to deformation. In this notation, the use of a comma preceding a lowercase subscript

denotes partial differentiation with respect to a global coordinate. Furthermore, a left superscript indicates the configuration in which a quantity occurs, and a left subscript indicates the configuration with respect to which a quantity is measured (t denotes the current, and 0 denotes the initial configuration).

To measure the magnitude of fibre stretches and rotations, the right Cauchy-Green deformation tensor, ${}^{t}_{0}\bar{C}_{ij}$, can be used:

$$ {}^{t}_{0}\bar{C}_{ij} = {}^{t}_{0}\bar{X}_{ki}\,{}^{t}_{0}\bar{X}_{kj} \tag{8.22} $$

Note that, in the above definitions, spatial coordinates in the reference configuration, ${}^{0}x_i$, are used as material coordinates (${}^{0}x_i = X_i$), and the Cartesian axes provide an orthogonal material basis for such coordinates. Also note that the body density, ${}^{t}\bar{\rho}$, may differ from the material density, $\bar{\rho}$, depending on the body deformation; however, it is often equal to the material density when in the initial configuration (${}^{0}\bar{\rho}$) assuming the material is not compressed or stretched in this configuration.

The stress and strain tensors can be defined in terms of the deformation gradients. With the TL FE method, the Green-Lagrange strain tensor, ${}^{t}_{0}\bar{E}_{ij}$, is used, which quantifies body strains with respect to the reference configuration of the material particles, enabling large deformations and rotations to be handled:

$$ \begin{aligned} {}^{t}_{0}\bar{E}_{ij} &= \frac{1}{2}({}^{t}_{0}\bar{u}_{i,j} + {}^{t}_{0}\bar{u}_{j,i} + {}^{t}_{0}\bar{u}_{k,i}\,{}^{t}_{0}\bar{u}_{k,j}) \\ &= \frac{1}{2}({}^{t}_{0}\bar{C}_{ij} - \delta_{ij}) \end{aligned} \tag{8.23} $$

As such, the work-conjugate second Piola-Kirchhoff stress measure, ${}^{t}_{0}\bar{S}_{ij}$, is also used, which quantifies body stresses in material directions with respect to the reference configuration. This is related to the Cauchy (true) stress measure, ${}^{t}\bar{\sigma}_{ij}$, which quantifies forces per unit area in the deformed configuration, as follows:

$$ {}^{t}_{0}\bar{S}_{ij} = {}^{t}\bar{J}\,{}^{0}_{t}\bar{X}_{im}\,{}^{t}\bar{\sigma}_{mn}\,{}^{0}_{t}\bar{X}_{jn} \tag{8.24} $$

where ${}^{0}_{t}\bar{X}_{ij} = {}^{t}_{0}\bar{X}^{-1}_{ij}$. In this transformation of the Cauchy stress, ${}^{0}_{t}\bar{X}_{ij}$ essentially scales and rotates the internal forces to map them from the current back to the reference configuration, while ${}^{t}\bar{J}$ transforms the area they act upon back to the reference configuration. The Cauchy stress, which is defined in spatial coordinates using the spatial basis, is therefore transformed so that it is defined with respect to the material basis (i.e. the spatial basis of material points in the initial reference configuration, as explained in Section 8.2.2).

The second Piola-Kirchhoff stresses can be computed directly from the Green-Lagrange strains using the relevant constitutive equation for the material of the body. For example, with a hyperelastic material, the stress-strain relationship is expressed using a strain energy density function, $W({}^{t}_{0}\bar{X}_{ij})$. Using the definition of strain energy density in terms of stresses and strains in Equation 8.8, stresses can be computed as:

$$ {}^{t}_{0}\bar{S}_{ij} = \frac{\delta W({}^{t}_{0}\bar{X}_{ij})}{{}^{t}_{0}\bar{E}_{ij}} \tag{8.25} $$

As an example, the stresses for the simple hyperelastic neo-Hookean material, which we have used for all of our simulations, can be computed as follows:

$$ {}^{t}_{0}\bar{S}_{ij} = \bar{\mu}(\delta_{ij} - {}^{t}_{0}\bar{C}^{-1}_{ij}) + \bar{\lambda}\,{}^{t}\bar{J}({}^{t}\bar{J} - 1)\,{}^{t}_{0}\bar{C}^{-1}_{ij} \tag{8.26} $$

where $\bar{\mu}$ and $\bar{\lambda}$ are the distributions of the material Lamé parameters over the body. Since ${}^{t}_{0}\bar{E}_{ij}$ and ${}^{t}_{0}\bar{S}_{ij}$ are symmetric, they can be used to define a Green-Lagrange strain vector, ${}^{t}_{0}\hat{\bar{E}}_{i}$, and a second Piola-Kirchhoff stress vector, ${}^{t}_{0}\hat{\bar{S}}_{i}$, respectively, like those in Equations 8.2 and 8.3:

$$ {}^{t}_{0}\hat{\bar{\mathbf{E}}} = \begin{pmatrix} {}^{t}_{0}\bar{E}_{11} & {}^{t}_{0}\bar{E}_{22} & {}^{t}_{0}\bar{E}_{33} & 2\,{}^{t}_{0}\bar{E}_{12} & 2\,{}^{t}_{0}\bar{E}_{23} & 2\,{}^{t}_{0}\bar{E}_{13} \end{pmatrix} \tag{8.27} $$

$$ {}^{t}_{0}\hat{\bar{\mathbf{S}}} = \begin{pmatrix} {}^{t}_{0}\bar{S}_{11} & {}^{t}_{0}\bar{S}_{22} & {}^{t}_{0}\bar{S}_{33} & {}^{t}_{0}\bar{S}_{12} & {}^{t}_{0}\bar{S}_{23} & {}^{t}_{0}\bar{S}_{13} \end{pmatrix} \tag{8.28} $$

8.2.7 Equilibrium of a Deformable Body

Using Equations 8.27 and 8.28, the dynamic equilibrium of a deformable body from Equation 8.18 can now be expressed as:

$$\int_{{}^0V} ({}^0\bar{\rho}\,{}^t\ddot{\bar{u}}_j + {}^t_0\bar{\kappa}\,{}^t\dot{\bar{u}}_j)\,{}^t\delta\bar{u}_i + {}^t_0\bar{S}_{ij}\,{}^t_0\delta\bar{E}_{ij}\,\mathrm{d}\,{}^0V = \int_{{}^0V} {}^t_0\bar{f}_{\mathrm{B}i}\,{}^t\delta\bar{u}_i\,\mathrm{d}\,{}^0V + \int_{{}^0A} {}^t_0\bar{f}_{\mathrm{S}i}\,{}^t\delta\bar{u}_{\mathrm{S}i}\,\mathrm{d}\,{}^0A + \sum_p {}^t f_i^{(p)}\,{}^t\delta u_i^{(p)} \tag{8.29}$$

Note that the body and surface forces are expressed as forces per unit of initial volume and area respectively since these values are known and don't need to be updated during the solution process.[3] The dynamic equilibrium for a deformable continuum in Equation 8.29 describes the time-varying deformation of a body, which can theoretically be solved or numerically approximated using a time-integration scheme. However, this still contains numerous partial differential equations with numerous unknown values that are difficult to solve analytically for complex objects like soft tissue.

Similar to the discretisation of a continuous surface into polygons interconnected at vertices, with the FE method, the complex domain of a body is discretised into elements, each of which occupy a simple subdomain of the body, that are interconnected at nodes on element boundaries. Simpler equilibrium equations can be formulated for these simpler element geometries, as described in Sections 8.2.8 to 8.2.12, which are then combined to approximate the equilibrium equations and solution for the entire continuum, as explained in Section 8.2.13. The equations formulated so far for an arbitrary deformable body continuum also hold for subdomains of the body, and therefore individual elements.

8.2.8 Element Shape Functions

To model the distributions of values across an element, such as a displacement distribution, shape functions are chosen to approximate the variation of values within an element in terms of the element nodal values. A shape function, $\check{h}_i$, is defined for each element node i that interpolates from a value of 1 at the corresponding nodal position, to a value of 0 at the positions of each other element node, with the condition that the sum of the shape functions always evaluates to 1 ($\sum_{i=1}^{n} \check{h}_i = 1$). Such functions can be easily defined in terms of element local coordinates. The distribution of a value across an element with n nodes, ${}^t\check{v}$, can be represented in terms of the shape functions as:

$${}^t\check{v} = \sum_{i=1}^{n} \check{h}_i\,{}^t v^{(i)} = \check{h}_i\,{}^t V_{\mathrm{E}i} \tag{8.30}$$

where ${}^t v^{(i)}$ is the value of ${}^t v$ at node i, and ${}^t V_{\mathrm{E}i}$ is a vector of element nodal values. The check notation (e.g. $\check{v}$) is used to denote a continuous function (distribution) of values for local coordinates $\bar{\zeta}_i$.

As shown by Figure 8.1, for a linear 8-node hexahedral element, a local coordinate system ζ_i for $1 \leq i \leq 3$ can be defined, and each orthogonal axis runs from -1 at the centre of a face, to 1 at the centre of the opposite face ($-1 \leq \zeta_i \leq 1$), meeting at the origin in the centre ($\zeta_i = 0$).[4] Numbering the nodes as shown in Figure 8.1 to ensure a positive Jacobian of the deformation gradient,[5] the shape functions can be concisely written as:

$$\check{h}_i = \frac{1}{8}(1 + \zeta_1\zeta_1^{(i)})(1 + \zeta_2\zeta_2^{(i)})(1 + \zeta_3\zeta_3^{(i)}) \tag{8.31}$$

[3] The body and surface forces could also be expressed as forces per unit of current volume and area respectively, provided the integrals are changed to match these domains.

[4] In local coordinate space, the spatial coordinates and basis of a material point remain fixed over time since elements don't deform geometrically in local coordinate space; for example, $\boldsymbol{\zeta} = (-1, -1, -1)$ at element node 1. Local coordinates can therefore be used to identify material points instead of material coordinates in global coordinate space.

[5] More detail on local coordinate systems and element node numbering can be found in the literature [Fel13].

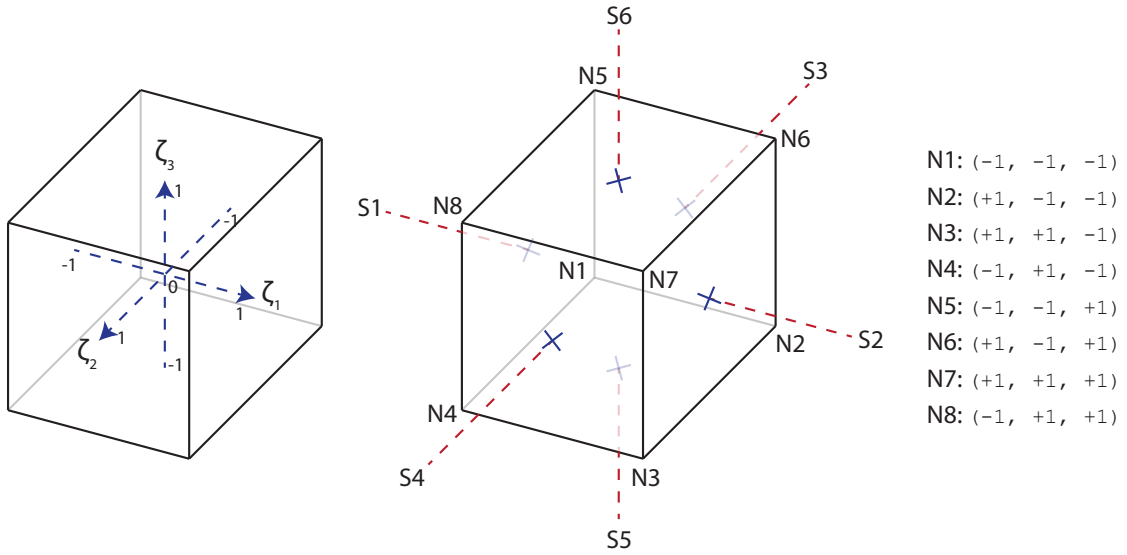

Figure 8.1: The local coordinate system, and node and surface numbering we use for 8-node hexahedral elements. The local coordinates of each node is also shown.

where $\zeta_j^{(i)}$ is the value of ζ_j at node i. Using these shape functions, a displacement distribution for an element can be defined:

$$ {}^t\check{u}_i = \sum_{j=1}^{8} \check{h}_j \, {}^t u_i^{(j)} = \check{H}_{ij} \, {}^t U_{\mathrm{E}j} \tag{8.32} $$

where $\check{H}_{ij}$ is the element interpolation matrix created from the shape functions, and ${}^tU_{\mathrm{E}i}$ is the element nodal displacement vector containing the displacements for all nodes connected to the element:

$$ \check{\mathbf{H}} = \begin{pmatrix} \check{\mathbf{H}}^{(1)} & \check{\mathbf{H}}^{(2)} & \cdots & \check{\mathbf{H}}^{(8)} \end{pmatrix} \tag{8.33} $$

$$ \check{\mathbf{H}}^{(a)} = \begin{pmatrix} \check{h}_a & 0 & 0 \\ 0 & \check{h}_a & 0 \\ 0 & 0 & \check{h}_a \end{pmatrix} \tag{8.34} $$

$$ {}^t\mathbf{U}_{\mathrm{E}}^{\mathrm{T}} = \begin{pmatrix} {}^t\mathbf{u}^{(1)\mathrm{T}} & {}^t\mathbf{u}^{(2)\mathrm{T}} & \cdots & {}^t\mathbf{u}^{(8)\mathrm{T}} \end{pmatrix} \tag{8.35} $$

$\check{H}_{ij}^{(a)}$ is the submatrix of the element interpolation matrix for element node a.

The shape functions can also be used to define a displacement distribution for each surface of an element, s, where $1 \leq s \leq 6$ for a hexahedron. Using the surface numbering in Figure 8.1, the element shape functions in Equation 8.31 can be modified to define shape functions that are

restricted to each such surface, $\check{h}^{(s)}_{\mathrm{S}i}$, for all ζ_i as follows:

$$\check{h}^{(1)}_{\mathrm{S}i} = \frac{1}{8}(1-\zeta_1^{(i)})(1+\zeta_2\zeta_2^{(i)})(1+\zeta_3\zeta_3^{(i)}) \tag{8.36}$$

$$\check{h}^{(2)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_1^{(i)})(1+\zeta_2\zeta_2^{(i)})(1+\zeta_3\zeta_3^{(i)}) \tag{8.37}$$

$$\check{h}^{(3)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_1\zeta_1^{(i)})(1-\zeta_2^{(i)})(1+\zeta_3\zeta_3^{(i)}) \tag{8.38}$$

$$\check{h}^{(4)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_1\zeta_1^{(i)})(1+\zeta_2^{(i)})(1+\zeta_3\zeta_3^{(i)}) \tag{8.39}$$

$$\check{h}^{(5)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_1\zeta_1^{(i)})(1+\zeta_2\zeta_2^{(i)})(1-\zeta_3^{(i)}) \tag{8.40}$$

$$\check{h}^{(6)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_1\zeta_1^{(i)})(1+\zeta_2\zeta_2^{(i)})(1+\zeta_3^{(i)}) \tag{8.41}$$

$\check{h}^{(s)}_{\mathrm{S}i}$ therefore always evaluates to a location on surface s. The element surface displacement distribution for an element surface can now be defined in terms of ${}^{t}U_{\mathrm{E}j}$ (like the element body displacement distribution in Equation 8.32):

$${}^{t}\check{u}^{(s)}_{\mathrm{S}i} = \sum_{i=1}^{8} \check{h}^{(s)}_{\mathrm{S}j}\, {}^{t}u_i^{(j)} = \check{H}^{(s)}_{\mathrm{S}ij}\, {}^{t}U_{\mathrm{E}j} \tag{8.42}$$

where $\check{H}^{(s)}_{\mathrm{S}ij}$ is the surface interpolation matrix, which can be created using Equation 8.33.

8.2.9 Element Strains

While Equation 8.32 expresses element displacements as a function of local coordinates, to compute the strains (Equation 8.23), it is necessary to express partial derivatives of an element displacement distribution with respect to material coordinates in the global coordinate space in which displacements are defined.[6] To do this, Equation 8.32 can be differentiated with respect to global material coordinates:

$${}^{t}_{0}\check{u}_{i,j} = {}_{0}\check{h}_{k,j}\, {}^{t}u_i^{(k)} \tag{8.43}$$

It is therefore necessary to compute the element shape function derivatives with respect to global coordinates. Using the chain rule, the shape function derivatives for an element can be expressed as:

$${}_{0}\check{h}_{i,j} = \frac{\partial \check{h}_i}{\partial \zeta_k}\frac{\partial \bar{\zeta}_k}{\partial\, {}^{0}x_j} \tag{8.44}$$

In matrix form, the term $\dfrac{\partial \bar{\zeta}_k}{\partial\, {}^{0}x_j}$ is the inverse transpose of the Jacobian matrix of global material with respect to the element local coordinates, ${}^{0}\check{J}_{\mathrm{E}ij} = \dfrac{\partial\, {}^{0}\check{x}_i}{\partial \zeta_j}$, which describes the transformation of tangent vectors (material fibres) from local to global material coordinates. As the global position (coordinates) of a point inside the domain of an element can be computed from local coordinates analogously to displacements in Equation 8.32, components of this Jacobian matrix can be easily computed:

$${}^{t}\check{J}_{\mathrm{E}_{ij}} = \frac{\partial\, {}^{t}\check{x}_i}{\partial \zeta_j} = \frac{\partial \check{h}_k}{\partial \zeta_j}\, {}^{t}x_i^{(k)} \tag{8.45}$$

Substituting into Equation 8.44, the shape function derivatives for an element can be expressed as:

$${}_{0}\check{h}_{i,j} = {}^{0}\check{J}^{-1}_{\mathrm{E}kj}\frac{\partial \check{h}_i}{\partial \zeta_k} \tag{8.46}$$

[6] Strains and stresses cannot be easily computed with respect to the element local coordinates as, in global coordinate space (in which displacements are defined), the basis is not necessarily orthogonal, particularly as it deforms with an element.

Using Equation 8.43, an element strain-displacement matrix, ${}^{t}_{0}\check{B}_{ij}$, can be constructed to relate element strains to element nodal displacements:

$$ {}^{t}_{0}\hat{\check{E}}_{i} = {}^{t}_{0}\check{B}_{ij}\, {}^{t}U_{\mathrm{E}j} \tag{8.47} $$

Based on the strain definition (Equation 8.23), element strains can be written in terms of the element nodal displacements as:

$$ {}^{t}_{0}\check{E}_{ij} = \frac{1}{2}\left({}_{0}\check{h}_{p,j}\, {}^{t}u^{(p)}_{i} + {}_{0}\check{h}_{p,i}\, {}^{t}u^{(p)}_{j} + {}_{0}\check{h}_{p,i}\, {}^{t}u^{(p)}_{k}\, {}_{0}\check{h}_{q,j}\, {}^{t}u^{(q)}_{k} \right) \tag{8.48} $$

The element strain-displacement matrix is therefore defined as:

$$ {}^{t}_{0}\check{\mathbf{B}} = \begin{pmatrix} {}^{t}_{0}\check{\mathbf{B}}^{(1)} & {}^{t}_{0}\check{\mathbf{B}}^{(2)} & \cdots & {}^{t}_{0}\check{\mathbf{B}}^{(8)} \end{pmatrix} \tag{8.49} $$

$$ {}^{t}_{0}\check{\mathbf{B}}^{(a)} = {}^{t}_{0}\check{\mathbf{B}}^{(a)}_{0} + {}^{t}_{0}\check{\mathbf{B}}^{(a)}_{1} \tag{8.50} $$

$$ {}^{t}_{0}\check{\mathbf{B}}^{(a)}_{0} = \begin{pmatrix} {}_{0}\check{h}_{a,1} & 0 & 0 \\ 0 & {}_{0}\check{h}_{a,2} & 0 \\ 0 & 0 & {}_{0}\check{h}_{a,3} \\ {}_{0}\check{h}_{a,2} & {}_{0}\check{h}_{a,1} & 0 \\ 0 & {}_{0}\check{h}_{a,3} & {}_{0}\check{h}_{a,2} \\ {}_{0}\check{h}_{a,3} & 0 & {}_{0}\check{h}_{a,1} \end{pmatrix} \tag{8.51} $$

$$ {}^{t}_{0}\check{\mathbf{B}}^{(a)}_{1} = \begin{pmatrix} {}^{t}_{0}\check{u}_{1,1}\, {}_{0}\check{h}_{a,1} & {}^{t}_{0}\check{u}_{2,1}\, {}_{0}\check{h}_{a,1} & {}^{t}_{0}\check{u}_{3,1}\, {}_{0}\check{h}_{a,1} \\ {}^{t}_{0}\check{u}_{1,2}\, {}_{0}\check{h}_{a,2} & {}^{t}_{0}\check{u}_{2,2}\, {}_{0}\check{h}_{a,2} & {}^{t}_{0}\check{u}_{3,2}\, {}_{0}\check{h}_{a,2} \\ {}^{t}_{0}\check{u}_{1,3}\, {}_{0}\check{h}_{a,3} & {}^{t}_{0}\check{u}_{2,3}\, {}_{0}\check{h}_{a,3} & {}^{t}_{0}\check{u}_{3,3}\, {}_{0}\check{h}_{a,3} \\ {}^{t}_{0}\check{u}_{1,1}\, {}_{0}\check{h}_{a,2} + {}^{t}_{0}\check{u}_{1,2}\, {}_{0}\check{h}_{a,1} & {}^{t}_{0}\check{u}_{2,1}\, {}_{0}\check{h}_{a,2} + {}^{t}_{0}\check{u}_{2,2}\, {}_{0}\check{h}_{a,1} & {}^{t}_{0}\check{u}_{3,1}\, {}_{0}\check{h}_{a,2} + {}^{t}_{0}\check{u}_{3,2}\, {}_{0}\check{h}_{a,1} \\ {}^{t}_{0}\check{u}_{1,2}\, {}_{0}\check{h}_{a,3} + {}^{t}_{0}\check{u}_{1,3}\, {}_{0}\check{h}_{a,2} & {}^{t}_{0}\check{u}_{2,2}\, {}_{0}\check{h}_{a,3} + {}^{t}_{0}\check{u}_{2,3}\, {}_{0}\check{h}_{a,2} & {}^{t}_{0}\check{u}_{3,2}\, {}_{0}\check{h}_{a,3} + {}^{t}_{0}\check{u}_{3,3}\, {}_{0}\check{h}_{a,2} \\ {}^{t}_{0}\check{u}_{1,1}\, {}_{0}\check{h}_{a,3} + {}^{t}_{0}\check{u}_{1,3}\, {}_{0}\check{h}_{a,1} & {}^{t}_{0}\check{u}_{2,1}\, {}_{0}\check{h}_{a,3} + {}^{t}_{0}\check{u}_{2,3}\, {}_{0}\check{h}_{a,1} & {}^{t}_{0}\check{u}_{3,1}\, {}_{0}\check{h}_{a,3} + {}^{t}_{0}\check{u}_{3,3}\, {}_{0}\check{h}_{a,1} \end{pmatrix} \tag{8.52} $$

${}^{t}_{0}\check{B}^{(a)}_{ij}$ is the submatrix of the strain-displacement matrix for element node a. It can be seen that ${}^{t}_{0}\check{B}_{1ij}$, and therefore ${}^{t}_{0}\check{B}_{ij}$, depends on the nodal displacements as well as shape function derivatives. This makes the strain-displacement relationship in Equation 8.47 nonlinear, enabling geometric nonlinearities to be handled using the nonlinear Green-Lagrange strain measure.

Rather than explicitly computing ${}^{t}_{0}\check{B}^{(a)}_{1ij}$ to formulate ${}^{t}_{0}\check{B}^{(a)}_{ij}$ in Equation 8.50, we can instead transform ${}^{t}_{0}\check{B}^{(a)}_{0ij}$ using the deformation gradient for the element, which is also required to compute stresses (defined in Equation 8.25). Based on the definition of the deformation gradient (Equation 8.20), an element deformation gradient can be formulated using Equation 8.43:

$$ {}^{t}_{0}\check{X}_{ij} = \delta_{ij} + {}_{0}\check{h}_{p,j}\, {}^{t}u^{(p)}_{i} \tag{8.53} $$

We can now compute ${}^{t}_{0}\check{B}^{(a)}_{ij}$ using the transpose of the element deformation gradient:

$$ {}^{t}_{0}\check{B}^{(a)}_{ij} = {}^{t}_{0}\check{B}^{(a)}_{0ik}\, {}^{t}_{0}\check{X}_{jk} \tag{8.54} $$

8.2.10 Multi-Material Element Stresses

As we use non-conforming models, the elements of which don't conform to the surfaces of the model surface mesh, an element may overlap multiple different mesh volumes enclosed by surfaces of the surface mesh, as explained in Section 7.2. Since different materials may be associated with such mesh volumes, rather than computing stresses for an element using a single constitutive equation, such stresses are computed using multiple weighted constitutive equations:

$$ {}^{t}_{0}\check{S}_{ij} = \sum_{m \in M} o^{(em)}\, {}^{t}_{0}\check{S}^{(m)}_{ij} \tag{8.55} $$

where M is the set of all mesh volumes, $o^{(em)} \in [0,1]$ is the proportion of overlap between the element and mesh volume m, and ${}^{t}_{0}\check{S}^{(m)}_{ij}$ are the element stresses for mesh volume m computed using the constitutive equation associated with m. Additional element muscle stresses, explained in Section 8.4, are also added to the element material stresses in Equation 8.55.

The element strains and deformation gradient in Equations 8.47 and 8.53 are defined in terms of local coordinates; therefore, by using these in the stress definition (Equation 8.25), element stresses for each mesh volume are also defined in terms of local coordinates. For example, the element stresses for a mesh volume represented by a hyperelastic neo-Hookean material can be computed as follows:

$$
\begin{aligned}
{}^{t}_{0}\check{S}^{(m)}_{ij} &= \frac{\delta W^{(m)}({}^{t}_{0}\check{X}_{ij})}{{}^{t}_{0}\check{E}_{ij}} && (8.56)\\
&= \mu^{(m)}(\delta_{ij} - {}^{t}_{0}\check{C}^{-1}_{ij}) + \lambda^{(m)}\,{}^{t}\check{J}({}^{t}\check{J} - 1)\,{}^{t}_{0}\check{C}^{-1}_{ij} && (8.57)
\end{aligned}
$$

where $\delta W^{(m)}$, and $\mu^{(m)}$ and $\lambda^{(m)}$ are the strain energy density function and element material Lamé parameters[7] respectively for the material associated with mesh volume m.

To simplify explanation, and for consistency with the description of the model creation process in Chapter 7, Equation 8.55 has been written in terms of mesh volumes. However, in practice, it is more efficient to associate weighted material models with elements, rather than weighted mesh volumes, since the number of material models will often be less than (but at most equal to) the number of mesh volumes. For example, the same material model (either with the same or different material parameters) may be associated with multiple mesh volumes overlapping an element. In this case, the materials for these mesh volumes can be combined to produce an average material; the proportions of overlap (weights) for these mesh volumes can be summed to produce a weight for the material model, and also used to compute weighted average material parameters for the material model. The constitutive equation for this material model can then be evaluated once using a single set of material parameters, rather than multiple times with different parameters for each mesh volume. With this formulation, M in Equation 8.55 would represent a set of material models, rather than mesh volumes.

8.2.11 Element Internal Virtual Work

For the computation of element internal virtual work, it is assumed that the relation for element strains in Equation 8.47 also holds to compute virtual strains in terms of element nodal values:

$$
{}^{t}_{0}\delta\hat{\check{E}}_{i} = {}^{t}_{0}\check{B}_{ij}\,{}^{t}\delta U_{\mathrm{E}j} \qquad (8.58)
$$

Using the definition of internal virtual work (Equation 8.10), the internal virtual work done by an element can be computed as:

$$
\begin{aligned}
\delta W_{\mathrm{Int}} &= \int_{{}^{0}V} {}^{t}_{0}\delta\hat{\check{E}}_{i}\,{}^{t}_{0}\hat{\check{S}}_{i}\,\mathrm{d}\,{}^{0}V\\
&= \int_{{}^{0}V} {}^{t}\delta U_{\mathrm{E}i}\,{}^{t}_{0}\check{B}_{ji}\,{}^{t}_{0}\hat{\check{S}}_{j}\,\mathrm{d}\,{}^{0}V \qquad (8.59)
\end{aligned}
$$

where ${}^{0}V$ here is the domain of the element.[8] Since the virtual displacements of element nodes are independent of the integral domain (i.e. they are constant, rather than a function of local

[7] While material parameter distributions could be used to vary material parameters over an element, such detail wouldn't be captured with constant strain elements, like the reduced-integration 8-node hexahedra we use.

[8] In the virtual work definitions in Section 8.2.4, integrals are formulated in terms of the global coordinate system, whereas the element local coordinate system is used in Equation 8.59. Conveniently, any coordinate system can be used to evaluate the integrals when computing the virtual work scalar, provided the same coordinate system is used to define the domain (e.g. volume) and each term over which the integral is being computed.

coordinates), Equation 8.59 can be rewritten as:

$$\delta W_{\text{Int}} = {}^{t}\delta U_{\text{E}i}\, {}^{t}F_{\text{E}i} \tag{8.60}$$

$${}^{t}F_{\text{E}i} = {}^{t}K_{\text{E}ij}({}^{t}U_{\text{E}i})\, {}^{t}U_{\text{E}j} = \int_{{}^{0}V} {}^{t}_{0}\check{B}_{ji}\, {}^{t}_{0}\hat{\check{S}}_{j}\, \mathrm{d}\, {}^{0}V \tag{8.61}$$

where ${}^{t}F_{\text{E}i}$ is the element internal force vector, the components of which are analogous to those of the element displacement vector defined in Equation 8.35, and ${}^{t}K_{\text{E}ij}({}^{t}U_{\text{E}i})$[9] is the element stiffness matrix that relates element internal forces to element nodal displacements.

The integral over an element in Equation 8.61 can be approximated using the Gaussian quadrature numerical integration scheme, which uses a weighted sum of function values at discrete integration points. The domain of integration with Gaussian quadrature is conventionally taken as $[-1, 1]$, which is also the domain of the element local coordinates for a hexahedron. Equation 8.61 is therefore firstly rewritten as a combination of 1D functions in terms of the local dimensions:

$${}^{t}F_{\text{E}i} = \int_{\zeta_1}\int_{\zeta_2}\int_{\zeta_3} {}^{t}_{0}\check{B}_{ji}\, {}^{t}_{0}\hat{\check{S}}_{j}\, {}^{0}\check{J}_{\text{E}}\, \mathrm{d}\zeta_3\, \mathrm{d}\zeta_2\, \mathrm{d}\zeta_1 \tag{8.62}$$

where ${}^{0}\check{J}_{\text{E}}$ is the determinant of the Jacobian matrix of global material with respect to element local coordinates (defined in Equation 8.45). This represents the change in volume from the local to global material configurations of an element, and is used to transform the domain of the original integral in Equation 8.61 from global material to local coordinates.

Gaussian quadrature can now be applied to Equation 8.62 using multiple summation:

$${}^{t}F_{\text{E}i} = \sum_{a=1}^{p}\sum_{b=1}^{q}\sum_{c=1}^{r} w_{1a}w_{2b}w_{3c}\left({}^{0}\check{J}_{\text{E}}\, {}^{t}_{0}\check{B}_{ji}\, {}^{t}_{0}\hat{\check{S}}_{j}\right)|_{\zeta_1=\xi_{1a},\zeta_2=\xi_{2b},\zeta_3=\xi_{3c}} \tag{8.63}$$

where p, q and r are the number of integration points in the ζ_1, ζ_2 and ζ_3 dimensions respectively, and ξ_{ij} is the j^{th} integration point in the i^{th} dimension, which is associated with weight w_{ij}. The number of integration points, along with their location and weights, depends on the order of the element, and whether full or reduced integration is used. For increased computational performance and accuracy with soft-tissue simulations, we use reduced-integration 8-node hexahedra, which have a single central integration point (i.e. $p = q = r = 1$, $\zeta_{11} = \zeta_{21} = \zeta_{31} = 0$, and $w_{11} = w_{21} = w_{31} = 2$). Equation 8.63 can therefore be simplified as:

$${}^{t}F_{\text{E}i} = 8\left({}^{0}\check{J}_{\text{E}}\, {}^{t}_{0}\check{B}_{ji}\, {}^{t}_{0}\hat{\check{S}}_{j}\right)|_{\zeta_a=0} \tag{8.64}$$

8.2.12 Element External Virtual Work

For the computation of element external virtual work, it is assumed that the relation for element displacements in Equation 8.32 also holds to compute virtual displacements in terms of element nodal values:

$${}^{t}\delta\check{u}_i = \check{H}_{ij}\, {}^{t}\delta U_{\text{E}j} \tag{8.65}$$

$${}^{t}\delta\check{u}^{(s)}_{\text{S}i} = \check{H}^{(s)}_{\text{S}ij}\, {}^{t}\delta U_{\text{E}j} \tag{8.66}$$

Using the definition of external virtual work (Equation 8.7), the external virtual work done by an element can be computed as:

$$\begin{aligned}\delta W_{\text{Ext}} &= \int_{{}^{0}V} {}^{t}\delta\check{u}_i\, {}^{t}_{0}\check{f}_{\text{B}i}\, \mathrm{d}\, {}^{0}V + \int_{{}^{0}A^{(s)}\in{}^{0}S} {}^{t}\delta\check{u}^{(s)}_{\text{S}i}\, {}^{t}_{0}\check{f}^{(s)}_{\text{S}i}\, \mathrm{d}\, {}^{0}A + \sum_{p=1}^{8} {}^{t}\delta u^{(p)}_{i}\, {}^{t}f^{(p)}_{i} \\ &= \int_{{}^{0}V} {}^{t}\delta U_{\text{E}i}\check{H}_{ji}\, {}^{t}_{0}\check{f}_{\text{B}j}\, \mathrm{d}\, {}^{0}V + \int_{{}^{0}A^{(s)}\in{}^{0}S} {}^{t}\delta U_{\text{E}i}\check{H}^{(s)}_{\text{S}ji}\, {}^{t}_{0}\check{f}^{(s)}_{\text{S}j}\, \mathrm{d}\, {}^{0}A + {}^{t}\delta U_{\text{E}i}\, {}^{t}R_{\text{CE}i}\end{aligned} \tag{8.67}$$

[9] Explicitly stating the stiffness matrix as a function of nodal displacements clarifies the fact that the element internal forces, and therefore the FE formulation, are nonlinear.

where ${}^{t}R_{\mathrm{CE}i}$ is the element concentrated force vector containing the total concentrated force for each element node. It is therefore assumed that such external point forces can only be applied at nodal points (there are 8 such points for a hexahedron). For the surface forces, ${}^{0}S$ is the set of all exposed element surface domains (i.e. element faces that form part of the body surface), where ${}^{0}S = \emptyset$ for elements that are completely surrounded by other elements. Like with element internal virtual work, Equation 8.67 can be rewritten in terms of element force vectors:

$$\delta W_{\mathrm{Ext}} = {}^{t}\delta U_{\mathrm{E}i}\, {}^{t}R_{\mathrm{E}i} \tag{8.68}$$

$${}^{t}R_{\mathrm{E}i} = {}^{t}R_{\mathrm{BE}i} + {}^{t}R_{\mathrm{SE}i} + {}^{t}R_{\mathrm{CE}i} \tag{8.69}$$

$${}^{t}R_{\mathrm{BE}i} = \int_{{}^{0}V} \check{H}_{ji}\, {}^{t}_{0}\check{f}_{\mathrm{B}j}\, \mathrm{d}\, {}^{0}V \tag{8.70}$$

$${}^{t}R_{\mathrm{SE}i} = \int_{{}^{0}A^{(s)} \in {}^{0}S} \check{H}^{(s)}_{\mathrm{S}ji}\, {}^{t}_{0}\check{f}^{(s)}_{\mathrm{S}j}\, \mathrm{d}\, {}^{0}A \tag{8.71}$$

where ${}^{t}R_{\mathrm{E}i}$ is the element external force vector, consisting of element body, surface and nodal point forces, denoted ${}^{t}R_{\mathrm{BE}i}$, ${}^{t}R_{\mathrm{SE}i}$ and ${}^{t}R_{\mathrm{CE}i}$ respectively.

Virtual work done by inertial and damping forces (Equation 8.17), δW_{ID}, can be also be formulated for an element:

$$\begin{aligned}\delta W_{\mathrm{ID}} &= \int_{{}^{0}V} {}^{t}\delta\check{u}_i\, {}^{0}\check{\rho}\, {}^{t}\ddot{\check{u}}_i\, \mathrm{d}\, {}^{0}V + \int_{{}^{0}V} {}^{t}\delta\check{u}_i\, {}^{t}_{0}\check{\kappa}\, {}^{t}\dot{\check{u}}_i\, \mathrm{d}\, {}^{0}V \\ &= \int_{{}^{0}V} {}^{t}\delta U_{\mathrm{E}i}\check{H}_{ki}\, {}^{0}\check{\rho}\check{H}_{kj}\, {}^{t}\ddot{U}_{\mathrm{E}j}\, \mathrm{d}\, {}^{0}V + \int_{{}^{0}V} {}^{t}\delta U_{\mathrm{E}i}\check{H}_{ki}\, {}^{t}_{0}\check{\kappa}\check{H}_{kj}\, {}^{t}\dot{U}_{\mathrm{E}j}\, \mathrm{d}\, {}^{0}V\end{aligned} \tag{8.72}$$

This can be rewritten in terms of the element mass, ${}^{0}M_{\mathrm{E}ij}$, and damping matrices, ${}^{t}C_{\mathrm{E}ij}$:

$$\delta W_{\mathrm{ID}} = {}^{t}\delta U_{\mathrm{E}i}(M_{\mathrm{E}ij}\, {}^{t}\ddot{U}_{\mathrm{E}j} + {}^{t}C_{\mathrm{E}ij}\, {}^{t}\dot{U}_{\mathrm{E}j}) \tag{8.73}$$

$$M_{\mathrm{E}ij} = \int_{{}^{0}V} {}^{0}\check{\rho}\check{H}_{ki}\check{H}_{kj}\, \mathrm{d}\, {}^{0}V \tag{8.74}$$

$${}^{t}C_{\mathrm{E}ij} = \int_{{}^{0}V} {}^{t}_{0}\check{\kappa}\check{H}_{ki}\check{H}_{kj}\, \mathrm{d}\, {}^{0}V \tag{8.75}$$

Note that, since the mass of an element doesn't change, the mass matrix is time independent (i.e. $M_{\mathrm{E}ij} = {}^{0}M_{\mathrm{E}ij} = {}^{t}M_{\mathrm{E}ij}$). For construction of the damping matrix, it is often very difficult to determine damping parameters for a body due to their high dependence on the material properties and mass of the body. However, with Rayleigh damping, a damping matrix can be approximated using the mass and stiffness matrices with experimentally determined or approximated scaling parameters, α and β:

$${}^{t}C_{\mathrm{E}ij} = \alpha M_{\mathrm{E}ij} + \beta\, {}^{t}K_{\mathrm{E}ij} \tag{8.76}$$

As for element internal virtual work, the integrals to compute the element external forces and mass matrix in Equations 8.69 and 8.74 can be approximated using Gaussian quadrature. This produces the element consistent external force vector, and the element consistent mass matrix since the same shape functions are also used to calculate the element stiffness matrix. Evaluation of the integrals using the element interpolation matrix with nodal values lumps forces to element nodes, and element virtual work is then computed from nodal forces and nodal virtual displacements. As this is the case, rather than evaluating the integrals, a technique called lumping can be used. This constructs an approximate element mass matrix and external force vectors by lumping masses to nodal points (producing a diagonal mass matrix), and directly lumping approximate body and surface forces at nodal points with the concentrated loads. Such approximations are commonplace in explicit FE simulations to make efficient computation of the small timesteps practical [Bat96].

Performance-wise, lumping is highly beneficial during the explicit time-integration procedure to compute updated nodal displacements, which requires the computation of the inverse mass and damping matrices. The computation of this procedure is therefore significantly reduced if the

mass and damping matrices are diagonal matrices. As such, we use the mass-lumping technique with mass-proportional Rayleigh damping:

$$M_{\mathrm{E}ij} = \frac{{}^0\rho\,{}^0V}{8}\delta_{ij} \tag{8.77}$$

$$C_{\mathrm{E}ij} = \alpha M_{\mathrm{E}ij} \tag{8.78}$$

where ${}^0\rho$ is the average element density, and ${}^0\rho\,{}^0V$ is the total mass of the element:

$${}^0\rho\,{}^0V = \int_{{}^0V} {}^0\check{\rho}\,\mathrm{d}\,{}^0V \tag{8.79}$$

Another significant performance advantage of these approximations is that, like the mass matrix, the damping matrix is now also time independent.

Since inertial forces are lumped, it makes sense to also use lumped external forces at the same points. These can be computed using Equation 8.69 with the following modified constant shape functions:

$$\check{h}_i = h_i = \frac{1}{8} \tag{8.80}$$

$$\check{h}^{(1)}_{\mathrm{S}i} = h^{(1)}_{\mathrm{S}i} = \frac{1}{8}(1-\zeta_1^{(i)}) \tag{8.81}$$

$$\check{h}^{(2)}_{\mathrm{S}i} = h^{(2)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_1^{(i)}) \tag{8.82}$$

$$\check{h}^{(3)}_{\mathrm{S}i} = h^{(3)}_{\mathrm{S}i} = \frac{1}{8}(1-\zeta_2^{(i)}) \tag{8.83}$$

$$\check{h}^{(4)}_{\mathrm{S}i} = h^{(4)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_2^{(i)}) \tag{8.84}$$

$$\check{h}^{(5)}_{\mathrm{S}i} = h^{(5)}_{\mathrm{S}i} = \frac{1}{8}(1-\zeta_3^{(i)}) \tag{8.85}$$

$$\check{h}^{(6)}_{\mathrm{S}i} = h^{(6)}_{\mathrm{S}i} = \frac{1}{8}(1+\zeta_3^{(i)}) \tag{8.86}$$

Computation of element lumped body and surface forces therefore consists of firstly computing the total body force, ${}^tf_{\mathrm{B}i}$, acting over the whole element (i.e. the sum of body forces acting anywhere on the domain of the element), and similarly the total surface force, ${}^tf^{(s)}_{\mathrm{S}i}$, for each element surface:

$${}^tf_{\mathrm{B}i} = \int_{{}^0V} {}^t_0\check{f}_{\mathrm{B}i}\,\mathrm{d}\,{}^0V \tag{8.87}$$

$${}^tf^{(s)}_{\mathrm{S}i} = \int_{{}^0A^{(s)}} {}^t_0\check{f}^{(s)}_{\mathrm{S}i}\,\mathrm{d}\,{}^0A \tag{8.88}$$

These forces are then distributed evenly to the relevant element nodes that are connected to the domain. For example, the total body force due to gravity is $(0, -G\,{}^0\rho\,{}^0V, 0)$, producing a force of $(0, \frac{-G\,{}^0\rho\,{}^0V}{8}, 0)$ at each element node.

8.2.13 Global System Equations

Using the equations for element work in Sections 8.2.11 and 8.2.12, the principle of virtual work (Equation 8.29) for a single, independent element can be written as:

$${}^t\delta U_{\mathrm{E}i}(M_{\mathrm{E}ij}\,{}^t\ddot{U}_{\mathrm{E}j} + C_{\mathrm{E}ij}\,{}^t\dot{U}_{\mathrm{E}j} + {}^tF_{\mathrm{E}i}) = {}^t\delta U_{\mathrm{E}i}\,{}^tR_{\mathrm{E}i} \tag{8.89}$$

We can now write the principle of virtual work for an entire model of connected elements. For mathematical purposes, the element quantities in Equation 8.89 must first be written in system

(model) rather than element degrees of freedom, with redundant terms for system degrees of freedom not directly influenced by the quantity set to 0. These quantities can then be summed to form the corresponding system quantities:

$$M_{ij} = \sum_e M_{\mathrm{E}ij}^{\mathrm{M}(e)} \tag{8.90}$$

$$C_{ij} = \sum_e C_{\mathrm{E}ij}^{\mathrm{M}(e)} \tag{8.91}$$

$${}^tF_i = \sum_e {}^tF_{\mathrm{E}i}^{\mathrm{M}(e)} \tag{8.92}$$

$${}^tR_i = \sum_e {}^tR_{\mathrm{BE}i}^{\mathrm{M}(e)} + \sum_e {}^tR_{\mathrm{SE}i}^{\mathrm{M}(e)} + {}^tR_{\mathrm{C}i} \tag{8.93}$$

where $M_{\mathrm{E}ij}^{\mathrm{M}(e)}$, $C_{\mathrm{E}ij}^{\mathrm{M}(e)}$, ${}^tF_{\mathrm{E}i}^{\mathrm{M}(e)}$, ${}^tR_{\mathrm{BE}i}^{\mathrm{M}(e)}$ and ${}^tR_{\mathrm{SE}i}^{\mathrm{M}(e)}$ are element quantities for element e written in system degrees of freedom. The system displacement vector, ${}^tU_i$, and concentrated force vector, ${}^tR_{\mathrm{C}i}$, contain quantities for each system node, and are independent of elements. Such quantities are therefore not included in the element summations.

The principle of virtual work for a model can now be written as:

$${}^t\delta U_i(M_{ij}\,{}^t\ddot{U}_j + C_{ij}\,{}^t\dot{U}_j + {}^tF_i) = {}^t\delta U_i\,{}^tR_i \tag{8.94}$$

By imposing a unit virtual displacement in turn for each degree of freedom, an equation of motion is formed for each degree of freedom. This system of equations forms the final equation of motion for the system, which can be written as:

$$\begin{aligned} M_{ij}\,{}^t\ddot{U}_j + C_{ij}\,{}^t\dot{U}_j + {}^tF_i &= {}^tR_i \\ \mathbf{M}\,{}^t\ddot{\mathbf{U}} + \mathbf{C}\,{}^t\dot{\mathbf{U}} + {}^t\mathbf{F} &= {}^t\mathbf{R} \end{aligned} \tag{8.95}$$

This system of second-order ordinary differential equations is uncoupled due to the diagonal mass and damping matrices. It can be numerically approximated over time using a time-integration method. Such methods derive expressions for the nodal displacement derivatives in terms of the nodal displacements at time $t + \Delta t$, which can then be substituted back into Equation 8.95 to solve for these displacements.

We use the explicit central difference time-integration method. This offers similar computational efficiency to the simple but inaccurate and highly unstable explicit Euler method, particularly when using constant diagonal mass and damping matrices, with accuracy and stability much closer to that of the computationally complex Runge-Kutta methods [BG05]. Being an explicit central difference scheme, both the current (at time t) and previous (known) displacements (at time $t - \Delta t$) are required to compute the updated displacements at time $t + \Delta t$. Using the central difference formulae, the displacement derivatives are expressed as:

$${}^t\dot{U}_i = \frac{1}{2\Delta t}({}^{t+\Delta t}U_i - {}^{t-\Delta t}U_i) \tag{8.96}$$

$${}^t\ddot{U}_i = \frac{1}{\Delta t}({}^{t+\frac{\Delta t}{2}}\dot{U}_i - {}^{t-\frac{\Delta t}{2}}\dot{U}_i) \tag{8.97}$$

$$= \frac{1}{\Delta t^2}({}^{t+\Delta t}U_i - 2\,{}^tU_i + {}^{t-\Delta t}U_i) \tag{8.98}$$

Substituting into Equation 8.95 gives:

$$\frac{1}{\Delta t}M_{ij}({}^{t+\Delta t}U_j - 2\,{}^tU_j + {}^{t-\Delta t}U_j) + \frac{1}{2\Delta t}C_{ij}({}^{t+\Delta t}U_j - {}^{t-\Delta t}U_j) + {}^tF_i = {}^tR_i \tag{8.99}$$

Rearranging this equation, the nodal displacements at time $t + \Delta t$ can be calculated. Since the mass and damping matrices are diagonal, the uncoupled updated displacements can be computed

independently for each degree of freedom without computationally complex inversion of these matrices:

$$ {}^{t+\Delta t}U_i = \frac{1}{\frac{C_{ii}}{2\Delta t} + \frac{M_{ii}}{\Delta t^2}} \left({}^{t}R_i - {}^{t}F_i + \frac{2M_{ii}}{\Delta t^2} {}^{t}U_i + \left(\frac{C_{ii}}{2\Delta t} - \frac{M_{ii}}{\Delta t^2} \right) {}^{t-\Delta t}U_i \right) \tag{8.100} $$

By defining time-independent vectors A_{1i}, A_{2i} and A_{3i}, Equation 8.100 may be rewritten in a simpler form:

$$ {}^{t+\Delta t}U_i = A_{1i}({}^{t}R_i - {}^{t}F_i) + A_{2i} {}^{t}U_i + A_{3i} {}^{t-\Delta t}U_i \tag{8.101} $$

where

$$ A_{1i} = \frac{1}{\frac{C_{ii}}{2\Delta t} + \frac{M_{ii}}{\Delta t^2}} \tag{8.102} $$

$$ A_{2i} = \frac{\frac{2M_{ii}}{\Delta t^2}}{\frac{C_{ii}}{2\Delta t} + \frac{M_{ii}}{\Delta t^2}} = \frac{2M_{ii}}{\Delta t^2} A_{1i} \tag{8.103} $$

$$ A_{3i} = \frac{\frac{C_{ii}}{2\Delta t} - \frac{M_{ii}}{\Delta t^2}}{\frac{C_{ii}}{2\Delta t} + \frac{M_{ii}}{\Delta t^2}} = \frac{C_{ii}}{2\Delta t} A_{1i} + \frac{1}{2} A_{2i} \tag{8.104} $$

8.3 Hourglass Control

To calculate nodal force contributions of an element, we approximate the integral over the element relating element deformations and strains using Gaussian quadrature (refer to Section 8.2.11). Fully integrated elements require a stiffness matrix that is rank sufficient, in which case the target element stiffness matrix rank $r = d_t - d_r$, where d_t is the degrees of freedom (DOF) of the element, and d_r is the number of rigid body modes. For typical 3D elements, with 3 DOF per node, and 3 translational and 3 rotational element DOF, $d_t = 3n$ and $d_r = 6$, where n is the number of element nodes. The number of required integration points such that an element stiffness matrix is rank sufficient depends on the value of r, with each integration point adding $\frac{1}{2}i(i+1)$ to the rank (therefore 3D integration points add 6 to the rank), up to a maximum of r.

With hexahedral elements, the same number of integration points are typically used through each local element dimension when the shape functions are the same in each such dimension; therefore, fully integrated 8-node hexahedra require 8 integration points ($2 \times 2 \times 2$). For improved efficiency and accuracy of incompressible soft-tissue simulations, we use reduced integration, whereby the next coarsest number of integration points is used. Reduced-integration 8-node hexahedra therefore require only 1 integration point ($1 \times 1 \times 1$), reducing the Gaussian quadrature computation by a factor of 8; however, the stiffness matrices of such elements, with $r = 18$, and an actual rank of 6, have a rank deficiency of 12.

Using reduced-integration elements overcomes the volume-locking limitations that can occur when simulating incompressible material like soft tissue, while greatly improving computational efficiency compared with fully integrated elements [TCO08]. However, hourglass effects (physically impossible stressless deformations, whereby no internal forces are produced to resist deformation) can occur with such elements, particularly with large deformations, due to the more approximate evaluation of element deformation. As mentioned in Section 3.3.1, a hourglass control technique is required to reduce the hourglass effects. This section explains the stiffness-based perturbation hourglass control technique we use, which adds artificial stiffness to elements to constrain the different hourglass modes based on element stiffness [FB81, JWM08].

8.3.1 Hourglass Modes

To counter hourglass effects, the hourglass modes resulting from reduced integration must first be determined. For an 8-node hexahedron, this can be done by rewriting the shape functions in Equation 8.31 to define them in terms of orthogonal base vectors:

$$\check{h}_p = \frac{1}{8}(1 + \zeta_1^{(p)}\zeta_1 + \zeta_2^{(p)}\zeta_2 + \zeta_3^{(p)}\zeta_3 + \zeta_2^{(p)}\zeta_3^{(p)}\zeta_2\zeta_3 + \zeta_1^{(p)}\zeta_3^{(p)}\zeta_1\zeta_3 + \zeta_1^{(p)}\zeta_2^{(p)}\zeta_1\zeta_2 + \zeta_1^{(p)}\zeta_2^{(p)}\zeta_3^{(p)}\zeta_1\zeta_2\zeta_3) \tag{8.105}$$

$$\check{h}_p = \frac{1}{8}(\Sigma_p + \Lambda_{1p}\zeta_1 + \Lambda_{2p}\zeta_2 + \Lambda_{3p}\zeta_3 + \Gamma_{1p}\zeta_2\zeta_3 + \Gamma_{2p}\zeta_1\zeta_3 + \Gamma_{3p}\zeta_1\zeta_2 + \Gamma_{4p}\zeta_1\zeta_2\zeta_3) \tag{8.106}$$

where Σ_p is the average base vector, defining the constant term in the shape functions, Λ_{ip} for $1 \leq i \leq 3$ are the volume base vectors, defining linear terms, and Γ_{jp} for $1 \leq j \leq 4$ are the hourglass base vectors, defining bilinear and trilinear terms:

$$\begin{aligned}
\boldsymbol{\Sigma}^{\mathrm{T}} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \\
\boldsymbol{\Lambda_1}^{\mathrm{T}} &= \begin{pmatrix} -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \end{pmatrix} \\
\boldsymbol{\Lambda_2}^{\mathrm{T}} &= \begin{pmatrix} -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \end{pmatrix} \\
\boldsymbol{\Lambda_3}^{\mathrm{T}} &= \begin{pmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{pmatrix} \\
\boldsymbol{\Gamma_1}^{\mathrm{T}} &= \begin{pmatrix} 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \end{pmatrix} \\
\boldsymbol{\Gamma_2}^{\mathrm{T}} &= \begin{pmatrix} 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \\
\boldsymbol{\Gamma_3}^{\mathrm{T}} &= \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix} \\
\boldsymbol{\Gamma_4}^{\mathrm{T}} &= \begin{pmatrix} -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \end{pmatrix}
\end{aligned} \tag{8.107}$$

Each of these base vectors represent a deformation mode (the element deformation produced by a base vector in isolation). The average base vector defines rigid body translations (no strain), the combination of only volume base vectors define rigid body rotations (no strain), and constant strain modes (constant normal and sheer strains across the element), and the hourglass base vectors give rise to linear strain modes.

As explained in Section 8.2.11, element internal forces (and therefore stresses and strains) are evaluated at integration points. Recall that strains are defined in terms of the shape function derivatives with respect to global material coordinates, which are subsequently defined in terms of the shape function derivatives with respect to element local coordinates, $\frac{\partial \check{h}_p}{\partial \zeta_i}$:

$$\frac{\partial \check{h}_p}{\partial \zeta_1} = \frac{1}{8}(\Lambda_{1p} + \Lambda_{2p}\zeta_2 + \Lambda_{3p}\zeta_3 + \Gamma_{1p}\zeta_2\zeta_3 + \Gamma_{2p}\zeta_3 + \Gamma_{3p}\zeta_2 + \Gamma_{4p}\zeta_2\zeta_3) \tag{8.108}$$

$$\frac{\partial \check{h}_p}{\partial \zeta_2} = \frac{1}{8}(\Lambda_{1p}\zeta_1 + \Lambda_{2p} + \Lambda_{3p}\zeta_3 + \Gamma_{1p}\zeta_3 + \Gamma_{2p}\zeta_1\zeta_3 + \Gamma_{3p}\zeta_1 + \Gamma_{4p}\zeta_1\zeta_3) \tag{8.109}$$

$$\frac{\partial \check{h}_p}{\partial \zeta_3} = \frac{1}{8}(\Lambda_{1p}\zeta_1 + \Lambda_{2p}\zeta_2 + \Lambda_{3p} + \Gamma_{1p}\zeta_2 + \Gamma_{2p}\zeta_1 + \Gamma_{3p}\zeta_1\zeta_2 + \Gamma_{4p}\zeta_1\zeta_2) \tag{8.110}$$

By evaluating these at the single central integration point used with reduced integration ($\zeta_i = 0$), it can be seen that the element strains depend only on the constant strain modes, while the linear strain modes represented by the hourglass base vectors are neglected. Such deformations (hourglass modes) therefore lead to zero strain, stress and internal forces. The stiffness-based hourglass control technique that we use adds artificial forces to resist these deformations.

8.3.2 Stiffness-Based Perturbation Hourglass Control

To counter hourglass effects, it is necessary to resist element displacements (deformations) moving in the directions of hourglass modes. Such resisting hourglass forces at element nodes can

therefore be computed based on element nodal velocities (deformation rates). Like with strains, the distribution of other element values, including velocity (displacement derivative), also neglect hourglass terms when evaluated at the central integration point, enabling hourglass forces to be computed by defining so-called hourglass velocities.

In Section 8.2.8, we have defined the full distribution of values across an element using the element shape functions and nodal values (refer to Equation 8.32). However, approximating such distributions by evaluating the shape functions at the central integration point produces linear distributions, neglecting the so-called hourglass distributions formed from the bilinear and trilinear terms. The element velocity distribution can therefore be defined in terms of a linear velocity distribution, ${}^t\dot{\check{u}}_{\mathrm{L}i}$, and a neglected hourglass velocity distribution that is orthogonal to this, ${}^t\dot{\check{u}}_{\mathrm{HG}i}$:

$$ {}^t\dot{\check{u}}_i = {}^t\dot{\check{u}}_{\mathrm{L}i} + {}^t\dot{\check{u}}_{\mathrm{HG}i} \tag{8.111} $$

This can be easily specialised to compute the velocity at element node p:

$$ {}^t\dot{u}_i^{(p)} = {}^t\dot{u}_{\mathrm{L}i}^{(p)} + {}^t\dot{u}_{\mathrm{HG}i}^{(p)} \tag{8.112} $$

Using a truncated Taylor series expansion, the linear approximation of the element velocity distribution can be expressed as:[10]

$$ {}^t\dot{u}_{\mathrm{L}i}^{(p)} = {}^t\dot{\check{u}}_i|_{\zeta_a=0} + {}^t_0\dot{\check{u}}_{i,j}|_{\zeta_a=0}({}^tx_j^{(p)} - {}^t\check{x}_j|_{\zeta_a=0}) \tag{8.113} $$

where ${}^t_0\dot{\check{u}}_{i,j}$ is the velocity gradient, defined analogously to the displacement gradient:

$$ {}^t_0\dot{\check{u}}_{i,j} = {}_0\check{h}_{p,j}\,{}^t\dot{u}_i^{(p)} \tag{8.114} $$

Since the element base vectors are orthogonal, the hourglass velocities may be expressed in terms of the hourglass base vectors:

$$ {}^t\dot{u}_{\mathrm{HG}i}^{(p)} = \frac{1}{\sqrt{8}}\,{}^t_0\dot{q}_{ij}\Gamma_{jp} \tag{8.115} $$

where the constant scale factor normalises the base vectors, and ${}^t\dot{q}_{ij}$ are the hourglass modal velocities corresponding to the hourglass modes. These can be defined by applying hourglass shape vectors, γ_{jp}, which define hourglass velocity patterns, to the velocity:

$$ {}^t_0\dot{q}_{ij} = \frac{1}{\sqrt{8}}\,{}^t\dot{u}_i^{(p)}\,{}_0\gamma_{jp} \tag{8.116} $$

where the shape vectors have also been normalised. Hourglass base vectors are orthogonal to the linear displacement distribution, and define hourglass patterns, while hourglass shape vectors are orthogonal to the linear velocity distribution, and define velocity patterns that lead to hourglassing.

Substituting Equations 8.113, 8.115 and 8.116 into 8.112, the following relationship involving the hourglass shape vectors can be derived:

$$ {}^t\dot{u}_i^{(p)} = {}^t\dot{\check{u}}_i|_{\zeta_a=0} + {}^t_0\dot{\check{u}}_{i,j}|_{\zeta_a=0}({}^tx_j^{(p)} - {}^t\check{x}_j|_{\zeta_a=0}) + \frac{1}{8}\,{}^t\dot{u}_i^{(q)}\,{}_0\gamma_{jq}\Gamma_{jp} \tag{8.117} $$

After multiplying by Γ_{jp}, and recognising the orthogonality of the base vectors, this equation can be rearranged as follows:

$$ {}^t\dot{u}_i^{(p)}\,{}_0\gamma_{jp} = {}^t\dot{u}_i^{(p)}\Gamma_{jp} - {}^t_0\dot{\check{u}}_{i,k}|_{\zeta_a=0}\,{}^tx_k^{(p)}\Gamma_{jp} \tag{8.118} $$

[10] The Taylor series expansion is defined in terms of global rather than local coordinates since the spatial position of a material point appears in the velocity calculation, which remains fixed in local coordinate space. However, similar to displacements, element global coordinates are defined in terms of local coordinates (refer to Equation 8.32).

Eliminating the nodal velocities by substituting the definition of the velocity gradient from Equation 8.114, an expression for the hourglass shape vectors can be derived:

$$ {}_0\gamma_{jp} = \Gamma_{jp} - {}_0\check{h}_{p,k}|_{\zeta_a=0}\, {}^t x_k^{(q)} \Gamma_{jq} \tag{8.119} $$

Since the hourglass shape vectors define the velocity patterns that lead to hourglassing, they can be used to define nodal hourglass resistance forces, ${}^t f_{\mathrm{HG}i}^{(p)}$. To do this, the definition of work rate (power) can be used with the hourglass modal velocities and work-conjugate forces, ${}^t_0 Q_{ij}$:

$$ {}^t \dot{u}_i^{(p)}\, {}^t f_{\mathrm{HG}i}^{(p)} = {}^t_0 Q_{ij}\, {}^t_0 \dot{q}_{ij} \tag{8.120} $$

$$ {}^t f_{\mathrm{HG}i}^{(p)} = \frac{1}{\sqrt{8}}\, {}^t_0 Q_{ij}\, {}_0\gamma_{jp} \tag{8.121} $$

${}^t_0 Q_{ij}$ can be defined in terms of displacements, and the lower bound of the maximum stiffness, ${}_0\check{K}_{\max}$, evaluated at the central integration point:

$$ {}^t_0 Q_{ij} = \kappa\, {}_0\check{K}_{\max}|_{\zeta_a=0}\, {}^t_0 q_{ij} \tag{8.122} $$

where κ is the hourglass stiffness parameter used to scale the resistance forces. As mass lumping is used, an approximation of the maximum stiffness can be easily computed from an approximation of the maximum frequency, ${}_0\check{\omega}_{\max}$ [FB81]:

$$ {}_0\check{K}_{\max} = \frac{{}^0\rho\, {}^0V}{8}\, {}_0\check{\omega}_{\max}^2 \tag{8.123} $$

$$ {}_0\check{\omega}_{\max} = \sqrt{8\frac{\check{\lambda}+2\check{\mu}}{\check{\rho}}\, {}_0\check{h}_{p,i}\, {}_0\check{h}_{p,i}} \tag{8.124} $$

where $\check{\rho}$, and $\check{\mu}$ and $\check{\lambda}$ are the distributions of material density and Lamé parameters over the element respectively (${}^0\rho\, {}^0V$ is the element mass, as defined in Section 8.2.12). Note that, since the hourglass forces in Equation 8.121 are defined in terms of the hourglass shape vectors, which are orthogonal to the linear velocity distribution, such forces affect only the neglected linear strain modes (produced from hourglass velocities). The constant strain modes (produced from linear velocities) that are captured at the single central integration point, and therefore included in the element internal force computations, are not affected.

Using Equations 8.121 to 8.124, and integrating Equation 8.116 with zero initial conditions (i.e. assuming that, in the initial configuration, there are no hourglass effects, and therefore no hourglass resistance forces) to compute ${}^t_0 q_{ij}$, the final equation to compute element nodal hourglass resistance forces can be written as:

$$ {}^t f_{\mathrm{HG}i}^{(p)} = \frac{\kappa\, {}_0\check{K}_{\max}|_{\zeta_a=0}}{8}\, {}^t u_i^{(q)}\, {}_0\gamma_{jq}\, {}_0\gamma_{jp} \tag{8.125} $$

Analogous to the element displacement and internal force vectors, an element hourglass force vector, ${}^t F_{\mathrm{HG}i}$, can be defined:

$$ {}^t\mathbf{F}_{\mathrm{HG}}^{\mathrm{T}} = \begin{pmatrix} {}^t\mathbf{f}_{\mathrm{HG}}^{(1)\mathrm{T}} & {}^t\mathbf{f}_{\mathrm{HG}}^{(2)\mathrm{T}} & \cdots & {}^t\mathbf{f}_{\mathrm{HG}}^{(8)\mathrm{T}} \end{pmatrix} \tag{8.126} $$

This is simply added to the element internal force vector, ${}^t F_{\mathrm{E}i}$. Regarding the hourglass stiffness parameter, κ, as a general rule, this should be chosen such that the total element hourglass energy doesn't exceed 10% of the total strain energy to prevent materials appearing overly stiff [YK11]. After experimentation, we have used fixed value of 0.0025 with the soft-tissue examples in this thesis. A better approach would be to dynamically calculate κ based on the current element stiffness and deformation, although this would lead to increased complexity and computational expense.

8.4 Muscle Contraction

To enable muscle stresses to be generated during simulations, a muscle contraction model has been implemented. This generates a muscle stress tensor for an element, ${}^{t}_{0}\check{S}_{\mathrm{C}ij}$:

$$ {}^{t}_{0}\check{S}_{\mathrm{C}ij} = \sum_{c \in C} o^{(ec)} \, {}^{t}_{0}\check{S}^{(c)}_{\mathrm{C}ij} \tag{8.127} $$

where C is the set of all muscles (contractile units) overlapping the element, e, and $o^{(ec)}$ is the proportion of overlap between the element and the muscle to weight the element muscle stresses. The muscle stress tensor for an element is simply added to the total stress tensor for that element (defined in Equation 8.55). The computation of such muscle stress tensors consists of two processes: the generation of active muscle stresses, and the generation of transversely isotropic passive muscle stresses. Due to the fibrous structure of muscles, passive resistance of a muscle increases as the muscle is stretched, but only in the fibre direction. As muscles exhibit such transversely isotropic behaviour with preferred deformation in the fibre direction, as well as an active stress component, this direction is used to compute an additional passive stress component.

The additional stress produced by muscle c is computed by adding the muscle active stress, ${}^{t}_{0}\check{S}_{\mathrm{A}ij}$, and additional passive stress, ${}^{t}_{0}\check{S}_{\mathrm{P}ij}$:

$$ {}^{t}_{0}\check{S}^{(c)}_{\mathrm{C}ij} = {}^{t}_{0}\check{S}^{(c)}_{\mathrm{A}ij} + {}^{t}_{0}\check{S}^{(c)}_{\mathrm{P}ij} \tag{8.128} $$

For a particular muscle, the active and additional passive stress components are computed as:

$$ {}^{t}_{0}\check{S}_{\mathrm{A}ij} = {}^{t}J \, {}^{t}\alpha \sigma_{\mathrm{A}} f_{\mathrm{A}}({}^{t}\lambda) \, {}^{0}\check{d}_i \, {}^{0}\check{d}_j \tag{8.129} $$

$$ {}^{t}_{0}\check{S}_{\mathrm{P}ij} = {}^{t}J \sigma_{\mathrm{P}} f_{\mathrm{P}}({}^{t}\lambda) \, {}^{0}\check{d}_i \, {}^{0}\check{d}_j \tag{8.130} $$

where ${}^{t}J$ is the Jacobian of the element deformation gradient, ${}^{t}\lambda$ is the muscle fibre stretch ratio, and ${}^{0}\check{d}_i$ is the element muscle fibre field[11]. Note that the fibre field in the initial configuration is used to compute the second Piola-Kirchhoff stresses. σ_{A} and σ_{P} are the active and passive muscle stress references respectively. σ_{A} represents the contraction strength of the muscle, the value of which is the maximum active stress magnitude that the muscle will produce under maximum contraction at optimal length. σ_{P} is a stress magnitude that represents the passive resistance strength of the muscle as it is stretched beyond equilibrium length in the fibre direction.

${}^{t}\alpha \in [0, 1]$ is used only for the computation of active muscle stresses, with no corresponding value for the computation of passive muscle stresses. This is the muscle contraction parameter (active stress scale factor) used to control the magnitude of contraction for the muscle, where a value of 0 represents no contraction, and 1 represents full contraction. It is varied over time, for example, according to a function (e.g. linear or cosine), to simulate muscle contraction. Active stresses therefore depend on the desired amount of muscle contraction, as well as the material and structure of muscles, whereas the transversely isotropic passive stresses depend only on the material and structure of muscles.

$f_{\mathrm{A}}({}^{t}\lambda)$ and $f_{\mathrm{P}}({}^{t}\lambda)$ are normalised stress curves that scale the active and passive stresses respectively depending on the fibre stretch ratio. We use functions based on those used by Rörle et al. [RP07], which, as shown by Figure 8.2, follow the tension-length properties observed by real

[11] While the element muscle fibre field is stated as a continuous function representing the distribution of fibre directions across the element, only the fibre directions at the integration points will be used in the evaluation of stresses (refer to Section 8.2.11). Since the reduced-integration 8-node hexahedral elements we use have a single central integration point, during model creation, we associate only a single fibre direction with each muscle overlapping the element (see Section 7.4).

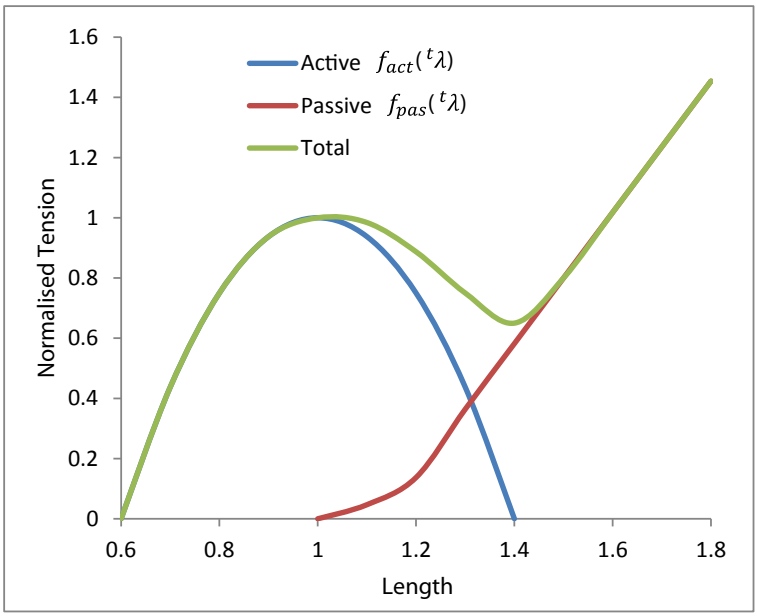

Figure 8.2: The normalised active, $f_{\mathrm{A}}({}^t\lambda)$, passive, $f_{\mathrm{P}}({}^t\lambda)$, and total muscle tension-length functions.

muscle. These functions are defined as:

$$f_{\mathrm{A}}({}^t\lambda) = \begin{cases} -\dfrac{25}{4\lambda_{\mathrm{A}}^2}\,{}^t\lambda^2 + \dfrac{25}{2\lambda_{\mathrm{A}}}\,{}^t\lambda - 5.25 & 0.6\lambda_{\mathrm{A}} \leq {}^t\lambda \leq 1.4\lambda_{\mathrm{A}} \\ 0 & \text{otherwise} \end{cases} \tag{8.131}$$

$$f_{\mathrm{P}}({}^t\lambda) = \begin{cases} 0 & {}^t\lambda \leq 1 \\ 0.05(\exp^{6.6({}^t\lambda - 1)} - 1) & 1 < {}^t\lambda \leq \lambda_{\mathrm{P}} \\ 2.18\,{}^t\lambda - 2.47 & \text{otherwise} \end{cases} \tag{8.132}$$

where λ_{A} is the optimal fibre stretch, which is normally similar to the stretch at which the passive stress increases sharply, λ_{P}. Such stress curves could be further customised for each muscle, for example, to produce a larger or smaller range of fibre stretch values at which active stresses are produced (rather than fixing this range as $0.6\lambda_{\mathrm{A}} \leq {}^t\lambda \leq 1.4\lambda_{\mathrm{A}}$). However, we haven't experimented with such fine levels of detail.

8.5 Boundary Conditions

To constrain models during simulations, boundary conditions must be set. It is possible to set nodes as rigid or sliding (bound by a surface) with our system (refer to Section 7.5 for details on the determination of such restricted nodes during model creation). Rigid nodes are simply fixed with zero displacement throughout simulations, and can therefore be used to model muscle attachments on the skull, or the roots of retaining ligaments. Sliding nodes are used to model material that is normally attached to, but can slide over a rigid surface; for example, in reality, superficial facial soft-tissue layers are able to slide over the stiff deep layers and skull [WMSH10], although retaining ligaments normally restrict the separation of these layers. This sliding phenomenon has often been neglected in previous work, in which nodes on the skull are simply treated as rigid nodes [SNF05, BJTM08].

As we're using non-conforming models, sliding nodes won't necessarily lie on the surface they are bound by. To constrain such nodes to follow the shape of the restricting surface, they are forced to maintain a fixed distance away from this surface. Friction is not considered. As shown

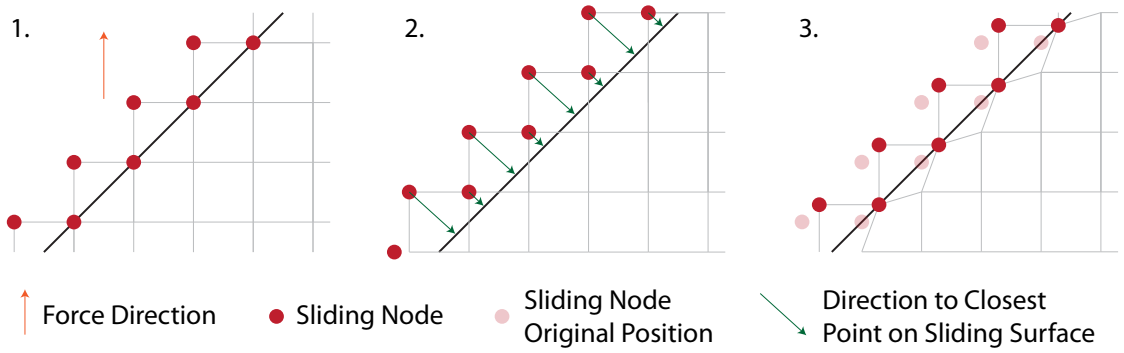

Figure 8.3: Sliding nodes moving along a sliding surface. The displacement of sliding nodes before (image 2) and after (image 3) the additional sliding displacement is shown. Note this is a 2D illustration of a 3D process.

by Figure 8.3, the displacement of a sliding node is updated after computation of a timestep to move this node to a position that is distance f (the fixed distance) away from the sliding surface. This additional displacement, $u_{\mathrm{R}i}$, is calculated as:

$$u_{\mathrm{R}i} = -b \cdot (f + b(d_j d_j)^{\frac{1}{2}}) \cdot \frac{d_i}{(d_j d_j)^{\frac{1}{2}}} \tag{8.133}$$

where

$$d_i = p_i - c_i \tag{8.134}$$

$$b = \begin{cases} 1 & d_j n_j < 0 \\ -1 & \text{otherwise} \end{cases} \tag{8.135}$$

p_i is the node position, c_i is the closest point on the surface to the node, and n_i is the surface normal at this point. To increase computational performance, a GPU-based semi-brute-force broad-phase collision detection algorithm with spatial subdivision has been implemented [AGA12] to prune the number of polygons to be tested when finding the closest surface position for each sliding node.

Due to the fixed-distance restriction with non-conforming elements, our sliding procedure can limit the ability of elements to adapt to the shape of the surface as they move, particularly with lower-resolution models and around highly curved areas. However, these constraints help to suitably restrict our models such that a higher timestep can be used. Alternatively, the penalty method has been used in related work [MHSH10], although the produced oscillatory movement can greatly decrease stability, and only penetration is constrained; for example, the movement of soft tissue away from the skull of a facial model would be unrestricted.

8.6 Animating the Surface Mesh

With conforming models, a surface mesh is attached to an FE model, with nodes and vertices sharing the same positions; therefore, vertex positions are automatically updated when nodal positions are computed. However, this isn't the case with the non-conforming voxel-based models that we use. As described in Section 7.6, we bind vertices to the voxel elements during model creation, and compute trilinear interpolation weights. To animate a surface mesh during simulation, the updated position of a bound vertex, ${}^t p_i$, can be computed by applying trilinear interpolation and extrapolation using the weights, w_j (one along each local axis, $1 \leq j \leq 3$), and the computed nodal positions for the element that it is bound to, ${}^t x_i^{(k)}$ for $1 \leq k \leq 8$. Assuming w_j is the weight of

node 1, ${}^t x_i^{(1)}$, and using the axes and node numbering defined in Figure 8.1, the updated position can be computed as:

$$\begin{aligned} {}^t p_i = {}& {}^t x_i^{(1)} w_1 w_2 w_3 \\ &+ {}^t x_i^{(2)} (1 - w_1) w_2 w_3 \\ &+ {}^t x_i^{(3)} (1 - w_1)(1 - w_2) w_3 \\ &+ {}^t x_i^{(4)} w_1 (1 - w_2) w_3 \\ &+ {}^t x_i^{(5)} w_1 w_2 (1 - w_3) \\ &+ {}^t x_i^{(6)} (1 - w_1) w_2 (1 - w_3) \\ &+ {}^t x_i^{(7)} (1 - w_1)(1 - w_2)(1 - w_3) \\ &+ {}^t x_i^{(8)} w_1 (1 - w_2)(1 - w_3) \end{aligned} \tag{8.136}$$

where

$${}^t x_i^{(k)} = {}^0 x_i^{(k)} + {}^t u_i^{(k)} \tag{8.137}$$

Such vertex positions are updated after all timesteps for a frame have been computed.

An alternate, more generic method of computing bound vertex positions would be to associate each such vertex with 4 nodes that form a tetrahedron, and use tetrahedral barycentric coordinates, in which case a vertex could be bound to a subsection of any 3D element. However, this approach requires an additional step during model creation to compute the element subsections to which vertices are bound, and additional memory to store the nodal indices of such subsections. This is unnecessary for our voxel-based models, for which trilinear interpolation weights can be easily computed and applied.

8.7 GPU Implementation

The TLED FE formulation we use is inherently parallel, making it suitable for GPU implementation. For example, the muscle stress (Equation 8.128) and internal nodal force calculations (Equation 8.61) can be done independently and in parallel for each integration point, and the anti-hourglass calculations (Equation 8.125) for each element. Using a lumped mass approximation and mass-proportional damping leads to uncoupled equations of motion (Equation 8.95), meaning the nodal displacement (Equation 8.100) and boundary condition calculations (Equation 8.133) can be done independently for each relevant node. The TL formulation also enables some variables to be precomputed.

Algorithm 8.2 shows the process to compute a timestep using our simulation system, which has been implemented using C++ with CUDA C API version 4.2 for use with the Fermi architecture. The main simulation system kernels can be grouped into two major procedures: computation of element nodal force contributions, and computation of nodal displacements. Various other kernels are also used within a timestep, for example, to initialise element stresses and internal nodal forces. The following sections provide an introduction to CUDA, before presenting details of our CUDA implementation, including memory organisation, the processing of the two main simulation system procedures, visualisation and interaction, and the optimisations for use with our voxel-based models.

8.7.1 Introduction to CUDA

GPUs are specialised for compute-intensive, massively parallel computation. They dedicate many more resources to ALUs than CPUs, rather than cache and control. They therefore use the SIMT (single instruction multiple thread) execution model, and require efficient memory usage to be used efficiently. To make use of GPUs with our simulation and visualisation system, we use CUDA, a parallel computing architecture developed by NVIDA [NVI12]. CUDA also supports

```
   // Element nodal force contributions:
 1 foreach element collection, M do
 2 |  if active stresses generated for M then
 3 |  |  kernel: foreach element in M do
 4 |  |  |  foreach integration point do
 5 |  |  |  |  Calculate deformation values;
 6 kernel: foreach contractile component do
 7 |  foreach integration point do
 8 |  |  Calculate muscle stresses;
 9 foreach element collection, M do
10 |  kernel: foreach element in M, m do
11 |  |  foreach integration point, i do
12 |  |  |  if active stresses generated with M then
13 |  |  |  |  Read deformation and stress values;
14 |  |  |  else
15 |  |  |  |  Calculate deformation values and initialise stress;
16 |  |  |  foreach material overlapping m do
17 |  |  |  |  Calculate material stresses;
18 |  |  |  Calculate internal nodal forces at i;
19 |  |  Add nodal force contributions for m;
20 |  if hourglass control enabled with M then
21 |  |  kernel: foreach element in M do
22 |  |  |  Calculate hourglass forces;
   // Nodal displacements:
23 kernel: foreach node, n do
24 |  if n not rigid then
25 |  |  Calculate nodal displacements;
26 foreach sliding constraint, C do
27 |  foreach surface in C do
28 |  |  kernel: foreach sliding node in C do
29 |  |  |  Perform broad-phase collision detection;
30 |  |  kernel: foreach sliding node in C do
31 |  |  |  Compute closest surface point;
32 |  kernel: foreach sliding node in C do
33 |  |  Update nodal displacements;
```

Algorithm 8.2: The GPU-based process to compute a timestep. The major kernels have been identified.

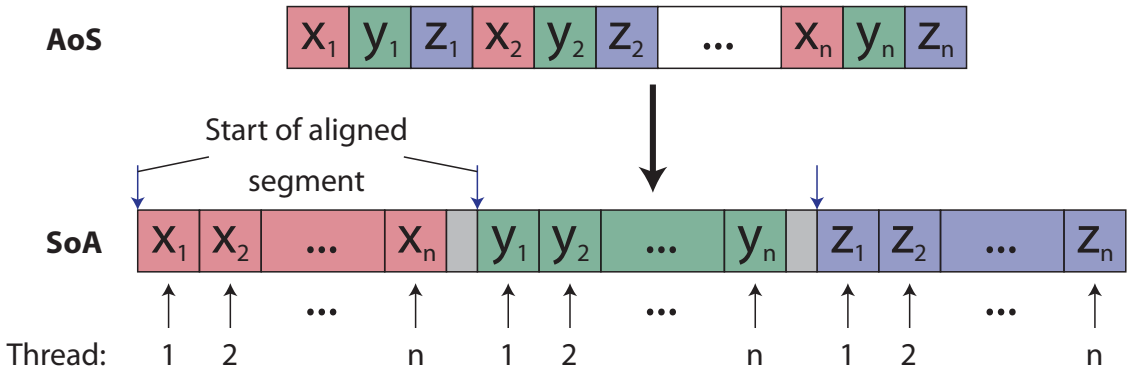

Figure 8.4: The storage of nodal positions in the array of structures (AoS) and structure of arrays patterns (SoA). Storing as SoA enables memory coalescing, for example, in a nodal kernel with consecutive threads accessing consecutive memory locations when accessing a nodal position.

interoperability with graphics libraries like OpenGL, enabling efficient visualisation of simulation data processed using CUDA.

GPUs consist of a number of streaming multiprocessors (SMs) that execute in parallel, and each SM contains multiple CUDA cores. Using CUDA, kernels (parallel sections of an application) are executed as a grid of equally sized thread blocks. A grid is executed on a GPU, a block is executed on an SM, and a thread is executed on a CUDA core. Threads are executed in warps (groups of 32 threads), and each CUDA core executes the same instruction on a different thread (SIMT paradigm). Depending on memory requirements, various blocks can reside on an SM, and an SM can quickly switch between warps to hide the latency of memory accesses.

Normally, most data on the GPU resides in high-latency global memory. Each SM has faster shared memory for sharing data between block threads. For local data, threads are allocated extremely fast registers, although, if exceeded, this spills into a section of global memory. With the Fermi architecture, the L1 and L2 cache can increase the efficiency of global memory accesses.

For optimal performance, it is important to balance memory requirements of a block with the ability to hide memory latency. Efficient access patterns should be used to achieve global memory coalescing and avoid shared memory bank conflicts. Also, branch divergence within a warp should be minimised, as this causes each branch to be executed in a serial fashion, with some threads idle during each branch execution (the same instructions are executed on each warp thread with the SIMT execution model). Similarly, uneven loops within a warp will result in idle threads waiting for the longest loop to complete.

8.7.2 Memory and Data Structures

During a CPU precomputation stage, all simulation values that relate only to the initial configuration, such as shape function derivatives and unscaled hourglass matrices, are precomputed. All simulation values are then copied to the GPU, where they remain throughout simulations to reduce the amount of slow CPU-GPU data transfer. All simulation variables are stored in global memory, which, with the L1 cache and lenient coalescing requirements, is now normally preferred over texture memory. As shown by Figure 8.4, data is stored using the structure of arrays pattern (e.g. $[[u_1, ..., u_n], [v_1, ..., v_n], [w_1, ..., w_n]]$) where possible, rather than array of structures (e.g. $[[u_1, v_1, w_1], ..., [u_n, v_n, w_n]]$), with each row of the arrays aligned to a memory block. This increases spatial locality of the cached accesses between consecutive warp threads, and enables memory coalescing.

Figure 8.5 shows the main classes used to organise and group simulation data based on functionality (e.g. to reduce branch divergence and uneven loops). As all nodes of our FE models share the same functionality, no grouping of nodes is required, and all nodal computations for a particular process, such as the calculation of displacements from forces, are computed in a single

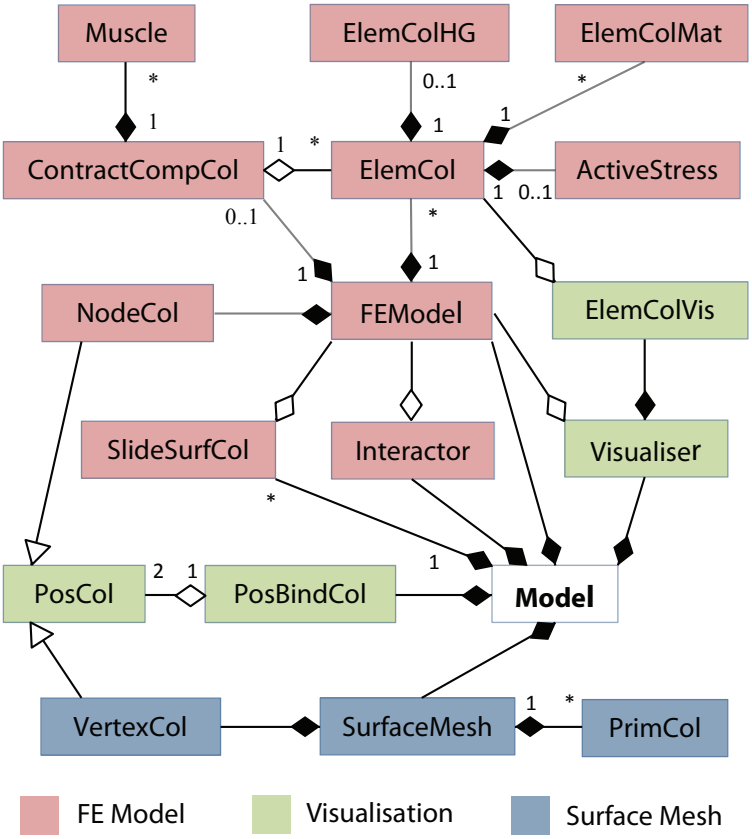

Figure 8.5: The relationships between the main classes used with our simulation system. Where not given, the multiplicities equal 1.

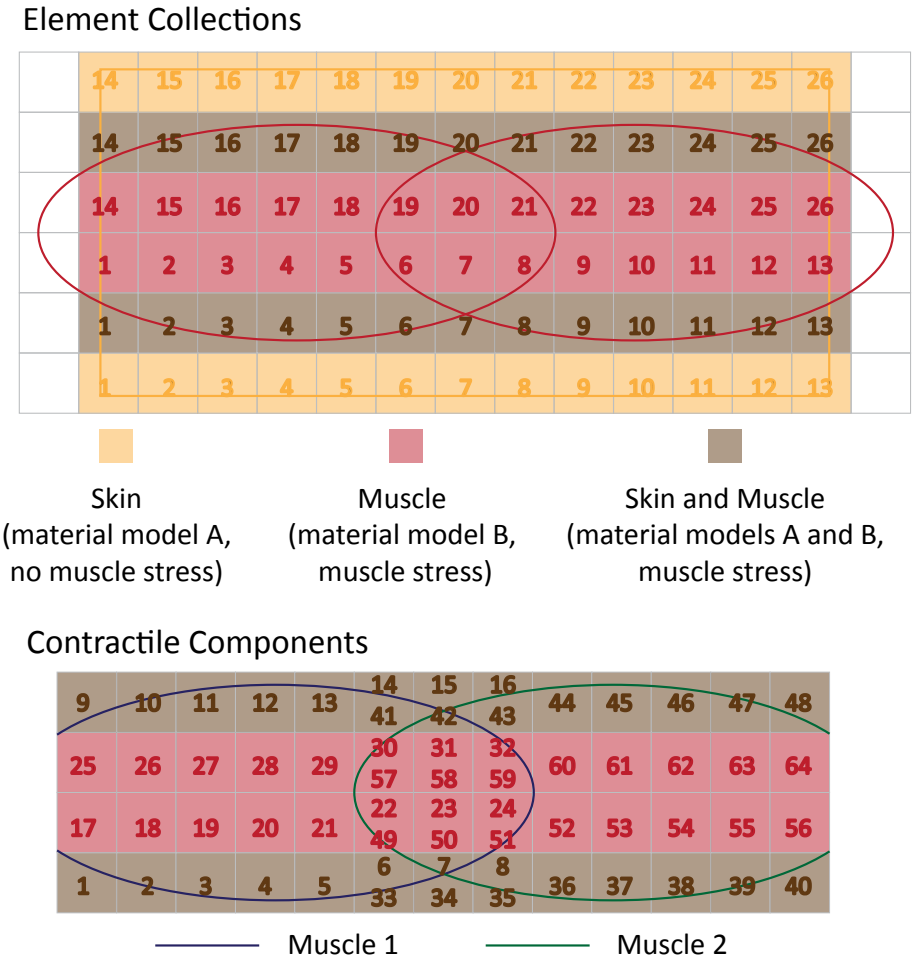

Figure 8.6: Element and contractile component groupings for a voxel-based model with 2 muscles and 2 material models, showing the element and contractile component orderings within these groups. Note this is a 2D illustration of a 3D process.

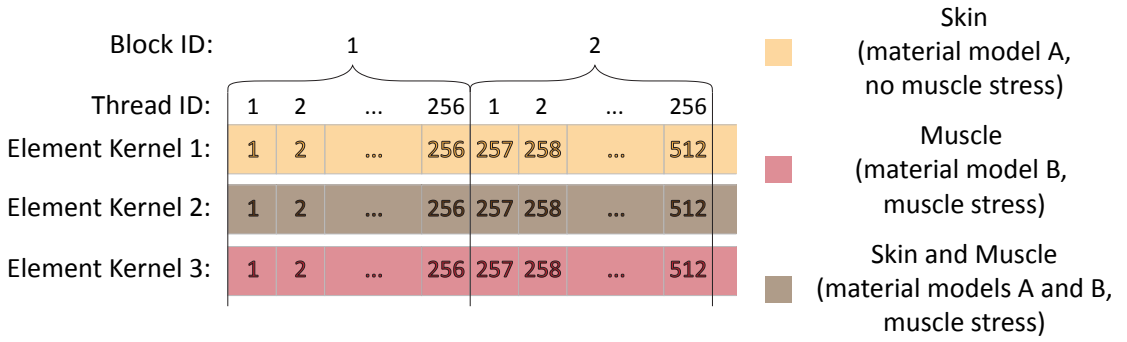

Figure 8.7: The organisation of the groups of elements into 1D grids of 1D blocks, each with 256 threads, for the execution of element kernels.

kernel call. On the other hand, element-related computation varies based on element type, material behaviour, and whether active muscle stresses are generated. Figure 8.6 shows an example of how elements are grouped to reflect this, forming element collections. All of our models contain a maximum of two element collections since all of the elements are reduced-integration 8-node hexahedra, and only neo-Hookean materials are used, but some elements overlap muscles. Element computations for a particular process, such as the computation of nodal force contributions, are computed using a separate kernel for each element collection. As each kernel processes elements with the same functionality, there is no branch divergence or uneven loops in such computations. Similarly, with the surface mesh, the processing of primitives varies (and is therefore grouped) depending on the type of primitive, and how these are rendered.

Contractile components are created to generate muscle stresses, and one component is created for each muscle that overlaps an element (as there may be multiple muscles that overlap a non-conforming element). As shown by Figure 8.6, while contractile components exhibit identical functionality, these are ordered firstly by the muscle, and then by the element collection and element within the collection that they reference, enabling several consecutive warp threads to access the same muscle data and spatially local element data. However, as each contractile component shares the same functionality on different data, element muscle stresses are computed using a single kernel call.

As we simply organise our processing components (such as nodes and elements) with a 1D sequential ordering, for execution of kernels, we typically use 1D grids organised into 1D blocks of 256 threads (see Figure 8.7). Consecutive kernel threads process consecutive components (e.g. threads $1...n$ process nodes $1...n$ respectively in a nodal kernel, or elements $1...n$ respectively in an element kernel) to increase spatial locality of memory accesses and enable global memory coalescing. While no detailed experiments have been performed regarding optimal block sizes for different kernels, a standard block size of 256 threads per block seems to produce the best performance for all of our kernels with our test GPU (NVIDIA GTX 680 2GB), giving reasonable balance between memory requirements of block, and the ability to fit multiple blocks on an SM.

8.7.3 Computation of Element Nodal Force Contributions

Firstly, muscle stresses are computed by launching a kernel for the contractile components. Additions of the stress values are performed atomically as there may be multiple contractile components per element. The element nodal force contributions kernel can then be launched for each element collection. For collections that overlap muscles, values relating to the deformation of elements that are used in the muscle and material stress computations, such as the deformation gradients, are computed in a separate kernel, and stored before the computation of muscle stresses. For other collections, these values are computed and used only in the element nodal force contributions kernel, which uses a function pointer to the relevant GPU function to compute or read such

values.

As well as obvious optimisations with efficient storage and computations using symmetric matrices like the right Cauchy-Green deformation tensors, when computing, for example, a deformation gradient, ${}^{t}_{0}\check{X}_{ij}$, efficient global memory accesses can be achieved by computing matrix multiplications in stages such that element and nodal values are only accessed once:

$$ {}^{t}_{0}\check{X}^{(a)}_{ij} = \begin{pmatrix} {}^{t}u^{(a)}_{1} \\ {}^{t}u^{(a)}_{2} \\ {}^{t}u^{(a)}_{3} \end{pmatrix} \begin{pmatrix} {}^{0}\check{h}_{a,1} & {}^{0}\check{h}_{a,2} & {}^{0}\check{h}_{a,3} \end{pmatrix} \tag{8.138} $$

$$ {}^{t}_{0}\check{X}_{ij} = \delta_{ij} + \sum_{a=1}^{n} {}^{t}_{0}\check{X}^{(a)}_{ij} \tag{8.139} $$

where a is the node number and n is the number of nodes.

For each material model used by a non-conforming element collection, a weight and average parameters are computed for each element. A pointer to the relevant GPU function that calculates material stresses is also stored, which enables easy integration of material types without modifying the element nodal force contributions kernel. The weighted combination of stresses calculated for each material model is added to any current (e.g. muscle) stresses. As this kernel makes use of many variables, to minimise both register spilling and multiple same-location global memory accesses, shared memory is used for temporary storage of the stress vectors being used by each thread in a block, and these are organised using the same structure of arrays pattern that is used for global memory storage.

When computing forces at an integration point, instead of computing the large strain-displacement matrix for the whole element, which would increase register spilling, rows, r, of the force vector can be constructed independently:

$$ {}^{t}F_{\mathrm{E}r} = \int_{{}^{0}V} {}^{t}_{0}\check{B}_{jr}\, {}^{t}_{0}\hat{\check{S}}_{j} \,\mathrm{d}\, {}^{0}V \tag{8.140} $$

Element nodal force contributions are computed using Gaussian quadrature with the forces computed at each integration point, and additions of these to the internal nodal forces are performed atomically, as nodes are usually attached to multiple elements. Hourglass forces can then be computed for each necessary element collection.

8.7.4 Computation of Nodal Displacements

A kernel is launched to update the unfixed nodal displacements for each node independently using the central difference time-integration scheme, and boundary conditions are then applied to update these according to boundary constraints. For sliding constraints (see Section 8.5), computation of the distance and direction of a node to the closest bound surface point is required. A GPU-based semi-brute-force broad-phase collision detection algorithm with regular spatial subdivision has been implemented to prune the number of polygons to be tested [AGA12], using which nodes are represented as AABBs with lengths of $2 \times d_q$, where d_q is the fixed distance for node q. For a particular sliding constraint, there may be several sliding surfaces, for which primitive and collision data is precomputed as these surfaces don't move.

After performing broad-phase collision detection, a kernel is executed for each surface to compute the closest surface point to each sliding node based on the broad-phase collisions, followed by a kernel to update the displacements for each sliding node based on these values. The former has a loop with a variable number of iterations per thread depending on the number of broad-phase collisions per surface. This could be avoided if the kernel launched a thread for each collision, rather than the group of collisions between a node and a surface, although race conditions could occur when comparing and overwriting the closest surface positions when there are multiple collisions per surface.

8.7.5 Visualisation and Output

As shown by Algorithm 8.3, before rendering a frame, kernels are executed to update nodal positions using the rest positions and displacements, and calculate the new positions of any bound vertices. Vertex normals can then be calculated, with contributions using atomic operations from each connected primitive (in a similar way element nodal force contributions are added to nodal forces). These updates are done after computation of a number of timesteps before outputting graphics frame data to screen. Simulation data at such intervals can also be saved to file for later playback and analysis in real time. During playback, simulation data is updated using values read from the file, requiring no modifications to the visualisation component to visualise this data, and enabling simulations to be continued after all frames have been read.

```
1  foreach timestep per frame do
2  |  Compute timestep (see Algorithm 8.2);
3  |  Update simulation parameters;
4  kernel: foreach nodes do
5  |  Update position using displacement;
6  kernel: foreach bound vertices do
7  |  Compute bound position;
8  foreach primitive collections, P do
9  |  kernel: foreach primitives in P do
10 |  |  Update vertex normals;
11 Update VBOs (see Algorithm 8.4) and render frame;
```

Algorithm 8.3: The process to compute a frame of animation.

The output intervals after a number of timesteps depend on the amount of time passed between the current and next frame (the current frame rate) if the animation is running in real time, or the desired simulation time between frames (the desired animation frame rate) otherwise. For example, to produce an animation at 30 frames per second (i.e. a frame is to be output approximately every 33ms), with a simulation timestep of 5μs, then approximately 6600 timesteps must be computed per frame. If the frame data computation (Algorithm 8.3) and rendering can all be computed in under 33ms, then the animation can run in real time at 30 frames per second. Otherwise, if a timestep can be computed in under 5μs (i.e. the simulation can be computed faster than real time), then the overhead of the additional frame data computation and rendering is preventing the animation from running in real time, and it may be possible to run the animation in real time at a lower frame rate. If a timestep can't be computed in under 5μs (i.e. the simulation takes longer than real time to compute), then the animation cannot run in real time at any frame rate.

For rendering, CUDA-OpenGL interoperability enables necessary simulation data on the GPU to be directly read by OpenGL. As shown by Figure 8.8, a vertex buffer object (VBO) is created for an FE model, which consists of all nodal positions, followed by all nodal colours, both of which are memory aligned for efficient writes using CUDA, and in an appropriate array of structures format for rendering. Two copies of nodal positions are therefore maintained, one of which as structures of arrays for efficient access using CUDA that is used to update data for the VBO. To visualise a vector, such as stresses, the magnitude is clamped and converted to a colour, which is written directly to the VBO. As the visualisation component is uncoupled from the simulation process, it can be easily extended to visualise other variables. A precomputed static index buffer object (IBO) contains a group of node indices to render each element in a collection. A similar approach is used for rendering a surface mesh, whereby a VBO contains vertex positions and normals, a static VBO contains texture data, and a static IBO indexes primitive vertices. Algorithm 8.4 shows the kernels used to update the dynamic VBOs for an FE model and surface mesh.

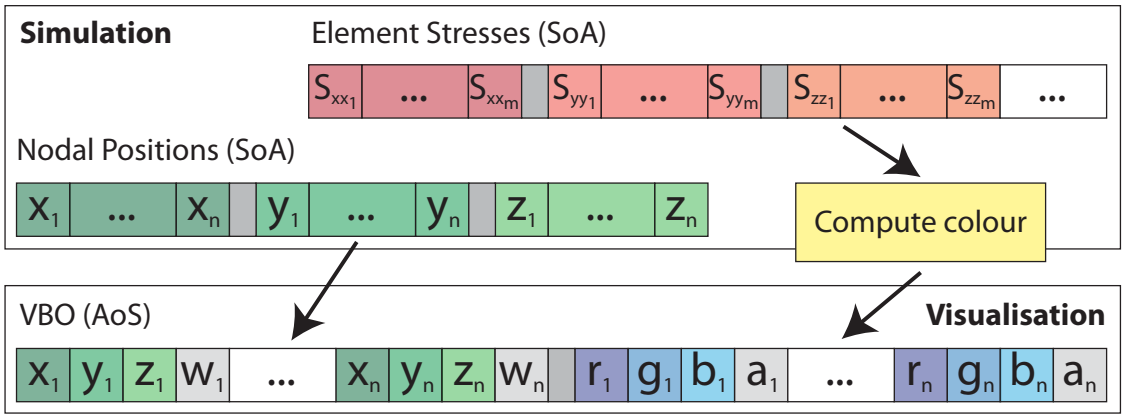

Figure 8.8: An illustration of the updates made to a vertex buffer object (stored in the array of structures format) for visualisation of nodal and element data (stored in the structure of arrays format).

```
1 kernel: foreach nodes do
2 |   Update position in FE model VBO;
3 |   if colour then
4 |   |   Compute colour from necessary values;
5 |   |   Update colour in FE model VBO;
6 kernel: foreach vertices do
7 |   Update position in surface mesh VBO;
8 |   Update normal in surface mesh VBO;
```

Algorithm 8.4: The process to update the VBOs before rendering a frame.

8.7.6 Interaction

As shown by Algorithm 8.3, after computation of a timestep, any updates to simulation parameters are computed and, if necessary, copied from host (main) memory to the GPU; for example, muscle contraction parameters are computed and copied to the GPU. Other boundary conditions, such as applied loads and displacements, can also be computed and set (either on the directly GPU, or on the CPU and then copied) at this point. For example, our system supports user interaction, whereby external forces are applied to a group of nodes. As shown by Algorithm 8.5, this involves two stages. When initialising an interaction (e.g. upon mouse button down), given a position, p_i, the nodes within a user-defined radius of p_i are computed. Efficiency of this stage could potentially be improved using broad-phase collision detection, like with sliding constraints (see Section 8.7.4). Forces are then applied to these nodes (e.g. upon mouse drag, with the mouse position updated once per frame to compute new forces). A similar approach could be taken with, for example, collision detection.

```
   // Find interaction nodes
 1 kernel: foreach nodes do
 2 |  Compute distance to p_i;
 3 Copy distances to host memory;
 4 Find closest node, n, to p_i;
 5 if n is within a user-defined range of p_i then
 6 |  kernel: foreach nodes do
 7 |  |  Compute distance to n;
 8 |  Copy distances to host memory;
 9 |  Determine interaction nodes (within radius of n);
10 |  Copy indices of interaction nodes to GPU;
   // Apply interaction node forces
11 Compute force direction, d_i, from n to p_i;
12 kernel: foreach interaction nodes do
13 |  Compute force (multiply d_i by node mass and user-defined strength);
```

Algorithm 8.5: The process to determine interaction nodes and compute interaction forces.

The interaction process involves both CPU and GPU computation, with memory copies between host memory and the GPU, and vice versa. Some tasks are difficult to compute using the GPU, such as computing the index of the closest node to a position. This requires both the node index and minimum distance to be updated before further distance tests are performed, which would cause race conditions with GPU threads, and probably require a costly lock to be implemented to serialise this portion of the code. Therefore, for this task, distances between each node and the position are computed on the GPU, and then copied to host memory to find the closest node. For efficiency, all memory copies during simulations are done via page-locked host memory.

8.7.7 Optimisation for Voxel-Based Models

Our system has been optimised for use with voxel-based models. For example, only a single set of the various element values, such as a single set of shape function derivatives (12 values) and one unscaled hourglass stiffness matrix (64 values), needs to be stored, rather than a set for each element. While greatly reducing memory usage, this also enables more efficient memory accesses as the same set of values are accessed by each thread, reducing the required number of slow global GPU memory accesses.

Elements and element nodes can also be easily efficiently numbered, which not only means that only 4 node indices per element need to be stored, from which the other 4 indices can be deduced, but the more efficient data storage of nodal values can also improve memory coalescing

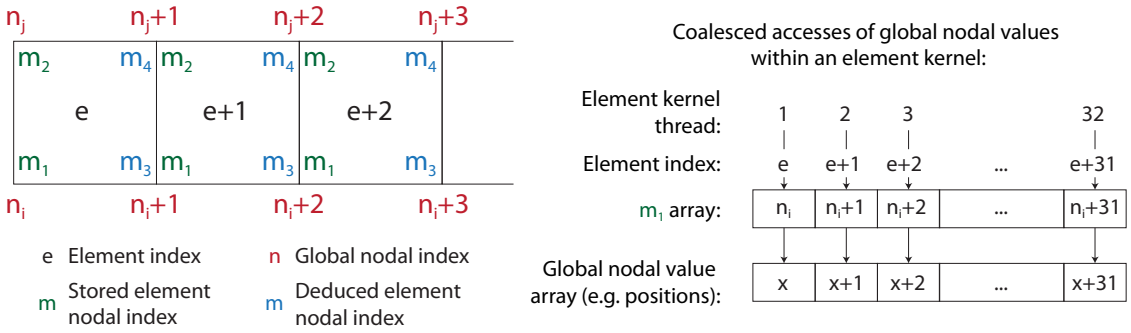

Figure 8.9: An example of efficient global memory accesses of nodal values during a kernel that loops over elements, when using a voxel-based model with efficient element and element node numbering. It is also shown that only half of the element nodal indices need to be stored, while the other half can be deduced.

and global memory cache hits, for example, during the element nodal force computations, as shown by Figure 8.9. Using such an approach to access element nodal values during a kernel that loops over elements, in the worst case scenario (when none of the neighbouring elements are connected in series), just 3×128B memory accesses per warp are required, as opposed to potentially 32×128B (when none of the nodal values are in the same memory block) with unordered and inefficient node organisation. While this value increases at sections where consecutive elements don't belong to the same element collection, all of our models have a maximum of two element collections (one of which overlaps muscles), and large sections of the models contain consecutive elements that belong to the same collection.

8.8 Summary

This chapter has presented our simulation process for simulating the multi-layered FE soft-tissue models generated using our model creation process (described in Chapter 7). By capturing detail such as skin layers, our models and simulation process are capable of simulating complex gross- and fine-scale behaviour, including wrinkling. The TLED FE method is used for the simulations; the TL formulation of the displacement-based FE method is used to approximate the governing equations of equilibrium for a deformable body, forming the equations of motion, which are approximated over time using the explicit central difference time-integration scheme. The requirement of a small but efficient simulation timestep with such a time-integration scheme is highly suited for simulating the complex nonlinear material behaviour of soft tissue under large deformations, as well as contact and sliding. Explicit methods are also inherently parallel, making them suitable for GPU implementation. To further increase simulation efficiency, the TL formulation enables some variables to be precomputed.

Since we use non-conforming models, which may overlap multiple volumes in the surface mesh, elements may be associated with multiple weighted material models, the stresses produced by which are combined when computing internal forces. The voxels of our simulation models are treated as reduced-integration 8-node hexahedra, which are computationally efficient, and overcome volume-locking limitations that occur when simulating incompressible materials like soft tissue with fully integrated 8-node hexahedra. A stiffness-based hourglass control technique is used to counter hourglass effects that occur with reduced-integration elements under large deformations.

The simulation process includes an anatomical muscle contraction model that generates muscle stresses for each muscle overlapping an element, weighted by the proportion overlap between the muscle and non-conforming element. This consists of generating both active and transversely isotropic passive muscle stresses that follow the tension-length properties of real muscle. The

processing of advanced boundary conditions is also part of the simulation process. Nodes of the simulation model can be fixed with zero displacement throughout simulations (rigid nodes), or slide along a rigid non-conforming surface (sliding nodes), enabling the sliding effect between superficial and deep soft tissue to be modelled. This sliding effect is often neglected with current physically based facial animation approaches [SNF05, BJTM08]. The final simulation step involves updating the positions of surface mesh vertices that are bound to the non-conforming simulation model using trilinear interpolation and extrapolation.

Our simulation and visualisation system has been implemented on the GPU using CUDA. This contains two major procedures: computation of element nodal force contributions, and computation of nodal displacements. Using the inherently parallel TLED formulation of the FE method with a lumped mass approximation and mass-proportional damping, the equations for computing these values are uncoupled, enabling them to be computed independently and in parallel for each element and node respectively. Simulations can be visualised during computation using CUDA-OpenGL interoperability, and can be interacted with. Our system has also been optimised to exploit the memory and performance advantages offered when simulating our voxel-based models on the GPU using the TLED FE method. As well as requiring storage of only a single set of the various element values, memory usage is further reduced, and the performance of memory accesses is increased, for example, by efficiently numbering elements and element nodes, leading to efficient organisation of nodal values in memory.

The flexible design and implementation of our simulation process enables new functionality to be easily integrated, such as for different element types, material models and boundary conditions, without affecting the rest of the simulation system. Our system can therefore be easily extended to support different simulation models. Examples demonstrating the complexity and flexibility of our simulation process, as well as some limitations, are discussed in detail with potential enhancements in the following chapter.

Chapter 9

Results, Evaluation and Discussion

Chapters 6 to 8 described our physically based animation approach, illustrating each stage using the forehead model introduced in Chapter 6. Simulating this model in a fully physically based manner, animations of gross- and fine-scale forehead movement, including expressive wrinkles, have been produced. The flexibility of our approach also enables animations of the forehead model using different simulation parameters, such as material properties and muscle contraction properties, to be easily created, as well as animations of different soft-tissue and generic soft-body models. This chapter presents some example models and animations of varying complexity to demonstrate our animation approach. Compared to current similar physically based facial animation approaches, our approach is able to animate fine details like wrinkles in a physically based manner, enabling the production of realistic-looking fine behaviour automatically as a result of the physics simulations without tedious and complex tuning by an animator.

Since this project involves producing realistic forehead animations, quantitative techniques can be used to help evaluate the animations, for example, by comparing animation data to data of real soft-tissue movement. As the main contribution of our animation approach is the simulation of forehead wrinkles (fine details), we have proposed a quantitative technique that can be used to measure and evaluate the accuracy of the simulated wrinkles. Following the approach of Flynn and McCormack to measure the maximum range and average roughness of wrinkles on a 2D profile of a flat skin surface [Fly07, FM08], our technique is also capable of measuring such statistics, but on curved forehead surfaces with complex wrinkling patterns.

Using the results and evaluation, various improvements to our animation approach and system have been identified, and experimental work has been done based on the most significant improvements. Such experimental work includes using thin shell elements that can be used instead of voxel elements to more accurately capture the thickness of the extremely thin outer skin layers without using an infeasible model resolution, improving the realism of wrinkling during simulations. Furthermore, the non-conforming elements can be smoothed near surface boundaries to conform more closely to the surfaces, increasing the accuracy of the models, and reducing the intensity of visual artefacts that occur when simulating a voxel-based model with sharp steps. For performance improvements, work towards a multi-GPU version of the simulation system has also been done.

This chapter firstly presents some results using our animation approach, including generated models and animations. The different example models are introduced, before being discussed further with animation examples. This is followed by a qualitative comparison of our approach to current soft-tissue and wrinkle animation approaches, and a description of our wrinkle measurement technique for quantitative evaluation of the wrinkle animations. Finally, possible improvements and future work related to our animation approach and system are presented. This includes a description of experimental work done towards these improvements.

Layer	ρ^* (kg/m^3)	Young's Modulus (MPa)	Poisson Ratio	Depth (mm)
Stratum Corneum	11,000	48	0.49	0.02
Dermis	11,000	0.0814	0.49	1.8
Hypodermis	11,000	0.034	0.49	Remains
Muscle	11,000	0.5	0.49	∼1
Tendon	11,000	24	0.49	∼1

Table 9.1: The neo-Hookean material properties used for the soft-tissue models. *Includes mass scaling.

9.1 Results

To demonstrate our animation approach, we have used a range of models of varying complexity. Our soft-tissue models can be grouped into three categories of increasing complexity:

1. Flat soft-tissue-block models (see Figures 9.1 to 9.3)
2. Angled and curved soft-tissue-block models (see Figures 9.8 to 9.10)
3. Forehead models (see Figure 9.15)

The surface meshes for all of the models were created in a similar manner to that for the forehead models using the techniques described in Section 6.4. The flat block models conform to the outer skin surfaces. Being cuboid shaped, and aligned with the voxel grid during model creation, they don't contain any sharp steps. Similar models have been frequently used in science and engineering fields to accurately study the behaviour of small areas of soft tissue [FM08, HMSH09].

The angled block models are not aligned with the voxel grid, and therefore contain the sharp steps that are present in more complexly shaped models, such as the curved block models that contain some curvature similar to a forehead. The forehead models capture the full shape and complexity of a forehead, like the forehead model introduced in Chapter 6. The parameters of this model have been varied to produce a range of results in this chapter, demonstrating the flexibility of our animation approach. Finally, a multi-material Stanford Armadillo model is used to demonstrate the applicability of our animation approach to generic soft bodies.

Table 9.1 shows the material properties that were used for the soft-tissue models (unless otherwise stated), based on those reported in literature, and discussed in Section 2.1. A constant total soft-tissue thickness of approximately 4.5mm was used for these models. We model three distinct soft-tissue layers: the stratum corneum (the thickest and most load-bearing layer of the epidermis [MTKL+02]), dermis and hypodermis. Current research on simulating wrinkles suggests that modelling such layers is necessary and sufficient to accurately simulate most wrinkling properties [MTKL+02, FM08]. As discussed in Section 6.4 for the forehead model, since the deep soft-tissue layers are tough and fairly rigid, these were not modelled, and the superficial layers simply slide over the skull or bone surface.

Muscles, and the galea aponeurotica (tendon) in the forehead model, are contained within the hypodermis layer. These structures are roughly 1mm thick through their thickest points, and have smooth edges to model the smooth blend between surrounding structures and connective tissue. Muscle parameters were estimated based on literature [RP07] and from testing. Each muscle was assigned an active and passive stress reference of 5MPa, an optimal fibre stretch of 1 (rest length), and a passive stretch parameter (λ_P in Equation 8.132) of 1.2. An estimated mass-proportional damping scale factor of 2Ns/m was used, and external forces that have little visual effect on the animations, such as gravity, were neglected.

Our example models are discussed further with animation examples in the following sections. Unless otherwise stated, each of the animations represent linear variations of muscle contraction parameters for the contracted muscles between 0 and 0.75 over 500ms. Mass and time scaling

Detail	Uniform Skin Block	Single Muscle Skin Block	Flat Forehead Block	30° Angled Forehead Block	Y Curved Forehead Block	XY Curved Forehead Block	Face	Armadillo
				Basic Model Statistics				
Nodes	18,910	146,410	514,710	541,198	533,372	542,799	629,178	19,698
Elements	16,200	129,600	459,000	465,600	463,800	466,288	503,530	15,107
Surface Vertices	2,864	16,512	17,085	17,085	18,571	20,199	11,571	48,421
Surface Polygons	5,064	31,728	32,774	32,774	35,586	38,584	22,239	93,840
				Detailed Model Statistics				
Element Size (mm^3)	0.5^3	0.5^3	0.5^3	0.5^3	0.5^3	0.5^3	0.5^3	2.5^3
Voxel Samples	4^3	4^3	4^3	4^3	4^3	4^3	4^3	10^3
Element Muscle Computations*	5,400	12,358	76,604	81,298	78,241	79,172	70,044	0
Rigid Nodes	310	1,210	3,810	5,270	4,809	5,296	9,563	596
Sliding Nodes	1,891	14,641	51,471	70,434	64,113	70,492	118,276	0
Sliding Surface Polygons	2	2	2	2	2,800	5,600	735	0
Bound Surface Vertices**	2,778	16,458	16,955	16,955	16,955	16,955	6,116	44,091
				Model Generation Performance (Single CPU Thread)				
Time (mins:secs)	0:13	1:30	4:10	4:59	5:19	5:58	6:00	0:41
Memory (MB)	130	1224	2487	2735	2919	3070	3174	144
				Model Simulation Performance (GPU)				
Timestep (ms)	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.15
Timestep Computation Time (ms)	0.57	1.43	4.04	4.35	5.10	7.47	6.28	0.14

Table 9.2: Statistics of the example models and simulations, using an Intel i7-3930K CPU and an NVIDIA GTX 680 GPU.
*$\sum_{e \in E} n^{(e)}$ where E is the set of all elements overlapping a muscle, and $n^{(e)}$ is the number of muscles overlapped by element e
**Includes skin, muscle and tendon vertices inside the model bounding box

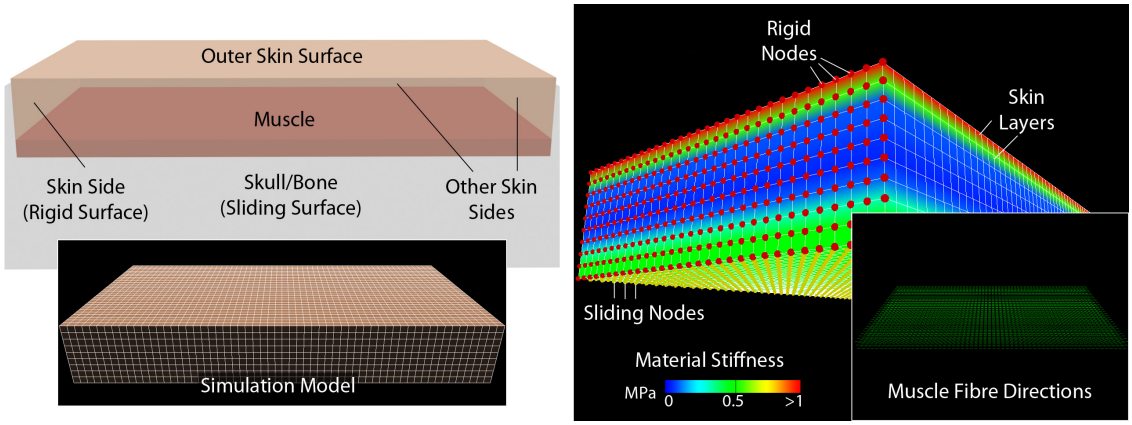

Figure 9.1: A flat soft-tissue-block model containing uniform soft-tissue layers throughout the model.

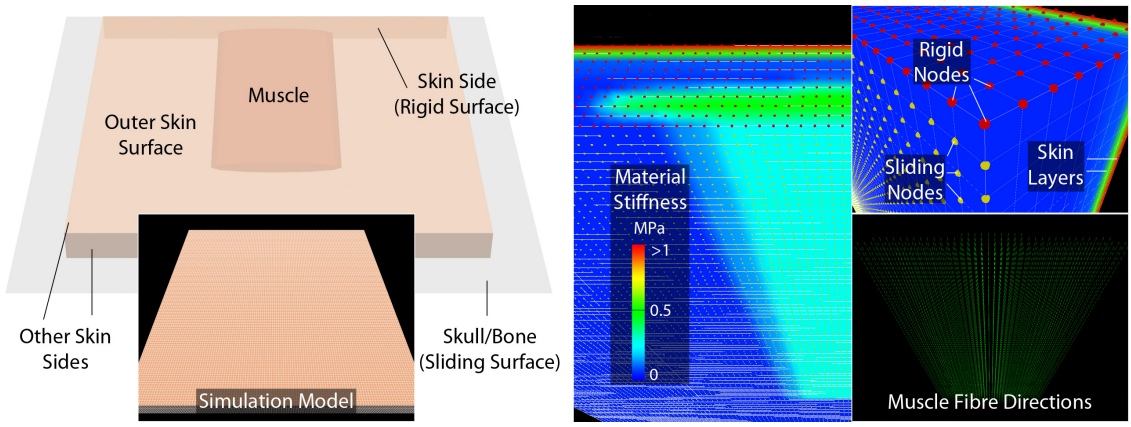

Figure 9.2: A flat soft-tissue-block model with a single muscle.

were used to increase stability and improve performance; for example, contractions over 500ms were produced by varying the contraction parameters over 50ms, and playing back the animations slower using time scaling (more discussion of mass and time scaling is presented in Section 9.4.1). Considering both mass and time scaling, such animations were played back 10x slower. Table 9.2 shows some model and performance statistics for our model and animation examples. Such complex, high-resolution models are necessary to capture the thin structures, such as skin layers, and simulate fine wrinkling behaviour.

9.1.1 Soft-Tissue-Block Animations

Figures 9.1 to 9.3 show some flat soft-tissue-block models, and Figures 9.4 to 9.6 show some animations using these models. The simplest model (Figures 9.1 and 9.4), with dimensions 15 × 30 × 4.5mm (length × width × height), contains uniform soft-tissue layers throughout the model with a single muscle spanning the width and length of the model. As such, the muscle acts as a separate layer, above which is the hypodermis, rather than being contained within the hypodermis. One end of the model is anchored using rigid nodes representing the muscle attachment. During simulation, the muscle contracts towards this end, whereas it would compress to the middle if it was unconstrained. The model also rests on a plane representing bone that is defined as a sliding surface.

This simple model resembles a very small area of soft tissue. Similar models are often used

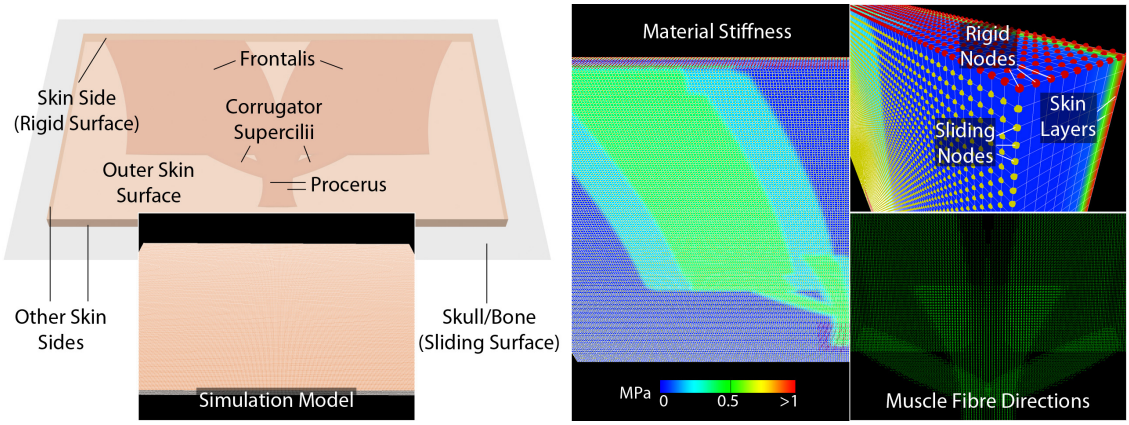

Figure 9.3: A flat soft-tissue-block model with a forehead-like muscle structure, containing the frontalis, procerus and corrugator supercilii muscles.

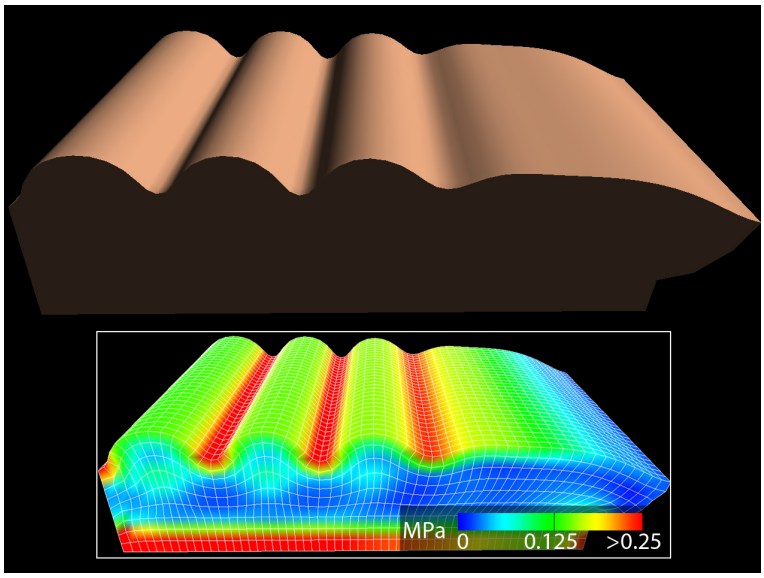

Figure 9.4: Animation of the flat uniform soft-tissue-block model under muscle contraction. The inset shows the stresses.

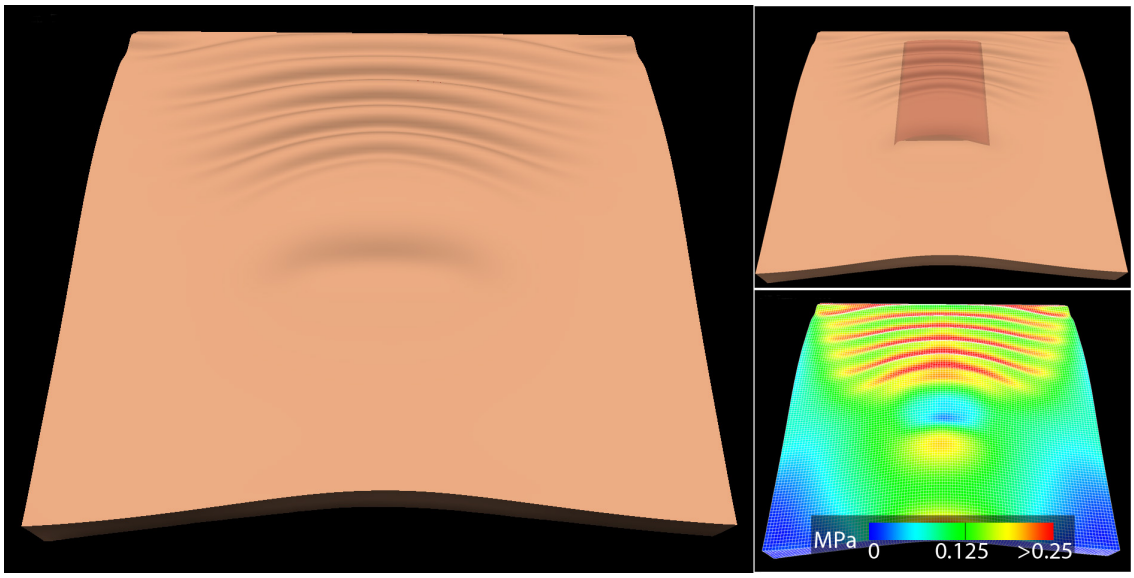

Figure 9.5: Animation of the flat single-muscle soft-tissue-block model under muscle contraction. The inset at the top right shows the deformed underlying muscle, and the inset at the bottom right shows the stresses.

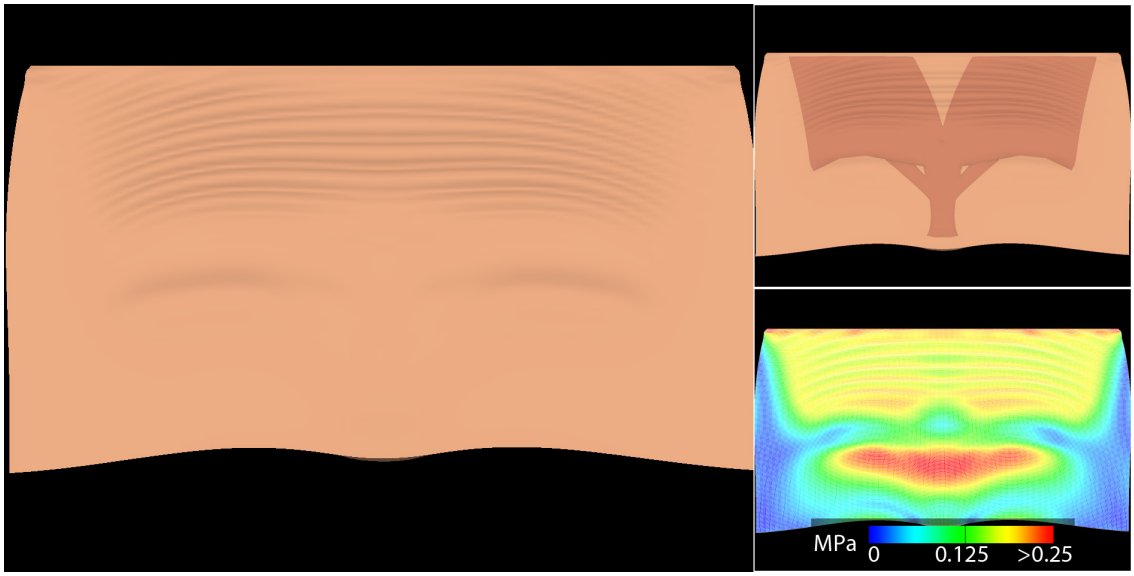

Figure 9.6: Animation of the flat forehead block model under contraction of the frontalis.

in science and engineering fields, whereby high-resolution models of very small areas of soft-tissue are used to accurately study the behaviour of soft-tissue [FM08, HMSH09]; however, the elements of such models would usually be sized appropriately such that they conform to all structures of the model (while the elements of the model in Figure 9.1 conform to the outer surfaces of the model, some overlap boundaries between skin layers and the muscle). These models are normally created and manually manipulated, for example, to set boundary conditions, using a complex FE modelling tool, whereas our model creation process automatically generates full, simulation-ready models.

Figure 9.7 shows some animation frames from the animation of this simple model, as well as the frames from an animation using a similar model but with thinner cuboid-shaped elements that more accurately capture the thickness of the epidermal layer. As well as the final result, the animation frames show the wrinkling patterns that are produced during the animations, with smaller wrinkles merging to form larger bulges when compressed further. These results are discussed further in Sections 9.2.1 and 9.5.1.

Other flat soft-tissue models include a larger $60 \times 60 \times 4.5$mm model with a single muscle that is embedded, and doesn't span the width and length of the model (Figures 9.2 and 9.5), and a $150 \times 85 \times 4.5$mm model with a forehead-like muscle structure, containing the frontalis, procerus and corrugator supercilii muscles (Figures 9.3 and 9.6). With the forehead block model, attachment areas on the skull surface for the procerus and corrugator supercilii muscles are modelled using rigid nodes. Due to the large dimensions of the forehead block model, compared to the single-muscle block model, a lower-resolution outer skin surface containing larger polygons was used for performance reasons. Since these polygons are larger than the element faces, this led to the production of some visual artefacts on the animated skin surface.

To test the effect on simulations of the sharp steps that are present in the complexly shaped forehead model, we first angled the flat forehead block model such that it wasn't aligned with the voxel grid during model creation (Figures 9.8, 9.11 and 9.12). Two such models were created and simulated, one of which angled with a gradient of 1 at 45° (therefore with uniform voxel steps), and the other angled at 30°. Similar wrinkling patterns to the flat forehead block model were produced, but with some artefacts due to the sharp steps in the voxel-based models (this issue is discussed further in Section 9.5.2). While the more complex angled model with non-uniform voxel steps produced slightly irregular and lower-quality wrinkling patterns, the animation of this model demonstrated that wrinkles can be simulated on such models. Increasing the complexity of these models, curvature around one axis (Figures 9.9 and 9.13) and then two axes (Figures 9.10 and 9.14) was added to the model angled at 30° such that it more closely resembles the shape of a real forehead. Like the angled models, the curved models also produced similar wrinkles to the flat forehead block model, but with some visual artefacts.

9.1.2 Facial Animations

Figure 9.15 shows the final forehead model that was introduced in Chapter 6, and Figure 9.16 shows some animations using this model with simulation parameter variations. Like the forehead-based soft-tissue-block model, this model contains the frontalis, procerus and corrugator supercilii muscles. Also, the skull surface has been split into multiple surfaces: the majority of the skull is a sliding surface, while the attachment area for the procerus and corrugator supercilii muscles is a rigid surface. Since the galea aponeurotica is modelled to anchor the frontalis during contraction, this muscle has no fixed attachment area.

With the forehead model, as only part of the facial model has been created, nodes along the boundaries to the remainder of the face have been set as rigid to provide the anchoring effect that connecting elements would provide with a full model. The size of the simulated area was carefully chosen to ensure that the magnitudes of propagated forces at boundary regions were small enough to ensure no visible artefacts appeared at such regions during simulations. A smaller area could have been simulated without producing artefacts by using a more advanced approach to constrain boundary regions, for example, by gradually increasing the stiffness of the material towards the boundaries, or only restricting the movement rather than completely fixing the positions of the

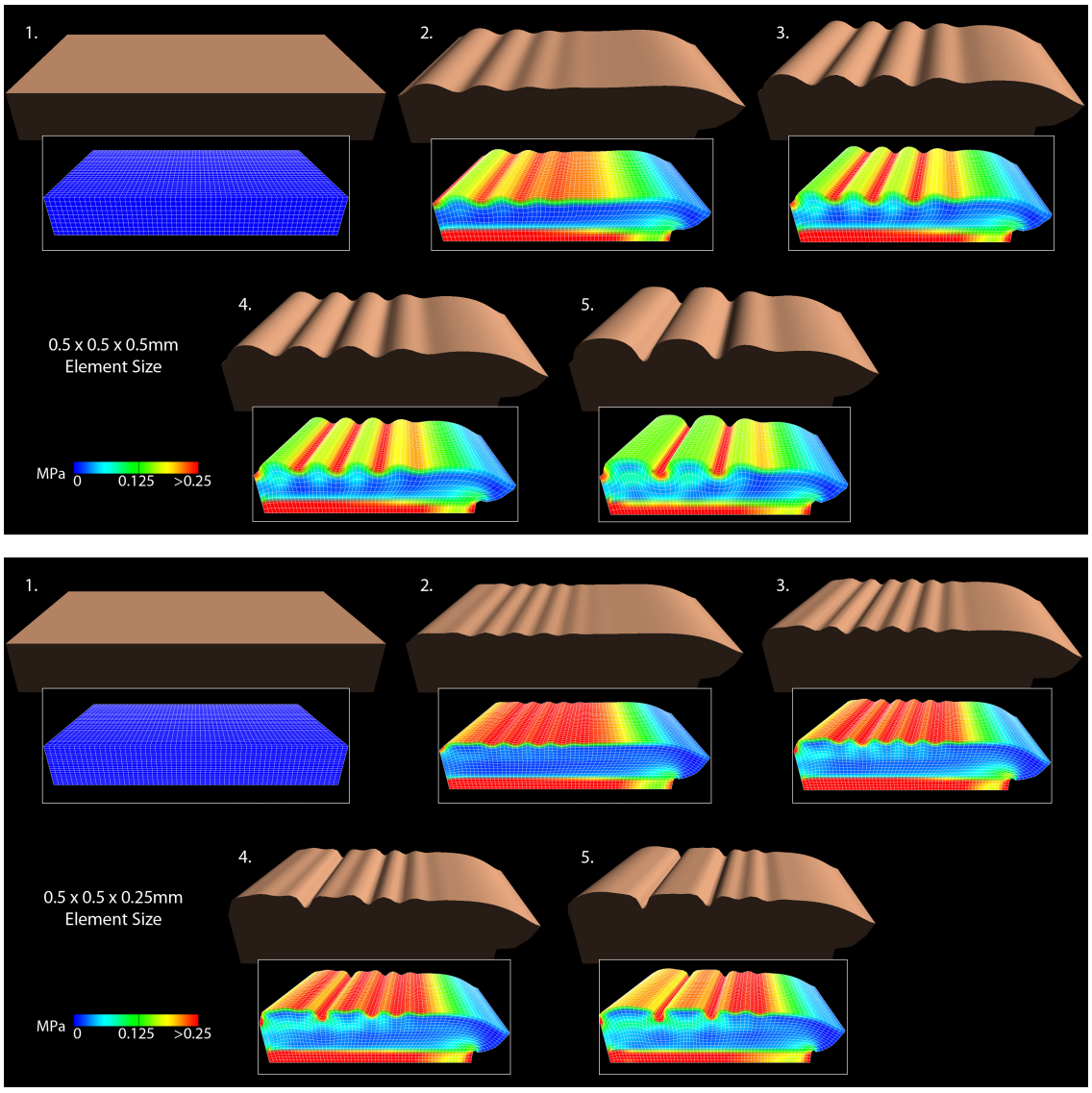

Figure 9.7: Animations of soft-tissue-block models with element heights of 0.5mm (top) and 0.25mm (bottom) under muscle contraction.

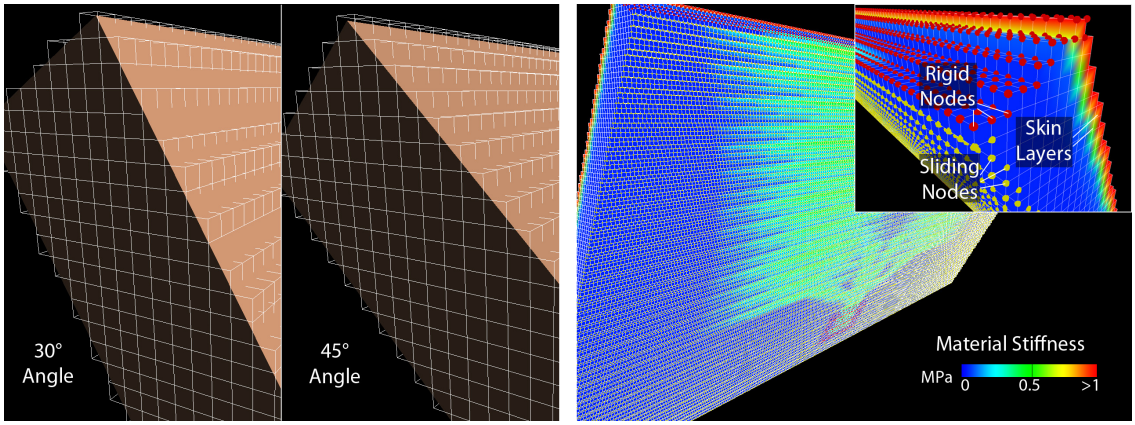

Figure 9.8: An angled soft-tissue-block model with a forehead-like muscle structure. The image on the left shows two such models: one angled at 30°, and one angled at 45°, while the image on the right shows the model angled at 30° in detail.

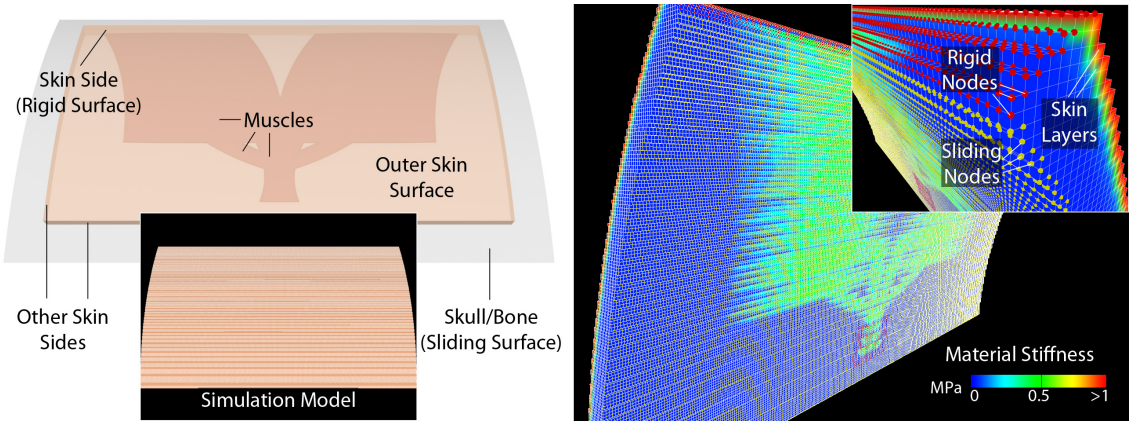

Figure 9.9: A soft-tissue-block model with a forehead-like muscle structure, curved around a single axis.

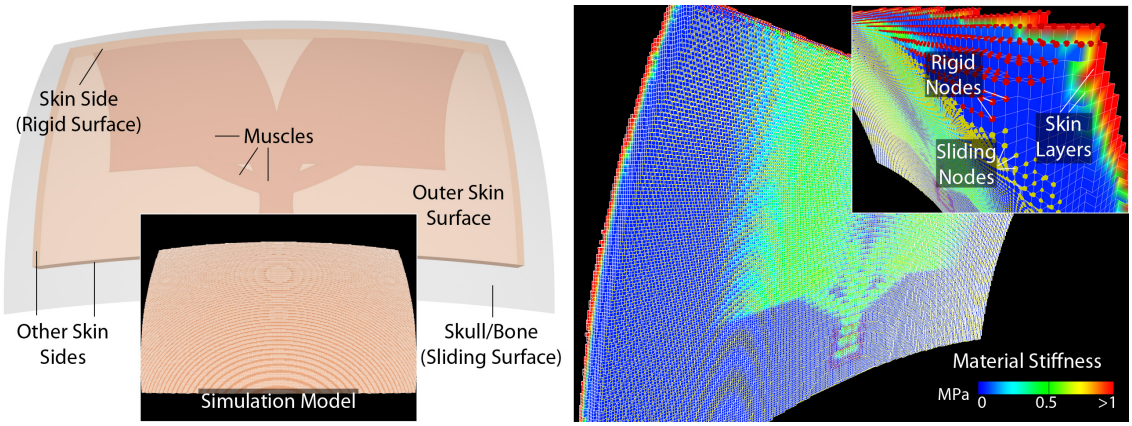

Figure 9.10: A soft-tissue-block model with a forehead-like muscle structure, curved around two axes.

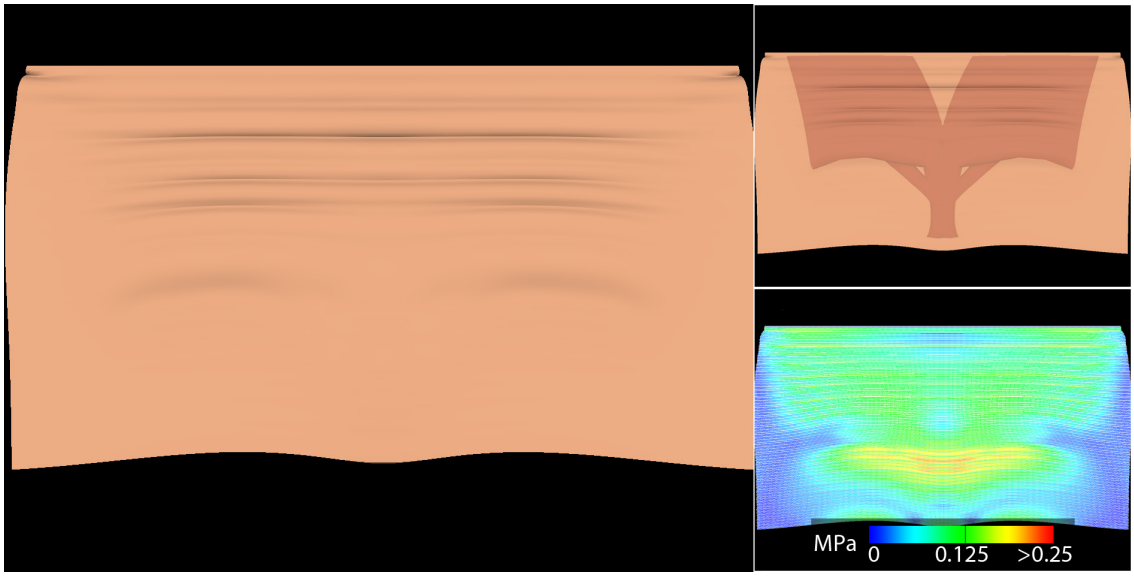

Figure 9.11: Animation of the forehead block model angled at 30° under contraction of the frontalis.

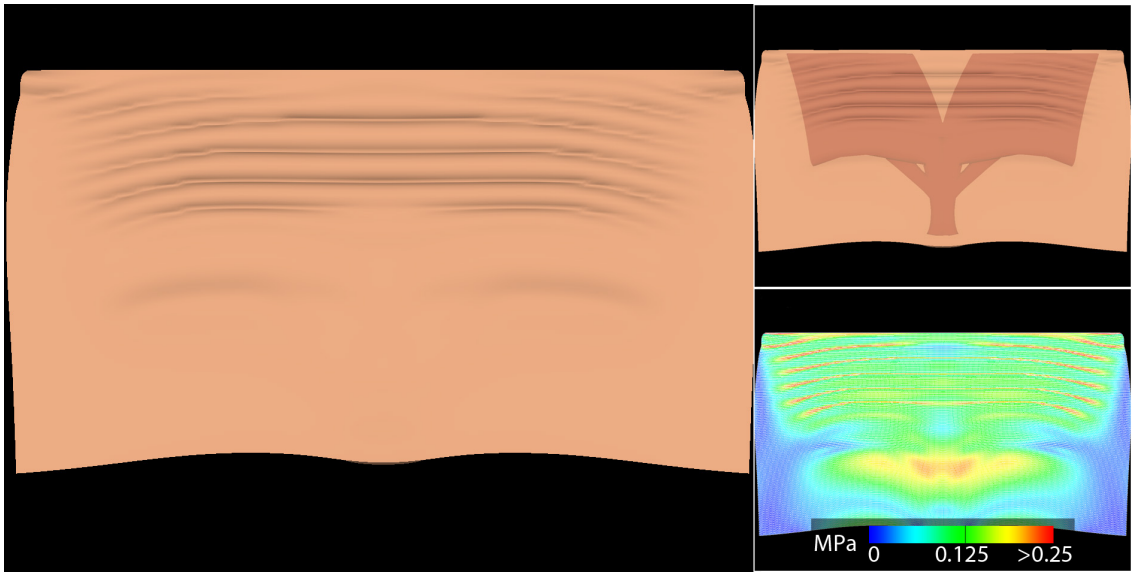

Figure 9.12: Animation of the forehead block model angled at 45° under contraction of the frontalis.

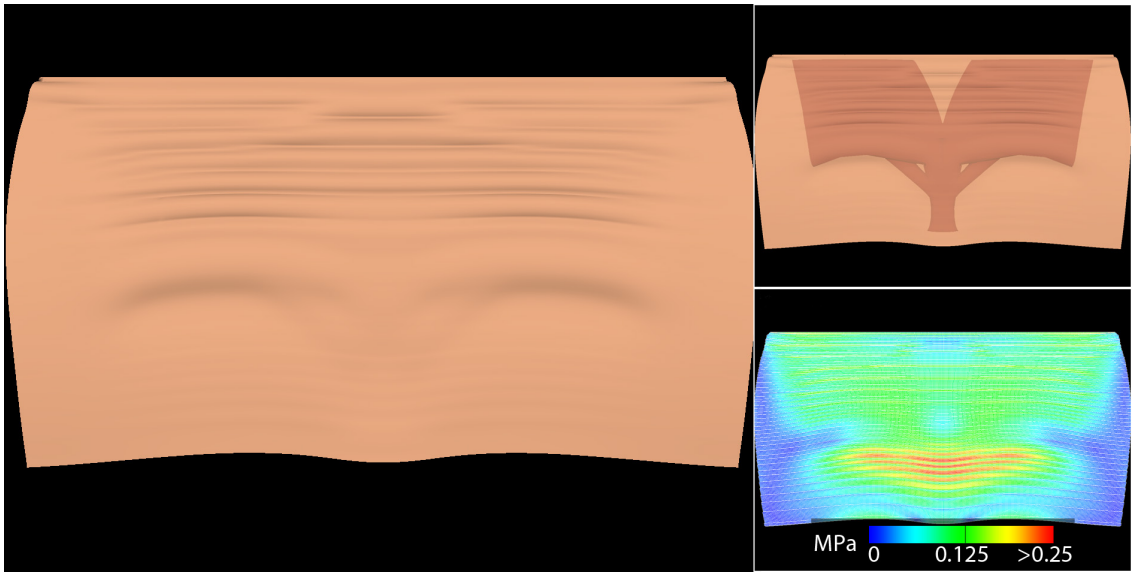

Figure 9.13: Animation of the forehead block model curved around a single axis under contraction of the frontalis.

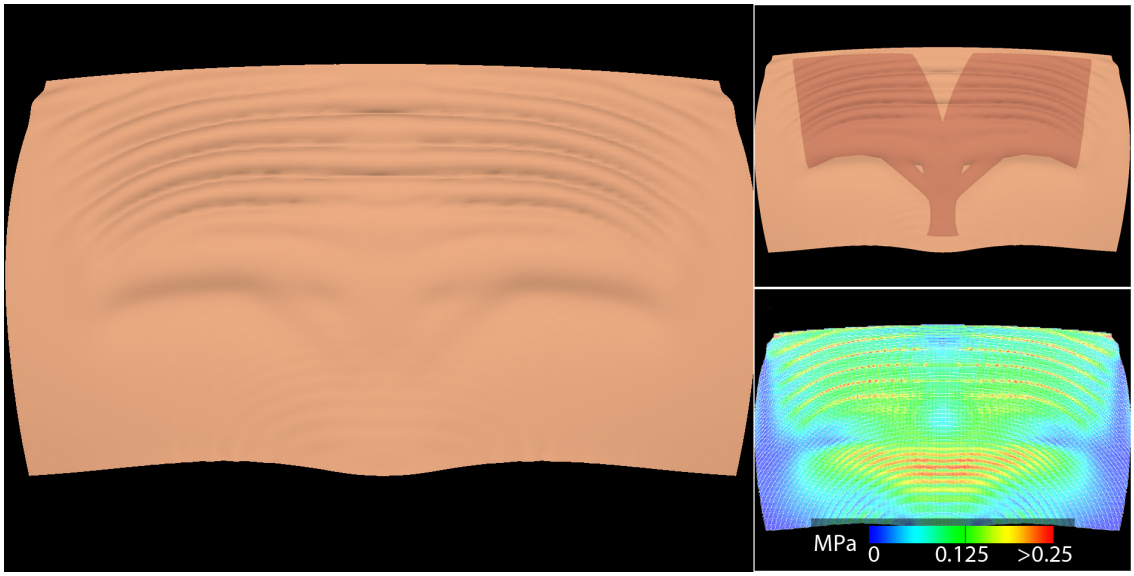

Figure 9.14: Animation of the forehead block model curved around a two axes under contraction of the frontalis.

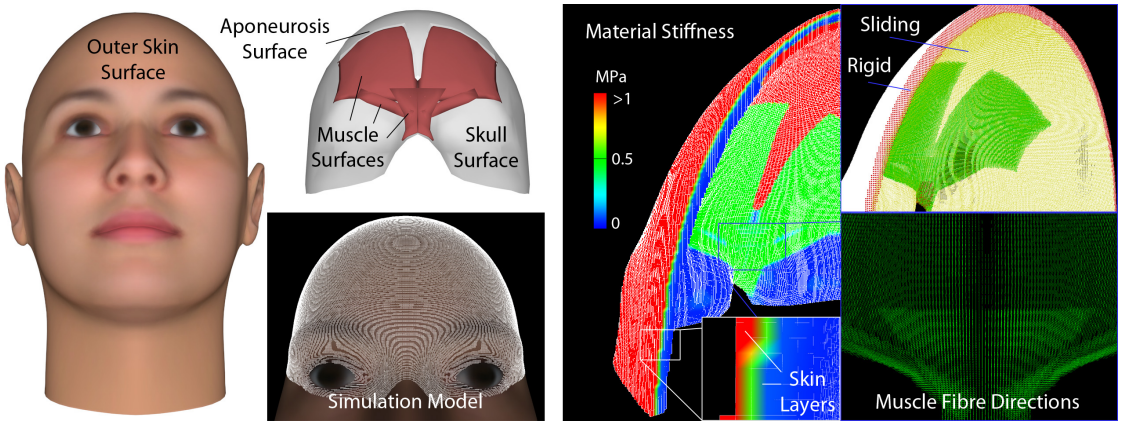

Figure 9.15: **Left:** Surfaces and the simulation model for a forehead including the frontalis, procerus and corrugator supercilii muscles. **Right:** Rear views of the forehead simulation model.

nodes along the boundaries; however, constraining large movements at the boundaries can limit the overall deformation of the entire model, making it appear overly stiff.

The various forehead animations in Figure 9.16 show that, by simply changing the material parameters of a model, or using a different muscle structure, it is possible to achieve different effects. When a lower epidermal stiffness is used (Subfigure 3), representing younger skin, fewer and shallower wrinkles are produced. With higher muscle stress references (Subfigure 4), the eyebrows are raised further, and the forehead skin is compressed more, producing more pronounced wrinkles. The passive anchoring effects of the procerus and corrugator supercilii muscles cause a dip in the wrinkles in the middle of the forehead that isn't present when these muscles aren't included (Subfigure 5). Thicker soft tissue (Subfigure 6) leads to the production of fewer but larger and deeper wrinkles.

In reality, facial muscle structures contain individual variations. One significant variation with muscles of the forehead is that in distance between the left and right frontalis [Sta09]. With the forehead model in Figure 9.15, there is a gap between these muscle bellies filled by the galea aponeurotica, and this gap widens towards the top of the head. With the forehead model in Subfigure 7 of Figure 9.16, this gap has been closed, leading to the production of wrinkles that span the length of the forehead, while the lower wrinkles just above the eyebrows still have a dip in the middle of the forehead due to the passive effects of the procerus and corrugator supercilii muscles. Finally, Figure 9.17 shows a detailed image of both the animated skin and underlying muscle surfaces for the forehead animation when using higher muscle stress references. A photograph of real wrinkles has been overlaid onto the animated skin surface for qualitative comparison. A higher-resolution skin surface was used to produce this animation, increasing the quality of animated wrinkles.

9.1.3 Computational Performance

Regarding model creation performance, the main bottleneck of the model generation process was the computation of parametric coordinates of NURBS volumes when computing muscle fibre directions. Using our unoptimised Newton-Raphson root-finding algorithm to compute these coordinates contributed to roughly 38% of the computation time. Also, while memory requirements are quite high, small sections of larger models could be generated independently, as computations of properties for each element and node are independent.

Using a simplified version of our GPU-based simulation system, early tests showed performance increases of over 130x with models of up to 250,000 4-node tetrahedral elements compared with our CPU-based solver. This made the implementation of the full system on the GPU an obvious choice. Using our current GPU-based system, optimisations when using voxel-based models have

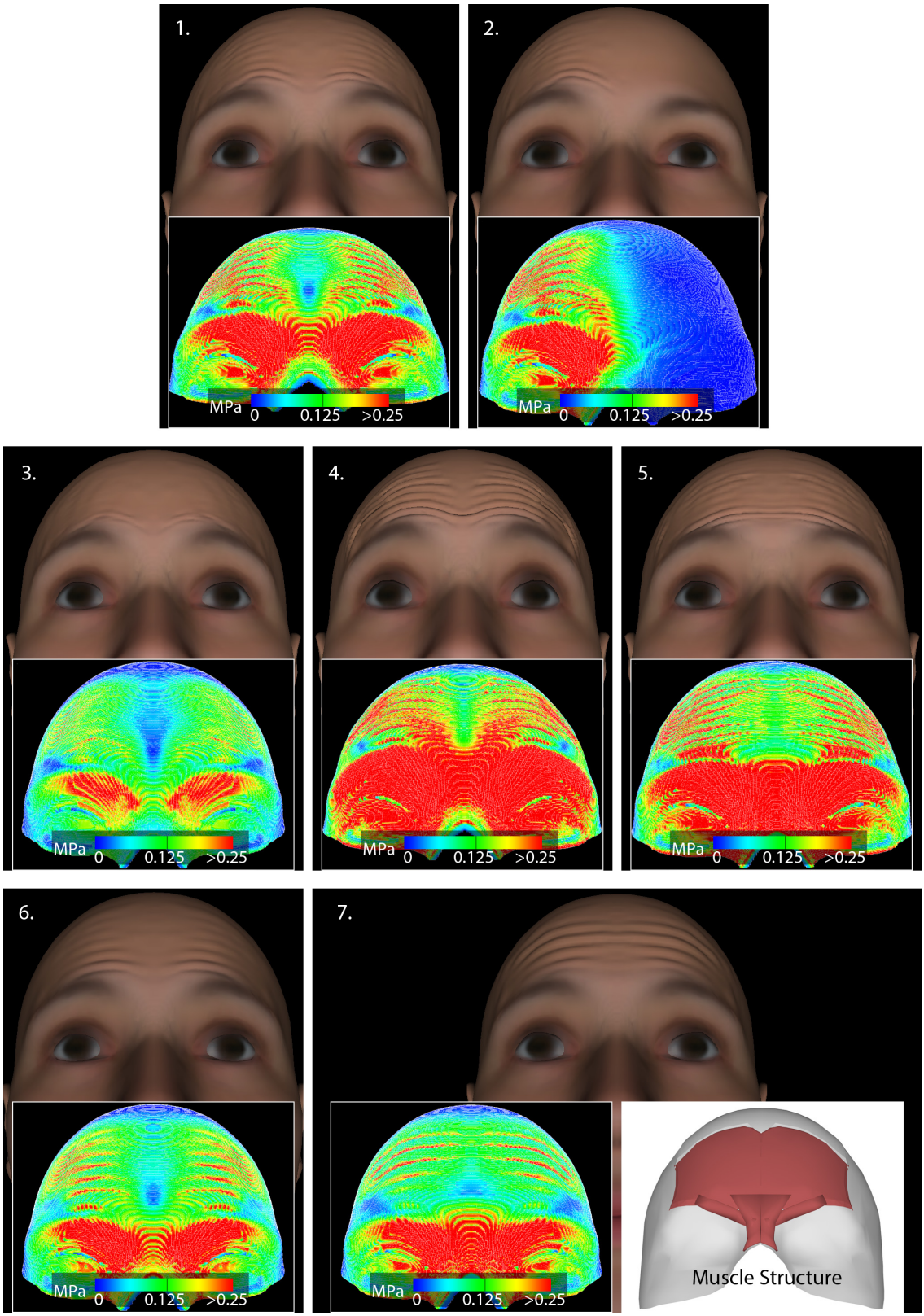

Figure 9.16: Variations of the forehead model under contraction of the frontalis. **1-2.** The standard forehead model as described. **3.** Lower epidermal stiffness of 24MPa, representing younger skin. **4.** Higher active and passive muscle stress references of 50MPa. **5.** Only the frontalis, and not the procerus and corrugator supercilii muscles, is modelled. **6.** Thicker hypodermis, resulting in a total soft-tissue thickness of 5mm. **7.** Thicker soft tissue, and a modified frontalis muscle such that there is no gap between the left and right muscle bellies, as shown by the inset.

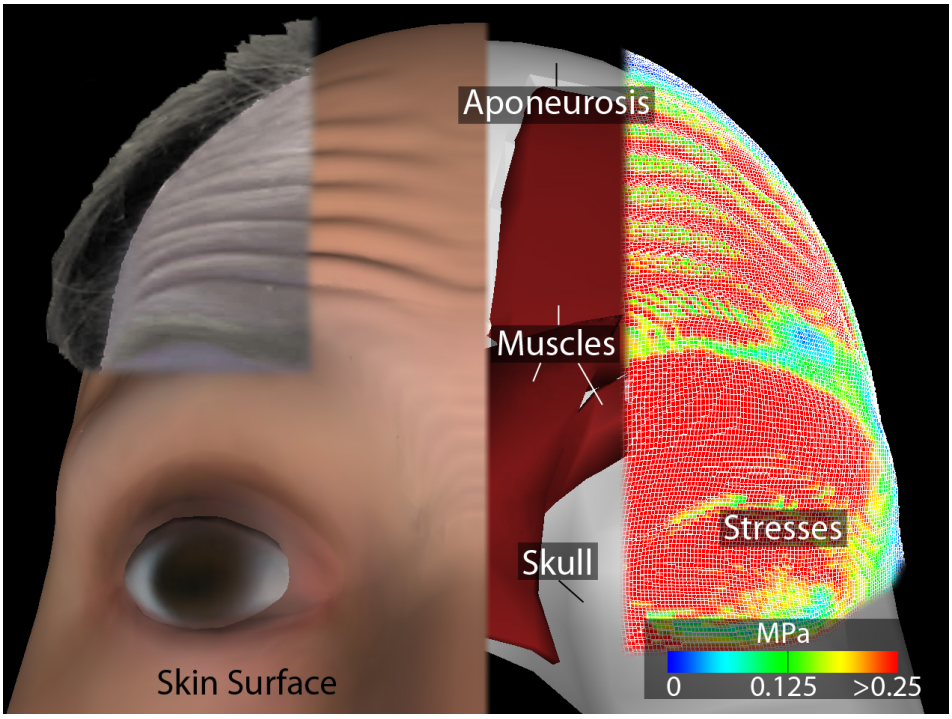

Figure 9.17: A detailed image of the forehead animation when using higher muscle stress references, showing the outer skin surface, underlying structures and stresses. A photograph of real wrinkles has been overlaid onto the skin surface.

led to performance increases, even of almost 2x with low-resolution models containing around 3,000 elements, compared with using a conforming hexahedral model.

Despite the performance advantages, the soft-tissue simulations aren't real time due to the required model complexity to capture the necessary detail for wrinkle simulation, such as skin layers. The computation time for producing one second of animation using our forehead model, with a simulation timestep of 0.005ms, and considering mass and time scaling, is roughly 126 seconds. One computational bottleneck is the processing of sliding constraints, which contributed to over 35% of the timestep computation time with the forehead simulations, containing 118,276 sliding nodes. Our implementation of this procedure can create uneven workload between GPU threads, and could be further optimised (see Section 8.7.4).

Since single-precision floating-point arithmetic offers much higher computational performance on current GPUs [NVI12], we used single precision for all floating-point values with our simulation system implementation. Experiments using an NVIDIA GTX 680 GPU showed these to offer sufficient accuracy with our simulations to produce the realistic-looking animations we desire, with very little visual difference between the simulations when using single- and double-precision values, while double-precision floating-point arithmetic was over 2x slower. On the other hand, inaccuracies and instabilities started to occur with our simulations when using single-precision values with a timestep lower than 1μs, although such a low timestep wasn't required for our simulations.

Finally, Figure 9.18 shows an example of a more generic multi-material object, and Table 9.3 lists the material properties that were used for the different parts of the model. This model and animation example demonstrates the flexibility of our model creation and animation approach. Fewer larger elements were used (but with a higher voxel sampling rate during creation), which are suitable for animation of gross object movement. With this lower-resolution model, real-time frame rates were achieved.

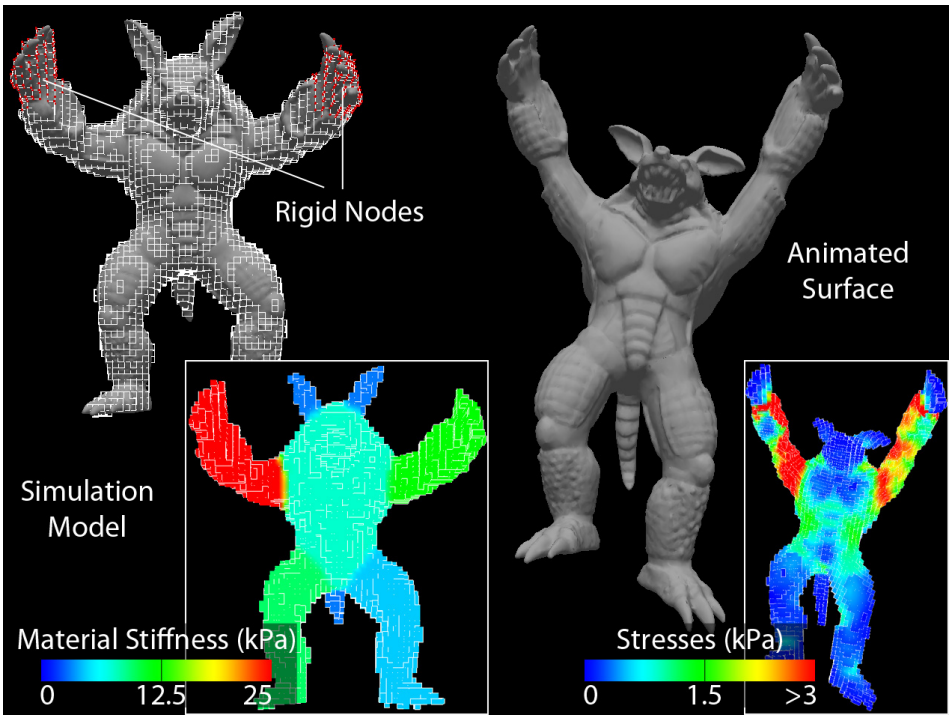

Figure 9.18: Animation of a multi-material Stanford Armadillo under gravity.

Body Part	ρ (kg/m^3)	Young's Modulus (MPa)	Poisson Ratio
Torso	1,000	0.0075	0.4
Left Arm	1,250	0.0125	0.4
Right Arm	750	0.025	0.4
Left Leg	1,250	0.005	0.4
Right Leg	750	0.01	0.4
Left Ear	1,000	0.003	0.4
Right Ear	1,000	0.003	0.4
Tail	1,000	0.003	0.4

Table 9.3: The neo-Hookean material properties used for the multi-material Stanford Armadillo model.

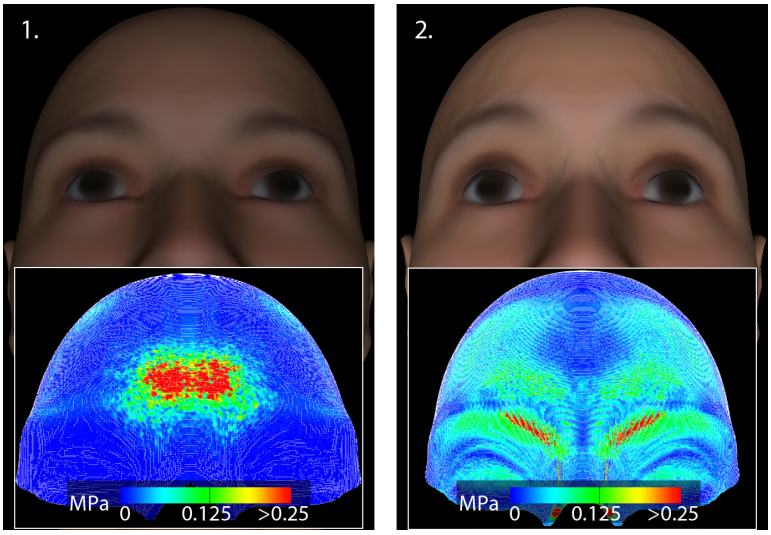

Figure 9.19: **1.** A less anatomically accurate forehead model, created using the earlier simpler version of our model creation process, under contraction of the frontalis. **2.** The current forehead model with skin represented as a single layer (material), meaning wrinkles aren't able to be simulated.

9.2 Qualitative Comparison to Other Soft-Tissue and Wrinkle Animation Approaches

Our animation approach can be compared to current techniques that involve soft-tissue and wrinkle animations. We compare to approaches in two categories of such techniques. The first category includes physically based soft-tissue animation approaches that simulate soft tissue, some of which include wrinkle simulation, in a purely physically based manner. The second category includes hybrid soft-tissue and wrinkle animation approaches that simulate soft tissue in a physically based manner, but use a geometric wrinkle model to layer wrinkles onto the simulation result. Such approaches have been used in computer graphics applications to add details to coarse physically based simulations [ZST05b]. While we only perform qualitative comparisons to other approaches, further evaluation against such approaches could be performed with quantitative comparisons using our wrinkle measurement approach in Section 9.3. Finally, a discussion between our physically based animation approach, and performance-capture facial wrinkle animation approaches is made, although no qualitative comparisons are made due to the complexities of capturing such data, particularly with fine details like wrinkles.

9.2.1 Physically Based Soft-Tissue Animation Approaches

Before being enhanced, an earlier version of our model creation process generated much simpler models [WM12]. This process determines whether a voxel is inside a volume, and consequently its material type, using a single sample at the voxel centre, and the skin and hypodermis are treated as a single rather than multi-layered material. With such models, all nodes on a restricting surface, such as the skull, are also set as fixed with zero displacement throughout the simulations, rather than being able to slide, and muscles contract towards a single centre of contraction, rather than along a fibre field.

As a comparison to our current forehead example model, Figure 9.19 shows animation results using a similar model that was also generated using our forehead surface mesh as input, and contains a similar number of nodes and elements, but was created using the earlier, simpler process. While it took just 1 minute 20 seconds to generate this model, using 2.95 GB RAM, it

lacks anatomical accuracy compared with our current, more advanced models, for example, with jagged boundaries and no blending of material properties between the non-overlapping structures, and the poor approximation of muscle fibre directions. Furthermore, while each timestep took 5.07s to simulate, no wrinkles were simulated as only a single skin layer was modelled, and stretching effects can be seen along the lateral edges of the frontalis due to poor approximation of muscle fibre directions. Compression of muscles towards a single point also reduced simulation stability.

To demonstrate the importance of using more anatomically accurate models to simulate soft-tissue behaviour, Figure 9.20 shows the equilibrium states of simple simulations involving a block of soft tissue under contraction of a single muscle using simulation models of different complexity and anatomical accuracy. The most anatomically accurate model was created using our current model creation process. As expected, best results are achieved by using the most anatomically accurate model with three layers of skin and connective tissue, accurate muscle fibre directions, and nodes that are able to slide across the skull/bone.

Modelling the skin and connective tissue as a layered structure seems to have the largest effect. As shown by Figure 9.20 (and also with a forehead animation example in Figure 9.19), when considering skin and connective tissue as a single layer (material), like with many current approaches [SNF05, BJTM08], no wrinkling is produced, even if the material properties are calculated as an average of those used with the layered models. Due to this, no wrinkles are produced when just a single sample at the centre of each voxel is used to calculate element properties, as there are no samples close enough to the outer skin surface to capture the material properties of the epidermis. When the muscle contraction direction is simply towards a central contraction point, rather than along a complex fibre field, wrinkles still form, although some unrealistic wrinkling effects seem to occur, particularly towards the muscle origin. On the other hand, when the bone is treated as a rigid rather than a sliding surface, wrinkles similar to those produced by the most anatomical model start to form, although the restricted movement prevents these wrinkles from forming fully.

Qualitatively, there are also similarities between our simulations and the highly accurate uniform skin-block simulations by Flynn and McCormack used for skin wrinkling studies [Fly07, FM08], particularly with our uniform soft-tissue-block model. As shown by Figure 9.7, similar wrinkling patterns occur, with relatively sharp furrows and wide bulges, and finer wrinkles under small loads grouping into larger bulges under higher loads. On the other hand, as we simulate less detailed models with lower accuracy, our simulations produce less detailed wrinkles; for example, the larger bulges are smooth, rather than containing finer furrows and bulges. However, visually, such fine details seem to be very difficult to see, particularly on forehead wrinkles. Such details are therefore typically neglected with computer graphics animations [ZST05b], or modelled using a texture map [BKN02].

Additionally, by using less detailed simulations, we are able to more easily and efficiently simulate much larger and complex areas, such as the entire forehead. For example, Flynn and McCormack used complex orthotropic viscoelastic materials with their uniform skin-block models, and these required fitting to experimental data to determine the complex parameters [Fly07, FM08]. They also used different element thicknesses for different skin layers, and a higher element resolution along the direction of wrinkling. Using such element size variations with our animation approach would be difficult as the elements of our complexly shaped models don't conform to the skin layers, and aren't aligned with the more complex and variable wrinkling patterns.

For performance comparison, taking into account the element sizes used by Flynn and McCormack, and the skin layer thicknesses presented in Table 9.1, an average element size of 0.36mm^3 can be computed. Using this element size, our soft-tissue models would contain over 2.5x as many elements, which would also lead to the requirement of a lower timestep due to the smaller element sizes, and likely the use of double-precision floating-point values due to the lower timestep, as discussed in Section 9.1.3. With the additional timesteps and longer timestep computation, as well as the use of additional thin shell elements to simulate the epidermis, and complex orthotropic viscoelastic materials, the computation time for producing one second of animation using a modified version of our forehead model containing this level of detail would likely be in the order of several hours rather than minutes, even with our GPU-based optimisations.

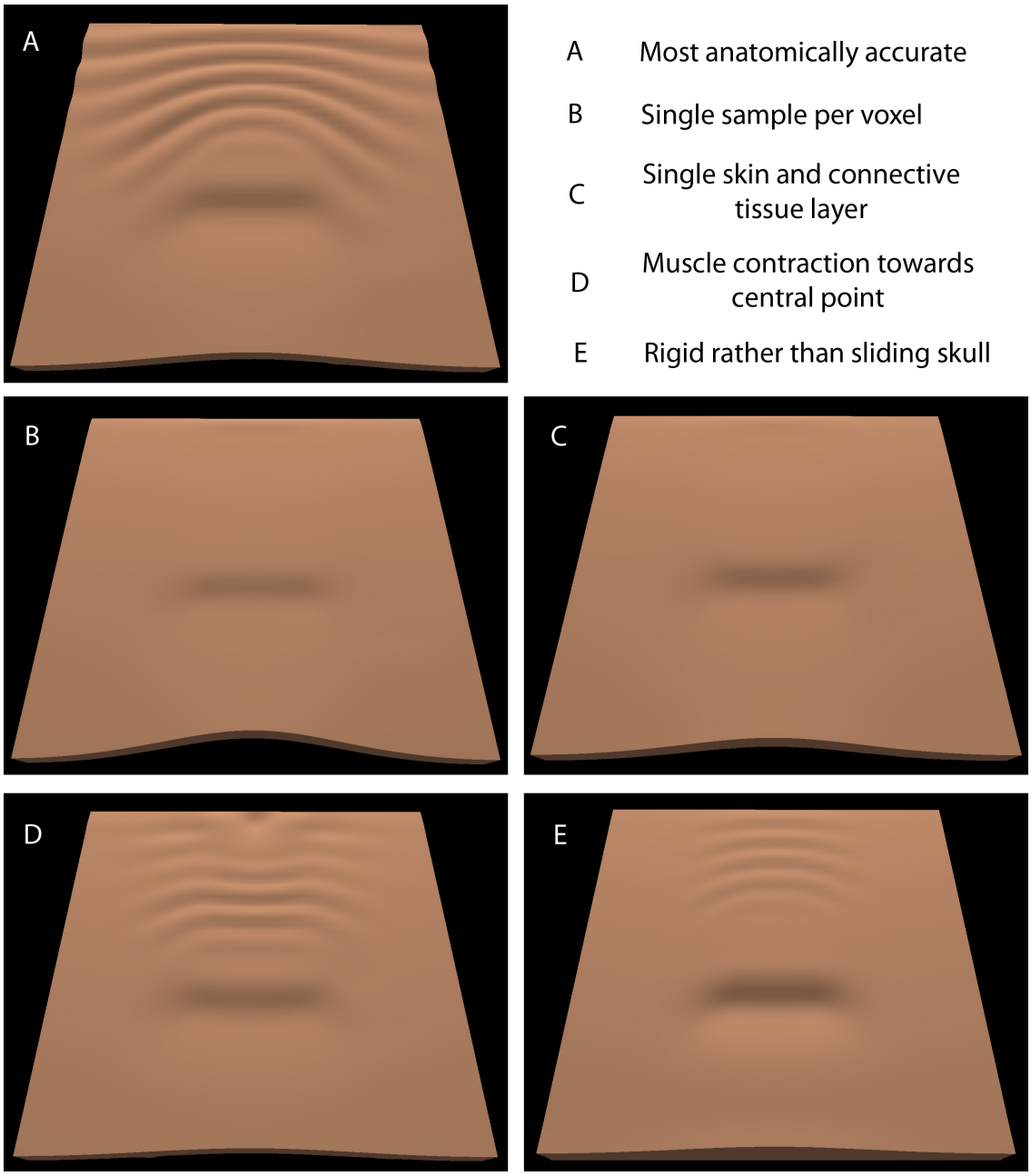

Figure 9.20: The equilibrium states of soft-tissue-block simulations under influence of a single muscle. The models for each simulation has different anatomical accuracy, with the most accurate being created using our current model creation process described.

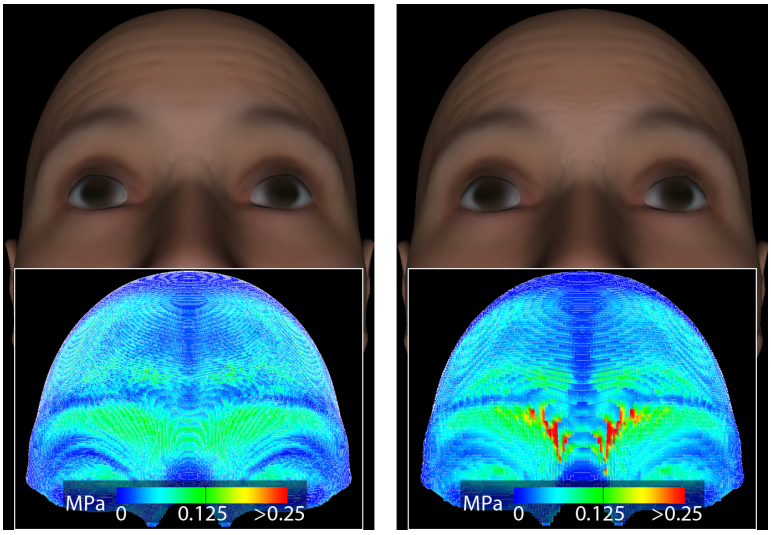

Figure 9.21: Forehead models with a single skin layer under contraction of the frontalis, where the wrinkles have been simulated using a geometric wrinkling technique. A lower-resolution simulation model was used for the animation on the right.

9.2.2 Hybrid Soft-Tissue and Wrinkle Animation Approach

There are currently no known approaches for producing animations of both large- and fine-scale soft-tissue movement, including wrinkles, over a large area, such as the forehead, in a physically based manner. However, various hybrid facial animation approaches simulate the gross facial movement using a physics-based technique, while wrinkles are produced using texture-mapping or geometric techniques [WKMMT99, ZST05b]. By using a fully physically based approach, we aim to advance upon such approaches, and produce more realistic fine details like wrinkles, with the additional advantage that additional manual configuration of a geometric technique isn't required to produce wrinkles. As part of the evaluation, we have therefore compared our fully physically based approach to a hybrid approach based on that proposed by Zhang et al. [ZST05b].

The facial animation approach by Zhang et al. uses an MS facial model with an extension of Waters' muscle model [Wat87] for animating gross facial movement. Three types of muscles are used: linear muscles are used for modelling the majority of facial muscles, sphincter muscles for the orbicularis oris and orbicularis oculi, and sheet muscles for the frontalis. Each muscle type has a different-shaped area of influence (AOI), and forces are applied to nodes inside a muscle's AOI based on their location within the AOI when the muscle is contracted. For animating wrinkles, each muscle type also has a wrinkling region inside its AOI. An amplitude is calculated for all nodes inside the wrinkling region based on their location within this region when the muscle is contracted, and the heights of such nodes are modulated during muscle contraction based on the skin surface deformation around the nodes.

With our hybrid approach, we use our physically based animation approach to simulate only the gross forehead movement without the wrinkles. This is done by representing the skin and connective tissue as a single layer (material), like with many current physically based facial animation approaches [SNF05, WHHM13], in which case skin wrinkles are unable to be simulated (as shown in Section 9.2.1). Following the approach by Zhang et al., we define similar regions, but use them only for animating wrinkles, as the nodal forces due to muscle contraction are computed using our FE muscle contraction approach. As such, the wrinkle regions are defined using points on the outer skin surface, rather than having attachment points below the skin surface.

Figure 9.21 shows an animation produced using our hybrid soft-tissue and wrinkle animation approach, with wrinkles simulated using the described geometric technique. Two rectangular wrinkle regions (analogous to the rectangular-based sheet muscle AOIs used by Zhang et al.) were

used, one for each side of the forehead approximating the shape of the frontalis. While forehead wrinkles are produced, it is likely that more realistic-looking wrinkles could be produced by tweaking the geometric wrinkling equation and parameters. There are therefore many complexities when designing geometric equations that produce realistic wrinkles without artefacts, and with configuring such equations to produce realistic-looking wrinkles. Furthermore, parameters, such as the wrinkling regions, would need to be altered to be used on a different facial model with different forehead dimensions, whereas, with the physically based approach, wrinkles are automatically simulated from the deformation of the physically based model. On the other hand, a lower-resolution physically based model can be used for better computational performance, since only the gross detail is simulated in a physically based manner (like with the multi-material Armadillo animation in Figure 9.18).

9.2.3 Performance-Capture Facial Wrinkle Animation Approaches

Various other state-of-the-art computer graphics approaches for animating faces and geometric facial wrinkles use performance-capture techniques [MES05, BBA+07]. While a comparison to these would be ideal, the setup requirements and configuration of such approaches, along with the required manual data cleanup, make this much less feasible and more time consuming. Furthermore, direct qualitative comparisons using our created forehead model with our physically based approach and a performance-capture approach, like with the other approaches in this section, would require the performance-capture data to be mapped to this fictional model, which would likely require a lot of manual work. Alternatively, a new physically based forehead model could be created, or qualitative comparisons using two different models could be made.

Therefore, a further disadvantage of performance-capture approaches is that the captured data can often only be easily used on the model for which it was captured. They are also limited to the captured data, and it can be difficult to combine captured data to create new expressions that haven't been captured. Some expressions are also very difficult to capture, particularly if fine details are required, such as wrinkles. On the other hand, while our physically based approach requires more work and tweaking to create a suitable facial surface mesh, such as the creation of surfaces for internal structures like muscles, once a suitable facial model has been created, any expressions can be simulated in a realistic manner according to the physics of the model. Slight changes to the model don't require new data to be captured, or captured data to be altered, and one can also easily create animations for models for which data cannot be easily captured, such as fictional characters.

9.3 Quantitative Evaluation of Wrinkles

For quantitative evaluation, it is necessary to measure statistics of wrinkles on the facial surface meshes, for example, to compare those wrinkles simulated with different models and simulation parameters, and real skin wrinkles. Flynn and McCormack measured the maximum range and average roughness of wrinkles on a flat skin surface with identical wrinkling patterns across the entire width of the surface [FM08]. Following this approach, we measure such statistics, but on curved forehead surfaces with complex wrinkling patterns.

To compute 3D surface wrinkle measurements, it is necessary to measure the deviations of the surface from its ideal form, which is simple for a flat surface. With a curved surface, a representation of the surface in its ideal form could be found, and the deviations measured against this, although this is a complex task for an arbitrary surface. Alternatively, the overall surface curvature could be removed such that its ideal form is a flat surface, although in some cases this is impossible without artefacts or cutting the surface (e.g. spherical surfaces).

Techniques for removing the overall curvature of surfaces are often used when examining surfaces in materials science, using techniques such as plane fit and flattening [VS10]. These involve subtracting fitted surfaces or curves from the surface or surface scan profiles. However, a simply

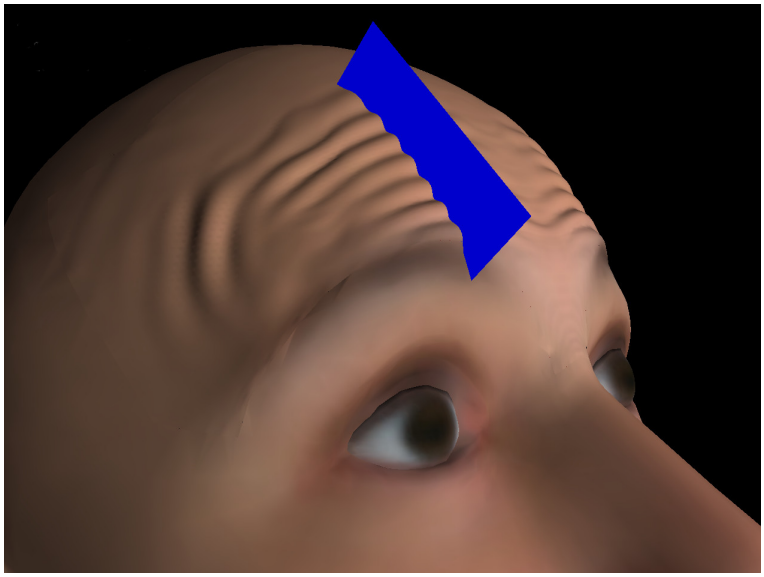

Figure 9.22: The main profile plane for the right frontalis.

subtracting a curved surface is likely to produce artefacts with our wrinkled surfaces, producing wider wrinkles on flatter areas, and thinner wrinkles on more curved areas.

As with the approach by Flynn and McCormack [FM08], we compute wrinkle measurements using 2D profiles of a 3D surface, as shown by Figures 9.22 and 9.23, which can be easily uncurved. While such 2D profiles don't provide information about an entire surface, they enable us to compare our simulations while avoiding the complications concerned with measuring wrinkles on a 3D surface.

9.3.1 Our Wrinkle Measurement Approach

Our wrinkle measurement approach consists of three main stages:

1. Computing the 2D surface profiles
2. Removing the overall curvature of the profiles
3. Computing the wrinkle measurements of the uncurved profiles

For the first stage, we precompute several planes to intersect the surface based on the initial undeformed muscles. These are the fixed estimated locations at which wrinkles should occur. Then, to measure wrinkles at any time during the simulation, the ends of the deformed muscles (origin and insertion) are used to determine the profile ends on the plane, between which wrinkles are measured. The profile can then be uncurved, and wrinkle measurements computed.

To compute the 2D profiles, as wrinkles normally form perpendicular to the muscle contraction direction, a main plane is created through the centre of the muscle based on the average muscle fibre direction, and a projection direction to the skin surface (see Figure 9.22). The points of intersection between the plane and muscle ends (origin and insertion) are projected along the plane to the skin surface to determine profile length. This profile measures the maximum wrinkling caused by contracting the muscle.

Using approximate initial muscle shapes and volumes, several secondary planes are also created to measure the wrinkles as they start to curve and fade towards the sides of the muscles (see Figure 9.23). We estimate the positions of the secondary planes at either the edges of the muscle, or the locations where the cross-sectional muscle area becomes less than 70% of the cross-sectional muscle area along the muscle centre (using the main plane). Figure 9.23 shows the main and secondary

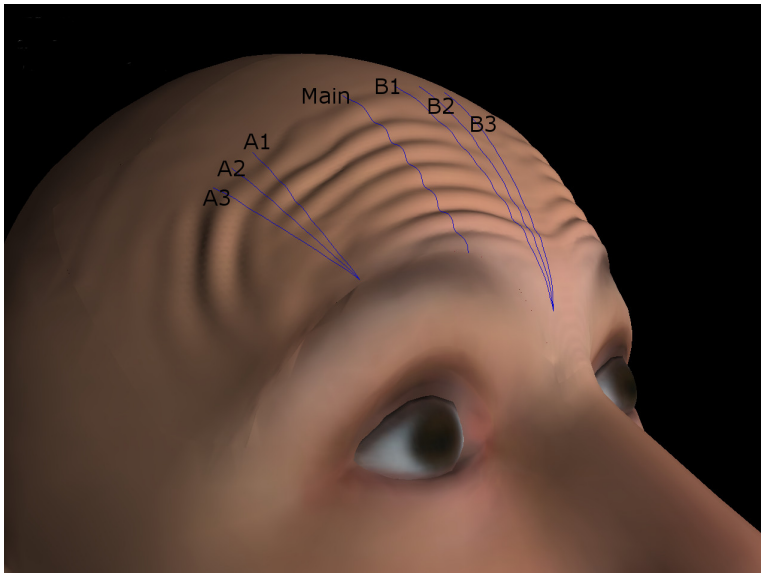

Figure 9.23: Main and secondary profiles (A1-A3 and B1-B3) for the right frontalis.

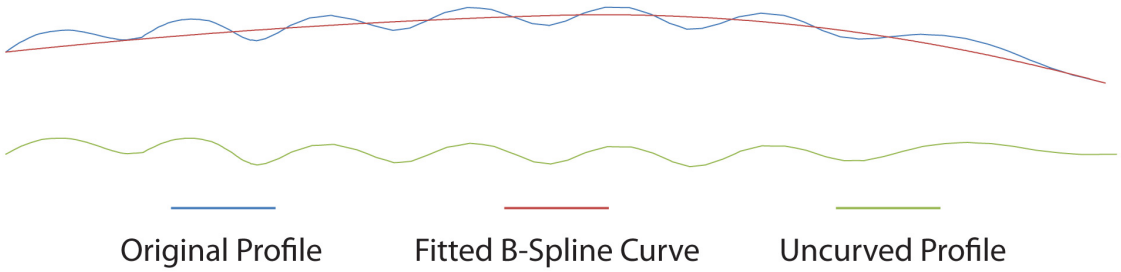

Figure 9.24: The main profile (original and uncurved) for the right frontalis (see Figure 9.23).

profiles for a forehead model. Finally, the main muscle planes are used to create an additional plane between a pair of muscles to measure the wrinkling produced by their combined contraction, such as a plane along the centre of the forehead to measure the wrinkles produced when both sides of the frontalis are contracted.

Once the profiles have been computed, the overall profile curvature is removed. This is done by fitting a quadratic B-spline curve with four control points to the profile, as shown by Figure 9.24. The total least squares regression implementation in the Geometric Tools (Wild Magic) library[1] is used to fit the B-spline. Such a curve can provide a reasonable approximation to the overall curvature of our forehead profiles. The B-spline now represents the x axis (essentially uncurving the B-spline), with the ends of the B-spline being assigned the coordinates $(0, 0)$ and $(B_{\mathrm{L}}, 0)$, where B_{L} is the length of the B-spline. Each profile point is projected onto the B-spline, producing a parametric coordinate. This is used to interpolate the x values of the B-spline ends, while the distance from the B-spline is used as the y value of the point. The resulting profile has no, or very little, overall curvature (see Figure 9.24).

Wrinkle measurements can be easily computed using the uncurved profiles. For example, the average roughness of a 2D profile requires computation of the area between the profile and a line representing the mean profile height, which is now simply the sum of the areas of multiple rectangles and triangles. The maximum range is simply computed using the maximum and minimum y values.

[1] http://www.geometrictools.com/

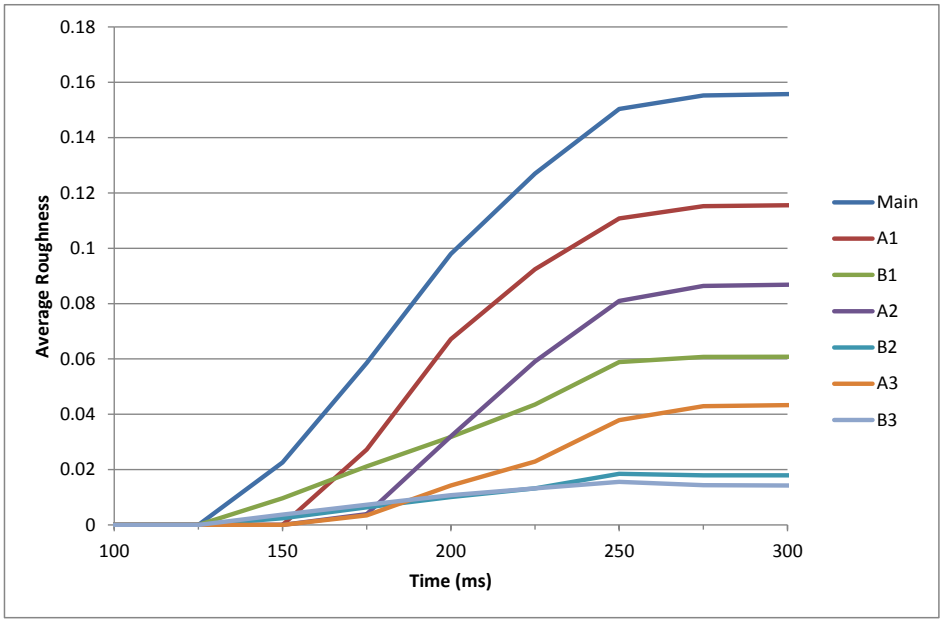

Figure 9.25: The average roughness of the right frontalis profiles for the forehead model (see Figure 9.23) during frontalis contraction.

9.3.2 Wrinkle Evaluation

To demonstrate our wrinkle measurement technique, Figure 9.25 shows the average roughness of the right frontalis profiles for the forehead model (labelled in Figure 9.23) during frontalis contraction. While a quantitative evaluation of our simulated forehead wrinkles against real forehead wrinkles would be desirable, this would require data of real forehead wrinkles. No such data has been found in literature, although approaches have been proposed to accurately measure skin wrinkles for wrinkle studies [Hat04, FM08]. Capturing such fine detail would require extremely high-precision measurement or performance-capture equipment[2]. Our wrinkle measurement approach could then be easily used to measure wrinkle statistics of surfaces, or even just 2D profiles constructed from the captured forehead data, and a comparison of these statistics to those computed for our simulated forehead wrinkles could be performed.

9.4 Discussion

The results presented in this chapter have demonstrated the ability of our animation approach to produce different animations of realistic large- and fine-scale soft-tissue behaviour, including wrinkles, on the face. Our model creation process can create any multi-layered FE model from any surface mesh that contains hole-free volumes (not just soft-tissue models). Such FE models can vary in complexity, from a complex face with many muscles, ligaments and tendons, to a simple generic object with a single surface and no muscles. The model examples demonstrate the complexity of models that can be created, which include skin layers, anatomical fibre directions, and advanced boundary conditions. Our simulation process can simulate such models efficiently on the GPU, and includes an anatomical muscle contraction model, and the processing of complex sliding boundary conditions. Also, by simply changing the material parameters of a model, it is possible to achieve different effects, such as the animation of different-aged skin. The animation examples show the flexibility of our approach to animate different human or non-human soft-tissue

[2]One possible capture system is a 3dMDface System (http://www.3dmd.com/3dMDface/), which can capture at a resolution of < 0.2mm, but only supports static captures.

structures, producing animations of both large and fine details using an accurate physics-based technique.

Various significant discussion points arise from our animation process and results, which are discussed in more detail in the following sections. We firstly discuss the use of mass and time scaling, compromising physical accuracy to increase stability and improve performance. This is followed by discussions relating to simulation model creation, since the animations produced depend highly on the model, such as muscle shape and size, and soft-tissue thickness and parameters, meaning more accurate models would likely enable more realistic animations to be produced. Finally, the flexibility of our animation system design and implementation is also discussed. Further discussion points related to experimental work are also covered in Section 9.5, including discussion on the modelling and simulation of the epidermal layer, the thickness of which cannot be accurately captured with hexahedral elements, and the artefacts present during simulations due to the sharp steps in the voxel-based models.

9.4.1 Mass and Time Scaling

With the soft-tissue animations, mass scaling was used to greatly increase the critical timestep, which is roughly proportional to the material density [TCO08]. The actual density of each material ($\sim$1100kg/m^3 [BJTM08]) was scaled by a factor of 10. As the magnitudes of the muscle contraction forces are dependent on fibre stretch rather than mass, the animation equilibrium states were not affected. If other forces were included, such as gravity, the magnitudes of which are affected by mass, these would need to be scaled to prevent the mass scaling from greatly affecting the equilibrium states. While mass scaling affects the transient response of dynamic simulations, producing a slower, more sluggish response, these effects can be reduced by scaling other parameters, such as reducing damping, and playing back the animations at a faster speed. For more accurate animations, mass scaling could be applied selectively to the necessary elements, reducing the amount of non-physical mass introduced [OSU05].

Time scaling was also used, whereby muscles were contracted at a faster speed, and the animations played back slower, reducing the number of timesteps over which the animations were run. While this can result in a more energetic transient response, these effects can be reduced by scaling other parameters, such as increasing damping. Using both mass and time scaling, damping parameters and the playback speed of animations were determined experimentally. As inertial effects are still relatively insignificant compared to muscle contraction forces when using such scaling parameters, the scaling techniques appeared to have little visual effect on the animations, particularly when played in real time.

9.4.2 Simulation Model Creation

In Figure 9.16, some artefacts can be seen on the forehead animations due to the inaccuracy of the surface mesh. The animations produced are highly dependent on the model and parameters; for example, the direction of movement, and wrinkling behaviour, such as number and size of wrinkles, depend highly on muscle shape, size and direction, thickness and parameters of skin layers and soft-tissue structures, and constrained nodes. It is therefore likely that much more realistic models and animations could be produced by using models generated from more accurate surface meshes (e.g. from CT and MRI data), rather than manually created surface meshes, as well as more accurate materials and muscle parameters. Models could be created using medical data with our model creation process, although such data would first require processing and clean up.

9.4.3 System Design Flexibility

Our creation process can also be used independently of our overall animation process. For example, it could be used to create a model for use with a different FE solver, which might use a different formulation of the FE method, or support different time-integration schemes or material models

that are more relevant for the simulation to be performed (not necessarily a soft-tissue simulation). All model data that would be required for simulation using such an FE solver is computed, including geometrical data, the elements associated with each material, and directions for elements associated with transversely isotropic materials (e.g. muscle elements), as well as nodes associated with boundary conditions. However, this data would need to be output in a format that is relevant for the solver to be used, and such data may be interpreted differently depending on the solver. For example, different nodal constraints may be required for sliding nodes, as the novel constraints we use are specific to our simulation process. Also, most solvers only support a single material per element, whereas elements in our models may be associated with multiple material models; therefore, an average material for each element may need to be computed.

Rather than requiring NURBS volume approximations of muscles, more automated techniques could be used to generate muscle fibre directions. While the directions would be difficult to automatically compute from element geometry, as elements of our models aren't aligned with the muscles, and don't have C^1 continuity [RP07, MHSH10], they could potentially be computed directly from the muscle surfaces. The approach by Aina [AZ10, Ain11] could be extended to generate a number of fibres along upper and lower muscle surfaces, the directions of which could be interpolated to compute the direction at a sample, producing a volumetric fibre field. However, this technique requires construction of geodesics, which are difficult to define on polygon surfaces. While defining NURBS surface muscle approximations requires additional manual work, this technique gives the user a great deal of control over the variation in the fibre directions throughout the muscle volume.

9.5 Experimental and Future Work

Although our animation approach produces realistic animations of large- and fine-scale forehead behaviour, including wrinkles, in a physically based manner, there are various improvements that can be made to our approach and system implementation. Some experimental work has also been done towards the most significant improvements. Such experimental work is presented in this section. Significant improvements include more accurately capturing the thickness of the thin epidermal layer, since it is infeasible to model the very low thickness of this layer with hexahedral elements. Furthermore, outer elements that represent the outer surfaces of models could be smoothed such that they conform better to these surfaces, rather than using entirely voxel-based structures with sharp steps. Performance-wise, our simulation system could be extended to make use of multiple GPUs.

9.5.1 Simulating the Epidermis

From Table 9.1 and Figure 9.15, it can be seen that, with the soft-tissue models, the epidermal layer (stratum corneum) was assigned a higher stiffness, and appears thicker than in reality. Due to the low thickness of this layer in relation to element size, the dermis dominates the outer layer of elements. When the material properties of these skin layers are combined for these elements, this results in a stiffness much lower than that of the epidermis alone. The high stiffness is therefore necessary to produce a large enough difference between the stiffness of the two outer layers of elements, which is necessary to simulate wrinkles [FM08]. As a consequence, when using the same material models for the dermis and epidermis, the outer layer of elements acts like a thick epidermal layer, which is unavoidable without using different-shaped elements that can capture the thickness of the epidermal layer.

Better capturing the epidermal thickness would probably produce more realistic wrinkling behaviour. For example, Figure 9.7, introduced in Section 9.1.1, shows animation results of two soft-tissue-block models: one with cube-shaped hexahedral elements, and the other with thinner, cuboid-shaped elements respectively (the top layer of elements containing the epidermal layer have the same material properties in both models). While both produce small wrinkles that merge to form larger bulges when compressed further, the latter animation produces finer, more realistic

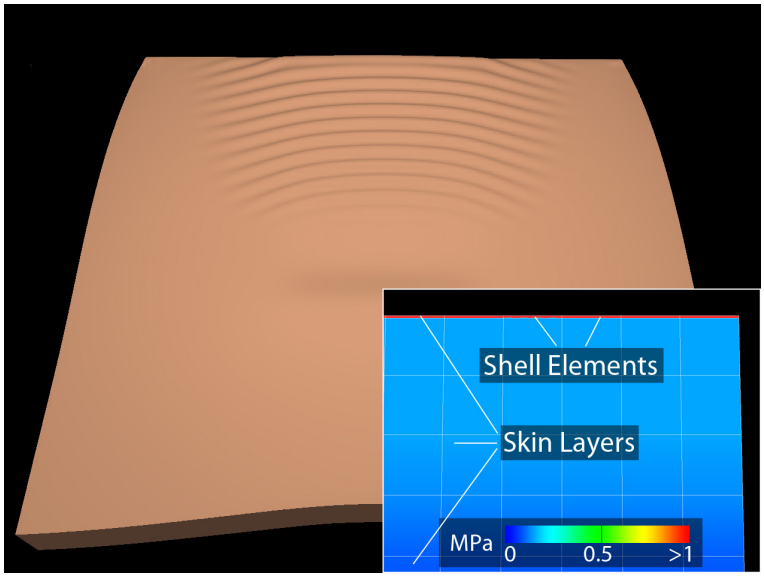

Figure 9.26: Animation of a flat single-muscle soft-tissue-block model, the epidermis of which is modelled using shell elements, under muscle contraction. The inset visualises the material stiffness, and shows the skin layers and shell elements.

wrinkles, and more detailed bulges under further compression. However, as solid elements (e.g. hexahedra) have optimal accuracy and stability when regularly shaped, producing inaccurate and unstable results when highly distorted, the thickness of the epidermal layer is far too thin to be feasibly captured using such elements.

To better model the thickness of the epidermal layer, functionality to simulate general 4-node quadrilateral shell elements using the simulation system has been experimented with. The theory behind such elements can be found in literature [BB80, Bat96]. Unlike the currently used solid elements, which have optimal accuracy and stability when regularly shaped, shell elements can have a thickness that is much lower than the other element dimensions. Such elements are therefore suited to modelling thin objects, like the epidermal skin layer. Shell elements have been successfully used in previous work to model epidermal layers [FM08], although such approaches focus on accurately simulating highly detailed models of small areas of skin and soft tissue for studying soft tissue behaviour. Current computer graphics approaches that efficiently produce realistic-looking animations of larger areas, such as the forehead, don't model detail like multiple skin layers [SNF05, MHSH10].

For the accuracy we desire, due to the low thickness of the epidermal layer, we simply attach the mid-surface nodes of the shell elements to the outer faces of the outer skin elements. The shell elements therefore share the same nodes as the solid elements, and do not require any additional nodal tying constraints. Figure 9.26 shows an example animation using a modified version of the flat single-muscle soft-tissue-block model from Figure 9.2, with which the epidermal layer is now modelled using shell elements. Compared to the animation of the original model in Figure 9.5, finer wrinkles are produced. Further experimentation would be required to see whether these merge to form larger bulges when compressed further, as in Figure 9.7.

To potentially improve simulation realism and accuracy further, shell elements could be used to create conforming simulation models that could be 'attached' to the current non-conforming simulation models. By converting the surface meshes to meshes containing quad rather than triangle primitives, these quads could be used as shell elements to create conforming models. A mapping would be needed to propagate the stresses and forces between elements and nodes of the non-conforming model, and those of the conforming shell element model. Using such an approach, gross movement could be efficiently simulated using the current non-conforming hexahedral sim-

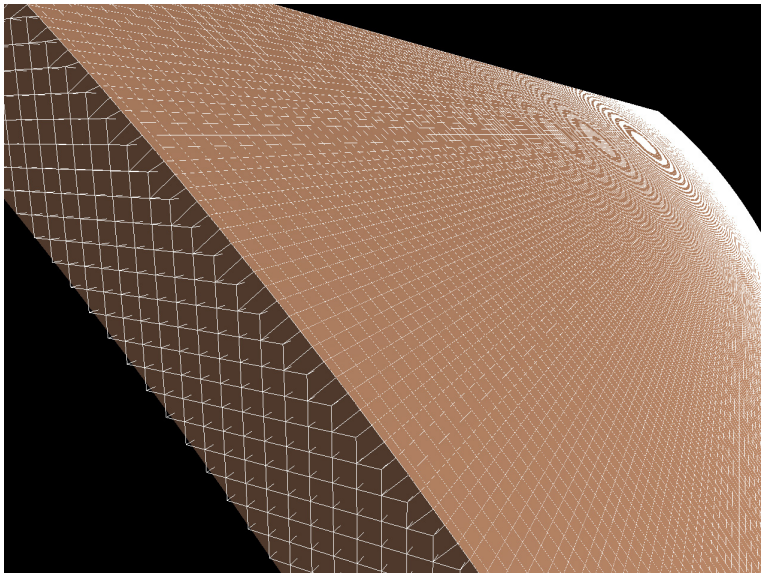

Figure 9.27: A curved soft-tissue-block model where the outermost voxel-shaped elements have been replaced by a new sheet of hexahedral elements, the outermost nodes of which lie on the outer skin surface.

ulation models, which could be propagated to the conforming shell element simulation models to hopefully simulate wrinkling. No known research has experimented with creating a mapping between two FE simulation models with non-conforming interfaces in this way.

9.5.2 Model Smoothing

A problem with non-conforming voxel models is that artefacts on the surface meshes and inaccurate stresses tend to be produced at locations where there is a sharp step in the models. Some techniques to improve the accuracy of such models by adapting them to conform more closely to the surface meshes have been experimented with. Only the adaptation of the outer elements that model the outer surfaces of a surface mesh has been considered so far. The inner voxel-shaped elements therefore still possess the advantages of non-conforming simulation models, whereas the outer elements ensure the model conforms more closely to the overall shape of the object being modelled.

A simple smoothing approach has been experimented with [AF07], which iteratively moves nodes of the outer elements based on neighbouring nodes. Using this approach, the intensity of artefacts was reduced (but not eliminated), although full simulations were unable to be computed since the more irregular element shapes resulted in decreased simulation stability and performance. Further work would be necessary to improve these issues. An alternative approach removes the outermost voxel-shaped elements of a model, and creates a sheet of new hexahedral elements, the outermost nodes of which lie on the outer surface of the surface mesh [SJ09], creating a model that, depending on the element resolution, can conform very closely to the outer surface (see Figure 9.27). However, this approach also produced models with some badly shaped elements that could become highly distorted during simulations, for example, having a high aspect ratio, or having two faces collapse and becoming coplanar. Such elements therefore had a large negative effect on the accuracy of the simulations, as well as stability and performance.

Further work would be necessary to determine whether such mesh adaptation techniques can be used to produce more accurate models and simulations without greatly affecting simulation stability and performance. For example, the two described approaches could be used together, creating new outermost elements that conform closely to the outer surface, and smoothing the adjoining elements. While this would result in models with fewer regularly shaped elements, the

elements that aren't regularly shaped should be less distorted. The material composition of the new and smoothed elements may also need to be altered since they will now overlap slightly different parts of the models (e.g. different proportions of the epidermal and dermal skin layers), which isn't currently considered.

9.5.3 Material Behaviour

While our soft-tissue animations use simple hyperelastic neo-Hooekan materials, such materials normally only fit material behaviour with reasonable accuracy under moderate strains. As high strains are produced during skin wrinkling, more accurate behaviour could be simulated by using, for example, Mooney-Rivlin or Ogden material models. Further skin behaviour could also be simulated, for example, by including anisotropic and viscoelastic material behaviour to model the preferred wrinkling direction parallel to Langer lines, and skin properties such as hysteresis, stress relaxation and creep. To model anisotropy in skin, like with muscles, each element overlapping an anisotropic skin layer would need to have one or multiple directions associated with that layer, for example, representing the directions of collagen fibres or Langer lines. The inclusion of biaxial natural tension with larger tension parallel to Langer lines would also help produce realistic anisotropic wrinkling effects. Such complex materials can be directly incorporated to the TLED FE method with little additional computational cost [TCC+09].

Modelling such complex material behaviour could also help to produce more realistic animations of different-aged soft tissue. As well as changes in material properties with age, such as the stiffness and thickness of skin layers, effects such as the decrease in natural tension, increase in anisotropy, and change in collagen fibre orientation could be modelled, each of which have an effect on wrinkling behaviour [FM10]. The increased appearance of wrinkle lines with age could also be modelled by considering skin plasticity [MTKL+02]. To produce a model of aged soft tissue with permanent ageing wrinkles, starting with an unaged model (such as that in Figure 9.15), a simulation of ageing involving numerous muscle contractions could be performed, with which, for example, the yield strength of the plasticity models could be decreased to simulate quick formation of the ageing wrinkles. From this, the initial deformation and material parameters for a series of different-aged faces could be determined.

9.5.4 Multi-GPU Simulation System

A program outline for a multi-GPU version of the simulation system has been produced. An implementation of this would be enable simulations to be scaled to run on any number of GPUs. Due to the complexity of the models being used, the potential performance increase gained by using such a system could be extremely beneficial when producing further simulation results, particularly if used on a HPC.

Using the multi-GPU system, as memory between GPUs isn't shared, parallelism across multiple GPUs would be less fine grained than that on a single GPU to reduce the amount of interaction and number of memory copies between each GPU. The system would simply split models into a number of smaller models that can be simulated independently on different GPUs, with a procedure that would apply boundary conditions after computation of each timestep to update nodes on the boundaries where the models have been split. Using such an approach, it would be relatively straight-forward to extend our current simulation system to simply run simulations of multiple FE models (representing parts of a larger model) on different GPUs, and apply boundary conditions between timesteps. Additionally, the models could be split non-uniformly according to the specification of each GPU, with the more powerful GPUs performing more of the computation. The same approach of splitting models into smaller independent models could also be used to parallelise the simulation system further, for example, using MPI to run simulations on a HPC with multiple GPUs attached to multiple nodes, increasing the number of GPUs that a simulation could be run on.

Chapter 10

Conclusion

This research has presented a fully physically based approach for efficiently producing realistic-looking animations of facial movement, focussing on the forehead. Modelling more physics-based behaviour than current computer graphics approaches, this includes animation of both gross- and fine-scale movement, such as expressive wrinkles, in a physically based manner. The animation approach consists of three main stages:

1. Creation of a surface mesh for an object
2. Creation of a suitable FE simulation model
3. Simulation and visualisation of the model over time

The surface meshes for the presented example models were manually created based on anatomy using 3D modelling tools, although a variety of other techniques could be used to create the surface meshes, such as by segmenting medical data. Once a suitable surface mesh has been created, this can be used with the novel automatic model creation process to create a simulation model that can be simulated using the advanced optimised simulation process.

The model creation process automatically creates animatable multi-layered non-conforming hexahedral (voxel-based) FE simulation models to which surface meshes are bound. Compared with tetrahedral elements, hexahedral elements suffer much less from volume locking when simulating incompressible material, such as soft tissue, and non-conforming models have model creation, performance and stability advantages over conforming models. Furthermore, the voxel-based models offer various simulation advantages compared to conforming hexahedral models, particularly regarding performance and stability. The model creation process involves voxelising a surface mesh, and computing model properties (such as skin layers, and element material and muscle properties) based on the proportions of overlap between the voxels and the volumes enclosed by the surfaces of the surface mesh. An efficient uniform sampling procedure is used to approximate such proportions of overlap. Muscle fibre fields are generated using the gradients of NURBS volumes (which are created automatically from NURBS surfaces) along the lengths of muscles. Boundary conditions are also computed, enabling nodes to be set as rigid (fixed) or sliding (bound by a surface) based on a collection of non-conforming surfaces. While this research focussed on generating soft-tissue models, the creation process can be used to generate any multi-layered FE model from any surface mesh.

The models generated using the model creation process can vary in complexity, from a complex face with many muscles, ligaments and tendons, to a simple generic object with a single surface and no muscles. By capturing detail such as skin layers, the models and simulation process are capable of simulating complex gross- and fine-scale behaviour, including wrinkling. To simulate the models, the TLED FE method is used with reduced-integration 8-node hexahedral elements, which are highly suited for efficiently simulating incompressible material like soft tissue. A stiffness-based hourglass control technique is used to counter hourglass effects that occur with reduced-integration

elements under large deformations. The simulation process includes an anatomical muscle contraction model that generates active and additional transversely isotropic passive stresses for each muscle overlapping an element. The processing of advanced boundary conditions to constrain nodes is also part of the simulation process. The boundary conditions enable the sliding effect between superficial and deep soft tissue to be modelled, which is often neglected with current physically based facial animation approaches [SNF05, BJTM08]. The GPU-based simulation and visualisation system has been implemented using CUDA, and optimised to exploit the memory and performance advantages offered when simulating voxel-based models on the GPU.

Example models and animations have demonstrated the ability of the animation approach to produce different animations of realistic large- and fine-scale soft-tissue behaviour, including wrinkles, on the face. Mass and time scaling is useful for such simulations to enable a much higher timestep to be used with little visual effect. A range of models and animations of varying complexity have been presented, demonstrating the suitability of the animation approach to a range of tasks. These include a range of flat soft-tissue-block models, the elements of which conform to the outer surfaces, such as a simple flat model consisting of uniform layers of soft tissue throughout the model, similar to those often used in science and engineering fields to study the behaviour of soft tissue. More complex curved soft-tissue-block models with forehead-like muscle structures, the elements of which don't conform to the outer surfaces, were also presented before applying the animation approach to a forehead model containing more complexly shaped surfaces and the galea aponeurotica. It was shown that, by simply changing the material parameters of a model, it is possible to achieve different effects, such as the animation of different-aged skin. While some visual artefacts are present due to the sharp steps in the voxel-based models, realistic-looking wrinkling patterns are produced when simulating each of the soft tissue models.

Qualitative comparisons to other soft-tissue and wrinkle animation approaches showed the advantages of using a detailed, fully physically based animation approach. The importance of using a layered soft-tissue structure to simulate soft tissue was demonstrated; when considering skin as a single layer, like with current FE facial animation approaches [SNF05, WHHM13], no wrinkles are produced. The wrinkling patterns produced by the simulations show similarities to those produced with highly accurate simulations for skin wrinkling studies [FM08]; for example, relatively sharp furrows and wide bulges are produced, and finer wrinkles under small loads group into larger bulges under higher loads. A quantitative evaluation approach has also been proposed to measure wrinkle statistics, including the average roughness and maximum range, on a curved surface with complex wrinkling patterns, like the forehead.

Despite increased complexity and computational requirements, various advantages of using a fully physically based animation approach were demonstrated when compared with a hybrid physically based facial animation approach that uses a geometric wrinkling technique to layer wrinkles onto the gross facial movement. There are many complexities when designing geometric equations that produce realistic wrinkles without artefacts, and with configuring such equations to produce realistic-looking wrinkles. A fully physically based approach enables realistic-looking wrinkles to be easily and automatically produced as a result of the physics-based simulations.

A model and animation example of a generic multi-material soft body has also been presented, showing the flexibility of the presented approach to animate different human or non-human soft-tissue structures, producing animations of both large and fine details using an accurate physics-based technique. Since the animations produced depend highly on the model, such as muscle shape and size, and soft-tissue thickness and parameters, models created from more accurate surface meshes would likely enable more realistic animations to be produced. Such surface meshes could be created, for example, using medical data. Provided these surface meshes contain hole-free surfaces, they can be directly used with the presented model creation process.

Various improvements to the animation approach and implementation have been discussed, and some experimental work has been completed towards the most significant improvements. Such significant improvements include using shell elements to more accurately model the thin epidermal layer. Shell elements can have a much lower thickness than hexahedral elements, and are therefore much more suited to modelling the epidermis. It was shown that using thinner elements to model the epidermis produced more detailed wrinkling patterns on soft-tissue-block

models. Furthermore, the outer elements that represent the outer surfaces of models could be smoothed such that they conform better to these surfaces, and multi-resolution models could be used, for example, to enable a higher resolution to be employed along the outer skin surface where wrinkles are produced. These improvements would likely reduce the intensity of artefacts, and improve the accuracy of the stresses produced at locations where there are sharp steps in the voxel-based models, while the inner voxel-shaped elements still possess the advantages of non-conforming models.

To improve simulation performance, models could be split and simulated in parallel on multiple GPUs, or even on a distributed system. Other potential improvements and future work include using more accurate material models to better simulate the response of soft-tissue, potentially integrating plasticity to simulate ageing wrinkles. Future work also includes extending the animation approach to apply it to the full face, rather than just the forehead, which would require consideration of the complex movement and collisions around the mouth and lips, and the modelling and movement of the jaw.

Bibliography

[3D 10] 3D World Magazine. Bridging the uncanny valley in CG. http://www.3dworldmag.com/2010/12/30/bridging-the-uncanny-valley/, 2010.

[3D 11] 3D World Magazine. LA Noire: The future of facial capture? http://www.3dworldmag.com/2011/05/20/la-noire-the-future-of-facial-capture/, 2011.

[AF07] P. Arbenz and C. Flaig. On Smoothing Surfaces in Voxel Based Finite Element Analysis of Trabecular Bone. In *Proc. LSSC*, pages 69–77, 2007.

[AGA12] Q. Avril, V. Gouranton, and B. Arnaldi. Fast Collision Culling in Large-Scale Environments Using GPU Mapping Function. In *Proc. EGPGV*, pages 71–80, 2012.

[AHTN01] Yoshimitsu Aoki, Shuji Hashimoto, Masahiko Terajima, and Akihiko Nakasima. Simulation of postoperative 3D facial morphology using a physics-based head model. *Vis. Comput.*, 17(2):121–131, 2001.

[Ain09] O. O. Aina. Generating anatomical substructures for physically-based facial animation. part 1: A methodology for skull fitting. *Vis. Comput.*, 25:617–625, 2009.

[Ain11] O. O. Aina. *Generating Anatomical Substructures for Physically-Based Facial Animation.* PhD thesis, Bournemouth University, UK, 2011.

[Asc08] U. M. Ascher. *Numerical methods for evolutionary differential equations.* SIAM, 2008.

[AT01] A. Aubel and D. Thalmann. Efficient Muscle Shape Deformation. In *Deformable Avatars*, pages 132–142. Springer US, 2001.

[Aut11] Autodesk Inc. *Maya Muscle*, 2011.

[AWM06] N. Z. Azmi, R. Wirza, and R. Mahmod. Modeling Expressive Wrinkle On Human Face. In *Proc. GRAPHITE*, pages 425–427, 2006.

[AZ10] O. O. Aina and J. J. Zhang. Automatic Muscle Generation for Physically-Based Facial Animation. In *SIGGRAPH Posters*, pages 105:1–105:1, 2010.

[BAG00] J. E. Bischoff, E. M. Arruda, and K. Grosh. Finite element modeling of human skin using an isotropic, nonlinear elastic constitutive model. *J. Biomech.*, 33(6):645–652, 2000.

[BAG04] J. E. Bischoff, E. M. Arruda, and K. Grosh. A rheological network model for the continuum anisotropic and viscoelastic behavior of soft tissue. *Biomech. Model. Mechanobiol.*, 3(1):56–65, 2004.

[Bar05] R. Baran. *Textbook of Cosmetic Dermatology.* Taylor & Francis, third edition, 2005.

[Bat86] K.-J. Bathe. Finite Element Procedures for Solids and Structures. http://ocw.mit.edu/resources/res-2-002-finite-element-procedures-for-solids-and-structures-spring-2010/index.htm, 1986.

[Bat96] K.-J. Bathe. *Finite Element Procedures*. Prentice Hall, 2nd edition, 1996.

[BB80] K. J. Bathe and S. Bolourchi. A Geometric and Material Nonlinear Plate and Shell Element. *Comput. Struct.*, 11(1–2):23–48, 1980.

[BB93] T. Belytschko and L. P. Bindeman. Assumed strain stabilization of the eight node hexahedral element. *Comput. Methods Appl. Mech. Eng.*, 105(2):225–260, 1993.

[BB98] J. Bonet and A. J. Burton. A simple averaged nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. *Commun. Numer. Methods Eng.*, 14(5):437–449, 1998.

[BB07] A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In *Proc. GRAPHITE*, pages 281–288, 2007.

[BBA+07] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross. Multi-Scale Capture of Facial Geometry and Motion. *ACM Trans. Graph.*, 26(3):33–1–33–10, 2007.

[BG05] M. M. I. Baig and T. Grätsch. Recommendations for practical use of numerical methods in linear and nonlinear dynamics. Technical report, Waltham, EUA: Simpson Gumpertz and Heger, Inc, 2005.

[BHN03] T. D. Bui, D. Heylen, and A. Nijholt. Improvements on a Simple Muscle-Based 3D Face for Realistic Facial Expressions. In *Proc. CASA*, pages 33–40, 2003.

[BHP06] R. Bade, J. Haase, and B. Preim. Comparison of Fundamental Mesh Smoothing Algorithms for Medical Surface Models. In *Proc. SimVis*, pages 289–304, 2006.

[BHS03] G. Bianchi, M. Harders, and G. Székely. Mesh Topology Identification for Mass-Spring Models. In *Proc. MICCAI*, pages 50–58, 2003.

[Bis06] J. E. Bischoff. Reduced Parameter Formulation for Incorporating Fiber Level Viscoelasticity into Tissue Level Biomechanical Models. *Ann. Biomed. Eng.*, 34(7):1164–1172, 2006.

[BJM11] G. G. Barbarino, M. Jabareen, and E. Mazza. Experimental and numerical study on the mechanical behavior of the superficial layers of the face. *Skin Res. Technol.*, 17(4):434–444, 2011.

[BJT+09] G. Barbarino, M. Jabareen, J. Trzewik, A. Nkengne, G. Stamatas, and E. Mazza. Development and validation of a three-dimensional finite element model of the face. *J. Biomech. Eng.*, 131(4):041006, 2009.

[BJTM08] G. Barbarino, M. Jabareen, J. Trzewik, and E. Mazza. Physically Based Finite Element Model of the Face. In *Proc. ISBMS*, pages 1–10, 2008.

[BKN02] Y. Bando, T. Kuratate, and T. Nishita. A Simple Method for Modeling Wrinkles on Human Skin. In *Proc. PG*, pages 166–175, 2002.

[BKTK00] L. Boissieux, G. Kiss, N. M. Thalmann, and P. Kalra. Simulation of Skin Aging and Wrinkles with Cosmetics Insight. In *Proc. Computer Animation and Simulation*, pages 15–27, 2000.

[BL11] B. Bickel and M. Lang. From Sparse Mocap to Highly Detailed Facial Animation. In *GPU Computing Gems Emerald Edition*, pages 413–426. Elsevier, 2011.

[Bli98] A. Blitzer. *Office-Based Surgery in Otolaryngology.* Thieme, 1998.

[BM06] S. K. Boyd and R. Müller. Smooth surface meshing for automated finite element model generation from 3D image data. *J. Biomech.*, 39(7):1287–1295, 2006.

[BMH01] J. Bonet, H. Marriott, and O. Hassan. Stability and comparison of different linear tetrahedral formulations for nearly incompressible explicit dynamic applications. *Int. J. Numer. Methods Eng.*, 50(1):119–133, 2001.

[Bor05] G. Borshukov. Making of The Superpunch. In *SIGGRAPH 2005 Courses*, 2005.

[BPMC95] S. E. Benzley, E. Perry, K. Merkley, and B. Clark. A Comparison of All Hexagonal and All Tetrahedral Finite Element Meshes for Elastic and Elasto-plastic Analysis. In *Proc. IMR*, pages 179–191, 1995.

[Bra06] H. Brannon. Glogau Classification of Photoaging. http://dermatology.about.com/od/wrinkles/a/glogau.htm, 2006.

[BRM+14] T. C. Baudouin, J.-F. Remacle, E. Marchandise, F. Henrotte, and C. Geuzaine. A frontal approach to hex-dominant mesh generation. *Adv. Model. Simul. Eng. Sci.*, 1:8:1–8:30, 2014.

[BSB+01] J. Brown, S. Sorkin, C. Bruyns, J.-C. Latombe, K. Montgomery, and M. Stephanides. Real-Time Simulation of Deformable Objects: Tools and Application. In *Proc. Computer Animation*, pages 228–236, 2001.

[BSSS10] E. Baker, M. Schuenke, E. Schulte, and U. Schumacher. *Head and Neck Anatomy for Dental Medicine.* Thieme, 2010.

[BTPB07] X. Bourdin, X. Trosseille, P. Petit, and P. Beillas. Comparison of tetrahedral and hexahedral meshes for organ finite element modeling: an application to kidney impact. Technical Report 07-0424, INRETS - LBMC (UMR_T9406 INRETS/UCBL1), 2007.

[BWL+10] L. Beldie, B. Walker, Y. Lu, S. Richmond, and J. Middleton. Finite element modelling of maxillofacial surgery and facial expressions – a preliminary study. *Int. J. Med. Robot. Comput. Assist. Surg.*, 6(4):422–430, 2010.

[Can92] S. A. Canann. Plastering - A new approach to automated, 3-D hexahedral mesh generation. Technical report, American Institute of Aeronautics and Astronics, 1992.

[CDA00] S. Cotin, H. Delingette, and N. Ayache. A Hybrid Elastic Model allowing Real-Time Cutting, Deformations and Force-Feedback for Surgery Training and Simulation. *Vis. Comput.*, 16(8):437–452, 2000.

[Che92] D. T. Chen. *Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method.* PhD thesis, Massachusetts Institute of Technology, USA, 1992.

[CLK01] B. Choe, H. Lee, and H.-S. Ko. Performance-Driven Muscle-Based Facial Animation. *J. Vis. Comput. Animat.*, 12(2):67–79, 2001.

[CLP03] Matthieu Chabanas, Vincent Luboz, and Yohan Payan. Patient specific finite element model of the face soft tissues for computer-assisted maxillofacial surgery. *Med. Image Anal.*, 7(2):131–151, 2003.

[CO82] W. A. Cook and W. R. Oakes. Mapping methods for generating three dimensional meshes. *Eng. Comput.*, 1:67–72, 1982.

[Coo98] Lee Cooper. *Physically Based Modelling of Human Limbs*. PhD thesis, The University of Sheffield, UK, 1998.

[Cou05] A. D. Coull. *A Physically-Based Muscle and Skin Model for Facial Animation*. PhD thesis, The University of Glasgow, UK, 2005.

[CP06] C. Chen and E. C. Prakash. Physically based facial expression synthesizer with performance analysis and GPU-aided simulation. In *Proc. CyberGames*, pages 171–176, 2006.

[CPL00] B. Couteau, Y. Payan, and S. Lavallée. The mesh-matching algorithm: an automatic 3D mesh generator for finite element structures. *J. Biomech.*, 33(8):1005–1009, 2000.

[CSI04] W.-Y. Choi, I.-H. Son, and Y.-T. Im. Locally refined tetrahedral mesh generation based on advancing front technique with optimization and smoothing scheme. *Commun. Numer. Methods Eng.*, 20(9):681–688, 2004.

[CTA+08] O. Comas, Z. A. Taylor, J. Allard, S. Ourselin, S. Cotin, and J. Passenger. Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA. In *Proc. ISBMS*, pages 28–39, 2008.

[Cul11] Culture.com. Shrek: Production Information. http://www.culture.com/articles/463/shrek-production-information.phtml, 2011.

[CZ92] D. T. Chen and D. Zeltzer. Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method. In *Proc. SIGGRAPH*, pages 89–98, 1992.

[Dat14] Database Center for Life Science. BodyParts3D/Anatomography. http://lifesciencedb.jp/bp3d/, 2014.

[DCKY02] F. Dong, G. J. Clapworthy, M. A. Krokos, and J. Yao. An Anatomy-Based Approach to Human Muscle Modeling and Deformation. *IEEE Trans. Vis. Comput. Graph.*, 8(2):154–170, 2002.

[DDP+13] T. T. Dao, S. Dakpé, P. Pouletaut, B. Devauchelle, and M. C. Ho Ba Tho. Facial mimics simulation using MRI and finite element analysis. In *Proc. EMBC*, pages 4585–4588, 2013.

[Der10] DermNet NZ. Facial lines and wrinkles. http://www.dermnetnz.org/site-age-specific/wrinkles.html, 2010.

[DGW11] C. Dick, J. Georgii, and R. Westermann. A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. *Simul. Model. Pract. Theory*, 19(2):801–816, 2011.

[DMB09] L. Dutreve, A. Meyer, and S. Bouakaz. Real-Time Dynamic Wrinkles of Face for Animated Skinned Mesh. In *Proc. ISVC*, pages 25–34, 2009.

[DN08] Z. Deng and J. Noh. Computer Facial Animation: A Survey. In *Data-Driven 3D Facial Animation*, pages 1–28. Springer London, 2008.

[DSVO11] C. Demirdover, B. Sahin, H. Vayvada, and H. Y. Oztan. The Versatile Use of Temporoparietal Fascial Flap. *Int. J. Med. Sci.*, 8(5):362–368, 2011.

[Duy06] A. Duysak. Triangle Propagation for Mass-Spring Chain Algorithm. In *Proc. ISCIS*, pages 306–315, 2006.

[DZ04] A. Duysak and Jian J. Zhang. Fast Simulation of Deformable Objects. In *Proc. IV*, pages 422–427, 2004.

[DZ05] A. Duysak and Jian J. Zhang. Fast Simulation of Facial Tissue Deformations Using Mass-Spring Chain Algorithm. In *Proc. TPCG*, pages 139–145, 2005.

[DZI03] A. Duysak, J. J. Zhang, and V. Ilankovan. Efficient modelling and simulation of soft tissue deformation using mass-spring systems. In *Proc. CARS*, pages 337–342, 2003.

[EF78] P. Ekman and W. Friesen. Facial action coding system: A technique for the measurement of facial movement. Technical report, Consulting Psychologists Press, 1978.

[EFH02] P. Ekman, W. Friesen, and J. Hager. *Facial Action Coding System: The Manual on CD ROM*. A Human Face, 2002.

[EM01] J. D. Edge and S. Maddock. Expressive visual speech using geometric muscle functions. In *Proc. EGUK*, pages 11–18, 2001.

[Enc11a] Encyclopdia Britannica, Inc. human skull. http://www.britannica.com/EBchecked/media/149110/Lateral-and-anterior-views-of-a-human-skull, 2011.

[Enc11b] Encyclopdia Britannica, Inc. muscle system, human: muscles of facial expression. http://www.britannica.com/EBchecked/media/119201/Muscles-of-facial-expression, 2011.

[Far10] M. A. Farage. *Textbook of Aging Skin*. Springer, 2010.

[FB81] D. P. Flanagan and T. Belytschko. A uniform strain hexahedron and quadrilateral with orthogonal hourglass control. *Int. J. Numer. Methods Eng.*, 17(5):679–706, 1981.

[Fel13] C. Felippa. Hexahedron Elements. Lecture Notes for Course ASEN 6367, The University of Colorado, 2013.

[FG08] P. J. Frey and P.-L. George. *Mesh Generation: Application to Finite Elements*. John Wiley and Sons, second edition, 2008.

[Fle00] P. Fleischmann. *Mesh Generation for Technology CAD in Three Dimensions*. PhD thesis, Vienna University of Technology, Austria, 2000.

[Fly07] Cormac Oliver Flynn. *The Design and Validation of a Multi-Layer Model of Human Skin*. PhD thesis, Institute of Technology, Sligo, 2007.

[FM99] N. T. Folwell and S. A. Mitchell. Reliable Whisker Weaving via Curve Contraction. *Eng. Comput.*, 15(3):292–302, 1999.

[FM08] C. Flynn and B. A. O. McCormack. Finite element modelling of forearm skin wrinkling. *Skin Res. Technol.*, 14(3):261–269, 2008.

[FM10] C. Flynn and B. A. O. McCormack. Simulating the wrinkling and aging of skin with a multi-layer finite element model. *J. Biomech.*, 43(3):442–448, 2010.

[Fra05] M. Fratarcangeli. Physically Based Synthesis of Animatable Face Models. In *Proc. VRIPHYS*, pages 32–39, 2005.

[Fra09] M. Fratarcangeli. *A Computational Musco-Skeletal Model for Animating Virtual Faces*. PhD thesis, Sapienza University of Rome, Italy, 2009.

[Fra12] M. Fratarcangeli. Position-based facial animation synthesis. *Comput. Animat. Virtual Worlds*, 23(3-4):457–466, 2012.

[Fra13] M. Fratarcangeli. *Computational Models for Animating 3D Virtual Faces*. PhD thesis, Linköping University, Sweden, 2013.

[Gee09] M. Geerligs. *Skin layer mechanics*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2009.

[Gha08] B. Ghali. *Algorithms for Nonlinear Finite Element-based Modeling of Soft-tissue Deformation and Cutting*. PhD thesis, McMaster University, Canada, 2008.

[Gla02] E. Gladilin. *Biomechanical Modeling of Soft Tissue and Facial Expressions for Craniofacial Surgery Planning*. PhD thesis, Free University of Berlin, Germany, 2002.

[GM97] S. Gibson and B. Mirtich. A Survey of Deformable Modeling in Computer Graphics. Technical report, MERL, 1997.

[Gol92] E. Goldfinger. *Human Anatomy for Artists: The Elements of Form*. Oxford University Press, 1992.

[GSZ11] J. Gregson, A. Sheffer, and E. Zhang. All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Comput. Graph. Forum*, 30(5):1407–1416, 2011.

[GW05] J. Georgii and R. Westermann. Mass-Spring Systems on the GPU. *Simul. Model. Pract. Theory*, 13(8):693–702, 2005.

[GW06] J. Georgii and R. Westermann. A Multigrid Framework for Real-Time Simulation of Deformable Volumes. *Comput. & Graph.*, 30(3):408–415, 2006.

[GZDH04] E. Gladilin, S. Zachow, P. Deuflhard, and H.-C. Hege. Anatomy- and physics-based facial animation for craniofacial surgery simulations. *Med. Biol. Eng. Comput.*, 42(2):167–170, 2004.

[Hat04] J. Hatzis. The wrinkle and its measurement: A skin surface Profilometric method. *Micron*, 35(3):201–219, 2004.

[Hen05] F. M. Hendriks. *Mechanical behaviour of human epidermal and dermal layers in vivo*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2005.

[HGS+07] C. J. Hughes, R. Grzeszczuk, E. Sifakis, D. Kim, S. Kumar, A. P. Selle, J. Chhugani, M. Holliman, and Y.-K. Chen. Physical Simulation for Animation and Visual Effects: Parallelization and Characterization for Chip Multiprocessors. In *Proc. ISCA*, pages 220–231, 2007.

[HHR+03] M. Harders, R. Hutter, A. Rutz, P. Niederer, and G. Székely. Comparing a simplified FEM approach with the mass-spring model for surgery simulation. *Stud. Health Technol. Inform.*, 94:103–109, 2003.

[Hil38] A. V. Hill. The Heat of Shortening and the Dynamic Constants of Muscle. *Proc. R. Soc. Lond. B*, 126(843):136–195, 1938.

[HKA+01] J. Haber, K. Kähler, I. Albrecht, H. Yamauchi, and H.-P. Seidel. Face to Face: From Real Humans to Realistic Facial Animation. In *Proc. IK*, pages 73–82, 2001.

[HMSH09] A. Hung, K. Mithraratne, M. Sagar, and P. Hunter. Multilayer Soft Tissue Continuum Model: Towards Realistic Simulation of Facial Expressions. In *Proc. WASET*, pages 134–138, 2009.

[Hun95] P. J. Hunter. *Myocardial constitutive laws for continuum models of the heart*, pages 303–318. Plenum Press, New York, 1995.

[HWHM11] Alice Pui-Lam Hung, Tim Wu, Peter Hunter, and Kumar Mithraratne. Simulating Facial Expressions using Anatomically Accurate Biomechanical Model. In *SIGGRAPH Asia 2011 Posters*, pages 29:1–29:1, Hong Kong, China, 2011. ACM.

[HWHM14] A. P. L. Hung, T. Wu, P. Hunter, and K. Mithraratne. A framework for generating anatomically detailed subject-specific human facial models for biomechanical simulations. *Vis. Comput.*, 2014.

[HWJM10] A. Horton, A. Wittek, G. R. Joldes, and K. Miller. A meshless Total Lagrangian explicit dynamics algorithm for surgical simulation. *Int. J. Numer. Methods Biomed. Eng.*, 26(8):977–998, 2010.

[Isl11] M. S. Islam. Improving the quality of hexahedral mesh generated by automatic mesh generators. *J. Nav. Archit. Mar. Eng.*, 8(2):121–128, 2011.

[ISS09] Y. Ito, A. M. Shih, and B. K. Soni. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *Int. J. Numer. Methods Eng.*, 77(13):1809–1833, 2009.

[ITF06] G. Irving, J. Teran, and R. Fedkiw. Tetrahedral and Hexahedral Invertible Finite Elements. *G. Models*, 68(2):66–89, 2006.

[IYYT93] T. Ishii, T. Yasuda, S. Yokoi, and J. Toriwaki. A Generation Model for Human Skin Texture. In *Proc. CGI*, pages 139–150, 1993.

[JMB00] T. Johansson, P. Meier, and R. Blickhan. A Finite-Element Model for the Mechanical Analysis of Skeletal Muscles. *J. Theor. Biol.*, 206(1):131–149, 2000.

[Joe12] B. Joe. Geompack++ Meshing Operations. Technical Report ZCS2012-01, Zhou Computing Services Inc., 2012.

[JWM08] G. R. Joldes, A. Wittek, and K. Miller. An efficient hourglass control implementation for the uniform strain hexahedron using the Total Lagrangian formulation. *Commun. Numer. Methods Eng.*, 24(11):1315–1323, 2008.

[JWM09] G. R. Joldes, A. Wittek, and K. Miller. Non-locking tetrahedral finite element for surgical simulation. *Commun. Numer. Methods Eng.*, 25(7):827–836, 2009.

[Käh03] K. Kähler. *A Head Model with Anatomical Structure for Facial Modeling and Animation*. PhD thesis, Saarland University, Germany, 2003.

[KCO+03] A. E. Kerdok, S. M. Cotin, M. P. Ottensmeyer, A. Galea, R. D. Howe, and S. L. Dawson. Truth Cube: Establishing Physical Standards for Real Time Soft Tissue Simulation. *Med. Image Anal.*, 7(3):283–291, 2003.

[Ken06] J. Kennard. Men and Wrinkles. http://menshealth.about.com/od/seniorhealth/a/Men_Wrinkles.htm, 2006.

[KGC+96] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. I. H. Parish. Simulating Facial Surgery Using Finite Element Models. In *Proc. SIGGRAPH*, pages 421–428, 1996.

[KGKG98] E. Keeve, S. Girod, R. Kikinis, and B. Girod. Deformable Modelling of Facial Tissue for Craniofacial Surgery Simulation. *Comput. Aided Surg.*, 3(5):228–238, 1998.

[KHS01] K. Kähler, J. Haber, and H.-P. Seidel. Geometry-based Muscle Modeling for Facial Animation. In *Proc. GI*, pages 37–46, 2001.

[KHS03] K. Kähler, J. Haber, and H.-P. Seidel. Reanimating the Dead: Reconstruction of Expressive Faces from Skull Data. *ACM Trans. Graph.*, 22(3):554–561, 2003.

[KHYS02] K. Kähler, J. Haber, H. Yamauchi, and H.-P. Seidel. Head shop: Generating animated head models with anatomical structure. In *Proc. SCA*, pages 55–63, 2002.

[KJW+10] H. Kim, P. Jürgens, S. Weber, L.-P. Nolte, and M. Reyes. A new soft-tissue simulation strategy for cranio-maxillofacial surgery using facial muscle template model. *Prog. Biophys. Mol. Biol.*, 103(2):284–291, 2010.

[KMBG09] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graph. Models*, 71(4):157–167, 2009.

[KMMTT91] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann. SMILE: A Multilayered Facial Animation System. In *Proc. IFIP WG 5.10*, pages 189–198, 1991.

[Kni01] D. M. Knize. *The Forehead and Temporal Fossa: Anatomy and Technique*. Lippincott Williams & Wilkins, 2001.

[Knu98] P. Knupp. Next-generation sweep tool: a method for generating all-hex meshes on two-and-one-half dimensional geometries. In *Proc. IMR*, pages 505–513, 1998.

[KPB08] A. V. Kumar, S. Padmanabhan, and R. Burla. Implicit boundary method for finite element analysis using non-conforming mesh or grid. *Int. J. Numer. Methods Eng.*, 74(9):1421–1447, 2008.

[KRG+02] R. M. Koch, S. H. M. Roth, M. H. Gross, A. P. Zimmermann, and H. F. Sailer. A Framework for Facial Surgery Simulation. In *Proc. SCCG*, pages 33–42, 2002.

[KSY08a] O. Kuwazuru, J. Saothong, and N. Yoshikawa. Evaluation of Aging Effects on Skin Wrinkle by Finite Element Method. *J. Biomech. Sci. Eng.*, 3(3):368–379, 2008.

[KSY08b] O. Kuwazuru, J. Saothong, and N. Yoshikawa. Mechanical approach to aging and wrinkling of human facial skin based on the multistage buckling theory. *Med. Eng. Phys.*, 30(4):516–522, 2008.

[Lam09] B. P. Lamichhane. Mortar Finite Elements for Coupling Compressible and Nearly Incompressible Materials in Elasticity. *Int. J. Numer. Anal. Model.*, 6(2):177–192, 2009.

[LBPH07] C. Lobos, M. Bucki, Y. Payan, and N. Hitschfeld. Techniques on mesh generation for the brain shift simulation. In *Proc. CLAIB*, pages 642–645, 2007.

[LBW00] M. Lai, S. Benzley, and D. White. Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. *Int. J. Numer. Methods Eng.*, 49(1–2):261–275, 2000.

[LC04] C. Larboulette and M.-P. Cani. Real-Time Dynamic Wrinkles. In *Proc. CGI*, pages 522–525, 2004.

[LCC+12] H. D. Lou, S. Chen, G. Chen, T. M. Xu, and Q. G. Rong. Patient-specific modeling of facial soft tissue based on radial basis functions transformations of a standard three-dimensional finite element model. *Chin. Med. J.*, 125(22):4066–4071, 2012.

[LCSP05] V. Luboz, M. Chabanas, P. Swider, and Y. Payan. Orbital and maxillofacial computer aided surgery: patient-specific finite element models to predict surgical outcomes. *Comput. Methods Biomech. Biomed. Eng.*, 8(4):259–265, 2005.

[LEG10] C. A. D. Leon, S. Eliuk, and H. T. Gomez. Simulating Soft Tissues using a GPU approach of the Mass-Spring Model. In *Proc. VR*, pages 261–262, 2010.

[LGK11] Rudy J. Lapeer, Paul D. Gasson, and Vasudev Karri. A Hyperelastic Finite-Element Model of Human Skin for Interactive Real-Time Surgical Simulation. *IEEE Trans. Biomed. Eng.*, 58(4):1013–1022, 2011.

[LGK+12] D. Lee, M. Glueck, A. Khan, E. Fiume, and K. Jackson. Modeling and Simulation of Skeletal Muscle for Computer Graphics: A Survey. *Found. Trends. Comput. Graph. Vis.*, 7(4):229–276, 2012.

[LJWD08] Y. Liu, S. Jiao, W. Wu, and S. De. GPU Accelerated Fast FEM Deformation Simulation. In *Proc. APCCAS*, pages 606–609, 2008.

[LKH04] Y. Liu, A. E. Kerdok, and R. D. Howe. A Nonlinear Finite Element Model of Soft Tissue Indentation. In *Proc. ISMS*, pages 67–76, 2004.

[LLT11] M.-F. Li, S.-H. Liao, and R.-F. Tong. Facial hexahedral mesh transferring by volumetric mapping based on harmonic fields. *Comput. Graph.*, 35(1):92–98, 2011.

[LLZ12] L. Li, Y. Liu, and H. Zhang. A Survey of Computer Facial Animation Techniques. In *Proc. ICCSEE*, pages 434–438, 2012.

[Lo12] S. H. Lo. Automatic merging of hexahedral meshes. *Finite Elem. Anal. Des.*, 55:7–22, 2012.

[LOU85] W. K. Liu, J. S.-J. Ong, and R. A. Uras. Finite element stabilization matrices-a unification approach. *Comput. Methods Appl. Mech. Eng.*, 53(1):13–46, 1985.

[LS86] W. F. Larrabee Jr. and D. Sutton. A finite element model of skin deformation. II. An experimental model of skin deformation. *The Laryngoscope*, 96(4):406–412, 1986.

[LST09] S.-H. Lee, E. Sifakis, and D. Terzopoulos. Comprehensive Biomechanical Modeling and Simulation of the Upper Body. *ACM Trans. Graph.*, 28(4):99:1–99:17, 2009.

[LT06] S.-H. Lee and D. Terzopoulos. Heads Up! Biomechanical Modeling and Neuromuscular Control of the Neck. *ACM Trans. Graph.*, 25(3):1188–1198, 2006.

[LTW93] Y. Lee, D. Terzopoulos, and K. Waters. Constructing Physics-Based Facial Models of Individuals. In *Proc. GI*, pages 1–8, 1993.

[LTW95] Y. Lee, D. Terzopoulos, and K. Waters. Realistic Modeling for Facial Animation. In *Proc. SIGGRAPH*, pages 55–62, 1995.

[LW08] F. Ledoux and J.-C. Weill. An Extension of the Reliable Whisker Weaving Algorithm. In *Proc. IMR*, pages 215–232, 2008.

[LYKL07] M. Li, B. Yin, D. Kong, and X. Luo. Modeling Expressive Wrinkles of Face For Animation. In *Proc. ICIG*, pages 874–879, 2007.

[Mar09] L. Maréchal. Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In *Proc. IMR*, pages 65–84, 2009.

[MBTF03] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra. In *Proc. IMR12*, pages 103–114, 2003.

[MC10] M. Müller and N. Chentanez. Wrinkle Meshes. In *Proc. SCA*, pages 85–92, 2010.

[MES05] S. Maddock, J. Edge, and M. Sanchez. Movement Realism in Computer Facial Animation. In *Proc. HCI Workshop on Human-animated Characters Interaction*, 2005.

[MHHR06] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position Based Dynamics. In *Proc. VRIPHYS 2006*, pages 71–80, Madrid, Spain, 2006.

[MHL+09] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De. Using the PhysX engine for physics-based virtual surgery with force feedback. *Int. J. Med. Robot. Comput. Assist. Surg.*, 5(3):341–353, 2009.

[MHS05] J. Mosegaard, P. Herborg, and T. S. Sørensen. A GPU Accelerated Spring Mass System for Surgical Simulation. *Stud. Health Technol. Inform.*, 111:342–348, 2005.

[MHSH10] K. Mithraratne, A. Hung, M. Sagar, and P. J. Hunter. An Efficient Heterogeneous Continuum Model to Simulate Active Contraction of Facial Soft Tissue Structures. In *Proc. WCB*, pages 1024–1027, 2010.

[Mit98] S. A. Mitchell. The All-Hex Geode-Template for Conforming a Diced Tetrahedral Mesh to any Diced Hexahedral Mesh. In *Proc. IMR*, pages 295–305, 1998.

[MJLW07] K. Miller, G. Joldes, D. Lance, and A. Wittek. Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Commun. Numer. Methods Eng.*, 23(2):121–134, 2007.

[MLC01] D. Ma, F. Lin, and C. K. Chua. Rapid Prototyping Applications in Medicine. Part 1: NURBS-Based Volume Modelling. *Int. J. Adv. Manuf. Technol.*, 18(2):103–117, 2001.

[Mor70] M. Mori. The Uncanny Valley. *Energy*, 7(4):33–35, 1970.

[MP95] J. Malmivuo and R. Plonsey. *Bioelectromagnetism - Principles and Applications of Bioelectric and Biomagnetic Fields*. Oxford University Press, 1995.

[MPRB05] E. Mazza, O. Papes, M. B. Rubin, and S. R. Bodner. Nonlinear elastic-viscoplastic constitutive equations for aging facial tissues. *Biomech. Model. Mechanobiol.*, 4(2-3):178–189, 2005.

[MS13] R. Mafi and S. Sirouspour. GPU-based acceleration of computations in nonlinear finite element deformation analysis. *Int. J. Numer. Methods Biomed. Eng.*, Early View, 2013.

[MSN+06] W. Mollemans, F. Schutyser, N. Nadjmi, F. Maes, and P. Suetens. Parameter Optimisation of a Linear Tetrahedral Mass Tensor Model for a Maxillofacial Soft Tissue Simulator. In *Proc. ISBMS*, pages 159–168, 2006.

[MSN+07] W. Mollemans, F. Schutyser, N. Nadjmi, F. Maes, and P. Suetens. Predicting soft tissue deformations for a maxillofacial surgery planning system: From computational strategies to a complete clinical validation. *Med. Image Anal.*, 11(3):282–301, 2007.

[MSNS05] W. Mollemans, F. Schutyser, N. Nadjmi, and P. Suetens. Very fast soft tissue predictions with mass tensor model for maxillofacial surgery planning systems. In *Proc. CARS*, pages 491–496, 2005.

[MTKL+02] N. Magnenat-Thalmann, P. Kalra, J. L. Lêvêque, R. Bazin, D. Batisse, and B. Querleux. A Computational Skin Model: Fold and Wrinkle Formation. *IEEE Trans. Inf. Technol. Biomed.*, 4(6):317–323, 2002.

[MTPS07] J. Mezger, B. Thomaszewski, S. Pabst, and W. Strasser. A Finite Element Method for Interactive Physically Based Shape Modelling with Quadratic Tetrahedra. Technical report, Tübingen University, 2007.

[MTPT88] N. Magnenat-Thalmann, E. Primeau, and D. Thalmann. Abstract muscle action procedures for human face animation. *Vis. Comput.*, 3(5):290–297, 1988.

[MTT98] R. J. Meyers, T. J. Tautges, and P. M. Tuchinsky. The Hex-Tet Hex-Dominant Meshing Algorithm as Implemented in CUBIT. In *Proc. IMR*, pages 151–158, 1998.

[NMK+06] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. *Comput. Graph. Forum*, 25(4):1–24, 2006.

[NMP+05] M. Nesme, M. Marchal, E. Promayon, M. Chabanas, Y. Payan, and F. Faure. Physically realistic interactive simulation for biological soft tissues. *Recent Res. Dev. Biomech.*, 2:1–22, 2005.

[NN99] J.-Y. Noh and U. Neumann. A Survey of Facial Modeling and Animation Techniques. Technical report, Univeristy of Southern Californina, 1999.

[NPF05] M. Nesme, Y. Payan, and F. Faure. Efficient, physically plausible finite elements. In *Proc. Eurographics*, 2005.

[NPP13] M. A. Nazari, P. Perrier, and Y. Payan. A Muscle Model Based on Feldman's Lambda Model: 3D Finite Element Implementation. In *Proc. CMBBE*, pages 457–458, 2013.

[NRP11] M. Nieser, U. Reitebuch, and K. Polthier. CUBECOVER – Parameterization of 3D Volumes. *Comp. Graph. Forum*, 30(5):1397–1406, 2011.

[NT98] L. P. Nedel and D. Thalmann. Real Time Muscle Deformations Using Mass-Spring Systems. In *Proc. CGI*, pages 156–165, 1998.

[NTHAB+98] V. Ng-Thow-Hing, A. Agur, K. Ball, E. Fiume, and N. McKee. Shape reconstruction and subsequent deformation of soleus muscle models using B-spline solid primitives. In *Proc. SPIE*, pages 423–434, 1998.

[NTHF02] V. Ng-Thow-Hing and E. Fiume. Application-specific muscle representations. In *Proc. GI*, pages 107–115, 2002.

[NVI12] NVIDIA Corporation. *NVIDIA CUDA C Programming Guide Version 4.2*, 2012.

[OMvO+03] C. W. J. Oomens, M. Maenhout, C. H. van Oijen, M. R. Drost, and F. P. Baaijens. Finite element modelling of contracting skeletal muscle. *Philos. Trans. R. Soc.*, 358(1437):1453–1460, 2003.

[Ope13] OpenStax College. Skeletal Muscle. http://cnx.org/contents/6df8aab3-1741-4016-b5a9-ac51b52fade0@3/Skeletal_Muscle, 2013.

[OS00] S. J. Owen and S. Saigal. H-Morph: an indirect approach to advancing front hex meshing. *Int. J. Numer. Methods Eng.*, 49(1–2):289–312, 2000.

[OSU05] L. Olovsson, K. Simonsson, and M. Unosson. Selective mass scaling for explicit finite element analyses. *Int. J. Numer. Methods Eng.*, 63(10):1436–1445, 2005.

[Owe98] S. J. Owen. A Survey of Unstructured Mesh Generation Technology. In *Proc. IMR*, pages 239–267, 1998.

[Par72] Frederick I. Parke. Computer Generated Animation of Faces. In *Proc. ACM '72*, pages 451–457, Boston, Massachusetts, USA, 1972. ACM.

[PB81] Stephen M. Platt and Norman I. Badler. Animating facial expressions. In *Proc. SIGGRAPH '81*, pages 245–252, Dallas, Texas, United States, 1981. ACM.

[PDA03] G. Picinbono, H. Delingette, and N. Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *G. Models*, 65(5):305–321, 2003.

[PF02] I. S. Pandzic and R. Forchheimer. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. Wiley-Blackwell, 2002.

[PO09] E. G. Parker and J. F. O'Brien. Real-Time Deformation and Fracture in a Game Environment. In *Proc. SCA*, pages 156–166, 2009.

[PPF+88] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, and O. C. Zienkiewicz. Finite element Euler computations in three dimensions. *Int. J. Numer. Methods Eng.*, 26(10):2135–2159, 1988.

[PSB+07] G. Pileicikiene, A. Surna, R. Barauskas, R Surna, and A. Basevicius. Finite element analysis of stresses in the maxillary and mandibular dental arches and TMJ articular discs during clenching into maximum intercuspation, anterior and unilateral posterior occlusion. *Stomatologija*, 9(4):121–128, 2007.

[PTKK09] E. C. Paes, H. J. Teepen, W. A. Koop, and M. Kon. Perioral Wrinkles: Histologic Differences Between Men and Women. *Aesthet. Surg. J.*, 29(6):457–472, 2009.

[PTVF07] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes: the art of scientific computing*. Cambridge University Press, third edition, 2007.

[Pus00] M. A. Puso. A highly efficient enhanced assumed strain physically stabilized hexahedral element. *Int. J. Numer. Methods Eng.*, 49(8):1029–1064, 2000.

[PW08] F. I. Parke and K. Waters. *Computer Facial Animation*. A K Peters, Ltd., second edition, 2008.

[QCY09] B. Qiao, G. Chen, and X. Ye. The Research of Soft Tissue Deformation Based on Mass-Spring Model. In *Proc. ICMA*, pages 4655–4660, 2009.

[RB02] M. B. Rubin and S. R. Bodner. A three-dimensional nonlinear model for dissipative response of soft tissue. *Int. J. Solids Struct.*, 39(19):5081–5099, 2002.

[RBM08] C. D. G. Reis, H. Bagatelo, and J. M. Martino. Real-time Simulation of Wrinkles. In *Proc. WSCG*, pages 109–116, 2008.

[RDP12] O. Röhrle, J. B. Davidson, and A. J. Pullan. A Physiologically Based, Multi-Scale Model of Skeletal Muscle Structure and Function. *Front. Physiol.*, 3:358, 2012.

[RG11] E. Ruiz-Gironés. *Automatic Hexahedral Meshing Algorithms: From Structured to Unstructured Meshes*. PhD thesis, Polytechnic University of Catalonia, Spain, 2011.

[RGRS09] E. Ruiz-Gironés, X. Roca, and J. Sarrate. A New Procedure to Compute Imprints in Multi-sweeping Algorithms. In *Proc. IMR*, pages 281–299, 2009.

[RGS10] E. Ruiz-Gironés and J. Sarrate. Generation of structured hexahedral meshes in volumes with holes. *Finite Elem. Anal. Des.*, 46(10):792–804, 2010.

[RL13] J. Ramos and C. Larboulette. A Muscle Model for Enhanced Character Skinning. In *Proc. WSCG*, pages 107–116, 2013.

[RP07] O. Röhrle and A. J. Pullan. Three-dimensional finite element modelling of muscle forces during mastication. *J. Biomech.*, 40(15):3363–3372, 2007.

[RS06] A. Ramos and J. A. Simões. Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur. *Med. Eng. Phys.*, 28(9):916–924, 2006.

[RS10] X. Roca and J. Sarrate. An automatic and general least-squares projection procedure for sweep meshing. *Eng. Comput.*, 26(4):391–406, 2010.

[RU12] Y. Rezaeitabar and I. Ulusoy. Automatic 3D segmentation of individual facial muscles using unlabeled prior information. *Int. J. CARS*, 7(1):35–41, 2012.

[San06] Manuel Sanchez. *Techniques for Performance-based, real-time Facial Animation.* PhD thesis, The University of Sheffield, UK, 2006.

[SBO06] M. A. Scott, S. E. Benzeley, and S. J. Owen. Improved many-to-one sweeping. *Int. J. Numer. Methods Eng.*, 65(3):332–348, 2006.

[SC99] Nicholas Ayache Stéphane Cotin, Hervé Delingette. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Trans. Vis. Comput. Graph.*, 5(1):62–73, 1999.

[Sch96] R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Eng. Comput.*, 12(3–4):168–177, 1996.

[Sch00] R. Schneiders. Algorithms for Quadrilateral and Hexahedral Mesh Generation. In *Proc. VKI-LS*, 2000.

[SERB98] A. Sheffer, M. Etzion, A. Rappoport, and M. Bercovier. Hexahedral Mesh Generation using the Embedded Voronoi Graph. In *Proc. IMR*, pages 347–364, 1998.

[SG05] H. Si and K. Gärtner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proc. IMR*, pages 147–163, 2005.

[She07] J. F. Shepherd. *Topologic and geometric constraint-based hexahedral mesh generation.* PhD thesis, The University of Utah, USA, 2007.

[Shi05] M. Shinya. Theories for Mass-Spring Simulation in Computer Graphics: Stability, Costs and Improvements. *IEICE Trans. Inf. Syst.*, E88-D(4):767–774, 2005.

[SJ09] J. F. Shepherd and C. R. Johnson. Hexahedral Mesh Generation for Biomedical Models in SCIRun. *Eng. Comput.*, 25(1):97–114, 2009.

[SK07] B. Stojanovic and M. Kojic. Modeling of Musculoskeletal Systems Using Finite Element Method. *J. Serbian Soc. Comput. Mech.*, 1(1):110–119, 2007.

[SKO+10] M. L. Staten, R. A. Kerr, S. J. Owen, T. D. Blacker, M. Stupazzini, and K. Shimada. Unconstrained plastering – Hexahedral mesh generation via advancing-front geometry decomposition. *Int. J. Numer. Methods Eng.*, 81(2):135–171, 2010.

[SLG+07] A. Sarti, C. Lamberti, R. Gori, G. Erbacci, L. Bassani, A. Bianchi, and C. Marchetti. Virtual Planning of Facial Reconstructions. *Imaging Decis. MRI*, 11(1):29–38, 2007.

[SLML01] J.-M. Schwartz, È. Langelier, C. Moisan, and D. Laurendeau. Non-linear Soft Tissue Deformations for the Simulation of Percutaneous Surgeries. In *Proc. MICCAI*, pages 1271–1272, 2001.

[SM06] T. S. Sørensen and Jesper Mosegaard. An Introduction to GPU Accelerated Surgical Simulation. In *Proc. ISBMS*, pages 93–104, 2006.

[SNF05] E. Sifakis, I. Neverov, and R. Fedkiw. Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data. *ACM Trans. Graph.*, 24(3):417–425, 2005.

[SPCM97] F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy-Based Modeling of the Human Musculature. In *Proc. SIGGRAPH*, pages 163–172, 1997.

[SR06] A. Seugling and M. Rölin. Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool. Master's thesis, Umeå University, Sweden, 2006.

[SSIF07] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid Simulation of Deformable Solids. In *Proc. SCA*, pages 81–90, 2007.

[SSLS10] M. L. Staten, J. F. Shepherd, F. Ledoux, and K. Shimada. Hexahedral Mesh Matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces. *Int. J. Numer. Methods Eng.*, 82(12):1475–1509, 2010.

[SSRMF06] E. Sifakis, A. Selle, A. Robinson-Mosher, and R. Fedkiw. Simulating Speech with a Physics-Based Facial Muscle Model. In *Proc. SCA*, pages 261–270, 2006.

[Sta09] Susan Standring. *Gray's Anatomy*. Churchill Livingstone, 40th edition, 2009.

[Syd09] L. Sydell. Building The Curious Faces Of 'Benjamin Button'. http://www.npr.org/templates/story/story.php?storyId=100668766, 2009.

[SZM12] L. Sun, G. Zhao, and X. Ma. Quality improvement methods for hexahedral element meshes adaptively generated using grid-based algorithm. *Int. J. Numer. Methods Eng.*, 89(6):726–761, 2012.

[TBM96] T. J. Tautges, T. Blacker, and S. A. Mitchell. The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing AllHexahedral Finite Element Meshes. *Int. J. Numer. Methods Eng.*, 39(19):3327–3349, 1996.

[TCC+09] Z. A. Taylor, O. Comas, M. Cheng, J. Passenger, D. J. Hawkes, D. Atkinson, and S. Ourselin. On modelling of anisotropic viscoelasticity for soft tissue simulation: Numerical solution and GPU execution. *Med. Image Anal.*, 13(2):234–244, 2009.

[TCO08] Z. A. Taylor, M. Cheng, and S. Ourselin. High-Speed Nonlinear Finite Element Analysis for Surgical Simulation Using Graphics Processing Units. *IEEE Trans. Med. Imaging*, 27(5):650–663, 2008.

[TE05] E. Tejada and T. Ertl. Large Steps in GPU-based Deformable Bodies Simulation. *Simul. Model. Prac. Theory*, 13(8):703–715, 2005.

[Teo09] L. Teo. The making of Avatar. http://www.techradar.com/news/video/the-making-of-avatar-658031, 2009.

[THMG04] Matthias Teschner, Bruno Heidelberger, Matthias Müller, and Markus Gross. A Versatile and Robust Model for Geometrically Complex Deformable Solids. In *Proc. CGI 2004*, pages 312–319, Crete, Greece, 2004. IEEE Computer Society.

[TL04] D. Terzopoulos and Y. Lee. Behavioral Animation of Faces: Parallel, Distributed, and Real-Time. In *SIGGRAPH 2004 Course Notes: Facial Modeling and Animation*, pages 119–128, 2004.

[TMSP80] J. T. Todd, L. S. Mark, R. E. Shaw, and J. B. Pittenger. The Perception of Human Growth. *Sci. Am.*, 242(2):132–144, 1980.

[TSB+05] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and Simulating Skeletal Muscle from the Visible Human Data Set. *IEEE Trans. Vis. Comput. Graph*, 11(3):317–328, 2005.

[TTC+05] C. P. Tsui, C. Y. Tang, C. L. Chow, S. C. Hui, and Y. L. Hong. Active Finite Element Method for Simulating the Contraction Behavior of a Muscle-Tendon Complex. *Adv. Mater. Res.*, 9:9–14, 2005.

[TTL+04] C. P. Tsui, C. Y. Tang, C. P. Leung, K. W. Cheng, Y. F. Ng, D. H. Chow, and C. K. Li. Active finite element analysis of skeletal muscle-tendon complex during isometric, shortening and lengthening contraction. *Biomed. Mater. Eng.*, 14(3):271–279, 2004.

[TW90] D. Terzopoulos and K. Waters. Physically-Based Facial Modeling, Analysis, and Animation. *J. Vis. Comput. Animat.*, 1(2):73–80, 1990.

[TZT09] C. Y. Tang, G. Zhang, and C. P. Tsui. A 3D skeletal muscle model coupled with active contraction of muscle fibres and hyperelastic behaviour. *J. Biomech.*, 42:865–872, 2009.

[UGÖ98] B. Uz, U. Güdükbay, and B. Özgüç. Realistic speech animation of synthetic faces. In *Proc. Computer Animation*, pages 111–118, 1998.

[VJMW11] K. Vemaganti, G. R. Joldes, K. Miller, and A. Wittek. Total Lagrangian Explicit Dynamics-Based Simulation of Tissue Tearing. In *Computational Biomechanics for Medicine: Soft Tissues and the Musculoskeletal System*, pages 63–72. Springer Science & Business Media, 2011.

[VS10] G. J. Vancso and H. Schönherr. *Scanning Force Microscopy of Polymers*. Springer, 2010.

[Wan10] W. Wang. *Muscle-based Facial Animation*. PhD thesis, Texas A&M University, USA, 2010.

[Wat87] K. Waters. A muscle model for animation three-dimensional facial expression. In *Proc. SIGGRAPH*, pages 17–24, 1987.

[WD04] A. M. Woodward and P. J. Delmas. Towards a Low Cost Realistic Human Face Modelling and Animation Framework. In *Proc. IVCNZ*, pages 11–16, 2004.

[WH04] W. Wu and P. A. Heng. A hybrid condensed finite element model with GPU acceleration for interactive 3D soft tissue cutting. *Comput. Animat. Virtual Worlds*, 15(3):219–227, 2004.

[WHHM13] T. Wu, A. P.-L. Hung, P. Hunter, and K. Mithraratne. Modelling facial expressions: A framework for simulating nonlinear soft tissue deformations using embedded 3D muscles. *Finite Elem. Anal. Des.*, 76:63–70, 2013.

[WKMMT99] Y. Wu, P. Kalra, L. Moccozet, and N. Magnenat-Thalmann. Simulating wrinkles and skin aging. *Vis. Comput.*, 15(4):183–198, 1999.

[WLBS95] D. R. White, M. Lai, S. E. Benzley, and G. D. Sjaardema. Automated Hexahedral Mesh Generation by Virtual Decomposition. In *Proc. IMR*, pages 165–176, 1995.

[WM12] M. Warburton and S. Maddock. Creating Animatable Non-Conforming Hexahedral Finite Element Facial Soft-Tissue Models for GPU Simulation. In *Proc. WSCG*, pages 317–325, 2012.

[WM13a] M. Warburton and S. Maddock. Creating Finite Element Models of Facial Soft Tissue. In *Proc. WSCG*, pages 215–224, 2013.

[WM13b] M. Warburton and S. Maddock. GPU Simulation of Finite Element Facial Soft-Tissue Models. In *Proc. TPCG*, pages 1–8, 2013.

[WM13c] M. Warburton and S. Maddock. Physically-Based Forehead Animation including Wrinkles. In *Proc. CASA*, 2013.

[WM13d] M. Warburton and S. Maddock. Physics-Based Soft-Tissue Modelling and Simulation. In *Proc. IUPS*, 2013.

[WM14] M. Warburton and S. Maddock. Physically-Based Forehead Animation including Wrinkles. *Comput. Animat. Virtual Worlds*, Early View, 2014.

[WMSH10] T. Wu, K. Mithraratne, M. Sagar, and P. J. Hunter. Characterizing Facial Tissue Sliding Using Ultrasonography. In *Proc. WCB*, pages 1566–1569, 2010.

[WN12] R. J. Warren and P. C. Neligan. *Plastic Surgery: Volume 2: Aesthetic Surgery.* Saunders, 3rd edition, 2012.

[WNR09] E. Wang, T. Nelson, and R. Rauch. Back to Elements - Tetrahedra vs. Hexahedra. Technical report, CAD-FEM GmbH, 2009.

[WRK+10] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien. Dynamic Local Remeshing for Elastoplastic Simulation. *ACM Trans. Graph.*, 29(4):49:1–49:11, 2010.

[WSO04] D. R. White, S. Saigal, and S. J. Owen. CCSweep: automatic decomposition of multi-sweep volumes. *Eng. Comput.*, 20(3):222–236, 2004.

[WT93] K. Waters and D. Terzopoulos. The Computer Synthesis of Expressive Faces. *Philos. Trans.: Biol. Sci.*, 335(1273):87–93, 1993.

[WT04] X. Wu and F. Tendick. Multigrid Integration for Interactive Deformable Body Simulation. In *Proc. ISMS*, pages 92–104, 2004.

[WT08] Chris Wojtan and Greg Turk. Fast Viscoelastic Behavior with Thin Features. *ACM Trans. Graph.*, 27(3):47:1–47:8, 2008.

[WTGT09] C. Wojtan, N. Thürey, M. Gross, and G. Turk. Deforming Meshes that Split and Merge. *ACM Trans. Graph.*, 28(3):76:1–76:10, 2009.

[WYX09] S. Wang, J. Yang, and K. Xie. A novel method for boundary condition computation in soft tissue deformation. In *Proc. CSIE*, pages 186–190, 2009.

[XLZH11] S. Xu, X. P. Liu, H. Zhang, and L. Hu. A Nonlinear Viscoelastic Tensor-Mass Visual Model for Surgery Simulation. *IEEE Trans. Instrum. Meas.*, 60(1):14–20, 2011.

[YGC10] J. Yin, G. J. Gerling, and X. Chen. Mechanical modeling of a wrinkled fingertip immersed in water. *Acta Biomaterialia*, 6(4):1487–1496, 2010.

[YK11] K. H. Yang and A. I. King. Modeling of the Brain for Injury Simulation and Prevention. In *Biomechanics of the Brain*, pages 91–110. Springer Science & Business Media, 2011.

[YKHG02] 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(9):1253–1262, 2002.

[YRTG10] Y. Yang, G. Rong, L. Torres, and X. Guo. Real-time hybrid solid simulation: spectral unification of deformable and rigid materials. *Comput. Animat. Virtual Worlds*, 21(3–4):151–159, 2010.

[YS03] S. Yamakawa and K. Shimada. Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. *Int. J. Numer. Methods Eng.*, 57(15):2099–2129, 2003.

[YZ05] X. S. Yang and J. J. Zhang. Modelling and Animating Hand Wrinkles. In *Proc. ICCS*, pages 199–206, 2005.

[ZBX05] Y. Zhang, C. Bajaj, and G. Xu. Surface Smoothing and Quality Improvement of Quadrilateral/Hexahedral Meshes with Geometric Flow. In *Proc. IMR*, pages 449–468, 2005.

[ZGHD00] S. Zachow, E. Gladilin, H.-C. Hege, and P. Deuflhard. Finite-Element Simulation of Soft Tissue Deformation. In *Proc. CARS*, pages 23–28, 2000.

[ZHB10] Y. Zhang, T. J. R. Hughes, and C. L. Bajaj. An Automatic 3D Mesh Generation Method for Domains with Multiple Materials. *Comput. Methods Appl. Mech. Eng.*, 199(5–8):405–415, 2010.

[ZHD06] S. Zachow, H.-C. Hege, and P. Deuflhard. Computer-Assisted Planning in Cranio-Maxillofacial Surgery. *J. Comp. Inf. Technol.*, 14(1):53–64, 2006.

[ZPS03] Y. Zhang, E. C. Prakash, and E. Sung. Efficient Modeling of An Anatomy-Based Face and Fast 3D Facial Expression Synthesis. *Comput. Graph. Forum*, 22(2):159–169, 2003.

[ZPS04] Y. Zhang, E. C. Prakash, and E. Sung. A New Physical Model with Multilayer Architecture for Facial Expression Animation Using Dynamic Adaptive Mesh. *IEEE Trans. Vis. Comput. Graph.*, 10(3):339–352, 2004.

[ZST05a] Y. Zhang, T. Sim, and C. L. Tan. From Range Data to Animated Anatomy-Based Faces: A Model Adaptation Method. In *Proc. 3DIM*, pages 343–350, 2005.

[ZST05b] Y. Zhang, T. Sim, and C. L. Tan. Simulating Wrinkles in Facial Expressions on an Anatomy-Based Face. In *Proc. ICCS*, pages 207–215, 2005.

[ZZM07] H. Zhang, G. Zhao, and X. Ma. Adaptive generation of hexahedral element mesh using an improved grid-based method. *Comput.-Aided Des.*, 39(10):914–928, 2007.