UNIVERSITY OF CALIFORNIA

Los Angeles

Biomechanical Modeling and Control of the Human Body for Computer Animation

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Sung Hee Lee

2008

The dissertation of Sung Hee Lee is approved.

Michael G. Dyer

Petros Faloutsos

Charles E. Taylor

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2008

To my mother and father, and to my wife

TABLE OF CONTENTS

1 Introduction 1

1.1 Contributions 5

1.2 Overview 11

2 Related Work 12

2.1 Muscle-Driven Motion Generation 12

2.2 Soft Tissue Deformation 14

2.3 Anatomical Modeling of the Human Body 15

2.4 Biological Joint Models 16

3 Biomechanical Modeling of the Neck-Head-Face Complex 19

3.1 Musculoskeletal Model 19

3.1.1 Musculoskeletal Model 20

3.1.2 Muscular Structure 23

3.2 Hill-Type Muscle Force Model 24

4 Neuromuscular Control of the Neck-Head System 27

4.1 System Overview 28

4.2 Reflex Control 30

4.3 Voluntary Control and Learning 31

4.3.1 Neural Networks 33

4.3.2 Setpoint Signal Generator 36

4.3.3 Head Motion Controller 38

4.4 Experiments and Results 39

4.4.1 Basic Simulations 39

4.4.2 Stiffness Control Experiments 40

4.4.3 Tracking Head Motion Capture Data 41
4.4.4 Gaze Behavior 43
4.4.5 Autonomous Multi-Head Interaction 45

5 Comprehensive Biomechanical Modeling and Control of the Body . . . 47

5.1 Musculoskeletal Modeling 48
5.1.1 Skeleton 48
5.1.2 Muscle Actuation Model 51
5.2 Soft Tissue Modeling and Simulation 60
5.2.1 Skin Surface Model Creation 60
5.2.2 Generation of an Embedded Simulation Mesh 63
5.2.3 Modeling Musculature and Skeletal Structure 67
5.2.4 Numerical Simulation and Time Integration 70
5.3 Control 72
5.3.1 System Overview 73
5.3.2 Computing Desired Accelerations 73
5.3.3 Generalized Force Computation 74
5.3.4 Computation of Muscle Activation Levels 75
5.3.5 Sensor Modeling 77
5.4 Experiments and Results 78

6 Spline Joints for Biological Joint Modeling 84

6.1 Geometric Preliminaries 85
6.2 Dynamics of General Scleronomic Joints 88
6.2.1 Creating New Joints 89
6.3 Formulation of Spline Joints 91
6.3.1 Curve Spline Joint 92
6.3.2 Multi-DOF Spline Joint 94

6.3.3 Derivatives of the Spline Joint Transformation 94
6.4 Designing Curve Spline Joints 96
6.4.1 Natural Parameterization 96
6.4.2 Data Fitting Algorithm 97
6.4.3 Smoothing Algorithm 99
6.5 Experiments and Results 100

7 Conclusion 107
7.1 Summary 107
7.2 Comparison Against Competing Approaches 108
7.3 Future Work 110

A CE Contribution to the Stiffness 115

B The Recursive Forward Dynamics Algorithm 116

C Parameters of the Musculoskeletal Model 118

References 135

List of Figures

1.1 Biomechanical modeling of the Head-Neck-Face complex 8

1.2 Biomechanical modeling of the human body 9

1.3 Snapshots of the femorotibial joint 10

3.1 The musculoskeletal model 21

3.2 Linearized Hill-type model 25

4.1 Face-Head-Neck System Architecture 28

4.2 The sub-controllers in the voluntary controller 32

4.3 A 3-layer neural network 34

4.4 Head motions created by different stiffness settings 40

4.5 Head orientation longitudinal angle during an impact simulation . . . 42

4.6 The sensorimotor controller following head motion capture data . . . 43

4.7 Head-Eye gaze behavior 44

4.8 Interaction between three autonomous face-neck-head systems 46

5.1 Modeled skeleton of the body 49

5.2 Closeup view of the upper body bones 50

5.3 Ventral view of the source geometry data of the modeled muscles . . . 52

5.4 Dorsal view of source geometry data of the modeled muscles 53

5.5 Ventral view of the lines-of-action of the uniaxial muscle actuators . . 54

5.6 Dorsal view of lines-of-action of the uniaxial muscle actuators 55

5.7 Piecewise line segment muscle models 57

5.8 Computing the PCSA of a muscle 59

5.9 Volumetric human body model 61

5.10 Detail from the chest of the anatomical model 63

5.11 Volumetric mesh cutting for the skin surface 65
5.12 Coarsening the volumetric mesh 66
5.13 Emdedding the skeleton into the volumetric mesh 71
5.14 Overview of the motion controller 73
5.15 Biomechanical body model holding dumbbells 79
5.16 Muscle coactivation 80
5.17 Snapshots of the flexing motion 81
5.18 Sensor delay model 82
5.19 Respiration 82
5.20 Snapshots of the jumping motion 83

6.1 An elliptic joint 90
6.2 The femorotibial joint 101
6.3 Knee spline joint inverse kinematics example 102
6.4 The scapulothoracic joint modeled with the spline surface joint 103
6.5 Example mechanisms modeled with spline joints 104
6.6 “SIGGRAPH Joint 1” 106
6.7 “SIGGRAPH Joint 2” 106

List of Tables

3.1 Physical parameters of the skeleton 20

3.2 List of the muscles modeled in the neck 24

5.1 List of the bones modeled in the body 48

5.2 List of the muscles modeled in the body 56

6.1 Frequently used symbols 87

C.1 Symbols used in the muscle parameter tables. 119

C.2 Bone parameters 119

C.3 Parameters of the muscles in the neck-head 121

C.4 Parameters of the muscles in the trunk 124

C.5 Parameters of the arm muscles 131

C.6 Parameters of the leg muscles 133

Acknowledgments

I am deeply grateful to my advisor, Professor Demetri Terzopoulos. His publications on computer animation fascinated me so much that I was compelled to pursue my PhD study in the field. I was very fortunate to be his student and learn from him what it takes to be a good researcher. Demetri encouraged me to set the highest standards and have ambitious goals in my research, and he enabled me to achieve my goals. This dissertation would never have been possible without his guidance and support.

I thank Professors Michael G. Dyer, Petros Faloutsos, and Charles E. Taylor for serving on my thesis committee and offering me their advice on how to improve my dissertation. I would especially like to thank Professor Faloutsos for his unwavering interest in my research and for his counsel on various issues that I encountered during my PhD study.

My appreciation also goes to my colleagues. I thank my friends and lab mates at NYU and UCLA, Wei Shao, Alex Vasilescu, Evgueni Parilov, Ari Shapiro, Anna Majkowska, Brian Allen, Gabriele Nataneli, Shawn Singh, Billy Hewlett, Kresimir Petrinec, and Mark Korthals for sharing their knowledge and ideas with me. They gave me insightful opinions on numerous aspects of computer graphics research and life in general. I also express my gratitude to Jinwook Kim and Sang Hoon Yeo for being great discussion partners, and to Ambarish Goswami, Brian Guenter, and Ari Shapiro for being great mentors during my internships at Honda Research, Microsoft Research, and Rhythm&Hues Studios.

I acknowledge with particular appreciation the contributions of Eftychios Sifakis. From him, I learned a great deal about the finite element simulation of soft tissues. Specifically, the work on modeling and simulating soft tissues reported in this thesis and in our co-authored paper [Lee et al. 2009] is primarily his. I also thank Yuencheng Lee for developing and implementing the biomechanical face model that I have used in my work.

The research reported herein was made possible in part by the financial support of the National Science Foundation under Grant No. IIS-0326388, by the IT Scholarship Program supervised by IITA, Republic of Korea, and by a Dissertation Year Fellowship from UCLA.

On a more deeply personal level, I thank my mother and father for the nurturing encouragement and trust that they have given me throughout my life. Most of all, I cannot adequately thank my loving wife Mina, the wonderful mother of our children, Ken and Kate, for her boundless love and support.

VITA

1974	Born, Seoul, South Korea
1996	B.S., Mechanical Engineering Seoul National University Seoul, South Korea
1996-1998	Military Service, Busan, South Korea
2000	M.S., Mechanical Engineering Seoul National University Seoul, South Korea
2000–2002	Research Engineer Samsung Advanced Institute of Technology Yong-In, South Korea
2004–2005	Research Assistant Computer Science Department New York University New York, New York
2005	Teaching Assistant Computer Science Department New York University New York, New York

2006–2007	Research Assistant Computer Science Department University of California, Los Angeles Los Angeles, California
2006	Research Intern Honda Research Institute USA Mountain View, California
2007–2008	Dissertation Year Fellowship Award University of California, Los Angeles Los Angeles, California
2007	Research Intern Microsoft Research Redmond, Washington
2008	Software Intern Rhythm & Hues Studios Los Angeles, California

PUBLICATIONS

S.-H. Lee and D. Terzopoulos (2008). Spline Joints for Multibody Dynamics. *ACM Transactions on Graphics*, **27**(3):22:1–8. Proceedings of the *ACM SIGGRAPH 2008 Conference*, Los Angeles, CA, August 2008.

S.-H. Lee and D. Terzopoulos (2008). Learning Neuromuscular Control for the Biomechanical Simulation of the Neck-Head-Face Complex. *The Learning Workshop*, Snowbird, UT, April 2008, 157–159.

S.-H. Lee and A. Goswami (2007). Reaction Mass Pendulum (RMP) : An explicit model for centroidal angular momentum of humanoid robots. Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, April 2007, 4667–4672.

S.-H. Lee and D. Terzopoulos (2006). Heads Up! Biomechanical Modeling and Neuromuscular Control of the Neck. *ACM Transactions on Graphics*, **25**(3):1188–1198. Proceedings of the *ACM SIGGRAPH 2006 Conference*, Boston, MA, August 2006.

S.-H. Lee, J. Kim, F.C. Park, M. Kim, and J.E. Bobrow (2005). Newton-Type Algorithms for Dynamics-Based Robot Movement Optimization. *IEEE Transactions on Robotics*, **21**(4):657–667.

S. Yeo, J. Kim, S.-H. Lee, F.C. Park, W. Park, J. Kim, C. Park, and I. Yeo (2004). A Modular object-oriented framework for hierarchical multi-resolution robot simulation. *Robotica*, **22**:141–154.

Abstract of the Dissertation

Biomechanical Modeling and Control of the Human Body for Computer Animation

by

Sung Hee Lee

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2008

Professor Demetri Terzopoulos, Chair

Realistic anatomical modeling capable of high-fidelity synthesis of human body shape and motion is a major challenge in computer animation. Despite significant progress in this domain, the detailed modeling of the human body has not received adequate attention because of its complexity. We develop a comprehensive biomechanical model of the human body, confronting the combined challenge of modeling and controlling more or less all of the relevant articular bones and muscles, as well as simulating the physics-based deformations of the soft tissues.

Emulating the relevant anatomy, our skeletal model comprises 75 bones (165 degrees of freedom), including the vertebrae and ribs, and it is actuated by 846 muscles, modeled as piecewise uniaxial Hill-type force actuators. To simulate the biomechanics of the soft tissues, we employ a coupled 3D finite element model with the appropriate constitutive behavior, in which are embedded the detailed anatomical arrangement and geometries of skin, muscle, and bone.

As a precursor to developing our comprehensive biomechanical model, we consider the highly important head-neck-face complex. Our head-neck model is characterized by appropriate kinematic redundancy (7 vertebrae) and muscle actuator redundancy (72 muscles). It presents us with a challenging motor control problem, even for the deceptively simple task of balancing the head in gravity atop the cervical spine. Our biologically inspired, neuromuscular controller provides the numerous muscle actuators with efferent activation signals, controlling the pose of the head through time, as well as the stiffness of the neck by coactivating mutually opposed muscles. Using machine learning techniques, the neural networks within the controller are trained offline to efficiently generate the online control signals necessary to synthesize various autonomous movements for the behavioral animation of the human head and face.

Our biomimetic modeling approach heightens the need to accurately model skeletal joints. Since the elementary joints conventionally used in physics simulation cannot produce the complex movement patterns of biological joints, we also introduce a new joint model, called “spline joints”, that can emulate biological joints more accurately. Spline joints can be efficiently simulated using minimal-coordinates-based dynamics algorithms.

CHAPTER 1

Introduction

Human beings are arguably the most important subject in computer animation. Computer animators must oftentimes depict all aspects of a human being, ranging from appearance to personality, and they must "bring the graphical character to life" by making it move in a believable manner.

The appearance of human characters in computer animation has become increasingly true to life as the supporting technologies have advanced. 3D scanning technology, such as the Cyberware scanner (www.cyberware.com), makes it possible to acquire realistic human shapes with little difficulty. The most advanced rendering techniques now enable the synthesis of the complex, translucent material properties of the skin [Weyrich et al. 2006] and hair [Moon et al. 2008]. As far as modeling and rendering the *appearance* of a human character is concerned, one could argue that this has passed the "Graphics Turing Test", in the sense that "computer generated imagery [has become] comparatively indistinguishable from real images" [McGuigan 2006].

As the appearance of a graphical character becomes more realistic, however, the audience becomes increasingly sensitive to its shortcomings, and the character is in danger of falling into the "uncanny valley". According to the uncanny valley hypothesis, due to the roboticist Mori [1970], people feel more comfortable with an anthropomorphic robot as it looks and moves more like a real person, but they can feel growing aversion to the robot as it becomes very similar to, yet remains oddly distinguishable from real humans. The validity of this hypothesis is unproven, but at least in computer animation it seems evident that the audience can feel revulsion when highly realistic human characters move unnaturally, as they often do in *Final Fantasy: The Spirits Within* (2001), *The Polar Express* (2004), and other recent computer-animated motion pictures. This suggests that realistic looking human characters require commensurately realistic animation.

The keyframing animation process, championed by leading animation studios such as Pixar, in which animators manually define the poses of characters at a multitude of key points in time and the computer interpolates these key poses to produce continuous movement, requires too much labor for it to be practical in creating large quantities of highly realistic human motion—as opposed to, say, animated toys and other fantasy characters. In a sense, it would be as unproductive for an animator to attempt to create realistic human motions manually as it would be to attempt to create realistic fluid animation by hand. The impracticality of keyframing in this context hinders the use of realistic human characters in today's computer animation.

Although motion capture technologies have contributed a great deal to creating realistic human animation, they too have their limitations. Many motion capture editing techniques [Bruderlin and Williams 1995; Witkin and Popovic 1995] treat motions as *data* and take a signal processing approach to the modification of captured human motions on an ad hoc basis, without consideration of how the motions are generated by the human body. Thus, there is no guarantee that the resulting motions can actually be generated by a real human. Purely through motion capture, it is even more difficult to create realistic, real-time animation in which a human character is interacting with an arbitrary virtual environment [Shapiro et al. 2007].

Therefore, we need new technologies to tackle the challenge of creating highly realistic human motions. Rather than using recorded motions or relying on keyframing, the approach of applying the *physical principles* of movement in the real world to synthesize the motion of virtual objects for computer animation has been very successful in animating solids, liquids, and gases. Physics-based animation employs the laws of physics to simulate the dynamic motion of rigid bodies [Baraff 1989], the deformations of nonrigid solids [Terzopoulos et al. 1987; Terzopoulos and Fleischer 1988], and the flow of fluids and gases [Foster and Metaxas 1997]. This has made it possible to realistically and practically animate the motions of these sorts of computer graphics objects.

The physics-based approach has also been applied to self-animating objects in order to automatically compute realistic, optimal motion trajectories [Witkin and Kass 1988], or to make an animated creature that interacts with its simulated physical environment [Armstrong and Green 1985; Wilhelms and Barsky 1985]. For the realistic animation of lower animals, such as marine animals, researchers have modeled even more advanced levels of motion generating mechanisms, such as muscle-based biomechanical and behavioral animation [Tu and Terzopoulos 1994; Funge et al. 1999].

Actuators inspired by biological muscles have been used for over two decades to generate expressions in facial animation [Waters 1987; Lee et al. 1995; Sifakis et al. 2005]. In other aspects of human animation, however, joint-actuating PD (proportional-derivative) servos have traditionally been used instead of muscles to produce articulated skeletal animation [Hodgins et al. 1995; Faloutsos et al. 2001]. Recently, in an effort to improve realism, researchers have been developing increasingly sophisticated biomechanical models of individual body parts, such as hands [Tsang et al. 2005a; Albrecht et al. 2003] and legs [Dong et al. 2002]. Although significant progress has been made in such localized modeling, the challenge of modeling of the entire body has not received adequate attention, no doubt because of the complexity of the human body. In prior work, the torso has been simplified to a relatively small number of components. Even in the most detailed anatomical models, such as [Zordan et al. 2004] and [Nakamura et al. 2005], many articular bones in the spine and ribs are grouped and treated as a single rigid object.

This dissertation addresses the biomechanical modeling and simulation of more or less the entire human body. By employing muscle actuators, we simulate the natural motion of the skeleton, which is driven by muscle forces. The conventional, joint-centered PD servo actuator model of anthropomorphic figures in physics-based animation cannot adequately capture the complex characteristics of the multiple muscles surrounding a skeletal joint. First, the magnitude of a muscle force differs depending on the length of the muscle as well as its rate of length change through time even under the same activation level. Second, a human controls both the motion and the stiffness of her body by controlling a large number of muscles, which cannot be approximated satisfactorily by the simplistic joint PD actuator model. In our work, we adopt a well-known muscle model used in the field of biomechanics, the Hill-type model, to represent the force generating property of a muscle.

Our approach features the comprehensive modeling, without oversimplification, of anatomical structures that contribute to the motion generation of the body. We model most of the articular bones in the body so that we can simulate the full range of motions that it is capable of producing, from pronounced motions such as the flexing of the arms to more subtle motions such as respiration and laryngeal movements. The greater number of bones that we model requires a commensurately increased number of muscles to adequately actuate and control the skeleton. Hence, we model most of the skeletal muscles in the body.

This musculoskeletal system is controlled by a hierarchical neuromuscular control architecture. A distinctive feature of the mammalian motor control architecture is that it is hierarchical [Kandel et al. 2000]—multiple neural organs, such as the cerebral cortex, basal ganglia, cerebellum, and spinal cord, participate in generating the signals finally transmitted by motor neurons that innervate the muscles. This suggests that simple, flat control strategies may be incapable of synthesizing a large repertoire of human motions. Hence, we propose a two-level control architecture comprising low-level feedback control and high-level feedforward control. The feedforward control determines the approximate activation levels of each muscle, while the feedback control corrects the activation levels according to the sensed errors.

Conventional elementary joint types used in simulation cannot adequately model biological joints since biological joints have nontrivial movement patterns due to the complex shapes of bones. Hence, to complement the human body modeling, we also introduce new a joint model, called *spline joints*, that can emulate the action of biological joints more accurately. Our spline joints can be efficiently simulated using dynamics algorithms based on the minimal coordinates formulation.

1.1 Contributions

To summarize, this thesis makes four primary contributions:

1. We develop a biomimetic, musculoskeletal model of the head-neck-face complex (Chapter 3). This is the first such biomechanical model developed in computer graphics.

2. We introduce a neuromuscular approach for controlling this biomechanical model (Chapter 4). The multilevel neuromuscular controller, which is trained using

machine learning techniques, is unique in the field.

3. We expand our model into a comprehensive biomechanical model of the entire body and address the associated control problem (Chapter 5). This is the most comprehensive and complex biomechanical human model ever implemented, particularly for computer animation.

4. We introduce spline joints as a novel technique for more accurately modeling skeletal joints (Chapter 6). The spline joint technique has other applications in computer animation.

Most of the above contributions have been reported in several publications [Lee and Terzopoulos 2006; Lee and Terzopoulos 2008a; Lee et al. 2009; Lee and Terzopoulos 2008b]. The following sections overview the contributions in greater detail.

The Head-Neck-Face Complex

Unlike the human face, the neck has been largely overlooked in the computer graphics literature. This may be due in part to the complexity of cervical anatomy and biomechanics. Yet the realistic modeling of the neck is a significant problem in human animation, because the neck determines the global movement of the head and face relative to the body. Indeed, the neck plays a crucial role in supporting the mass of the head, balanced in gravity, atop the cervical spine while generating the controlled head movements that are essential to so many aspects of human behavior.

We will develop the first biomechanical model of the human head-neck musculoskeletal system for computer animation. In particular, we model the head and each vertebra in the cervical spine as a dynamic rigid body with appropriate mass distribution and three rotational degrees of freedom (DOF), coupling the bones with joints to emulate the biological assembly of interest. The resulting articulated multibody system is actuated by contractile muscles. Each actuator is also modeled biomechanically as a simplified Hill-type muscle model, which is frequently used in biomechanics research. The complexity of the musculoskeletal model, especially its kinematic and muscular redundancy, which imitates that of its biological counterpart, confronts us with a challenging control problem. We believe that the best way to tackle this problem is via an approach inspired by biological motor control mechanisms, all the more so because our long-term goal is to create lifelike characters that are able to synthesize a broad range of human motions. Hence, we develop a novel neuromuscular control model for human (head) animation that emulates the relevant biological motor control mechanisms.

Emulating the mammalian motor control architecture, we will take a hierarchical approach, proposing a bi-level motor control architecture whose lower level corresponds to *reflex* (or *feedback*) *control* in the human body, and whose upper level corresponds to *voluntary* (or *feedforward*) *control*. Our hierarchical head-neck controller provides the inputs to the numerous muscle actuators necessary to maintain the stability of the cervical spine and autonomously generate a variety of head movements. In addition to head pose and movement, it regulates the stiffness of the head-neck multibody system by controlling the tone of mutually opposed neck muscles. Taking a machine learning approach, the neural networks within our neuromuscular controller are trained offline to efficiently generate the online pose and tone control signals necessary to synthesize a variety of autonomous movements for the behavioral animation of the human head and face.

Figure 1.1 illustrates our implementation of the above ideas, and more, as a self-animating virtual human neck, head, and face. In a simulated physical environment with gravity, our autonomous system naturally selects, alters, and maintains head pose

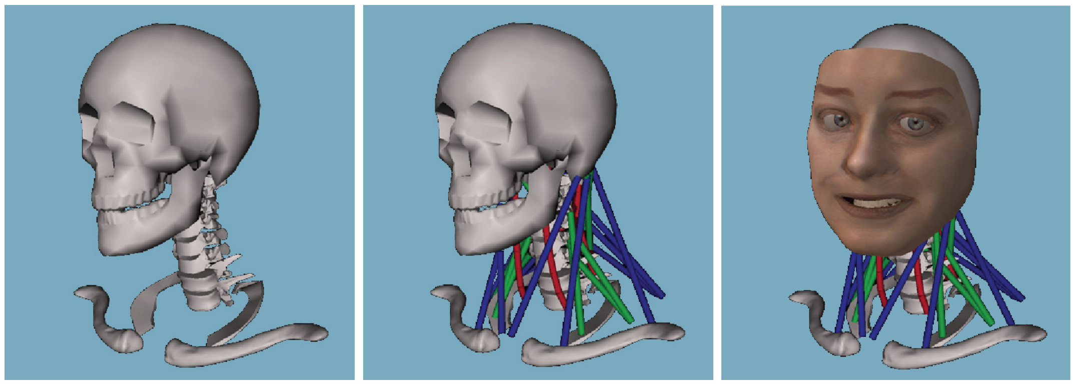


Figure 1.1: Our biomechanical head-neck system comprises the skeleton (left), Hill-type actuator muscles in three layers (center), and a biomechanical face (right).

and gaze direction, and it can adjust its stiffness to appropriately respond to external disturbances.

The Entire Body

Next, we develop a highly-detailed biomechanical model of the human body for computer animation. Our model features a musculoskeletal system with a full complement of muscle actuators plus a coupled 3D finite element simulation of soft tissue deformations (Figure 1.2).

In particular, we confront the challenge of modeling more or less all of the relevant articular bones, creating a physics-based skeletal model that consists of **75 bones** and **165 DOFs** (degrees of freedom), with each vertebral bone and most ribs having independent DOFs. To be properly actuated and controlled, our detailed bone model requires a comparable level of detail with respect to muscle modeling. We incorporate **846 muscles**, which are modeled as piecewise line segment simplified Hill-type force actuators. We also develop an associated physics-based animation controller that computes accelerations to drive the musculoskeletal system toward a sequence of tar-

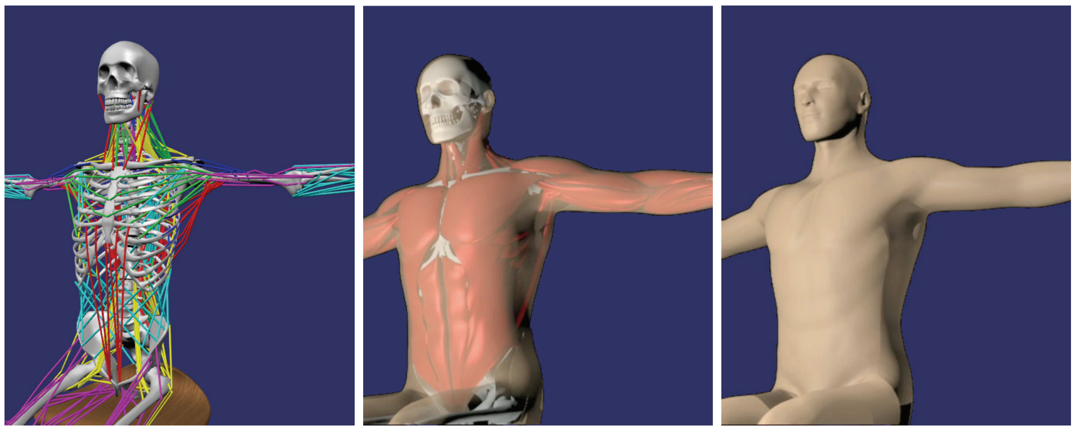


Figure 1.2: Our biomechanical human body is characterized by the comprehensive modeling of the relevant tissues for motor control. The skeleton is driven by Hill-type muscle actuators modeled with piecewise line segments (left). The motion of the skeleton and the activation level of each muscle deforms the inner soft tissue (center) as well as the skin (right).

get key poses set by an animator, and then computes the required activation signal for each muscle through inverse dynamics.

Our volumetric human body model incorporates detailed skin geometry, as well as the active muscle tissues, passive soft tissues, and skeletal substructure. Driven by the skeletal motion and muscle activation inputs, a companion simulation of a volumetric, finite element model of the soft tissue introduces the visual richness of more detailed, 3D models of the musculature. Specifically, we achieve robust and efficient simulation of soft tissue deformation within the finite element framework by decoupling the visualization geometry from the simulation geometry. A total of **354,000 Body-Centered-Cubic (BCC) tetrahedra** are simulated to create detailed deformation of the embedded high-resolution surfaces of the skin and each of the muscles.

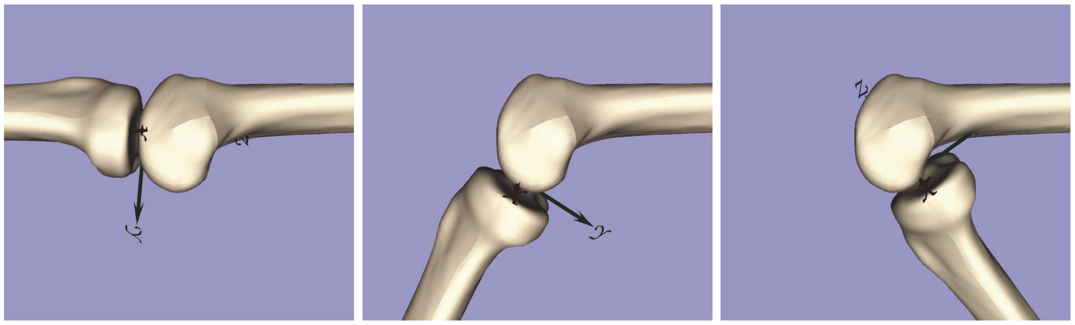


Figure 1.3: A spline joint can model complex biological joints much more accurately than is possible using conventional joint models. In this figure, the femorotibia joint is modeled using a spline joint. The rotation axis of the tibia moves as the joint is flexed and extended.

Spline Joints

Traditionally, only a few types of elementary joints have been used to model articulated multibody systems for physics-based animation, robotics, and human movement research. These are the *lower pairs* [Reuleaux 1876]; i.e., the prismatic, helical, cylindrical, planar, and spherical joints, and their compounds, such as the universal joint. The lower pairs, which are used for modeling most mechanical or biological systems, are characterized by one or more fixed axes of rotation or translation, and thus can only model simplistic joint behaviors.

However, more complex joints are common in biological systems. Due to the complicated shapes of bones, biological joints usually produce non-trivial movement patterns. For example, the femorotibial joint (Figure 1.3) undergoes both rotation and translation as it is flexed and extended [Kapandji 1974]; it cannot be accurately approximated by a lower pair. While it may be reasonable to approximate biological joints, such as those in the neck, by lower pairs when one is interested only in macroscopic joint articulation [Lee and Terzopoulos 2006], the more accurate analysis and simulation of biological joint motion is important in biomechanics research and medical applications such as virtual surgery simulation or prosthetics design [Delp et al. 1990].

We will introduce the *spline joint*, a novel joint model that can emulate complex biological joints. Our key idea is to use splines to model arbitrary, complex joint motions. Specifically, we will formulate the 1-DOF curve spline joint as the product of exponentials of a twist multiplied by a spline basis function, thus defining an arbitrary twice differentiable spline motion curve on $\mathrm{SE}(3)$(special Euclidean group representing rotations and translations in $\mathbb{R}^3$) that is free of singularities. We will furthermore present geometric data-fitting and smoothing algorithms for 1-DOF spline joint design. Since higher-dimensional, analytically differentiable splines on $\mathrm{SE}(3)$ are not yet known, we formulate an *n*-DOF spline joint as the product of six exponentials of a basis twist multiplied by an *n*-parameter spline function.

Figure 1.3 illustrates our application of the spline joint model to modeling the femorotibia joint. As can be seen from the figure, the spline joint can model the translation of the axis of rotation of the tibia during the flexion and extension of the joint.

1.2 Overview

The remainder of the dissertation is organized as follows: Chapter 2 surveys related work. Chapter 3 presents our biomechanical musculoskeletal model of the head-neck-face complex. Chapter 4 introduces a biologically inspired, hierarchical controller for head-neck-face complex. Chapter 5 addresses the biomechanical modeling and control of the entire body. Chapter 6 develops the spline joint model. Finally, Chapter 7 concludes the dissertation, summarizing our work, comparing it to competing approaches and discussing several promising avenues for future research.

CHAPTER 2

Related Work

Our work cuts across physical and biological modeling and simulation in computer animation, as well as related fields such as anatomy, biomechanics, and control. In this chapter, we review relevant prior work on muscle-driven approaches for controlling human characters (Section 2.1), the simulation of the soft tissues (Section 2.2), and anatomical modeling of the human body (Section 2.3). We also review prior research on modeling biological joints (Section 2.4).

2.1 Muscle-Driven Motion Generation

In contrast to facial animation where muscle actuators have been used for over two decades to generate expressions [Waters 1987], PD servos have traditionally been used instead of muscles to produce articulated skeletal animation [Hodgins et al. 1995; Faloutsos et al. 2001]. Komura et al. [2000; 1997] computed optimal feedforward muscle activation levels given several key poses of human lower extremities for solving inverse kinematics or "physiological retargeting" of the motion. These references and [Tsang et al. 2005a] are relevant to our work in that they perform inverse dynamics to compute necessary muscle activation level for Hill-type muscle models. However, their controllers are not as comprehensive as ours, inasmuch as they disregard muscle coactivation and must solve expensive space-time optimization problems online, making them impractical for interactive, autonomous animation. Also [Tsang et al. 2005a] and [Komura et al. 1997] disregard feedback control. It should be noted that when one applies control input computed from inverse dynamics to a system without incorporating feedback control, the system can easily become unstable even under the slightest disturbance.

Not surprisingly, there exists a large biomechanics literature on human motor control mechanisms. Traditionally, biomechanics researchers have attempted to interpret motor control strategy as an optimization process and have devoted effort to understanding the optimality criteria [Crowninshield 1978; Pandy et al. 1990]. Recently, some researchers have adopted robot control theories to human motor control. Sapio et al. [2005] proposed a task-level feedback control framework in the simulation of goal-directed human motion. Thelen et al. [2003] used static optimization along with feedforward and feedback controls to drive the kinematic trajectory of a musculoskeletal leg model toward a set of desired kinematics, and reported that the muscle excitations computed by their method were similar to measured electromyographic patterns. Our motion controller takes a similar approach, but to enable the musculoskeletal system to respond more naturally to external forces, we also modeled the response times of biological sensors in Chapter 5.

With the advent of artificial neural networks, researchers have adopted the technique to the study of human motor learning. For example, Kawato et al. [1987] constructed a hierarchical neural network that learns inverse dynamics of a simple arm model. This forward simulation/learning model is biomimetic but computationally expensive. Kim and Hemami [1998] performed a similar study with a simplistic human head and torso model. Grzeszczuk et al. [1998] applied artificial neural networks and the backpropagation learning algorithm to training feedforward controllers for dynamic objects, among them a locomotion controller for a biomechanical dolphin model.

A unique feature of muscle is that its stiffness increases with increasing neural signal. Consequently, by coactivating agonist and antagonist muscles, humans and other animals can increase stiffness while maintaining pose. They effectively use such tone control to mitigate instability under external loads or to increase the accuracy of the limbs in motor tasks. It is also well known that coactivation occurs when humans learn new motions. Hogan [1984] studied tone (a.k.a. impedance) modulation by coactivating agonist and antagonist muscles. In computer animation, Neff and Fiume [2002] proposed a joint-actuated control technique in which they attached two opposing PD feedback controllers to every joint of an articulated anthropomorphic figure, controlling the tension and relaxation of the resulting body motion by modulating the two proportional feedback gains. Their work falls short of our richly muscle-actuated model in that it does not include feedforward control and its joint controllers cannot accurately model the characteristics and functions of real muscles, especially when these muscles span multiple joints as many muscles in the trunk do. Allen et al. [2007] proposed an analytic method to determine time-varying feedback gains of the PD-servos to control timing and tension of a character's motion.

2.2 Soft Tissue Deformation

In the anatomy-based modeling approach, Chen and Zeltzer [1992] introduced the biomechanical modeling of muscles for computer animation, modeling muscle tissue with large finite elements and simulating muscle deformation by applying a Hill-type force in the muscle. Lee et al. [1995] used multi-layer mass-spring-damper meshes with embedded muscle actuators to model the soft tissues of the face and synthesize facial expressions.

Parametric muscle models have been proposed that deform geometrically, and they have been used to simulate skin shape change due to the bulging of underlying muscles. Scheepers et al. [1997] and Wilhelms and Gelder [1997] used various geometric primitives to model muscles and kinematically defined their deformations due to joint motions.

Recently, more sophisticated muscle deformation methods have been proposed. Ng-Thow-Hing [2001] used B-spline solids for the efficient modeling and simulation of certain muscles. Pai et al. [2005] developed a strand muscle model for the fast simulation of individual muscles. Irving et al. [2004] enabled the robust simulation of soft tissue deformation by introducing invertible finite elements. Our soft tissue model is based on the work of Sifakis et al. [2005], where soft tissues are modeled using finite elements and material properties of muscles are embedded in surrounding elements.

While most literature focuses on either form or function of the muscle, some literature deals both aspects. Albrecht et al. [2003] proposed an anatomy-based hand animation system where they modeled two types of muscles—geometric muscle for simulating muscle deformation and pseudo-muscle for actuating bones—but their controller is manually-tuned.

2.3 Anatomical Modeling of the Human Body

Although the significant progress has been made in modeling various body parts, including the biomechanical modeling of the face [Lee et al. 1995; Sifakis et al. 2005], the hand [Albrecht et al. 2003; Tsang et al. 2005b], and the leg [Dong et al. 2002], the comprehensive modeling of the human body has not been attempted due to its complexity. In prior work, the torso has been simplified to a relatively small number of components. Monheit and Badler [1991] proposed a purely kinematic spine and torso model, where the total bending angle is distributed to each joint according to weighting parameters. Even in the most detailed anatomical models, such as [Zordan et al. 2004] and [Nakamura et al. 2005], many articular bones in the spine and ribs are grouped and treated as a single rigid body.

In biomechanics, there has been some research effort in creating detailed anatomical model for the human body. For example, Vasavada et al. [1998] constructed a 3D human neck muscle model and measured the moment-generating capacity of each muscle. They visualized human neck motion in their work, but the movement is generated kinematically, with no dynamics.

As an alternative to current muscle models developed by the biomechanics community (see, e.g., [Delp et al. 2007]), which unfortunately do not model a sufficient number of muscles to control our skeletal system, we estimated muscle parameters—e.g., attachment points and physiological cross sectional areas—by analyzing commercially available muscle geometry data, without attempting to reduce the number of muscles. For this purpose, we used the Ultimate Human Model (www.cgcharacter.com/ultimatehuman.html), which is a purely geometric model of the human body.

2.4 Biological Joint Models

Researchers have endeavored to create complex models of biological joints, but no prior effort has provided a suitably complex joint model that can be used in dynamics simulation. Delp et al. [1990] and Maciel et al. [2002] modeled the knee as a revolute joint whose joint axis translates along a parametric curve. Shao and Ng-Thow-Hing [2003] proposed a general framework for modeling complex joints by composing elementary joint components. Their method is useful for forward kinematics, but they too did not consider inverse kinematics or dynamics.

Our work is related to research on spline curves for rotation. The splines on $\mathrm{SO}(3)$ (special orthogonal group representing rotations in $\mathbb{R}^3$) that smoothly interpolate rotations are rather involved. Shoemake [1985] proposed the interpolation of rotations using quaternions. Various techniques have been developed to achieve optimal spline curves in $\mathrm{SO}(3)$ that minimize the tangential acceleration [Gabriel and Kajiya 1985; Barr et al. 1992; Kim et al. 1995; Ramamoorthi and Barr 1997; Park and Ravani 1997]. Since the interest is in interpolating the rigid motion of a single body, they all rightfully divide the problem on $\mathrm{SE}(3)$ into an interpolating rotation in $\mathrm{SO}(3)$ and translation in $\mathbb{R}^3$. In contrast, we are interested in the articulated motion of a jointed, multibody system. Hence, we treat rigid-body motion as a screw motion, without decomposing it into rotation and translation. We adopt the spline algorithms on $\mathrm{SO}(3)$ proposed by Kim et al. [1995] and extend their methods to $\mathrm{SE}(3)$ for use in our curve spline joints. Their approach provides a simple form of the derivatives of the spline curve, which makes it easier to compute the dynamics of spline joints.

Kry and Pai [2003] introduced a continuous surface contact simulation technique in a minimal-coordinates dynamics framework. Since they handle general topological surfaces obtained by subdivision, the computation of derivatives is more involved. By contrast, our method features the efficient computation of derivatives thanks to the structure of the joint representation. Tändl and Kecskeméthy [2007] use the Frenet frame of spline curves in $\mathbb{R}^3$ for the dynamics simulation of simple mechanisms. The orientation of the Frenet frame is determined entirely by the spline curve, whereas the orientation of the frame in our model is independent of its position along the curve, which enables us to model arbitrary rotations along the motion curve. Moreover, we use C^2-continuous splines, whereas they must use C^4-continuous splines.

Our spline joint can easily be incorporated into current dynamics algorithms. Featherstone [1987] developed an articulated-body dynamics algorithm for multiple-DOF joints, such as the universal joint. One can use this dynamics algorithm without modification to simulate spline joints (see Appendix B).

CHAPTER 3

Biomechanical Modeling of the Neck-Head-Face Complex

The neck has a complex anatomical structure and plays the important role in supporting the head in balance while generating the controlled head movements that are essential to so many aspects of human behavior. In this chapter, we introduce a biomechanical model of the human head-neck system. Emulating the relevant anatomy, our model is characterized by appropriate kinematic redundancy (7 cervical vertebrae coupled by 3-DOF joints) and muscle actuator redundancy (72 neck muscles arranged in 3 muscle layers). Modeled as a uniaxial Hill-type muscle, each muscle actuator exerts a force on the skeleton. We will consider the control of this complex musculoskeletal system in the next chapter. In the current chapter, Section 3.1 details our biomechanical, musculoskeletal model, including the skeletal model and the structure of the muscle actuators and Section 3.2 explains the uniaxial Hill-type neck muscle model.

3.1 Musculoskeletal Model

Our musculoskeletal model comprises a model of the skeleton and a model of the muscles of the neck, which we will describe in turn.

Bone	**Mass**	k_s: **x,z-axes**	k_s: **y-axis**
Skull	3.5	50	25
C1–C7	0.21	50–70	25–35

Table 3.1: Physical parameters of the skeleton. The masses are in kilograms. The k_s quantities are in $\mathrm{N \cdot m/rad}$. The k_d are set to 10% of the corresponding k_s. The y axis is in the vertical direction.

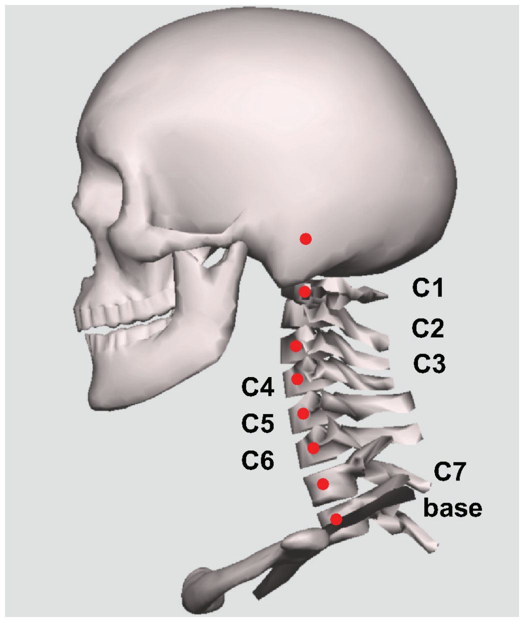
3.1.1 Musculoskeletal Model

The relevant skeletal structure is modeled as an articulated multibody system. It includes a base link, seven cervical bones, C1–C7, and a skull, as shown in Figure 3.1(a). In the human spine, disks are sandwiched between adjacent vertebrae, allowing 6-DOF motion. By carefully locating pivot points as in [Kapandji 1974], we simplified each joint to a 3-DOF rotational joint. To each joint angle, we attach a rotational damped spring in order to model the stiffness of the ligaments and disks, as follows: $\tau_s = -k_s(q - q_0) - k_d\dot{q}$, where q is the joint angle, q_0 is the joint angle in the natural, rest configuration, k_s is the spring stiffness, and k_d is the damping coefficient. The linear damping increases the stability of the system. Table 3.1 specifies the physical parameters of the skeleton.

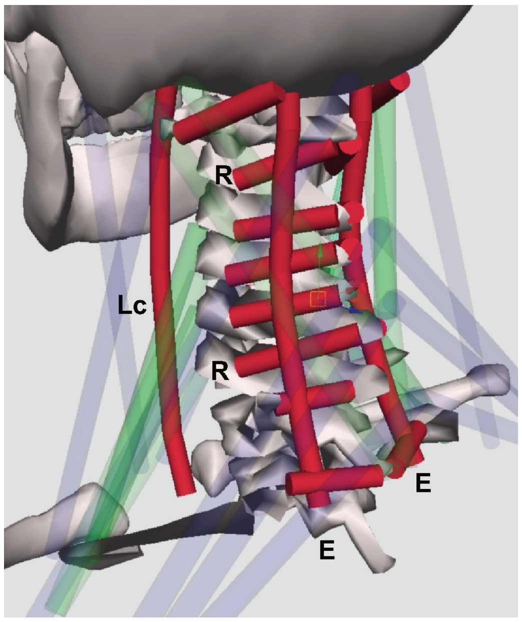
The equations of motions of the skeletal system are

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{K}_s\mathbf{q} + \mathbf{K}_d\dot{\mathbf{q}} - \mathbf{P}(\mathbf{q})\mathbf{f}_P = \mathbf{P}(\mathbf{q})\mathbf{f}_C + \mathbf{J}(\mathbf{q})^T\mathbf{f}_e, \tag{3.1}$$

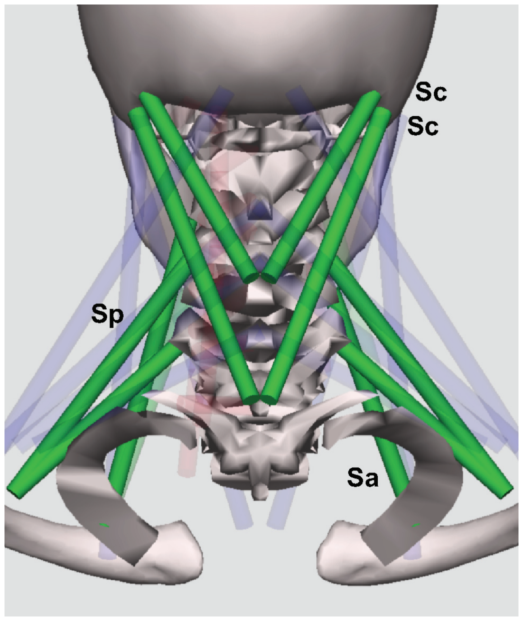
where $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are 24-dimensional vectors containing all the joint angles (generalized coordinates), the angular velocities, and the angular accelerations, respectively. Since our muscle model is massless and purely force-based, the mass of the head is incorporated into the skull and the mass of the neck is distributed among the cervical vertebrae. $\mathbf{M}(\mathbf{q})$ denotes the inertia matrix of the skeleton. The vector $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ rep-



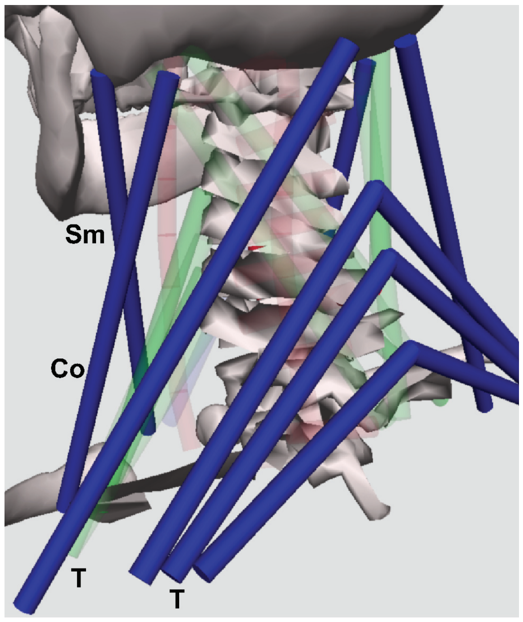
(a) Skeleton model.



(b) Deep muscles.



(c) Intermediate muscles.



(d) Superficial muscles.

Figure 3.1: The musculoskeletal model. (a) The red dots represent the pivots of the eight joints of the cervical column. The pivots of vertebra C2 to C7 are in their supporting bones. Geometric mesh data were acquired from www.3dcafe.com. The deep muscle layer (b), intermediate muscle layer (c), and superficial muscle layer (d) of the neck are shown. Table 3.2 details the muscles and attachments.

resents the Coriolis forces, centrifugal forces, and gravity. The diagonal stiffness $\mathbf{K}_s$ and damping $\mathbf{K}_d$ matrices are due to the aforementioned rotational springs. Since the equations of motion (3.1) are expressed in joint space, $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix that transforms the external force $\mathbf{f}_e$ into joint torques. The muscle forces are divided into passive, elastic forces $\mathbf{f}_P$ produced by the muscles' material properties as they are stretched, and active, contractile forces $\mathbf{f}_C$ generated by the muscles in response to the neural control signal. $\mathbf{P}(\mathbf{q})$ is the moment arm matrix (Section 3.1.1.1). We compute $\ddot{\mathbf{q}}$ in (3.1) using Featherstone's dynamics algorithm and numerically integrate through time to obtain $\dot{\mathbf{q}}$ and $\mathbf{q}$ using the explicit Euler method.

3.1.1.1 Computation of the Moment Arm Matrix

The moment arm matrix $\mathbf{P}(\mathbf{q})$ maps muscle forces to joint torques, i.e., $\tau = \mathbf{P}(\mathbf{q})\mathbf{f}_C$, where $\tau = [\tau_1, \ldots, \tau_n]^T$ is the vector of joint torques created by $\mathbf{f}_C$ and n is the number of joints. The moment arm matrix is computed using the principle of virtual work [Delp and Loan 1995]. Let $\mathbf{l}_j$ be the vector from the origin to the insertion of muscle j. Let $\delta l_j = \langle \dot{\mathbf{l}}_j, \mathbf{l}_j / \|\mathbf{l}_j\| \rangle$ and $\delta \mathbf{l} = [\delta l_1, \cdots, \delta l_m]^T$, where m is the number of muscles. The principle of virtual work $\langle \mathbf{f}_C, \delta \mathbf{l} \rangle = \langle \tau, \delta \mathbf{q} \rangle$ yields the relation $\mathbf{P}(\mathbf{q})^T \delta \mathbf{q} = \delta \mathbf{l}$. If we set $\delta \mathbf{q}$ to be the i-th basis vector $\mathbf{e}_i$ in the joint space, then the resulting $\delta \mathbf{l}$ is the same as the i-th row of $\mathbf{P}$. Thus, we can compute $\mathbf{P}(\mathbf{q})$ as follows:

```
Require: $\mathbf{q}$
1: Update the transformation matrix of each bone
2: for $i = 1$ to $n$ do
3:     Set $\dot{\mathbf{q}} = \mathbf{e}_i$
4:     Compute generalized velocity of each transformation matrix
5:     Compute $\delta \mathbf{l}$ as defined above
```

```
6:  Set the $i$-th row of $\mathbf{P}$ to $\delta\mathbf{l}$
```

3.1.2 Muscular Structure

There are more than 20 types of muscles in the neck, and there are many muscles of each type. Individual muscles often have multiple origins and insertions. Since it would be difficult and computationally very costly to model all the muscles accurately, we were motivated to reduce the number of muscles modeled. In an effort to minimize the total number of actuators in the synthetic musculoskeletal system, we first attempted to model only the major superficial muscles of the neck. We discovered, however, that even though these muscles outnumbered the total number of degrees of freedom of the system, the system was uncontrollable, apparently because most of the major muscles span multiple bones. The solution was to dauntlessly emulate the considerable muscular redundancy of the target biological system.

Consulting references on anatomy [Warfel 1985; Kapandji 1974], we incorporated 72 individual muscles into the musculoskeletal model, as shown in Figure 3.1(b)–(d). The neck muscles are arranged in three layers—deep, intermediate, and superficial. In the deep layer (Figure 3.1(b)), there are a total of 48 muscles, which improve controllability. Six muscles are attached across each cervical joint, such that they cover the 3 DOFs of the joint. This increases, if not guarantees, controllability and affords greater freedom to model the major muscles of the intermediate and superficial layers, each of which include 12 muscles arranged as shown in Figure 3.1(c) and (d).

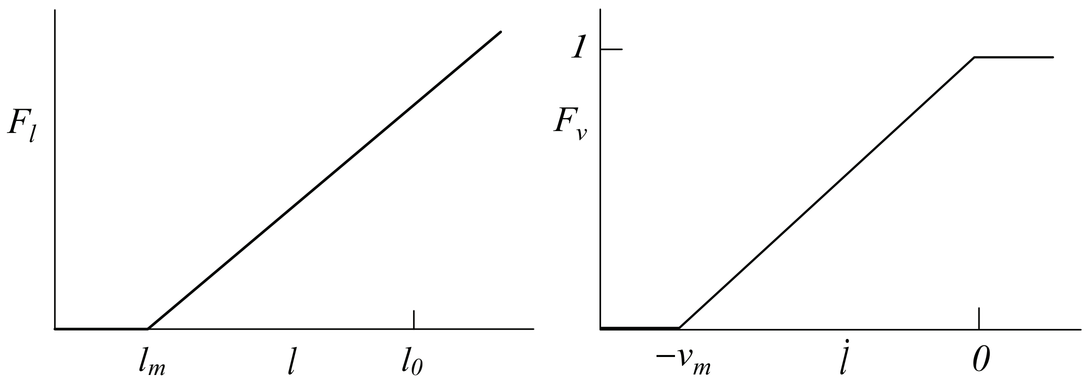
Notwithstanding the rather large number of modeled muscles, note that we have disregarded many of the muscles of the neck, such as the muscles attached to the hyoid bone, in an effort to simplify our model. Table 3.2 details the muscular structure of our biomechanical system.

Layer	Muscle	#m	Origin / Insertion	w
Deep	Longus colli (Lc)	16	adjacent vertebrae (anterior vertebral bodies)	1.0
	Erector (E)	16	adjacent vertebrae (behind transverse pro)	1.0
	Rotator (R)	16	adjacent vertebrae (transverse pro / spinous pro)	1.0
Inter-mediate	Scalenus anterior (Sa)	4	base (lateral) / C5 C3 (transverse pro)	2.0
	Scalenus posterior (Sp)	4	base (lateral) / C6 C4 (transverse pro)	2.0
	Splenius capitis (Sc)	4	C7 C5 (spinous pro) / skull (superior nuchal line)	2.0
Super-ficial	Sternomastoid (Sm)	2	base (sternum) / skull (mastoid pro)	3.0
	Cleidooccipital (Co)	2	base (clavicle) / skull (superior nuchal line)	3.0
	Trapezius (T)	8	base (posterior) / C6 C4 C2 (behind spinous pro) skull (external occipital prot)	3.0

Table 3.2: The subset of neck muscles that are modeled and their origins/insertions. Legend: number of muscles (#m); strength weight factor (w); process (pro); protuberance (prot).

3.2 Hill-Type Muscle Force Model

To model each muscle actuator, we employ a popular muscle model in biomechanics research, which is known as a Hill-type model. Good introductions to this model can be found elsewhere [Ng-Thow-Hing 2001; Winters and Crago 2000]. If we assume that the length of the tendon remains constant as the muscle is stretched, the muscle force comes from two sources: A parallel element (PE), which passively produces a restoring force f_P due to the material elasticity of the muscle, and a contractile element (CE), which actively generates a contractile force f_C in response to excitation from the motor neurons. The total muscle force is: $f_m = f_P + f_C$.



(a) Force-length relation (b) Force-velocity relation

Figure 3.2: Linearized Hill-type model.

The PE is modeled as a uniaxial exponential spring:

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

where k_s and k_c are *elastic coefficients*, k_d is the *damping coefficient*, $e = (l - l_0)/l_0$ is the *strain* of the muscle, with l and l_0 its *length* and *slack length*, respectively, and $\dot{e} = \dot{l}/l_0$ is the *strain rate* of the muscle. Since f_P is determined by the state of the musculoskeletal system rather than by its neural activation, it is not treated as a control input in (3.1).

The contractile force from the CE is typically expressed as

$$f_C = aF_l(l)F_v(\dot{l}), \tag{3.2}$$

where $0 \le a \le 1$ is the *activation level* of the muscle (i.e., the input signal from the motor neuron innervating the muscle). F_l denotes the *force-length relation* (i.e., the muscle force as a function of its length) and F_v denotes the *force-velocity relation* (i.e., the muscle force as a function of its shortening velocity).

We use a simple, linearized Hill-type model with F_l and F_v as shown in Figure 3.2. In particular, $F_l(l) = \max(0, k_{\max}(l - l_m))$, where $k_{\max}$ is the maximum stiffness of a fully activated muscle and l_m is the minimum length at which the muscle can produce force, and $F_v(\dot{l}) = \max(0, 1 + \min(\dot{l}, 0)/v_m)$, where v_m is the maximum contraction velocity under no load. Per [Ng-Thow-Hing 2001], we set $l_m = 0.5l_0$ and $v_m = 8l_0$ sec^{-1}. The coefficient k_c is set to 7 for all the muscles. The coefficients k_s, k_d, and $k_{\max}$ for each muscle are scaled by its *strength weight factor* w, which is set roughly proportional to the cross sectional area of the muscle. Table 3.2 specifies the strength weight factors and attachment sites of the muscles.

Note that the original Hill model includes a negative stiffness range as the muscle is stretched. This range is seldom reached in everyday movement (see Ch. 7 of [Winters and Crago 2000] and references therein). It is known that negative stiffness can destabilize musculoskeletal systems such as ours. We have avoided this by modifying the model. Even though our $F_l(l)$ increases monotonically (the same F_l was used in [Hogan 1984]), the difference relative to the original Hill model is modest, because the stretch of the neck muscles is limited by the constrained motions of the bones.

CHAPTER 4

Neuromuscular Control of the Neck-Head System

The anatomically consistent biomechanical head-neck model developed in the previous chapter confronts us with a challenging motor control problem, even for the relatively simple task of balancing the mass of the head in gravity atop the cervical spine. In this chapter, we develop a novel neuromuscular control model for human head animation that emulates the relevant biological motor control mechanisms. Incorporating low-level reflex and high-level voluntary sub-controllers, our hierarchical controller provides input motor signals to the numerous muscle actuators. In addition to head pose and movement, it regulates the stiffness of the head-neck multibody system by controlling the tone of mutually opposed neck muscles. Neural networks within our neuromuscular controller are trained offline to efficiently generate the online pose and tone control signals necessary to synthesize a variety of autonomous movements for the behavioral animation of the human head and face.

The following sections provide a functional overview of our face-head-neck animation system (Section 4.1) and develop our hierarchical, neuromuscular control framework, comprising the reflex controller (Section 4.2) and the voluntary controller (Section 4.3) plus its associated control learning algorithms, as well as present our experiments and results (Section 4.4).

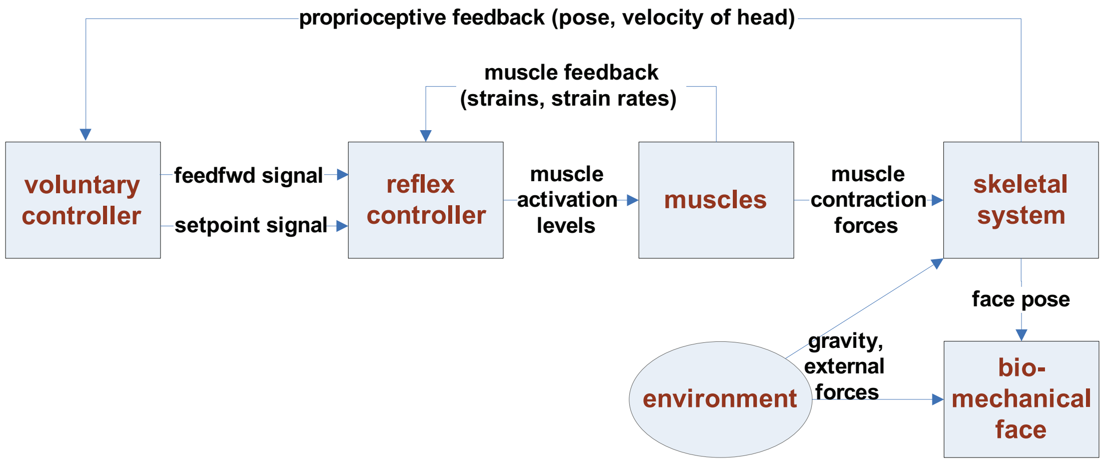


Figure 4.1: Face-Head-Neck System Architecture.

4.1 System Overview

Figure 4.1 shows the overall architecture of our head-neck system model, which comprises the skeleton, muscles, and hierarchical controller. The higher-level voluntary sub-controller generates a feedforward control signal and a setpoint control signal. The dynamic feedforward signal is generated to attain the desired pose and tone. The kinematic setpoint signal specifies the desired strain and strain rate of each muscle, as well as the magnitude of the feedback gain. Comparing the strain and strain rate against their desired values, the lower-level reflex controller generates a feedback signal and adds it to the feedforward signal, thus determining the activation level of each muscle. Given an input activation signal, each muscle generates a contraction force depending on its length and velocity. Finally, the skeleton produces articulated motion in response to the internal muscle forces and external environmental forces, such as gravity and applied forces. Physics-based animation is achieved by numerically integrating the equations of motion of the biomechanical model through time.

Since the output signal from the voluntary controller normally changes more slowly than that of the reflex controller, we run the two controllers at different speeds. The hierarchical structure offers a practical advantage in view of the fact that the computational cost of the voluntary controller is significantly higher than that of the reflex controller. In our system, the voluntary controller updates every 40 milliseconds whereas the reflex controller updates once per integration time step; i.e., approximately every millisecond. Including the control computations, our simulation runs about 10 times slower than real time on a PC with a 3.2 GHz Mobile Intel Pentium 4 CPU and 1 GB of RAM.

Although this chapter does not dwell on facial animation, we have augmented the realism of our biomechanical head-neck model for the demonstrations that we present in Section 4.4 by coupling a biomechanical face model (the lower right box in Figure 4.1) to the front of the skull as shown in Figure 1.1. This expressive, behaviorally-capable face model [Terzopoulos and Lee 2004] is an improved version of the second-generation biomechanical model reported in [Lee et al. 1995]. Conceptually, the face model decomposes hierarchically into several levels of abstraction related to the (FACS) control of facial expression, the anatomy of facial muscle structures, the histology and biomechanics of facial tissues, as well as facial geometry and appearance. Like our biomechanical model of the neck, the face model is muscle-driven. Its 44 facial muscles are arranged in an anatomically consistent manner within the bottom layer of a synthetic facial soft tissue. The tissue is modeled as a lattice of uniaxial viscoelastic units assembled into multilayered prismatic elements with epidermal, dermal, sub-cutaneous fatty tissue, fascia, and muscle layers. The elements enforce volume preservation constraints and model contact response against the bone substrate. Expressive facial tissue deformations are animated by numerically simulating the physical response of the element assembly to the stresses induced by appropriately coordinated facial muscle contractions. The face simulation runs at real-time, interactive rates on the aforementioned PC.

4.2 Reflex Control

The reflex controller generates a neural activation level a for each muscle by summing the feedforward signal a_f generated by the voluntary controller with an internally-generated feedback signal a_b that is computed by comparing the strain and strain rate of each muscle with their desired values. In terms of its biological basis, our reflex controller emulates the stretch reflex in human motor control, which is believed to be modulated by the gamma motor neural signal and is activated when the muscle is elongated beyond the desired length [Kandel et al. 2000]. The length and velocity of the muscle are measured by its proprioceptive sensory organs, among them the spindles inside the muscle.

Our reflex control model is as follows:

$$\begin{aligned} a_b &= s\left(k_p(e-e_d)+k_v\,\mathrm{sat}_m(\dot{e}-\dot{e}_d)\right), \\ a &= \min(1,\max(0,a_f+a_b)), \end{aligned} \tag{4.1}$$

where k_p and k_v are proportional and derivative gains, s is the feedback gain scaling factor, and e and $\dot{e}$ are the muscle's strain and strain rate, respectively (given in Section 3.2). Note that s along with the desired strain e_d and desired strain rate $\dot{e}_d$ are determined by the setpoint signal generated by the voluntary controller. In our experience, a large derivative feedback force overwhelms the proportional feedback force and tends to make the system unstable, so we employ the function

$$\mathrm{s}at_m(x) = \begin{cases} x & \text{if } |x| < m, \\ m\ \mathrm{s}gn(x) & \text{otherwise,} \end{cases}$$

which saturates its input at the value m (we set m to 2.0). With this saturated derivative feedback, we found that we can use a reasonable derivative gain $k_v = 0.05$ relative to the proportional gain $k_p = 8$ without having to decrease the integration time step.

4.3 Voluntary Control and Learning

A distinctive feature of human motor control is that one can increase the stiffness or tone of the body by coactivating opposing (agonist and antagonist) muscles. Humans are known to use coactivation to increase their stability when subjected to external disturbances or to improve accuracy when performing certain difficult motor control tasks. From the mechanical perspective, higher tone can be advantageous, because it increases the stiffness of the musculoskeletal system, thus improving robustness against perturbation, though at the cost of increased energy consumption. However, the issue of tone control has been more or less neglected in animation research [Neff and Fiume 2002]. Biomechanics researchers have suggested that humans can independently control the coactivation and movement [Yamazaki et al. 1994]. To emulate this feature of human motor control, we have designed our voluntary controller to be capable of controlling the pose and the tone of the neck independently.

In our voluntary controller (Figure 4.2), the pose signal $\mathbf{a}_p$ and tone signal $\mathbf{a}_t$ are independently generated by two neural networks, and the feedforward signal is obtained

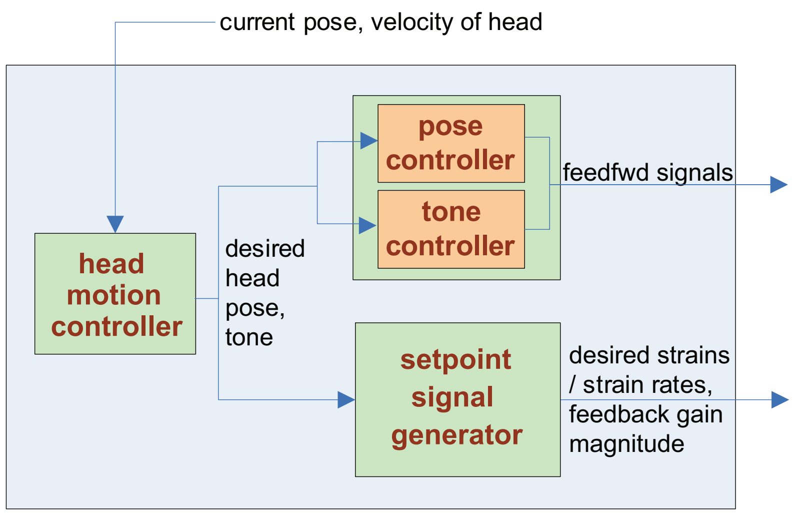

current pose, velocity of head
pose controller
tone controller
feedfwd signals
head motion controller
desired head pose, tone
setpoint signal generator
desired strains / strain rates, feedback gain magnitude

Figure 4.2: The sub-controllers in the voluntary controller.

by summing the two signals:

$$\mathbf{a}_f = \mathbf{a}_p + \mathbf{a}_t.$$

This separation is possible because the tone signal is computed to be orthogonal to the pose signal, in the sense that the tone signal does not affect the pose of the system.

Another distinctive feature is that through trial and error, humans and other animals are able to learn how to control their muscles in order to move effectively and efficiently. This can be regarded as an optimization process that solves for the necessary neural input to the muscles required to achieve a desired motion [Grzeszczuk and Terzopoulos 1995]. Throughout this incremental learning process, the brain generates increasingly more appropriate motor signals to accomplish the desired motion and it becomes decreasingly dependent on feedback control. From the robotics perspective, this feedforward signal enables the animal to use lower feedback gains, which enhances the naturalness of the motion, among other factors. Similarly, the voluntary controller in our system generates its feedforward signal through machine learning. In particular, we solve offline for optimal neural inputs that achieve sampled target poses and tones, and use them to train neural network controllers to efficiently output optimal solutions online [Grzeszczuk et al. 1998].

4.3.1 Neural Networks

Since the computational structure of artificial neural networks is based on insights into biological nervous systems, we employ them in our pose and tone controllers. Moreover, the well-known function approximating ability of neural networks is attractive and compatible with our training strategy. Our offline learning process generates sample input-output training pairs by solving repeated optimization problems, as we will explain in the subsequent two sections, and then it trains neural networks on numerous such precomputed pairs, thus obtaining a suitable function approximator. It takes less than 10 hours to train each neural network on our 3.2 GHz CPU PC. Once trained, the neural network can approximate suitable outputs for particular inputs orders of magnitude faster than one can hope to do by solving the associated optimization problem. This makes the trained neural network suitable for online use, especially for interactive animation.

Figure 4.3 shows the fully connected, feedforward neural network that we employed for our pose and tone controllers. The inputs to the neural network are the normalized three components of the quaternion coordinate **h** (orientation) of the head. Each neuron is modeled as a sigmoid function, $y = \tanh(b + \sum_{i=1}^{k} w_i x_i)$, where b is a bias term and the w_i are the weights of the inputs x_i from the k neurons in the previous layer. The output of the neural network is the normalized pose signal a_p (or tone signal a_t). The dimension of the network output vector is 72, the total number of muscles. We use a 3-layer network with two hidden layers of sizes 20 and 40 neurons. The trainable parameters of the network are the weights and bias terms associated with the

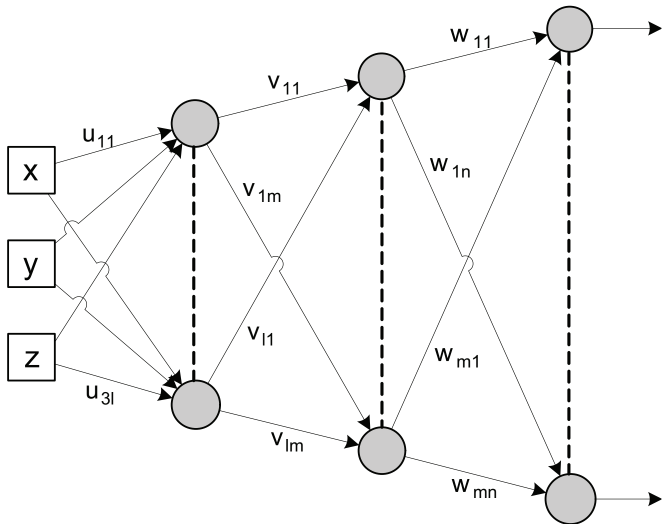


Figure 4.3: A 3-layer neural network.

neurons, and they are computed using the backpropagation learning algorithm, as in [Grzeszczuk et al. 1998]. Although free and commercial neural network packages are available, we used our own simple implementation.

4.3.1.1 Pose Controller

To train the pose controller neural network, we randomly sample the head pose space. For the *i*-th sample pose $\mathbf{h}_d^i$, the desired pose signal $\mathbf{a}_p^i$ is the solution of the constrained optimization problem

$$\mathbf{a}_p^i = \operatorname*{argmin}_{\mathbf{a}} \|\mathbf{f}_C^w\|^2 \tag{4.2}$$

$$\text{subject to} \quad \mathbf{c}(\mathbf{q}_d^i, \mathbf{0}) + \mathbf{K}_s\mathbf{q}_d^i - \mathbf{P}(\mathbf{q}_d^i)\mathbf{f}_p = \mathbf{P}(\mathbf{q}_d^i)\mathbf{f}_C, \quad \mathbf{a} \in [0,1]^m.$$

Eq. (4.2) minimizes weighted muscle contraction forces $\mathbf{f}_C^w = \mathbf{W}^{-1}\mathbf{f}_C$, where $\mathbf{W} = \mathrm{d}iag(w_1, \ldots, w_m)$ for the m muscles. The strength weight factors w_i (see Table 3.2) encourage muscle forces in proportion to muscle strengths. The primary constraint in the minimization stems from (3.1) with $\ddot{\mathbf{q}} = \dot{\mathbf{q}} = \mathbf{0}$ (to maintain $\mathbf{h}_d^i$ statically), $\mathbf{f}_e = \mathbf{0}$ (no external forces other than gravity), and with the joint angles $\mathbf{q} = \mathbf{q}_d^i$ provided by the setpoint signal generator to yield the desired $\mathbf{h}_d^i$ (i.e., $\mathbf{h}_d^i = \mathbf{g}(\mathbf{q}_d^i)$, where $\mathbf{g}(\cdot)$ is the forward kinematics function). To solve (4.2), we use DONLP2 [Spellucci], which is based on the sequential equality constrained quadratic programming method. On the order of $20,000 \approx N$ training pairs $\{\mathbf{h}_d^i, \mathbf{a}_p^i\}_{i=1}^N$ are generated offline to train $\mathbf{n}_p$ using backpropagation.

Given a desired head pose $\mathbf{h}_d$, the trained pose controller network efficiently computes a feedforward signal online to maintain $\mathbf{h}_d$ with minimal muscle contraction forces $\mathbf{f}_C$:

$$\mathbf{a}_p = \mathbf{n}_p(\mathbf{h}_d).$$

Given the form of the objective function, $\mathbf{n}_p$ cannot coactivate opposing muscles to increase musculoskeletal stiffness.

4.3.1.2 Tone Controller

Due to muscle redundancy, there are usually many combinations of muscle coactivations that can increase tone. It remains an open research problem as to how humans choose opposing muscle coactivations. Instead of formulating some explicit stiffness criterion that the musculoskeletal system should maximize, our intuitive assumption is that to achieve maximum stiffness one maximizes the muscle contraction forces while not actuating the musculoskeletal system. Similarly to $\mathbf{n}_p$ above, the tone neural network $\mathbf{n}_t$ is trained offline with on the order of $20,000 \approx N$ training pairs $\{\mathbf{h}_d, \mathbf{a}_t\}_{i=1}^N$, where the maximum tone signal $\mathbf{a}_t^i$ is obtained by solving the constrained optimization problem

$$\mathbf{a}_t^i = \underset{\mathbf{a}}{\operatorname{argmax}} \|\mathbf{f}_C^w\|^2 \quad \text{subject to} \quad \mathbf{P}(\mathbf{q}_d^i)\mathbf{f}_C = \mathbf{0}, \quad \mathbf{a} \in [0,1]^m.$$

Given a desired head orientation $\mathbf{h}_d$ and *tone parameter c*, the tone signal is computed online using the trained network $\mathbf{n}_t$ as

$$\mathbf{a}_t = c\,\mathbf{n}_t(\mathbf{h}_d). \tag{4.3}$$

Since we should have $a_f = a_p + a_t \leq 1$, then $0 \leq c \leq 1 - \max(a_p)$.

It may at first seem surprising that arbitrary tone can be achieved by simply scaling the output of $\mathbf{n}_t$. However, this is to be expected because the resulting muscle force $\mathbf{f}_C$ is constrained to lie in the null space of $\mathbf{P}(\mathbf{q})$, thus it does not contribute to the generalized force τ. Furthermore, this is possible because the muscle force and the neural signal are linear in the Hill-type model (3.2); hence, scaling the neural signal retains the muscle force in the null space of $\mathbf{P}(\mathbf{q})$. Note that, aside from c, the tone signal $\mathbf{a}_t$ depends only on the configuration of the system $\mathbf{q}_d$. It is not affected by the external force field (gravity) or by the global orientation of the system, whereas the pose control signal does have such dependencies.

4.3.2 Setpoint Signal Generator

Given a desired head pose $\mathbf{h}_d$, the setpoint signal generator computes the desired strain $\mathbf{e}_d$ and strain rate $\dot{\mathbf{e}}_d$ of each muscle. The former is given by

$$\mathbf{e}_d = \mathbf{n}_g(\mathbf{h}_d).$$

Unlike the pose and tone controllers, we do not implement the function $\mathbf{n}_g$ as a neural network. Rather, it entails the solution of the constrained optimization problem

$$\mathbf{q}_d = \underset{\mathbf{q}}{\operatorname{argmin}} \|\mathbf{q}^v\|^2 \quad \text{subject to} \quad \mathbf{h}_d = \mathbf{g}(\mathbf{q}), \tag{4.4}$$

where $\mathbf{q}^v = \mathbf{V}^{-1}\mathbf{q}$ with $\mathbf{V} = \mathrm{d}iag(v_1, \ldots, v_n)$ and n the number of joints. Here, v_i is the weighting factor of joint q_i, which we set to the range of the joint in accordance with [Hay and Reid 1988], and $\mathbf{g}(\cdot)$ is the forward kinematics function. Having computed $\mathbf{q}_d$ (i.e., the smallest joint angles that achieve $\mathbf{h}_d$), we then obtain $\mathbf{e}_d$ from $\mathbf{g}(\mathbf{q}_d)$.

Finally, we compute the desired strain rate as

$$\dot{\mathbf{e}}_d = \frac{\mathbf{n}_g(\mathbf{h}_d(t+\Delta t)) - \mathbf{n}_g(\mathbf{h}_d(t))}{\Delta t},$$

where $\mathbf{h}_d(t)$ and $\mathbf{h}_d(t+\Delta t)$ are the desired orientation of the head at time t and at the subsequent time step $t+\Delta t$, respectively.

Although simple, the objective in (4.4) yields natural looking results. We did not implement the setpoint signal generator as a neural network for several practical reasons. First, due to its simplicity, (4.4) can be solved faster online than by using a neural network. We solve (4.4) using the gradient descent method, which typically achieves the solution within 3 iterations. Second, this direct computation yields an accurate result, whereas a neural network would incur some error. The error issue is potentially crucial here, as the setpoint signal serves as a reference signal for feedback control in the reflex controller.

4.3.3 Head Motion Controller

At the topmost level of our control hierarchy is a voluntary controller that produces movements which take the head from a current orientation to a desired new orientation. It does its job by providing a series of commands to the neck feedforward and setpoint signal generators to modify the pose/tone of the biomechanical system. We will discuss two approaches next.

Interpolation: Given quaternion representations of initial $\mathbf{h}_i(t_i)$ and desired final $\mathbf{h}_f(t_f)$ orientations of the head, a natural trajectory $\mathbf{h}_d(t)$ from $t_i \leq t \leq t_f$ may be computed as the spherical linear interpolation $\mathbf{h}_d(t) = \mathrm{slerp}(r(t), \mathbf{h}_i, \mathbf{h}_f)$. The interpolation parameter $r(t)$ is determined so that the time derivative of r is bell shaped; i.e., $\dot{r}(t_n) = 1 - \cos(2\pi t_n)$, where $t_n = (t - t_i)/(t_f - t_i)$. The head motion controller also modulates the tone c in (4.3) and feedback gain scaling factor s in (4.2) by comparing the actual and desired orientations of the head. If the total accumulated error over a time window exceeds a threshold, the controller increases the tone and feedback gain gradually until the error falls below threshold. By decreasing the error threshold, the neck maintains the pose better and is stiffer. Conversely, by increasing the error threshold, the neck produces more relaxed motion and allows greater perturbation during the movement.

Sensorimotor Control: Although the interpolation generator produces reasonable head-neck motion for the purposes of character animation, an approach that is more consistent with biological control mechanisms is sensorimotor control. At every command-generating instant of the voluntary controller, a desired head orientation and velocity command are generated on the fly based on sensory feedback. For example, given initial $\mathbf{h}_i(t_i)$ and desired final $\mathbf{h}_f(t_f)$ orientations of the head, the sensorimotor controller initiates a head movement towards $\mathbf{h}_f(t_f)$. The inertia of the head yields a natural angular acceleration. During movement, the instantaneous head angle error $\|\mathbf{h}_f(t_f) - \mathbf{h}(t)\|$ is sensed at a fast rate and corrective "steering" is applied to continually reduce the error. When the error decreases to below some threshold, the sensorimotor controller begins to slow the head so that it comes to rest in pose $\mathbf{h}_f(t_f)$.

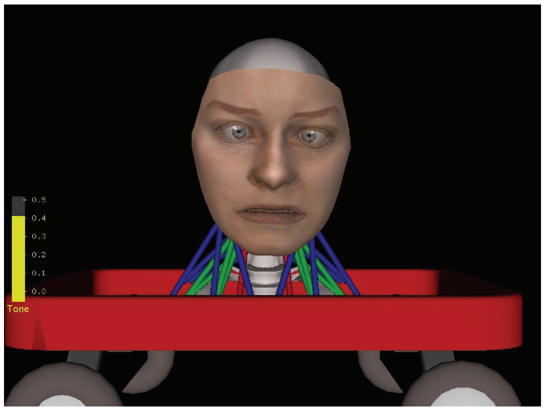
4.4 Experiments and Results

We have conducted several experiments with our biomechanical, neuromuscular face-head-neck animation system, ranging from the investigation of stiffness control to the creation of behavioral animation.

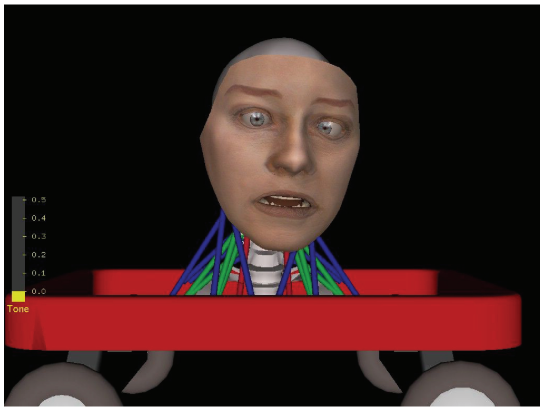
4.4.1 Basic Simulations

Even with the rotational springs (which represent ligaments and disks) attached to each cervical joint, the skeletal system appropriately collapses in gravity, exhibiting the expected passive dynamics. Without active control, the complete musculoskeletal system appropriately collapses as well, albeit in a more damped manner. However, simulating the passive dynamics of the musculoskeletal system was crucial for adjusting the parameters of the 72 muscles. Since each muscle's stiffness and damping parameters are not known precisely and, even if they were, since we cannot model all of the muscles in the neck (thus our actuators must also assume the roles of neighboring unmodeled muscles), we cannot naively use empirical data reported in the biomechanics literature. Hence, we tuned the muscle parameters in our model by visually assessing the plausibility of the resulting passive dynamics.

With the feedforward and feedback control networks trained, we ascertained the im-



(a) Both tone *c* and feedback gains *s* are modulated.



(b) No tone control; only feedback gains *s* are modulated.

Figure 4.4: Different head motions result depending on the tone control. Snapshots (a) and (b) are taken at the same time with identical perturbations of the red wagon.

portance of feedforward control by turning it off and attempting to animate the head using only feedback control. With the feedback gain set at its nominal value, feedback control alone fails to maintain the upright stance of the cervical spine with the head in balance. However, feedback control is important for maintaining the stability of the musculoskeletal system.

4.4.2 Stiffness Control Experiments

In a different experimental scenario, we apply perturbations to the base link of the head-neck system that are analogous to riding on a vehicle over a bumpy road (Figure 4.4). As the head motion controller senses excessive error between the desired and the actual orientation of the head, it gradually increases the feedback gain *s* (to its maximum value of 3.0) and tone *c* (to its maximum value of 0.4) until the error drops below an acceptable threshold or until the maxima are reached. Not surprisingly, the head wobbles less when both the tone and feedback gain are increased, compared to increasing the feedback gain alone. However, we also observed that increasing the tone alone is insufficient to suppress the wobble. This implies that reflexive stiffness also plays an important role in the overall stiffness of the musculoskeletal system. Appendix A discusses reflexive stiffness and intrinsic stiffness.

In a second set of perturbation experiments, we apply with a ball external impacts to the head under various tone conditions (Figure 4.5). After impact, the head motion controller issues head stopping commands to the lower-level neuromuscular controllers; i.e., set the desired pose to the current pose and the desired velocity to zero. When the head approaches stationariness, the controller issues a command for the head to return to its original upright pose. Since the stiffness of the musculoskeletal system is greater when it increases its tone by coactivating opposing muscles, it is less perturbed by the same impact. This illustrates the fact that even passive human motion differs markedly depending on the internal state of muscle activation.

4.4.3 Tracking Head Motion Capture Data

Section 4.3.3 discussed interpolation and sensorimotor voluntary controllers. Figure 4.6 shows an example of our sensorimotor controller tracking head motion capture data from the CMU database (mocap.cs.cmu.edu—subject #79, motion #83 (shaving)). The sequence of head orientations from the motion capture data are set as target head orientations to the head controller. The head controller computes the desired head angular velocity as $\dot{x}_d(t) = (x_d(t+d) - x(t))/d$, and the desired orientation as $x_d(t+\Delta t) = x(t) + \dot{x}_d(t)\Delta t$, where $x(t)$ is the angular representation of the head orientation at time t, and d is the time in which the system is allowed to reach the target $x_d(t+d)$. In this example, we set $d = 10\Delta t$ with Δt = 0.033 sec. Figure 4.6 reveals that our dynamic head-neck system follows the motion capture data while smoothing noise in the data.

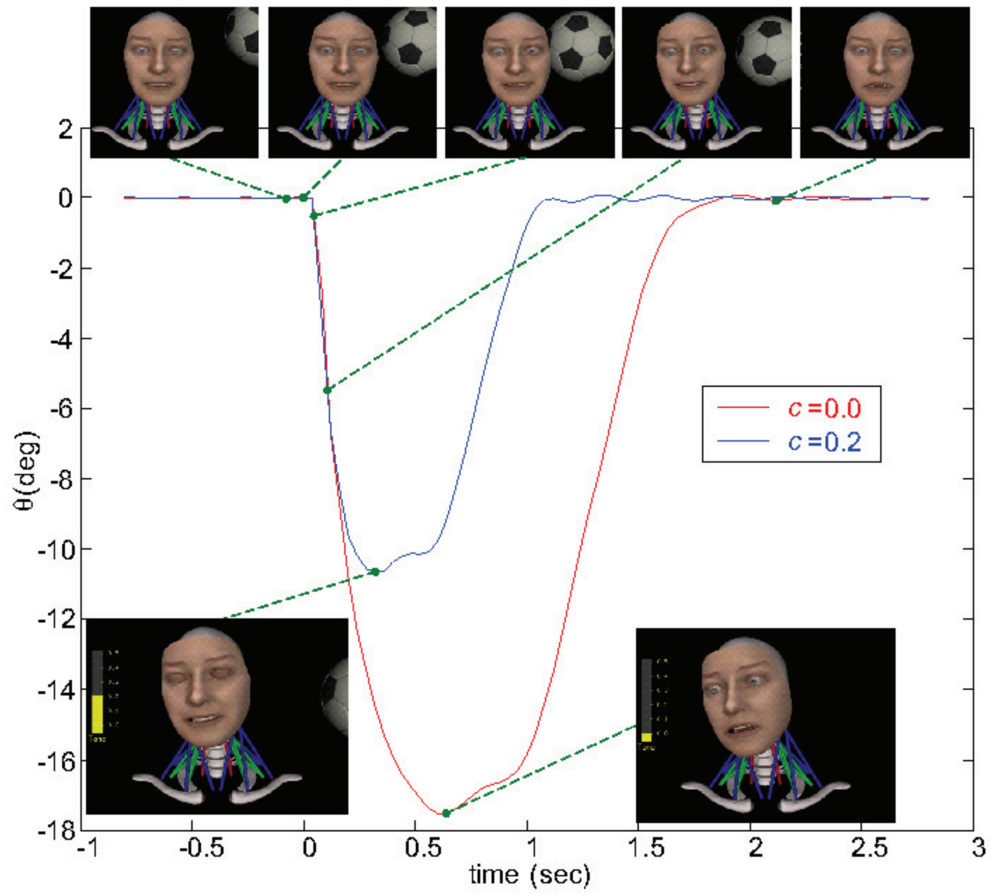


Figure 4.5: Head orientation longitudinal angle θ over time during an impact simulation. When controlled with zero tone signal, $c = 0$, (red), the head is perturbed more by the impact with the ball than when controlled with tone $c = 0.2$ (blue). All snapshots except for the lower left one are sampled from the zero tone (red) case.

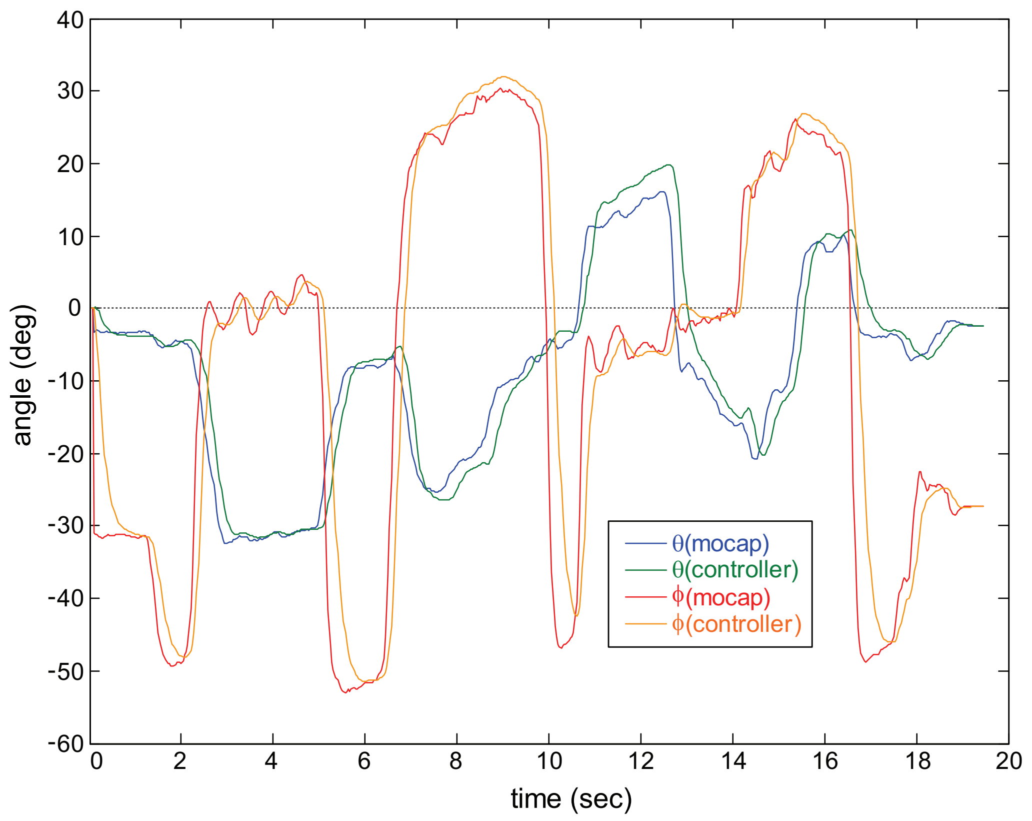


Figure 4.6: An example of the sensorimotor controller following head motion capture data. The time plots compare the longitudinal θ and latitudinal ϕ angles of the synthetic head (controller) and real head (mocap).

4.4.4 Gaze Behavior

Human vision is foveated. The foveal region of the retina, which spans roughly 5 degrees of visual arc, is specialized for high-acuity, color vision. To see an object clearly, gaze-shifting eye movements are usually needed to direct the eye to the visual target. Since the resulting eye motion disrupts vision, these movements are executed as quickly as possible and are called *saccadic* eye movements. As a visual target moves closer, the two eyes must also converge onto the target; these are called *vergence* eye movements. The oculomotor system, which positions the eyes relative to the head, and

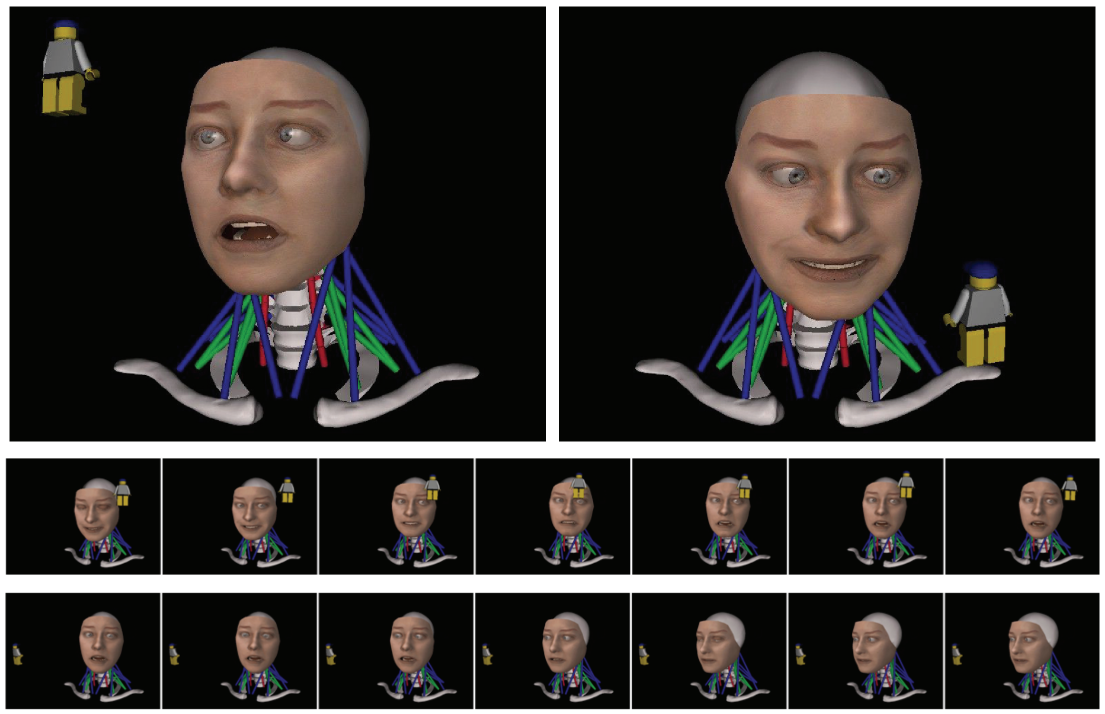


Figure 4.7: Head-Eye gaze behavior. (Top) snapshots of the model gazing at a visual target (the doll) in different directions and reacting to the location and movement of the target with different facial expressions. (Bottom) two sequences illustrating typical head-eye movements to gaze at the visual target.

its interaction with head movement has been the subject of intense research (see, e.g., [Carpenter 1988]).

Given the significantly greater mass of the head relative to the eye, head dynamics are much more sluggish than eye dynamics. For example, in a voluntary head-eye movement to direct the gaze at an off-axis visual target in the horizontal plane, the eye movement is an initial high-speed saccade in the direction of the head movement, presumably to facilitate rapid search and visual target localization, followed by a slower return to orbital center, compensating for the more sluggish head movement that follows.

As Figure 4.7 shows, our biomechanical model can synthesize coordinated head-eye movements that emulate at least the primary head-eye movement phenomena reported in the literature. When we present a moving visual target (the doll) to the model, the eyes are directed to make a saccadic ocular rotation (with maximum angular velocity of 200 degrees/sec) to point in the direction of the visual target relative to the head. Simultaneously, the head motion sub-controller of the neck neuromuscular controller issues a high-level command to rotate the head in the direction of the gaze. As the head executes the desired rotation via the low-level physical simulation, the eyes make a continuous compensatory movement such that they remain directed at the visual target. Figure 4.7 (top) shows the head gazing at the target in two different directions. Employing a rule-based behavior routine, the biomechanical face automatically synthesizes baby-like facial expressions as the eyes and head track the target. It appears awed when the doll is situated above the head, pleased when the doll is around eye level and held still, and angry when the doll is shaken.

4.4.5 Autonomous Multi-Head Interaction

Figure 4.8 illustrates three autonomous face-head-neck systems interacting in a multi-way behavioral facial animation scenario, which was inspired by a more primitive demonstration in [Terzopoulos and Lee 2004] not involving neck models. In our version, each of the faces is supported by our head-neck musculoskeletal system, which automatically synthesizes all of the head motions necessary to sustain a highly dynamic multi-way interaction. As in the above demonstration, the synthesized head movements must cooperate with eye movements in order to direct the gaze at visual targets in a natural manner. The middle head in the figure acts as a “leader” synthesizing random expressions and alternating its attention between the other two heads, which act as “followers”. Once a follower has the leader’s attention, the follower will observe the leader’s expression and engage in expression mimicking behavior. How-

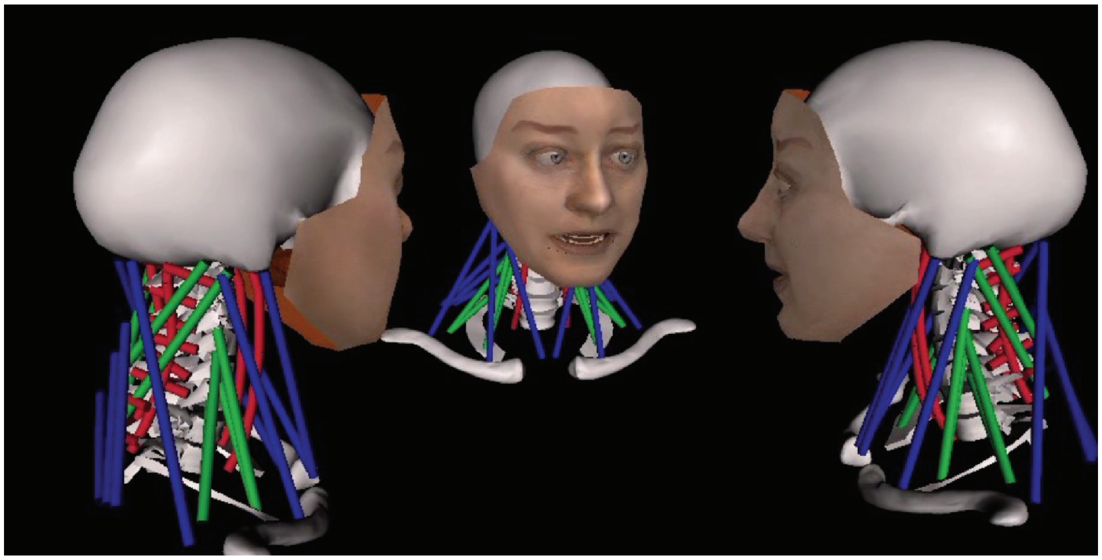


Figure 4.8: Autonomous behavioral-based interaction between three face-head-neck systems.

ever, excessive mimicking will lead to behavior fatigue—the follower will lose interest in the leader and attend to its fellow follower. A complete explanation of the behavioral modeling is beyond the scope of this chapter; reference [Terzopoulos and Lee 2004] provides additional details.

CHAPTER 5

Comprehensive Biomechanical Modeling and Control of the Body

In this chapter, we present a comprehensive biomechanical model of the human body, confronting the combined challenge of modeling and controlling more or less all of the relevant articular bones and muscles, as well as simulating the physics-based deformations and bulging of the soft tissues. In particular, our dynamic skeletal model comprises 75 bones and 165 articular degrees of freedom, including those of each vertebra and most of the ribs. To be properly actuated and controlled, the skeletal model requires comparable attention to detail with respect to muscle modeling. We incorporate a full complement of actuators, a total of 846 muscles, each of which is modeled as a piecewise line segment Hill-type force actuator. We present the details of our musculoskeleton model in Section 5.1. To simulate the biomechanics of the active muscular tissues and passive soft tissues, we also apply a coupled finite element model with the appropriate constitutive behavior, in which are embedded the detailed 3D anatomical geometries of skin, muscle, and bone, as described in Section 5.2. In Section 5.3, we develop an associated physics-based animation controller that computes accelerations to drive the elaborate musculoskeletal system toward a sequence of target key poses set by an animator, and then computes the required activation signal for each muscle actuator through (hybrid) inverse dynamics. Finally, Section 5.4 presents our experiments

Head/Neck
Cervical vertebrae (7×3), Skull (3), Mandible (1), Hyoid (1), Thyroid (1)
Trunk
Pelvis (6), Lumbar vertebrae (5×3), Thoracic vertebrae (12×3), Sternum (2), Ribs (19×1), Costal cartilages (10×3), Clavicle-Scapulas (2×2)
Arm
Humerus (3), Ulnar (1), Radius (1), Hand (2)
Leg
Femor (3), Tibia-Fibula-Patella (1), Foot (2)

Table 5.1: List of the bones modeled in the body. A total of 75 bones with 165 DOFs are modeled. The numbers in parentheses indicate the DOFs of each bone.

and results.

5.1 Musculoskeletal Modeling

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

5.1.1 Skeleton

The skeleton is modeled as an articulated, multi-body system. As shown in Table 5.1 and Figure 5.1, we individually modeled most of the articular bones in human body (except the fingers). The system has a total of 75 bones with 165 DOFs. Among these, 139 DOFs are associated with the head-neck-trunk region (Figure 5.2). In particular, all the vertebrae in the lumbar, thoractic, and cervical regions are modeled as individual rigid bodies interconnected with 3-DOF joints. The first 10 ribs are modeled to rotate independently from the spine along axes running through costotransverse and

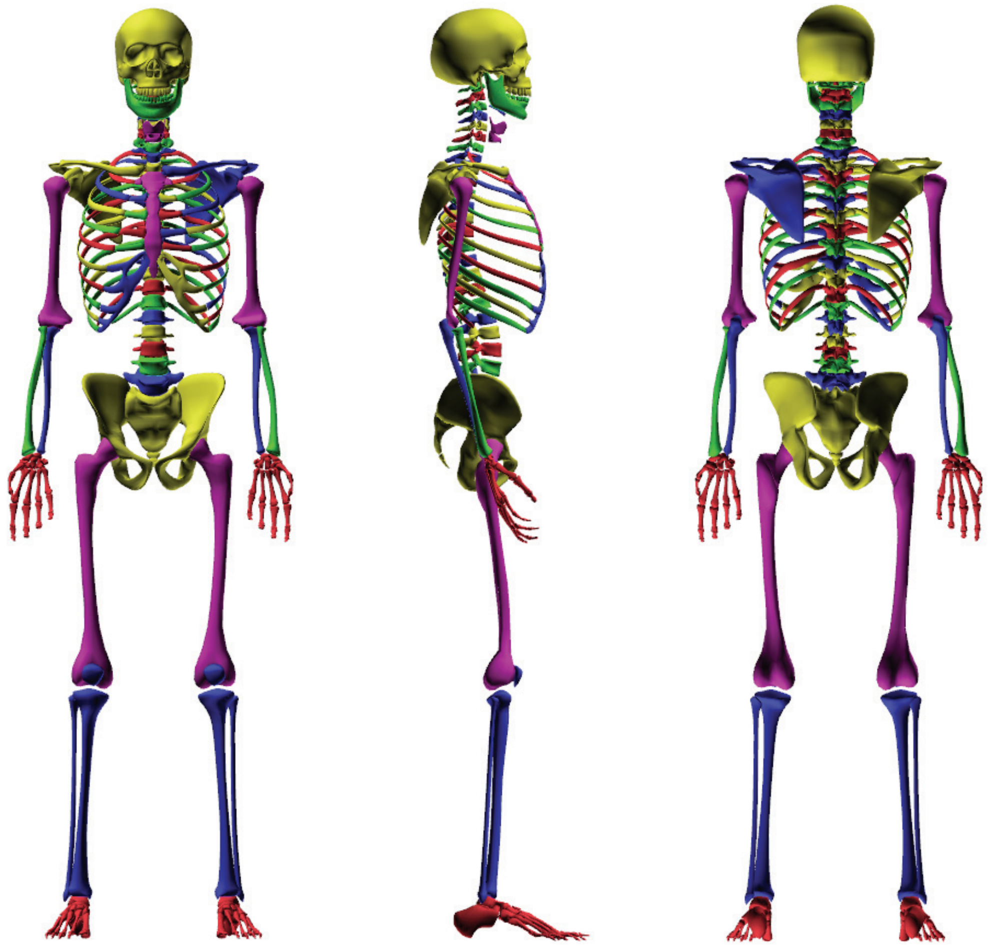


Figure 5.1: Ventral (a), lateral (b), and dorsal (c) views of the modeled skeleton of the body. Most of the articular bones are individually modeled. There is a total of 75 bones articulated by 165 DOFs, of which 139 DOFs are in the head-neck-trunk region of the upper body. Neighboring bones with the same color are treated as a single rigid body. The parameters of the skeleton are given in Table C.2.

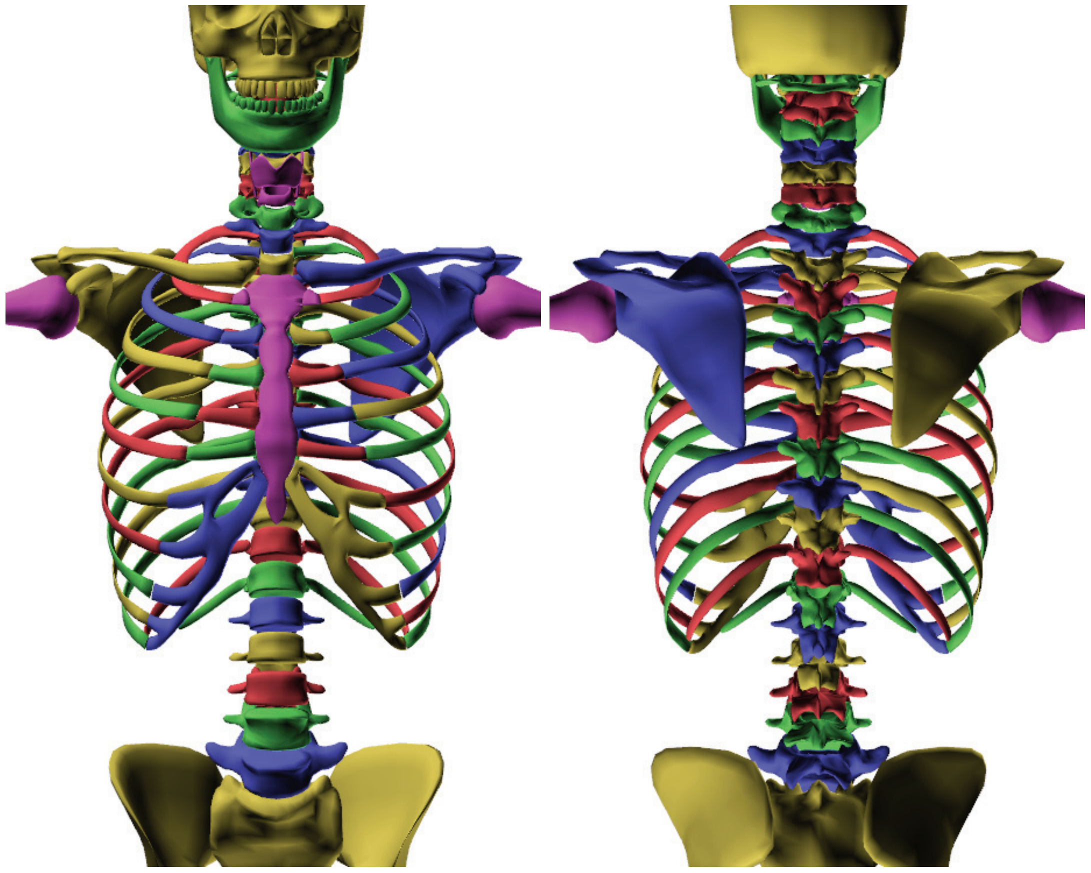

(a) (b)

Figure 5.2: Closeup ventral (a) and dorsal (b) views of the upper body bones.

costovertebral joints [Kapandji 1974]. The 11th and 12th ribs are rigidly attached to their parent vertebrae. Although costal cartilages are flexible bodies, we model them as rigid bodies, and the flexibility of the cartilages are substituted by 3-DOF joints connecting to the sternum and springs connecting to the ribs. By not modeling the joints between the costal cartilages and ribs, we maintained the skeleton as an open-loop system so that we can use a fast Articulated Body Method for simulating dynamics. The hyoid bone and the thyroid cartilages are also modeled as rigid bodies. Even though the actual bones are not jointed to vertebrae, we modeled them for simplicity as child links of nearby vertebrae, with rotational joints positioned about 5cm posterior to the parent bones.

Since we are less concerned about modeling highly coordinated motions of the clavicle, scapula, and humerus, we simplified the model of the shoulder; the clavicle has 2 DOFs and the scapula is rigidly attached to the clavicle. Nevertheless, this allows plausible movement for a modest range of upper arm motions. A more accurate modeling of the scapula will be discussed in Section 6.5. Also, for simplicity, the patella is assumed to be rigidly attached to the tibia.

The inertial properties of the skeleton are approximated from the dense volumetric mesh of the surrounding soft tissues. We associated the inertial parameters of each volumetric element to the nearest bone so that each bone's inertial tensor is augmented by the inertial parameters of the associated soft tissues. The modeling parameters of each bone, such as the hierarchy, inertia, joint position, and joint axis, are given in Table C.2.

5.1.2 Muscle Actuation Model

A precise modeling of the muscle parameters is prerequisite to correctly computing the activation levels of each muscle. We model most of the skeletal muscles. By analyzing the muscle geometries (Figure 5.3 and 5.4) in the commercially available Ultimate Human Model (www.cgcharacter.com/ultimatehuman.html), which is a purely geometric model of the human body, we created a total of 846 biomechanical muscles, modeled using piecewise line segment (PLS) models (Figure 5.5 and 5.6). Table 5.2 lists the modeled muscles. It may at first seem that we modeled more muscles than necessary, but given the high number of DOFs of our skeletal model, this is only about 2.6 times the minimum number of muscles required to actuate the system assuming only one agonist/antagonist muscle pair per DOF.

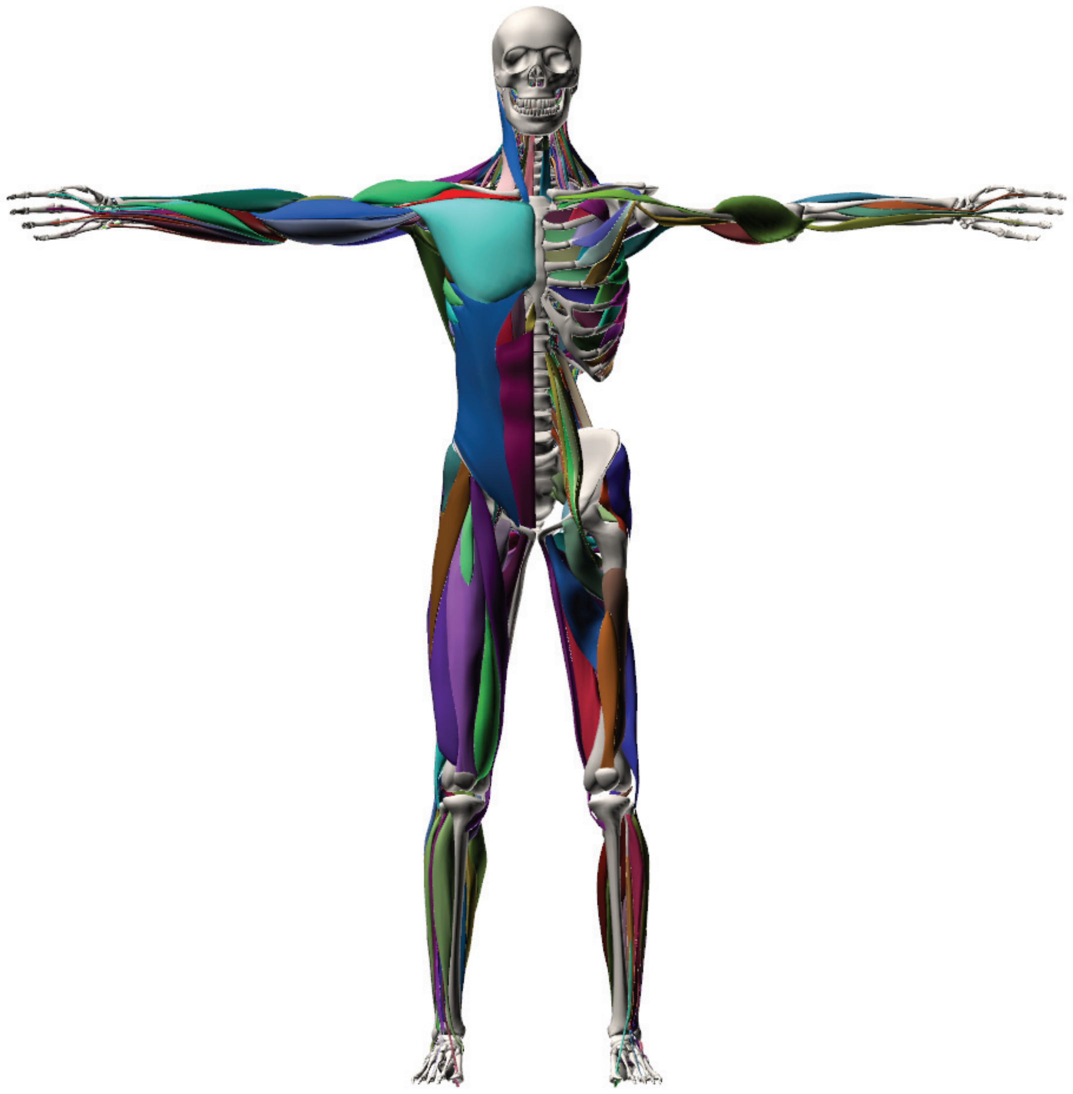


Figure 5.3: Ventral view of source geometry data of the modeled muscles. The superficial muscles are shown on the right side of the body, while deeper muscles are shown on the left side.

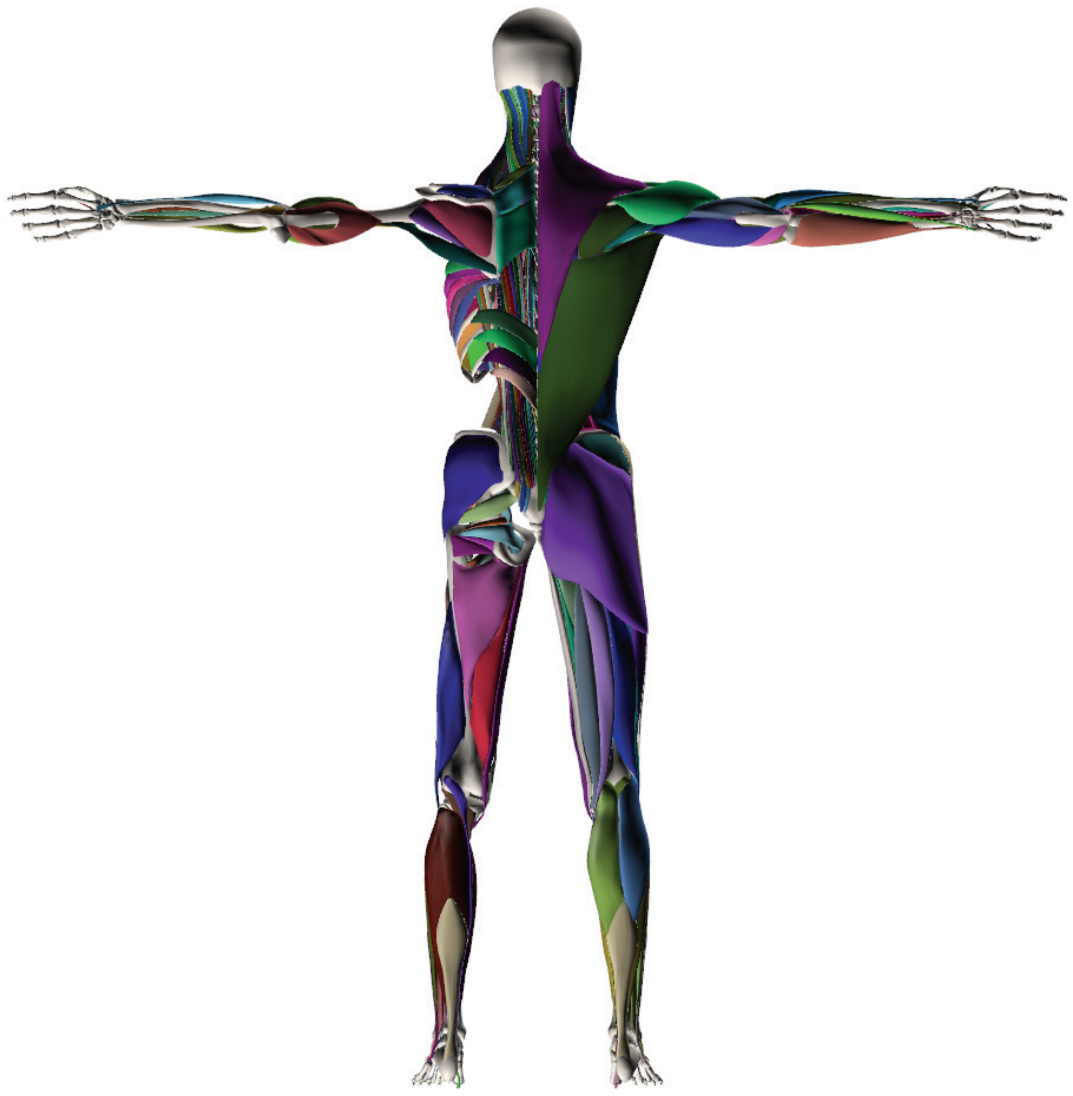


Figure 5.4: Dorsal view of source geometry data of the modeled muscles. The superficial muscles are shown on the right side of the body, while deeper muscles are shown on the left side.

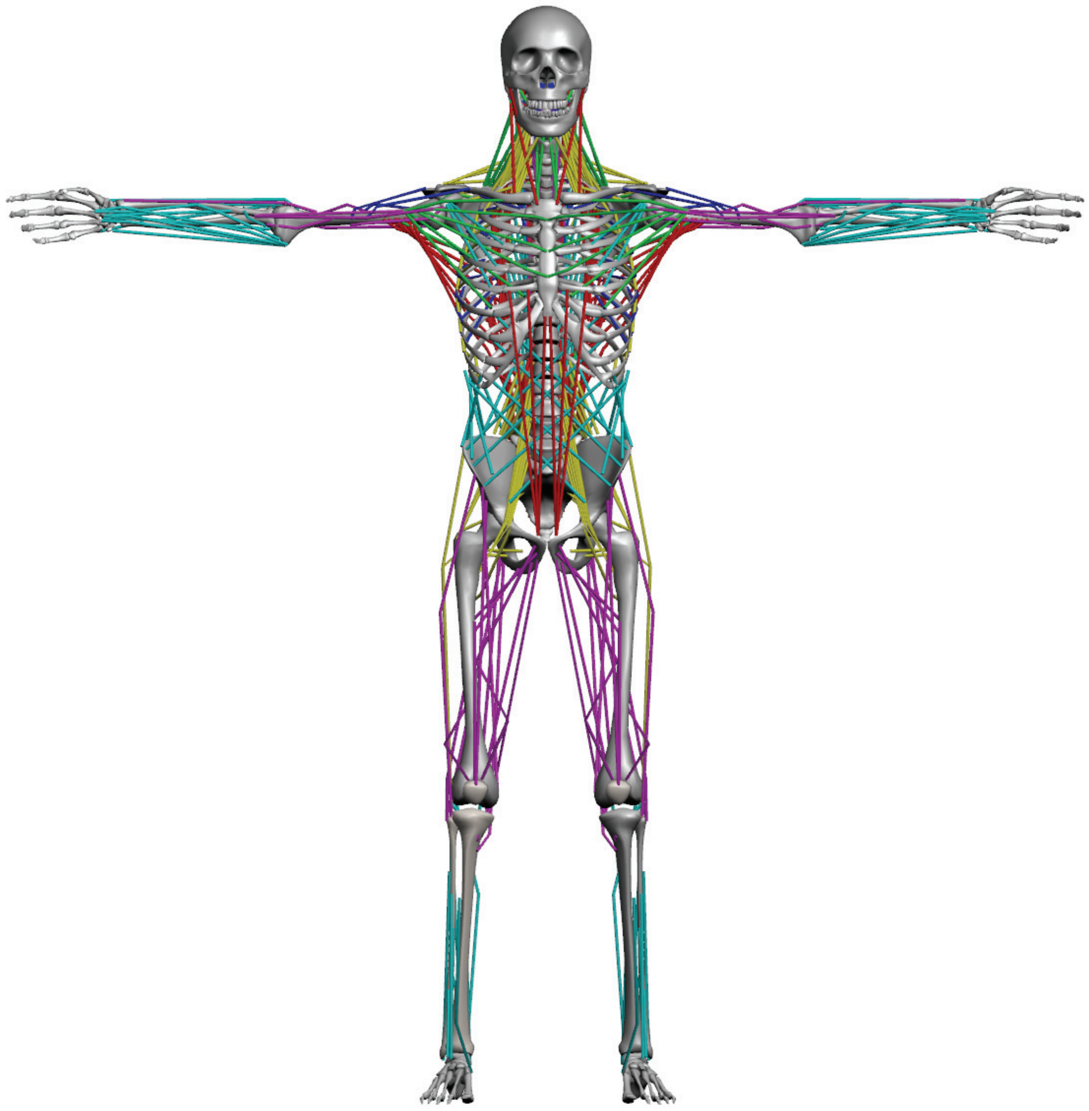


Figure 5.5: Ventral view of the lines-of-action of the uniaxial muscle actuators in the body. A total of 846 muscle forces are modeled. The parameters of the muscles are given in Appendix C.

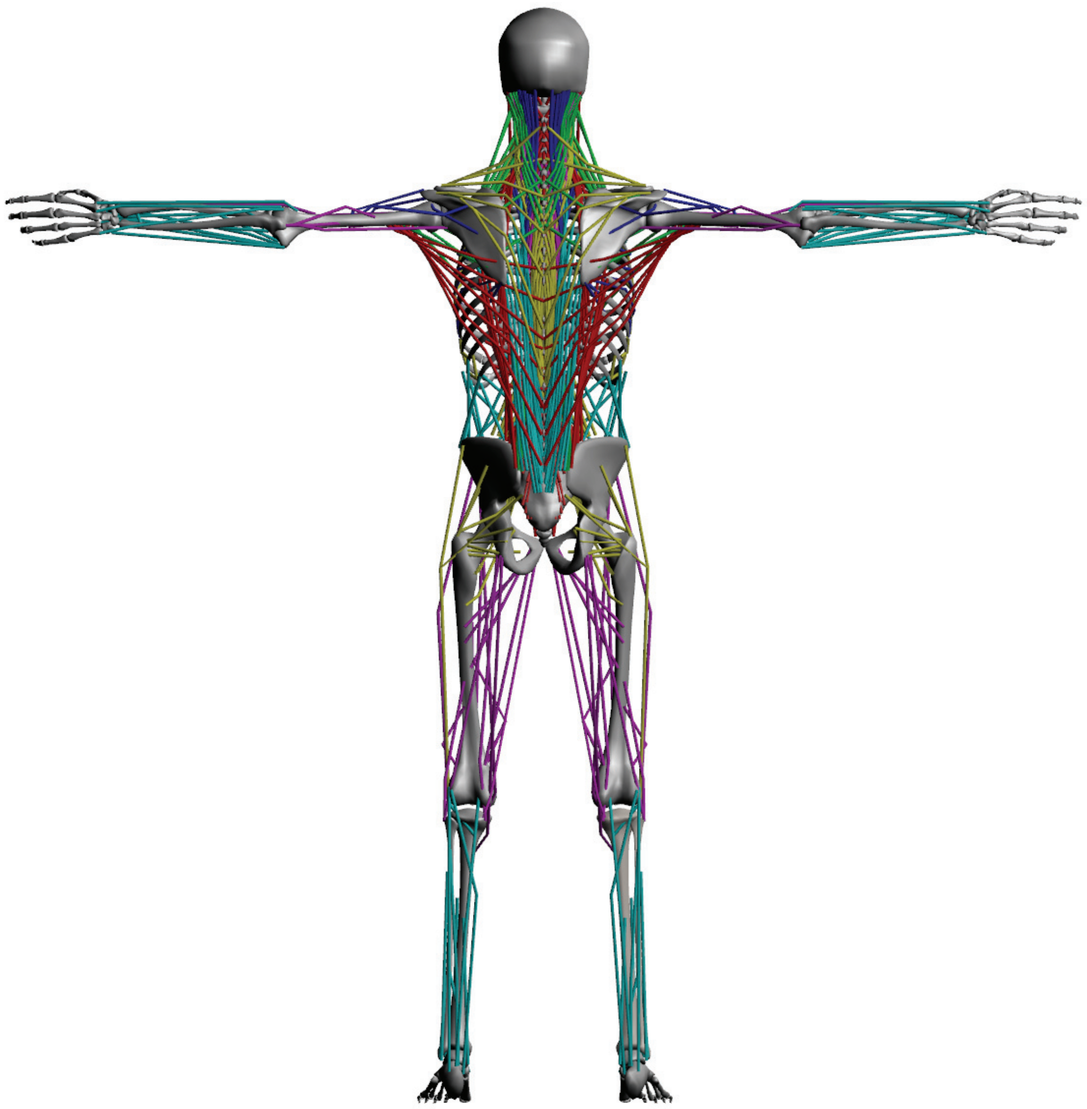


Figure 5.6: Dorsal view of the lines-of-action of the uniaxial muscle actuators in the body.

Head/Neck muscles (264)
Iliocostalis thoracis (2), Interspinalis (12), Intertransversi (14), Rotatores (10), Semispinalis thoracics (4), Trapezius (4), Masseter (2), Iliocostalis cervicis (8), Longissimus capitis (16), Longissimus cervicis (10), Semispinalis cervicis (12), Splenius capitis (10), Semispinalis capitis (18), Longus capitis (8), Geniohyoid (2), Longus colli (14), Obliquus capitis (4), Omohyoid (2), Rectus capitis posterior (4), Rectus capitis anterior (2), Scalenes (28), Sternocleidomastoid (4), Sternohyoid (2), Sternothyroid (2), Stylohyoid (2), Rectus capitis lateralis (2), Levator scapulae (16), Multifidus (36), Thyrohyoid (2), Mylohyoid (2), Splenius cervicis (8), Rhomboid minor (2)

Trunk muscles (412)
External/Internal obliques (22), Rectus abdominis (6), Iliocostalis lumborum (24), Iliocostalis thoracis (18), Interspinalis (14), Intertransversi (14), Multifidus (114), Rotatores (30), Semispinalis thoracics (8), Spinalis thoracis (14), Trapezius (10), External intercostal (20), Pectoralis minor (6), Serratus anterior (16), Subclavius (2), Longissimus thoracis (48), Serratus posterior inferior (10), Internal intercostal (20), Rhomboid major (2), Quadratus lumborum (10), Iliocostalis cervicis (2)

Arm muscles (41)
Pectoralis major (6), Latissimus dorsi (7), Biceps brachii (2), Brachioradialis, Brachialis, Coracobrachialis, Triceps Brachii (3), Infraspinatus, Deltoid (3), Supraspinatus, Teres major, Extensor carpi radialis brevis, Extensor carpi ulnaris, Extensor carpi radialis longus, Supinator, Flexor carpi ulnaris, Palmaris longus, Flexor digitorum superficialis (4), Pronator teres (2), Flexor carpi radialis, Pronator quadratus

Leg muscles (44)
Gluteus (3), Gemellus (2), Piriformis, Obturator (2), Rectus femoris, Sartorius, Quadratus femoris, Adductor brevis, Adductor longus, Gracilis, Adductor magnus (2), Extensor digitorum longus, Peroneus brevis, Peroneus longus, Peroneus tertius, Vastus (3), Semimembranosus, Semitendinosus, Pectineus, Iliopsoas (6), Soleus, Extensor hallucis longus, Flexor hallucis longus, Flexor digitorum longus, Gastrocnemius (2), Plantaris, Popliteus, Biceps femoris (2), Tensor fascia latae, Tibialis anterior

Table 5.2: List of the muscles modeled in the body. A total of 846 muscles are modeled. The numbers in parentheses indicate the number of multiple muscle forces of each kind. See Appendix C for the modeling parameters of the muscles.

5.1.2.1 Piecewise Line Segment Modeling

We modeled broad muscles such as the trapezius and latissimus dorsi using multiple PLS models (Figure 5.7(a)). Although the external obliques and internal obliques are

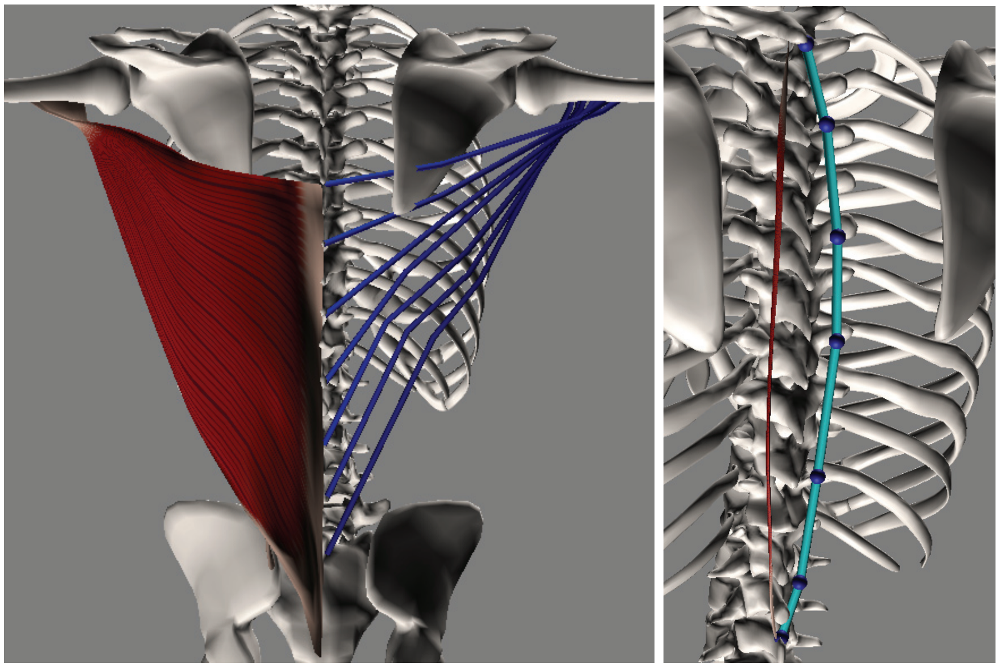

(a) (b)

Figure 5.7: Piecewise line segment muscle models. (a): Broad muscles such as the Latissimus dorsi in the figure are modeled using multiple line segments. (b): When a deep muscle spans many vertebrae, one in every two or three vertebrae is assigned a fixation point (blue sphere) to conform the muscle to the movement of the vertebrae.

abdominal muscles organized in different layers, we constructed PLS models of the left (right) external obliques and right (left) internal obliques as if they were connected.

In the case of deep muscles, we can determine the via points for the PLS model without much difficulty, because deep muscles are positioned close to bones and the distance from a muscle to neighboring bones does not change much as the bones move. When a deep muscle spans many vertebrae, via points are set to only a portion of bones for computational efficiency (Figure 5.7(b)). Modeling the superficial muscles using PLS models is more complicated, since superficial muscles are rather far from bones, making it harder to determine via points, and their relative position to the bones vary significantly as bones move. For superficial muscles, we selected the a few (typically

2-4) via points subjectively, making sure that the resulting PLS muscle model deforms convincingly as the skeleton moves.

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

5.1.2.2 Hill-type Muscle Force Model

The force generating characteristic of the PLS muscle is modeled as a *linearized Hill-type model* in Section 3.2. Assuming that the length of the tendon is constant, we model a muscle force as the sum of the forces from a contractile element (CE) and a parallel element (PE).

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

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

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

The CE force is expressed as

$$f_C = aF_l(l)F_v(\dot{l}), \tag{5.1}$$

where $0 \leq a \leq 1$ is the activation level of the muscle. The force-length relation F_l is modeled as $F_l(l) = \max(0, k_{\max}(l - l_m))$, where $k_{\max}$ is the maximum stiffness of a fully activated muscle and l_m is the minimum length at which the muscle can produce

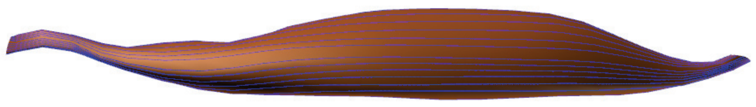


Figure 5.8: The PCSA of a muscle is calculated as its volume divided by its mean fiber length. Blue lines indicate the fiber direction of the biceps brachii.

force, and $F_v(\dot{l}) = \max(0, 1 + \min(\dot{l}, 0)/v_m)$, where $v_m(\geq 0)$ is the maximum contraction velocity under no load. We set $l_m = 0.3l_0$, $v_m = 8l_0$ sec^{-1}, and $k_c = 6$ for all the muscles. The coefficients k_s, k_d, and $k_{\max}$ for each muscle are scaled by the physiological cross sectional area (PCSA) of the muscle. The PCSA of a muscle is calculated by dividing its geometric volume by its mean fiber length (Figure 5.8). Appendix C lists the relevant parameters of each muscle, such as the PCSA, rest length, and the attachment points.

5.1.2.3 Equations of Motion

It is important to note that even if we had faithfully modeled the majority of the skeletal muscles, the sternum and the costal cartilages cannot be controlled by muscles. This is natural because they are in fact moved only passively by the motion of the neighboring ribs; hence, we dub them *passive joints* in this chapter. This does not pose any problem in forward dynamics simulation, but it does make it tricky for inverse dynamics; i.e., computing the muscle forces needed to generate the desired accelerations. We will address this problem in 5.3.3.

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

$$\mathbf{M}(\mathbf{q}) \begin{bmatrix} \ddot{\mathbf{q}}_m \\ \ddot{\mathbf{q}}_p \end{bmatrix} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{P}(\mathbf{q})\mathbf{f}_C \\ \mathbf{0} \end{bmatrix} + \mathbf{J}^T \mathbf{f}_e, \tag{5.2}$$

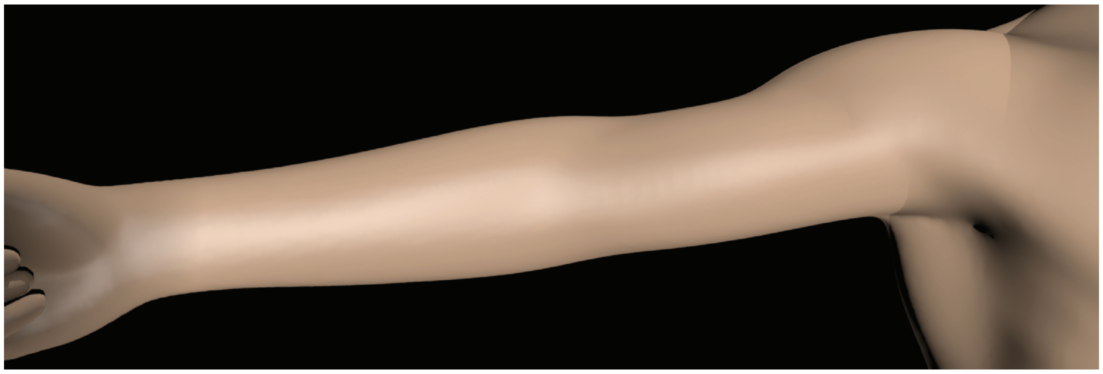
where $\mathbf{q}_m$ are the generalized coordinates of the muscle-driven joints, $\mathbf{q}_p$ are those of the passive joints, and $\mathbf{q} = [\mathbf{q}_m^T, \mathbf{q}_p^T]^T$ is the state vector, $\mathbf{M}$ is the mass matrix, and $\mathbf{c}$ accounts for the forces from connecting tissues and muscle parallel elements (f_P), as well as gravity, Coriolis forces, and centrifugal forces. The Jacobian matrix $\mathbf{J}(\mathbf{q})$ transforms the applied external force $\mathbf{f}_e$ into joint torques. The moment arm matrix $\mathbf{P}$ transforms the muscle force $\mathbf{f}_C$ to the joint space torque and is defined as $\mathbf{P}^T = d\mathbf{l}/d\mathbf{q}$, where $\mathbf{l}$ is the vector of the lengths of each muscle. The algorithm to compute $\mathbf{P}$ is detailed in Section 3.1.1.1.

5.2 Soft Tissue Modeling and Simulation

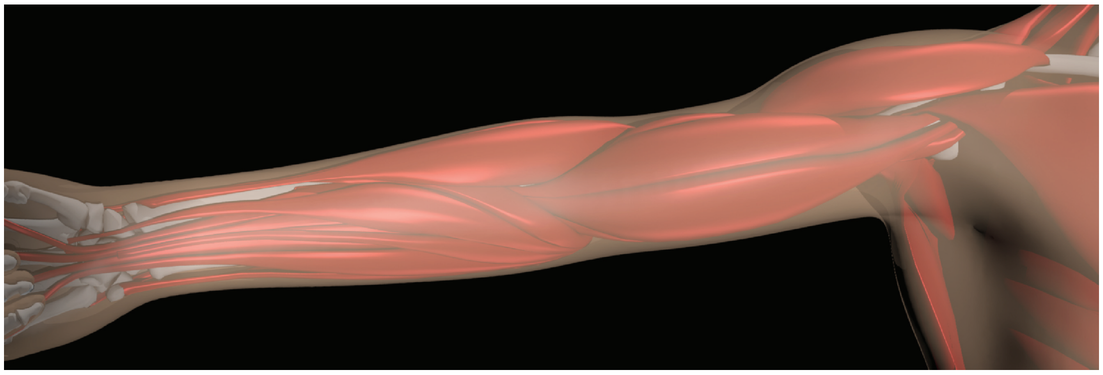
In our human body model, the muscle activation parameters are computed using a PLS idealization of the musculature. Driven by these muscle action inputs, a companion simulation of a volumetric, finite element model of the musculoskeletal structure introduces the visual richness of more detailed, 3D musculature models (Figure 5.9).

5.2.1 Skin Surface Model Creation

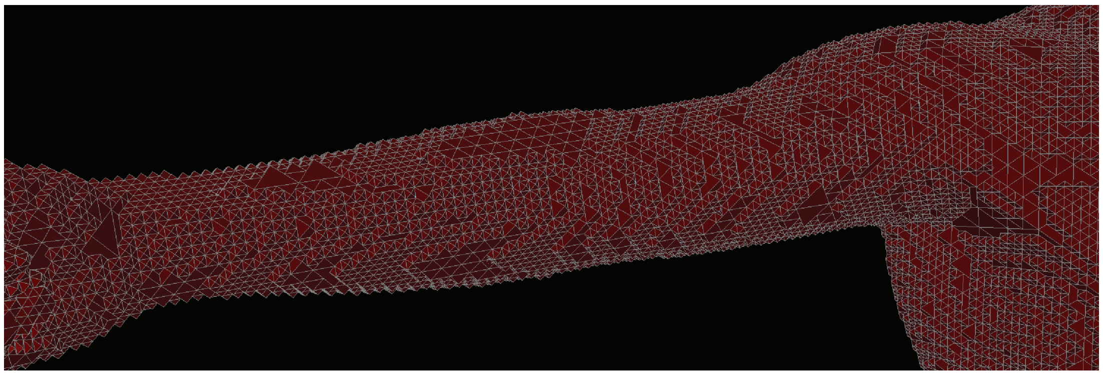
The first step in the construction of our physics-based soft tissue model is the creation of a high quality skin surface geometry. The initial high resolution skin surface mesh that serves as our prototype is created by subdividing the original mesh of the Ultimate Human Model. Although rich in geometric detail, it does not clearly define the volu-



(a) Skin Visualization Geometry



(b) Anatomical Bone and Soft Tissue Geometry



(c) Simulation Mesh

Figure 5.9: Our volumetric human body model incorporates (a) detailed aspects of skin geometry and (b) active muscle tissues, passive soft tissues, and skeletal substructure. The skin surface is discretized into a 300K-triangle mesh. Resolving this surface detail with a fully tetrahedralized mesh through the bounded volume would make any form of finite element simulation impractical. We overcome this difficulty by decoupling the visualization geometry from the simulation geometry, by creating an *embedded model*. To this end, an adaptive, BCC tetrahedralized mesh is superimposed on the soft tissue volume (c). This mesh embeds the high-resolution surface representation by means of barycentric interpolation of the surface nodes from the nodes of the tetrahedral simulation mesh.

metric boundary of the body, since it is not a closed, intersection-free surface, having openings at the eye sockets, spurious intersections near the ears, etc. Additionally, the mesh is rather poorly conditioned for the purposes of a volumetric physics-based simulation, with elements exhibiting aspect ratios as high as 60:1, while the ratio of the longest to the shortest edge in the mesh is in excess of 1000:1. This would hinder the time integration of the resulting physics-based model, as well as collision processing. We ameliorate these shortcomings as follows:

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

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

- The meshing algorithm of [Molino et al. 2003] is utilized to convert the level set implicit model of the flesh into a well-conditioned tetrahedralized volumetric mesh. We create a moderately adaptive tetrahedral mesh with element diameters ranging from 1mm to 10mm, resolving the surface geometry at an average resolution of 3–5mm. The resulting mesh has an overwhelming 6.2 million tetrahedral elements; however, instead of directly using this model for simulation, we keep only the *triangulated surface* of this tetrahedralized volume (i.e., its topological boundary), which we use in the context of an embedded simulation framework described below. The interior structure of this tetrahedral mesh is

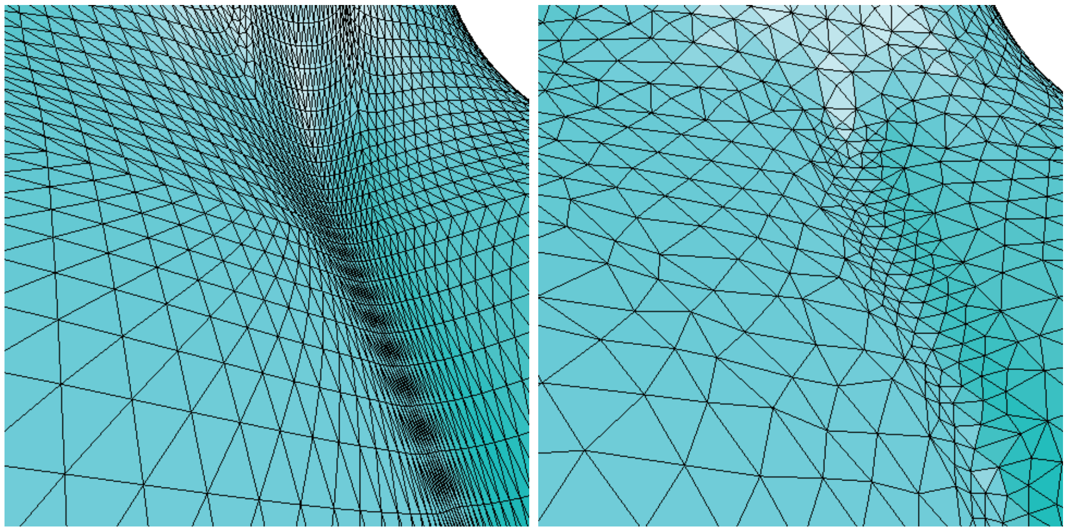


Figure 5.10: Detail from the chest of the anatomical model. The Ultimate Human Model skin surface is riddled with sliver elements (left), compared to our well-conditioned surface mesh (right).

discarded.[1]

The resulting triangulated skin geometry has 302K triangles, with an average diameter of 4–5mm and a maximum aspect ratio of 3.8:1 (Figure 5.10). This well-conditioned mesh provides an excellent starting point for physics-based flesh modeling and greatly helps collision handling.

5.2.2 Generation of an Embedded Simulation Mesh

Having created a high-quality surface representation, our next goal is to generate a *volumetric* simulation mesh on which the governing equations of the soft tissue will be defined. One possibility would be simply to create a tetrahedral mesh directly from the implicit surface representation of the flesh volume that was previously created. In fact, we have already described how we created such a tetrahedral mesh with the purpose

[1]Note that a different method for meshing implicit surfaces into triangle meshes could have been used instead, provided that it can create a high quality, well conditioned mesh.

of using its boundary triangle mesh as the skin surface representation. Nevertheless, there are a number of challenges with using such a mesh as the simulation geometry. The mesh resolves the skin surface at a resolution of 3–5mm, with an interior element size of 10mm. At this resolution the mesh consists of 6.2 million tetrahedra, which is at least an order of magnitude greater than the maximum practical size for a nonlinear finite element flesh simulation.

One possible remedy would be to use more aggressive adaptivity; increasing the maximum element size to 20–30mm could lead to 8 to 27 times fewer *interior* elements. Unfortunately, at the given level of surface resolution, as many as 1–1.5 million tetrahedra would be incident to the surface where high refinement would be necessary. In practice, we found that even with very aggressive adaptivity in the interior of the flesh, the minimum mesh size attainable without coarsening near the surface would be approximately 2 million tetrahedra. Finally, even if we tolerated a lower surface resolution for most of the skin surface, there are certain regions that need to be adequately resolved due to high-curvature features (e.g., fingers, toes, face) or to facilitate collision handling (e.g., inner thighs, elbow, armpit). Since the size of the time-step (for explicit integration methods) and the conditioning of the mesh (for explicit and implicit methods) depend on the smallest element size, having highly resolved portions of the mesh would come at the cost of reduced simulation performance.

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

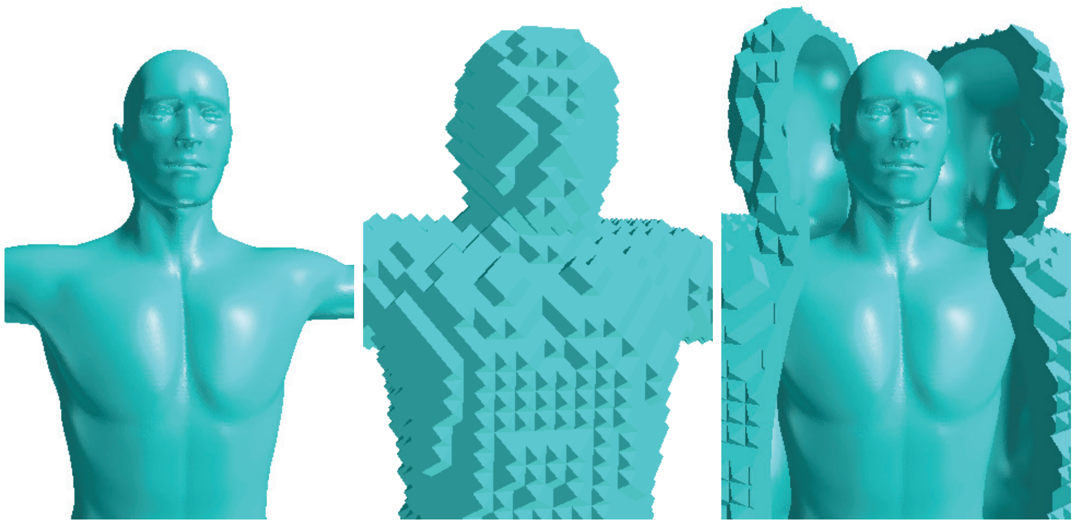


Figure 5.11: Volumetric mesh cutting for the skin surface. Left: Triangulated skin geometry. Center: Background BCC-tetrahedralized material volume. Right: After cutting along the skin surface, the material volume has been separated into an embedded volumetric body model, and its outer mold of unused material (sliced and peeled open for illustration purposes).

We start by generating a Body-Centered-Cubic (BCC) tetrahedral lattice (see [Molino et al. 2003] for details), which completely covers the volume spanned by the human body, as seen in Figure 5.11 (center). We use a uniform size of 7mm for the tetrahedral elements at this step. Subsequently, we use the algorithm of [Sifakis et al. 2007a] to *cut* this background tetrahedral along the triangulated surface of the skin into two separate parts—the fragment interior to the skin surface, which corresponds to the human body volume, and the exterior part, which forms a negative “mold” enclosing the body, as seen in Figure 5.11 (right). We discard this exterior volume as it is irrelevant to our simulation. The interior volume is exactly the soft tissue model that we wish to simulate. The cutting algorithm of [Sifakis et al. 2007a] provides us with the subset of the original tetrahedral mesh that intersects this volume (Figure 5.12 (center)), and an embedded skin surface geometry in terms of a triangle mesh whose vertices are barycentrically embedded into the tetrahedra of the embedding volume.

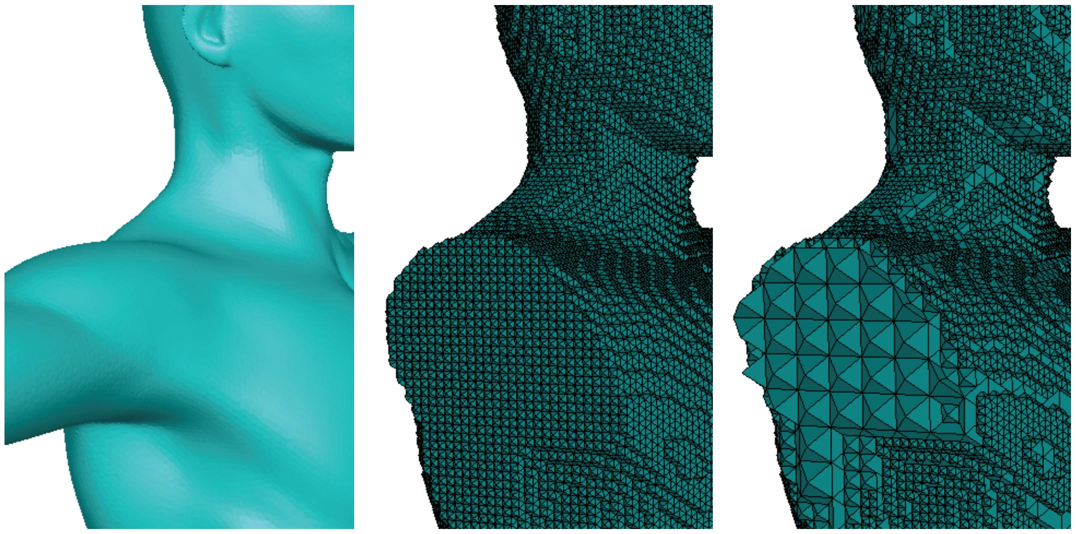


Figure 5.12: Coarsening the volumetric mesh. Left: Visual reference. Center: Uniform embedding mesh at a resolution of 7mm. Right: The embedding mesh after 2 steps of adaptive coarsening, down to an element resolution of 28mm. T-junctions are visible at the boundaries between refinement levels.

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

Since the embedding mesh thus created originates from a uniform resolution lattice, its total number of elements (3.8 million) is still prohibitively high. Leveraging the highly regular structure of the underlying BCC lattice, we proceed to adaptively coarsen this mesh by reversing the process of a *red refinement* as defined in [Molino et al. 2003].

The inverse of this process, *red coarsening*, collapses eight child tetrahedra into one, similar to each of the child tetrahedra with a edge ratio of 2:1. The criterion for coarsening is that all eight child tetrahedra must be present in the embedding mesh and that none of them has been duplicated by the cutting algorithm of [Sifakis et al. 2007a]. After recursively coarsening for a maximum of two levels (i.e., a tetrahedron size of 28mm), we obtain the final simulation mesh consisting of 354,000 tetrahedra (Figure 5.12 (right)). Due to the nature of our refinement process, T-junctions are present at the boundaries between different levels of refinement. These special points are simulated in a straightforward fashion using the framework of [Sifakis et al. 2007b]. We found that the small overhead of simulating T-junctions is well offset by the low tetrahedron count and high level of adaptivity obtained.

5.2.3 Modeling Musculature and Skeletal Structure

The tetrahedral simulation mesh created in the previous step does not strictly conform to the geometry of muscles or bones. Consequently, such features will be *embedded* in the simulation mesh, in analogy to the treatment of the high-resolution skin surface. As a first step, we use the geometry of the muscles to modulate the material properties assigned to each simulation element. A number of randomly generated points (we used between 10^3 and 10^4 points, depending on element size) are uniformly distributed in each simulation tetrahedron, indicated as colored dots in Figure 5.13 (left). We check whether each of these sample points is located inside any muscle volume, in which case the direction of the muscle fiber field at the given location is recorded and associated with the sample point. These samples are depicted as red and orange vector fields in Figure 5.13 (left), corresponding to two distinct muscles intersecting a simulation element. Points not inside any muscle volume are considered as locations of passive flesh or fatty tissue (displayed as blue dots in Figure 5.13). Using these sample points, we compute a *muscle density* $d_i \in [0,1]$, denoting the fraction of the simulation element covered by the *i*-th muscle. Similarly, d_{passive} denotes the fraction of the simulation element covered by passive flesh (outside any muscle). Consequently, these densities satisfy $d_{\text{passive}} + \sum d_i = 1$. Finally, we average the fiber directions of the sample points inside the *i*-th muscle and normalize the result to unit length to obtain a representative fiber direction $\mathbf{f}_i$ for this muscle with respect to the simulation element in question.

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

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

$$\Psi(\mathbf{F}) = d_{\text{passive}}\Psi_{\text{passive}}(\mathbf{F}) + \sum d_i \Psi_i(\mathbf{F}).$$

The passive flesh is modeled as an isotropic, quasi-incompressible Mooney-Rivlin material [Bonet and Wood 1997], leading to the following formula for Ψ_{passive}:

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

In this definition, $J = \det\mathbf{F}$ is the volume change ratio, $\hat{\mathbf{F}} = J^{-1/3}\mathbf{F}$ is the deviatoric component of the deformation gradient and $\hat{\mathbf{C}} = \hat{\mathbf{F}}^T\hat{\mathbf{F}}$ is the deviatoric Cauchy strain tensor. We used the values $\mu_{01} = 0.06$MPa, $\mu_{10} = 0.02$MPa for the elasticity moduli while the bulk modulus κ (a measure of incompressibility of the tissue) was set to 10MPa. The constitutive model for active muscles is the sum of the isotropic contribution Ψ_{passive} and an anisotropic muscle term:

$$\Psi_i = \Psi_{\text{passive}} + \Psi_{\text{muscle}}(\lambda_i),$$

where $\lambda_i = \|\hat{\mathbf{F}}\mathbf{f}_i\|$ is the along-fiber contraction ratio of the i-th muscle in the simulation element, and the function Ψ_{muscle} is defined via its first derivative:

$$\frac{\partial \Psi_{\text{muscle}}(\lambda)}{\partial \lambda} = \frac{\sigma_{\max}}{\lambda_{\text{opt}}} f_{\text{total}}(\lambda).$$

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

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

We circumvent these problems by using *soft constraints* and applying them to *embedded* locations rather than true nodes of the simulation mesh, as follows: A set of particles is uniformly sampled from the surface of each bone. These particles (displayed as blue dots in Figure 5.13, right) are rigidly constrained to the respective bone. We then duplicate each of these particles with the locations that they have in the undeformed configuration of the body. The duplicated particles (illustrated with a slight displacement as green dots in Figure 5.13, right) are barycentrically embedded into the simulation element with which they overlap. Finally, each particle attached to a bone is connected with its duplicate embedded counterpart using a zero rest-length spring. This embedded treatment of skeletal attachments allows us to decouple the resolution of the simulation mesh from the resolution of the skeletal geometry and define the attachment regions as arbitrarily point-sampled surfaces.

5.2.4 Numerical Simulation and Time Integration

Our musculoskeletal simulation model contains features such as T-junctions, hybrid descriptions using embedded collision geometries and embedded point sets for skeletal attachments, as well as soft constraints implemented as zero-length springs. We use the hybrid simulation framework of [Sifakis et al. 2007b], which accommodates such simulation elements in the context of either explicit or implicit time-integration schemes. In particular, T-junctions and points embedded in the simulation mesh are

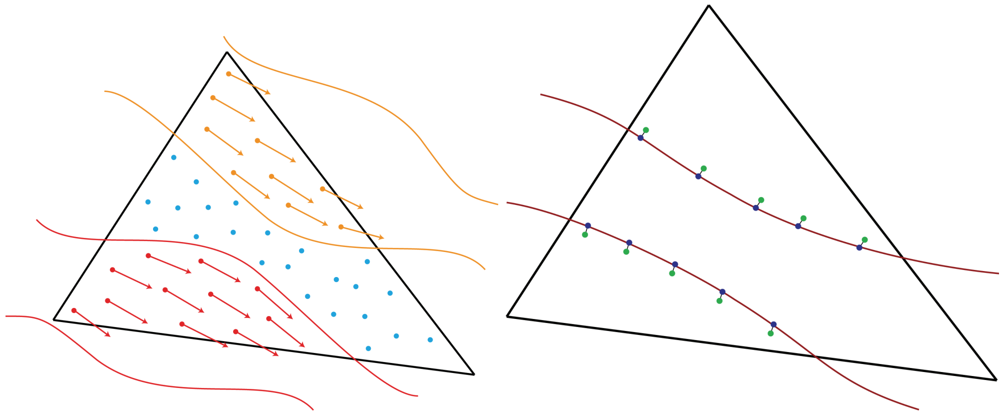


Figure 5.13: Emdedding the skeleton into the volumetric mesh. Left: Randomly scattered point samples in a simulation element are classified as belonging to active muscles (red and orange points, with their associated fiber directions) or passive flesh (blue points). Right: A rigid bone (the area between the brown outlines) is sampled on its surface (blue points). These sample points are connected with zero length springs to barycentrically embedded locations (green points) in the simulation mesh.

naturally handled without compromising the symmetry or definiteness of the linear systems arising from the finite element discretization of the simulation mesh. Additionally, the zero-length springs used to enforce soft constraints are handled fully implicitly in the context of Newmark-type integration schemes, alleviating any timestep restrictions that could possibly arise from stiff constraint springs. For the examples illustrated in this chapter, we used the quasistatic time integration scheme of [Teran et al. 2005b], which provides for the robust handling of extreme deformation and element inversion, both of which are frequent occurrences in our context. We obtained simulation times of 1–4 minutes per frame, running on a single core of a 3.0Ghz Intel Xeon workstation, mostly depending on the rate of change of muscle activation and the velocity of the skeleton.

5.3 Control

Ideally, our comprehensive biomechanical model of the human body would be controlled by a generalized version of the biomimetic, hierarchical control architecture that we proposed in Chapter 4 to animate the neck-head complex. However, the increased complexity of the entire biomechanical body bodes commensurately increased controller complexity. To avoid the "curse of dimensionality", it would be sensible to perform a natural modularization, developing additional specialized neural network controllers for the trunk and each limb, plus a higher level controller to coordinate the collection of specialized controllers. The success of our specialized head-neck controller suggests that such a modular approach is feasible and promising. This is an intriguing research topic, but it is in itself another PhD-worthy challenge, which is beyond the scope of this thesis.

In this section, we develop a simpler whole-body controller that tackles a more modest yet still interesting and useful control task. We will take an in inverse-dynamics, *computed muscle force control* approach; i.e., we will first employ the equations of motion (5.2) of the musculoskeletal system to compute the muscle forces necessary to produce some desired motion—say, by specifying the desired joint angles over time via motion capture data—under applied external forces, and then we will apply the computed muscle forces to produce the final simulation. To specify the 6 DOFs of the root node of the model, the pelvis, we either constrain our character to remain in a sitting pose with its pelvis fixed or we allow our inverse dynamics algorithm to compute a fictitious force that drives the pelvis through a series of desired target poses.

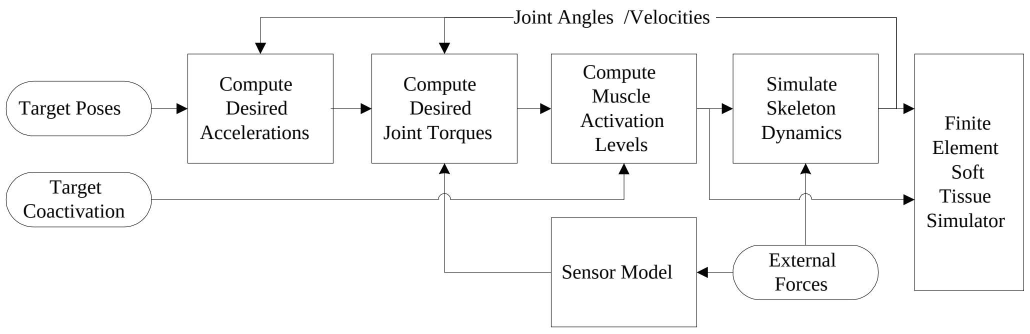

Joint Angles /Velocities
Target Poses
Target Coactivation
Compute Desired Accelerations
Compute Desired Joint Torques
Compute Muscle Activation Levels
Simulate Skeleton Dynamics
Finite Element Soft Tissue Simulator
Sensor Model
External Forces

Figure 5.14: Overview of the motion controller.

5.3.1 System Overview

Figure 5.14 shows a schematic of our motion controller. The inputs to the controller are the target pose and muscle coactivation of agonist and antagonist muscles of the body. Although the pose/coactivation inputs can be dense time series, such as motion capture data, we use sparse key frames (at an approximate rate of 1 frame/sec) as inputs in our experiments. Given the inputs and the current state, the motion controller determines the desired acceleration of the joints, and then computes the required muscle activation levels in order to achieve the desired accelerations and target coactivation. Then the articulated body dynamics of the skeletal system are simulated and the joint angles and muscle activation levels are sent to soft tissue simulator. We use an explicit Euler time integration method with a simulation time step of around 0.1ms. The control signals are updated every 5–10 simulation time steps.

5.3.2 Computing Desired Accelerations

Given target positions and orientations of the head and chest (T1 bone) as well as target joint angles of the arms and legs, the motion controller performs inverse kinematics to determine the desired angles of all the muscle-driven joints.

For the vertebrae, we determine the joint angles such that the sum of the squared norm of the joint angles is minimized under the constraint that they produce the target pose of head and chest. Regarding rib motion, we define a suitable periodic function that computes desired joint angles.

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

$$\ddot{\mathbf{q}}^* = k_p(\mathbf{q}^* - \mathbf{q}) + k_v(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}), \tag{5.3}$$

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

5.3.3 Generalized Force Computation

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

The hybrid dynamics algorithm [Featherstone 1987] is an efficient algorithm that serves this purpose. Through simple modification of the Articulated Body algorithm, we provide desired accelerations for some joints and desired input torques for others and, in time linear in the number of joints, we can compute the required joint torques for the former and the resulting accelerations of the latter. In other words, the algorithm computes inverse dynamics for some joints and forward dynamics for others in linear time.

We set the generalized forces of the passive joints to zero and set the desired accelerations of the muscle-driven joints. The hybrid dynamics algorithm computes the required generalized forces for the muscle-driven joints. Mathematically, the algorithm solves for τ^* in

$$\begin{bmatrix} \tau^* \\ \mathbf{0} \end{bmatrix} = \mathbf{M}(\mathbf{q}) \begin{bmatrix} \ddot{\mathbf{q}}_m^* \\ \ddot{\mathbf{q}}_p \end{bmatrix} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{J}^T \mathbf{f}_e, \tag{5.4}$$

where $\ddot{\mathbf{q}}_m^*$ is the desired acceleration of the muscle-driven joints. Next, we determine muscle activation levels that generate the generalized forces.

5.3.4 Computation of Muscle Activation Levels

One of the most distinguishing features of muscles is that their stiffness varies according to the activation level. In addition to pose, humans control stiffness by exploiting the redundancy of muscle actuators, and stiffness is an important aspect of motion style.

We introduce a new method to determine muscle activation levels. Our method achieves the desired stiffness by explicitly computing agonist and antagonist muscle activations.

The agonist muscle activation level $\mathbf{a}_g$ is determined by solving the following optimization problem:

$$\mathbf{a}_g = \underset{\mathbf{a}}{\operatorname{argmin}} \frac{1}{2} \sum_i (w_i a_i)^2 \text{ such that } \mathbf{P}\mathbf{f}_C = \tau^*, \ \mathbf{a} \geq 0, \tag{5.5}$$

where w_i is the weight factor.[2] We did not enforce an upper bound inequality constraint, i.e., $\mathbf{a} \leq 1$, because muscles should not reach their maximum state in normal situations.

We define the antagonist muscle activation as generating an opposing force; i.e., we perform the optimization only to change the sign of generalized force:

$$\mathbf{a}_n = \underset{\mathbf{a}}{\operatorname{argmin}} \frac{1}{2} \sum_i (w_i a_i)^2 \text{ such that } \mathbf{P}\mathbf{f}_C = -\tau^*, \ \mathbf{a} \geq 0. \tag{5.6}$$

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

$$\mathbf{a} = (1+\gamma)\mathbf{a}_g + \gamma \mathbf{a}_n. \tag{5.7}$$

For computational efficiency, we divide the muscles into four groups, and perform separate optimizations for the muscles in each of the two arms, the two legs, the head-neck, and the torso. First, we optimize the activation levels of the arm, leg and head-neck muscles. Naturally, muscles that cross the torso to other body parts apply torques on joints in the torso. Therefore, we determine the activation level of the torso muscles

[2]Muscle weight factors are determined such that muscle activation levels are regularized. For sample cases, if a certain muscle's activation level exceeds 1 from solving (5.5), we increase the muscle's weight factor in order to lower its activation level and increase those of synergistic muscles.

so that the net torque is the same as the desired torque. In this way, the coactivation parameter can also be specified on a per-body-part basis.

Section 4.3 also solved two optimization problem in order to control both pose and stiffness, but their optimization problems are different—one computes agonist muscle activation similarly to ours, while the other finds a vector in the null-space of the moment arm matrix. In contrast, our new method solves one optimization problem twice, but with different equality constraints. Our method promises to be better for implementing a learning machine (say, neural networks) that can determine muscle activation levels, since just a single trained machine can determine both agonist and antagonist muscle activation levels. Additionally, in our method the stiffness is determined by the desired acceleration as well as the state of the system, while in Section 4.3 the maximum stiffness signal is determined solely from the pose.

5.3.5 Sensor Modeling

Since we apply exact control input, the resulting motion follows the target motion very well. However, this is not always desirable in computer animation. Since this method computes the required control input to create the desired motion under any given external force, the resulting motion can easily lose its natural response to external forces, making the movement look too stiff. For example, even if a ball hits the character, his motion would not be affected at all since we have computed control input to nullify the external force from the ball.[3]

Sensor Modeling can avoid this problem by differentiating the external force input to

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

the controller from the actual applied external force. The response time delay in the sensors and actuators of biological control systems yields a natural response to external forces. We use the following simple delay model which produces plausible reactive motion:

$$\frac{d}{dt}\tilde{\mathbf{f}}_e = \sigma(\mathbf{f}_e - \tilde{\mathbf{f}}_e), \tag{5.8}$$

where $\tilde{\mathbf{f}}_e$ is the *sensed* external force that is used to compute muscle force and σ is the time lag coefficient. To apply the sensor model, we replace $\mathbf{f}_e$ in (5.4) with $\tilde{\mathbf{f}}_e$ computed above. Even if the actual delay model for human sensing should be more complex, our sensor model is simple and allows intuitive control of the character's response.

5.4 Experiments and Results

Our experiments with the comprehensive, biomechanical body model range from simulating dumbbell curls to creating respiratory movement. We have produced several simulations which demonstrate that the control algorithms can actuate the elaborate musculoskeletal system in a controlled manner in order to track various input key-pose sequences.

Figure 5.15 shows a sample still frame from one of our animation experiments. As can be seen from the figure, the skin shows natural deformations caused by the activation of the muscles and the motion of the skeleton. The embedded volumetric muscles also show credible, volume-preserving bulging, which suggests that our embedded model approach is a promising technique for accurately estimating deformation of muscles.

Figure 5.16 shows a close-up of the shoulder region with different coactivations of agonist/antagonist muscle pairs. Naturally, high coactivation produces more muscle bulging, as can be seen from the deformed skin surface.

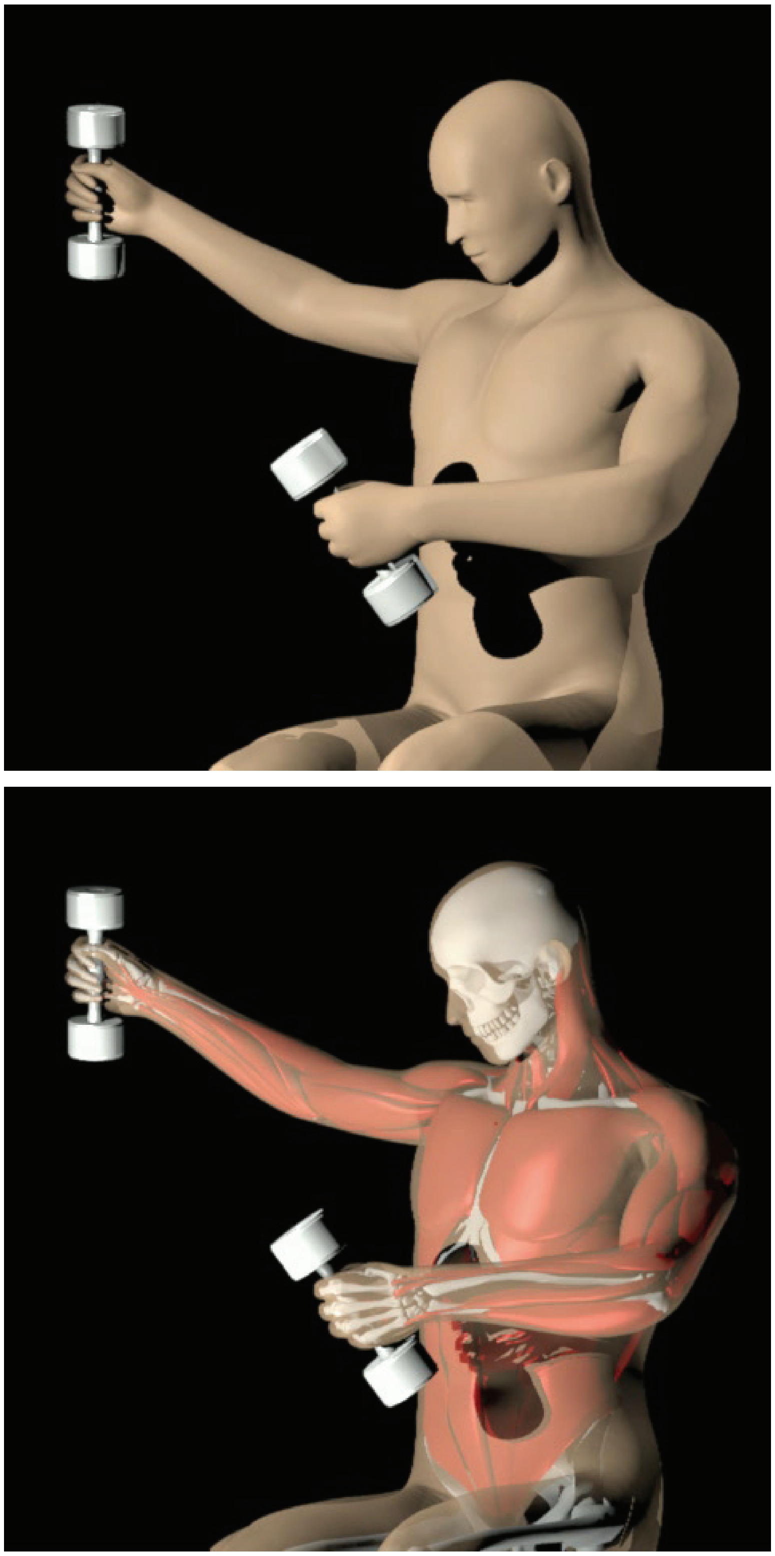


Figure 5.15: Biomechanical body model holding dumbbells. The snapshot shows that our soft tissue deformation technique produces realistic deformations of the skin surface (top) and of the embedded volumetric muscles (bottom).

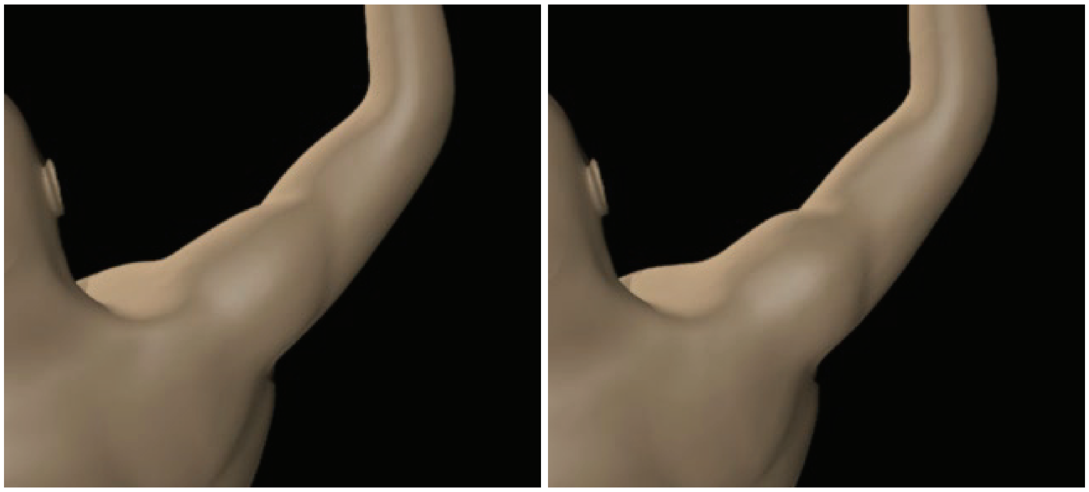


Figure 5.16: Compared to zero muscle coactivation (top), higher coactivation results in greater muscle bulging and stiffness in shoulder (bottom).

Figure 5.17 demonstrates arm flexing motions with dumbbell loads. A “dumbbell curl” animation in Figure 5.18 shows the effects of the sensor modeling in the motion controller. When the mass of the dumbbells suddenly increases, the synthetic character shows some natural failure of pose control before returning to the target pose. Without sensor modeling, the target pose would have been perfectly maintained regardless of the unexpected change of external forces.

Figure 5.19 shows the snapshots of an autonomous breathing animation in which plausible respiratory movement is produced by the intercostal muscles.

We also experimented with the creation of a jumping simulation (Figure 5.20). To balance the character, the inverse dynamics algorithm computes a fictitious exernal force which is applied to the pelvis. The remainder of the biomechanically simulated body is driven by the motion controller, which follows input target poses acquired from recorded motion data.

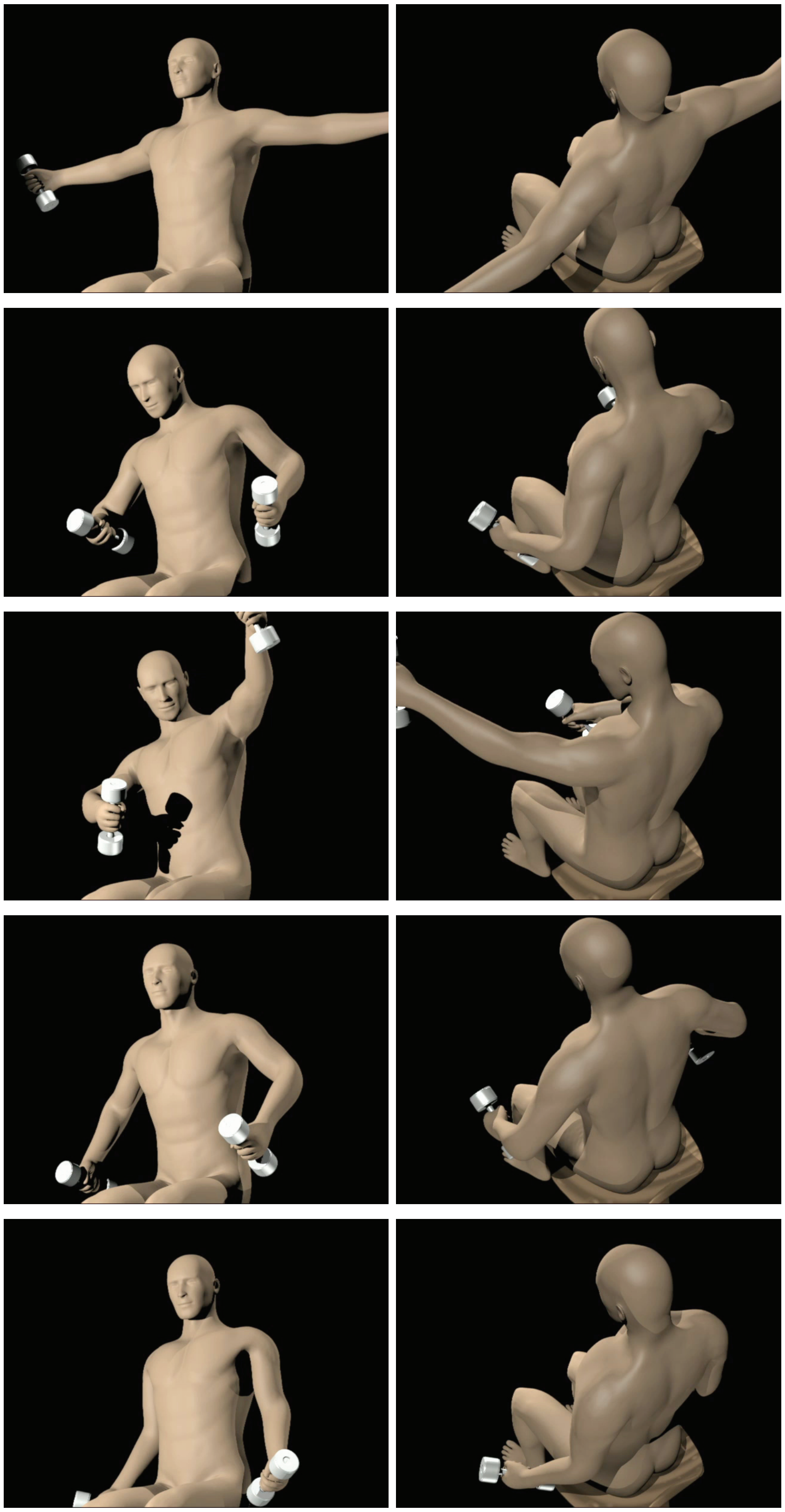


Figure 5.17: The motion controller drives the musculoskeletal system to track a sequence of target poses.

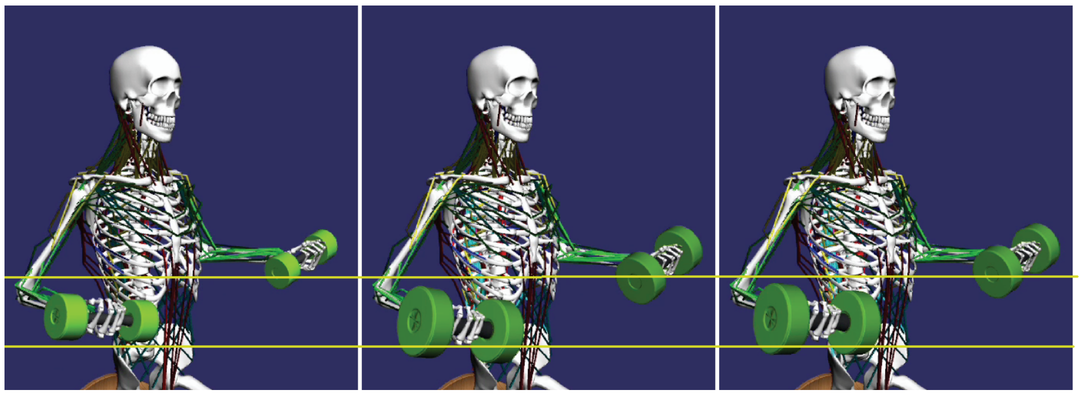


Figure 5.18: When the mass of the dumbbells increases suddenly, the arms show a natural failure to maintain the pose and are lowered for an instant (center). Soon, they return to the desired, original pose (right). Horizontal lines are drawn for the easier observation of the difference.

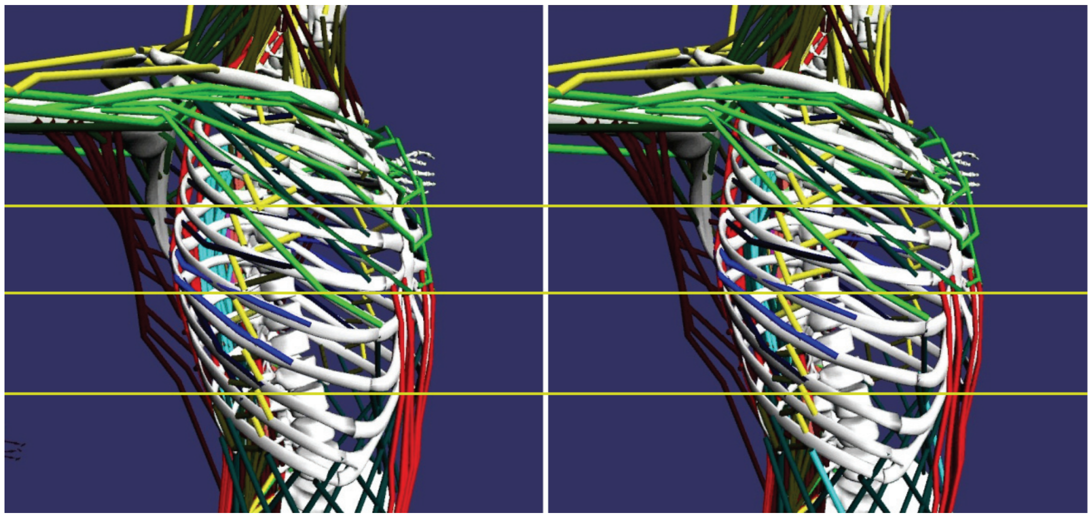


Figure 5.19: Normal breathing. The ribs in inspiration (right) are elevated than those in expiration (left). Horizontal lines are drawn for the easier observation of the differences.

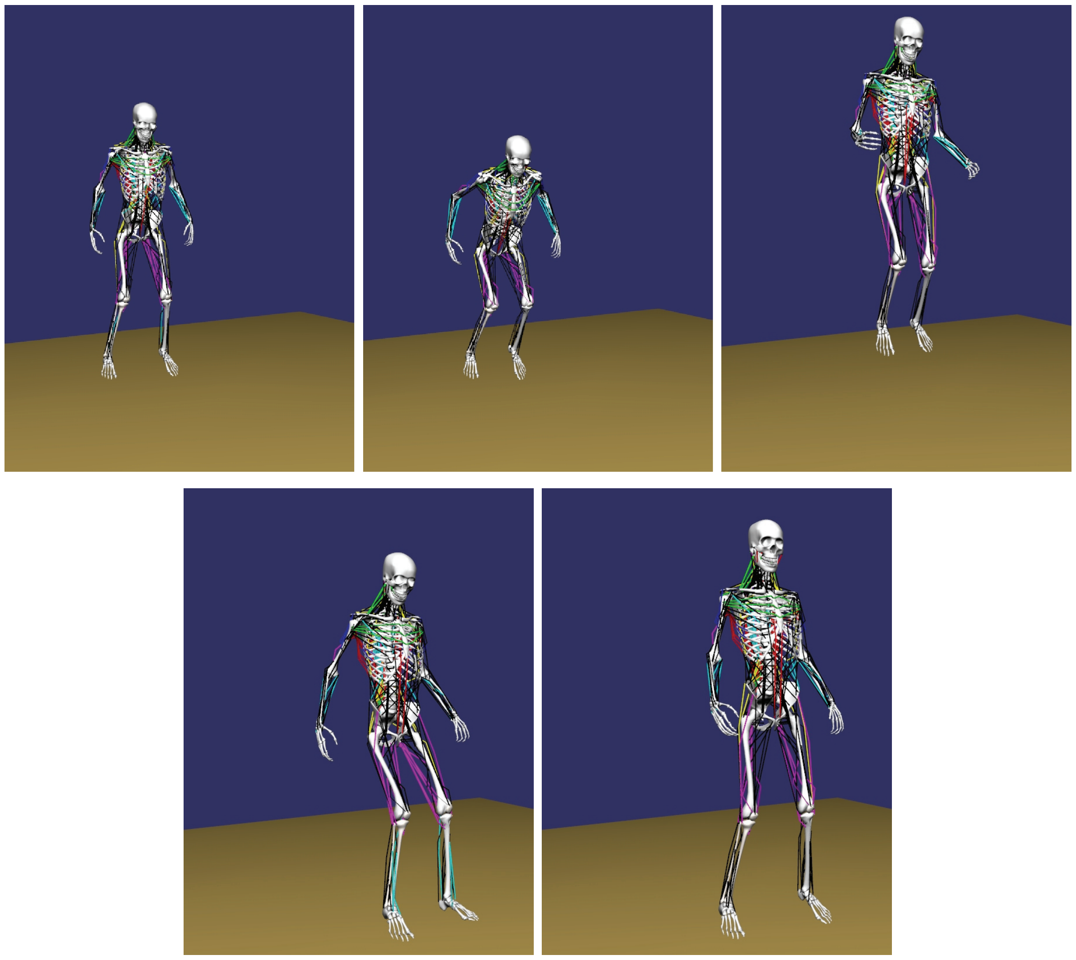


Figure 5.20: Snapshots of the jumping motion.

CHAPTER 6

Spline Joints for Biological Joint Modeling

In this chapter, we introduce *spline joints*, a novel joint model that can more accurately emulate complex biomechanical joints such as the knee and shoulder. Spline joints can model general scleronomic constraints for multibody dynamics based on the minimal coordinates formulation. The main idea is to introduce spline curves and surfaces in the modeling of joints: We model 1-DOF joints using splines on SE(3), and construct multi-DOF joints as the product of exponentials of splines in Euclidean space. We present efficient recursive algorithms to compute the derivatives of the spline joint, as well as geometric algorithms to determine optimal parameters in order to achieve the desired joint motion. Other than biological joints, the spline joints can also be used to create interesting new simulated mechanisms for computer animation.

This chapter is organized as follows: After presenting the mathematical tools that we use in this chapter (Section 6.1), we derive the kinematics and dynamics equations for a general scleronomic joint (Section 6.2). Section 6.3 defines the spline joint and Section 6.4 presents its data fitting algorithm. Finally, Section 6.5 presents our experiments and results.

6.1 Geometric Preliminaries

This section briefly introduces geometric tools derived from Lie group theory that we will use in this chapter. Readers who are familiar with differential geometry can skip this section. Additional details can be found in [Murray et al. 1994].

Given a moving body frame $\mathbf{T}(t) = (\mathbf{R}, \mathbf{p}) \in \mathrm{SE}(3)$, where $\mathbf{R} \in \mathrm{SO}(3)$ denotes rotation and $\mathbf{p} \in \mathbb{R}^3$ translation, its *generalized velocity* expressed in the instantaneous body frame (hence dubbed the *body velocity*) is defined as a *twist*

$$\hat{\mathbf{v}} = \mathbf{T}^{-1}\dot{\mathbf{T}} = \begin{bmatrix} [\omega] & \upsilon \\ 0 & 0 \end{bmatrix}, \tag{6.1}$$

which is an element of $\mathrm{se}(3)$, the Lie algebra of $\mathrm{SE}(3)$, where ω and υ are, respectively, the angular and linear velocities of $\mathbf{T}$ expressed in the body frame. The 3×3 skew-symmetric matrix of ω is denoted as $[\omega]$. We also represent the twist $\hat{\mathbf{v}}$ as a vector $\mathbf{v} = [\omega^T, \upsilon^T]^T$. The $\vee$ operator maps a twist to the corresponding twist coordinate; i.e., $\hat{\mathbf{v}}^\vee = \mathbf{v}$.

Given $\mathbf{T} \in \mathrm{SE}(3)$ and $\mathbf{g} = [\omega^T, \upsilon^T]^T \in \mathrm{se}(3)$, the adjoint mapping $\mathrm{Ad}_{\mathbf{T}} : \mathrm{se}(3) \mapsto \mathrm{se}(3)$ is defined as $\mathrm{Ad}_{\mathbf{T}}\, \hat{\mathbf{g}} = \mathbf{T}\hat{\mathbf{g}}\mathbf{T}^{-1}$, or in matrix form as

$$\mathrm{Ad}_{\mathbf{T}}\, \mathbf{g} = \begin{bmatrix} \mathbf{R} & 0 \\ [\mathbf{p}]\,\mathbf{R} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \omega \\ \upsilon \end{bmatrix}. \tag{6.2}$$

The adjoint mapping is used in the coordinate transformation of twists. As we will see in Section 6.2, the body velocity of frame $\{i-1\}$ expressed in frame $\{i\}$ is written as

$${}^{i}\mathbf{v}_{i-1} = \mathrm{Ad}_{\mathbf{G}_i^{-1}}\mathbf{v}_{i-1}, \tag{6.3}$$

where $\mathbf{G}_i$ is the configuration of frame $\{i\}$ with respect to $\{i-1\}$.

Another useful operator is the Lie bracket $\mathrm{ad}_{\hat{\mathbf{g}}} : \mathrm{se}(3) \mapsto \mathrm{se}(3)$ and it occurs when (6.2) is differentiated. The Lie bracket is defined as $\mathrm{ad}_{\hat{\mathbf{g}}_1} \hat{\mathbf{g}}_2 = \hat{\mathbf{g}}_1 \hat{\mathbf{g}}_2 - \hat{\mathbf{g}}_2 \hat{\mathbf{g}}_1$, or

$$\mathrm{ad}_{\mathbf{g}_1} \mathbf{g}_2 = \begin{bmatrix} [\omega_1] & 0 \\ [\upsilon_1] & [\omega_1] \end{bmatrix} \begin{bmatrix} \omega_2 \\ \upsilon_2 \end{bmatrix}. \tag{6.4}$$

The generalized force $\mathbf{f} = [\mu^T, \eta^T]^T$ is an element of $\mathrm{se}^*(3)$, the dual space of $\mathrm{se}(3)$, where $\mu \in \mathbb{R}^3$ represents a moment and $\eta \in \mathbb{R}^3$ a force. In matrix form, the corresponding dual adjoint mappings $\mathrm{Ad}^*_{\mathbf{T}} : \mathrm{se}^*(3) \mapsto \mathrm{se}^*(3)$ and $\mathrm{ad}^*_{\mathbf{g}} : \mathrm{se}^*(3) \mapsto \mathrm{se}^*(3)$ are the transposes of $\mathrm{Ad}_{\mathbf{T}}$ and $\mathrm{ad}_{\mathbf{g}}$; i.e.,

$$\mathrm{Ad}^*_{\mathbf{T}} = \mathrm{Ad}^T_{\mathbf{T}}, \qquad \mathrm{ad}^*_{\mathbf{g}} = \mathrm{ad}^T_{\mathbf{g}}. \tag{6.5}$$

One can easily verify that $\mathrm{Ad}^{-1}_{\mathbf{T}} \mathbf{g} = \mathrm{Ad}_{\mathbf{T}^{-1}} \mathbf{g}$ and $\mathrm{ad}_{\mathbf{g}} \mathbf{g} = 0$.

For all $\mathbf{g} \in \mathrm{se}(3)$, $e^{\hat{\mathbf{g}}}$ is an element of $\mathrm{SE}(3)$. There exists a closed-form formula of the exponential map $\exp : \mathrm{se}(3) \mapsto \mathrm{SE}(3)$ [Murray et al. 1994]. Note that the derivative of the exponential map is not trivial; $e^{\hat{\mathbf{g}}(t)} \frac{d\hat{\mathbf{g}}(t)}{dt} \neq \frac{d}{dt} e^{\hat{\mathbf{g}}(t)} \neq \frac{d\hat{\mathbf{g}}(t)}{dt} e^{\hat{\mathbf{g}}(t)}$ in general. However, in the case where the rigid motion is due to a constant twist, its derivative takes the following simple form:

$$\frac{d}{dt} e^{\hat{\mathbf{s}}\rho(t)} = e^{\hat{\mathbf{s}}\rho(t)} \hat{\mathbf{s}} \dot{\rho}(t) = \hat{\mathbf{s}} e^{\hat{\mathbf{s}}\rho(t)} \dot{\rho}(t). \tag{6.6}$$

Finally, using the notations defined above, the Newton-Euler equations of the motion

q; $\mathbf{q}$	Generalized coordinate; vector of generalized coords
$\breve{q}_i$	The i^{th} knot
$\dot{x}$	Differentiation of x with respect to time t
x'	Differentiation of x with respect to $\mathbf{q}$
$\mathbf{T}_i$	The configuration of frame $\{i\}$ w.r.t. the inertial frame
$\mathbf{G}_i$	The configuration of frame $\{i\}$ w.r.t. its parent frame
$\breve{\mathbf{G}}_i$	The i^{th} control frame
$\mathbf{z}_i$	The i^{th} control twist
$\mathbf{v}_i$	The body velocity of $\mathbf{T}_i$
$\mathbf{u}_i$	The body velocity of $\mathbf{G}_i$
$\mathbf{S}_i$	The joint Jacobian of frame $\{i\}$

Table 6.1: Frequently used symbols.

of a rigid body are expressed in a simple form as follows [Park et al. 1995]:

$$\mathbf{f} = \mathbf{J}\dot{\mathbf{v}} - \mathrm{ad}^*_{\mathbf{v}}\mathbf{J}\mathbf{v}, \tag{6.7}$$

where $\mathbf{f} \in \mathrm{se}^*(3)$ is the generalized force applied to the rigid body and $\mathbf{v} \in \mathrm{se}(3)$ is the generalized velocity. The *generalized inertia* $\mathbf{J} \in \mathbb{R}^{6\times 6}$ of the rigid body has the following structure:

$$\mathbf{J} = \begin{bmatrix} \mathscr{I} & m[\mathbf{r}] \\ m[\mathbf{r}]^T & m\mathbf{I} \end{bmatrix}, \tag{6.8}$$

where m is the mass, $\mathscr{I} \in \mathbb{R}^{3\times 3}$ is the rotational inertia matrix, $\mathbf{r} \in \mathbb{R}^3$ is the position of the center of mass, and $\mathbf{I}$ is the identity matrix. Eq. (6.7) is coordinate-invariant; i.e., it holds with respect to any coordinate frame.

Table 6.1 presents a list of symbols that we will use frequently.

6.2 Dynamics of General Scleronomic Joints

Based on the geometric tools introduced in Section 6.1, we will now derive the kinematics and dynamics equations for general scleronomic joints. Our derivation holds for any scleronomic joint, including the lower pair joints.

Assuming that link i of a multibody system is connected to its parent link $i-1$ via a joint, the configuration $\mathbf{T}_i \in \mathrm{SE}(3)$ of the body frame $\{i\}$ of i with respect to the inertial reference frame is

$$\mathbf{T}_i = \mathbf{T}_{i-1}\mathbf{G}_i, \tag{6.9}$$

where $\mathbf{T}_{i-1}$ is the configuration of $\{i-1\}$, and $\mathbf{G}_i$ denotes the relative configuration of $\{i\}$ with respect to $\{i-1\}$, which we will call the *joint transformation*. It is determined by the action of the joint and, for scleronomic joints, it is determined entirely by the *joint coordinate* $\mathbf{q}_i \in \mathbb{R}^n$, where n is the number of DOFs of the joint; i.e., $\mathbf{G}_i = \mathbf{G}_i(\mathbf{q}_i)$. The *generalized joint velocity* generated by the action of the joint is

$$\hat{\mathbf{u}}_i = \mathbf{G}_i^{-1}\dot{\mathbf{G}}_i. \tag{6.10}$$

Substituting (6.9) into (6.1), we can express the body velocity $\mathbf{v}_i$ as the velocity of link $i-1$ plus that of the joint:

$$\mathbf{v}_i = {}^{i}\mathbf{v}_{i-1} + \mathbf{u}_i. \tag{6.11}$$

For scleronomic joints,

$$\mathbf{u}_i(\mathbf{q}_i) = \mathbf{S}_i(\mathbf{q}_i)\dot{\mathbf{q}}_i, \tag{6.12}$$

$$\text{where} \quad \mathbf{S}_i(\mathbf{q}_i) = \left(\mathbf{G}_i^{-1}\frac{d\mathbf{G}_i}{d\mathbf{q}_i}\right)^{\vee}. \tag{6.13}$$

The $6 \times n$ *joint Jacobian* $\mathbf{S}_i$ is a mapping from the time derivatives of the joint coordinates $\dot{\mathbf{q}}_i$ to the generalized joint velocity $\mathbf{u}_i$. Using (6.13), the time derivative of $\mathbf{v}_i$ can be expressed as

$$\dot{\mathbf{v}}_i = {}^i\dot{\mathbf{v}}_{i-1} + \mathrm{ad}_{\mathbf{v}_i}\mathbf{S}_i\dot{\mathbf{q}}_i + \dot{\mathbf{q}}_i^T\nabla\mathbf{S}_i\dot{\mathbf{q}}_i + \mathbf{S}_i\ddot{\mathbf{q}}_i, \tag{6.14}$$

where $\nabla\mathbf{S}_i$ is the *joint Hessian*. Finally, from (6.7), the Newton-Euler equations of the motion of link i and the joint force or torque τ are as follows:

$$\mathbf{f}_i = \mathbf{J}_i\dot{\mathbf{v}}_i - \mathrm{ad}^*_{\mathbf{v}_i}\mathbf{J}_i\mathbf{v}_i + {}^i\mathbf{f}_{i+1} - \mathbf{f}_{e,i}, \tag{6.15}$$

$$\tau_i = \mathbf{S}_i^T\mathbf{f}_i, \tag{6.16}$$

where $\mathbf{f}_i \in \mathrm{se}^*(3)$ is the generalized force applied by link $i-1$ to link i and $\mathbf{f}_{e,i}$ is the external force (e.g., gravity) on link i. Note that ${}^i\mathbf{f}_{i+1} = \mathrm{Ad}^*_{\mathbf{G}^{-1}_{i+1}}\mathbf{f}_{i+1}$ expresses $\mathbf{f}_{i+1}$ relative to frame $\{i\}$.

Based on the above Lie group theoretic formulation of the kinematics and dynamics equations of general scleronomic joints, we derive in Appendix B an $\mathrm{O}(n)$ recursive forward dynamics algorithm for simulating such joints.

6.2.1 Creating New Joints

According to (6.9)–(6.16), the joint transformation as well as its Jacobian and Hessian are necessary for the kinematic analysis and dynamic simulation of the system. Hence, to create a new joint, it suffices to define a twice-differentiable joint transformation and its derivatives. Before defining new joints, consider the joint transformation, Jacobian, and Hessian of some simple joints.

The **helical joint** with pitch h has a joint transformation in the form of an exponential

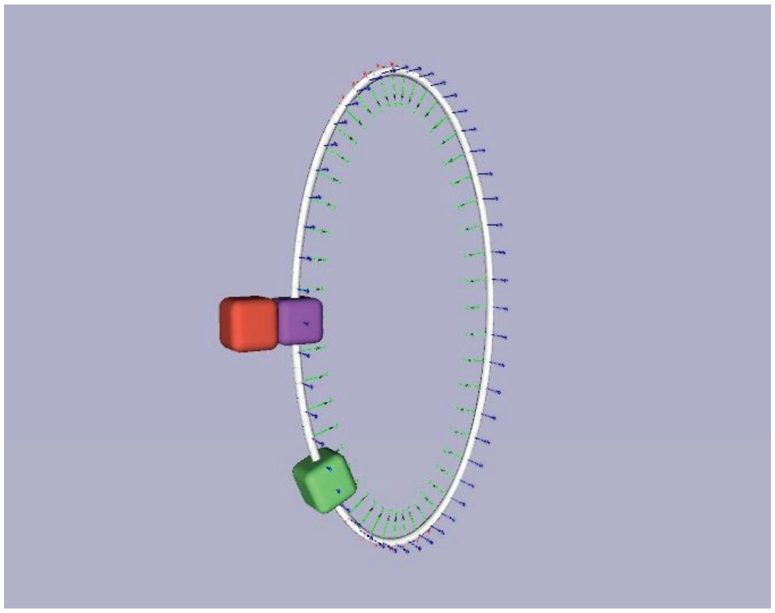


Figure 6.1: An elliptic joint. The child link (green) is constrained to slide along the ellipse attached to the parent link (purple).

of a twist of a screw parameter $\mathbf{s} = [0,0,1,0,0,h]^T$ multiplied by a joint coordinate $q \in \mathbb{R}$; i.e., $\mathbf{G}(q) = \mathbf{G}(0)e^{\hat{\mathbf{s}}q}$. Using (6.13), we can derive the joint Jacobian and Hessian as $\mathbf{S} = \mathbf{s}$ and $\nabla\mathbf{S} = \mathbf{0}$. Thus, the joint Jacobian of the helical joint is constant and it is actually the screw parameter of the joint.

The **universal joint** can be modeled as the product of two revolute joints, $\mathbf{G} = \mathbf{G}(0)e^{\hat{\mathbf{s}}_0 q_0}e^{\hat{\mathbf{s}}_1 q_1}$. Here, the joint Jacobian is $\mathbf{S} = \left[\mathrm{Ad}^{-1}_{e^{\mathbf{s}_1 q_1}}\mathbf{s}_0 \mathbf{s}_1\right] := [\mathbf{s}_0^* \; \mathbf{s}_1]$, where $\mathbf{s}_0^*$ is the instantaneous screw parameter at given q_1, and an element of the Hessian is $d\mathbf{s}_0^*/dq_1 = \mathrm{ad}_{\mathbf{s}_0^*}\mathbf{s}_1$. Note that the joint Jacobian is not constant, but is a function of the joint coordinates, which is typically the case for multi-DOF joints.

Next, consider the creation of a non-conventional joint; for example, an **elliptic joint** (Figure 6.1) that constrains the child link to slide along an ellipse with its orientation tangential to the ellipse. The joint transformation $\mathbf{G}(q)$ can be written as follows:

$$\mathbf{G}(q) = \mathbf{H}\begin{bmatrix} e^{[\mathbf{k}]\phi(q)} & (a\sin q, -b\cos q, 0)^T \\ 1 & 0 \end{bmatrix}, \tag{6.17}$$

$$\phi(q) = \operatorname{atan2}(b\sin q, a\cos q), \tag{6.18}$$

where a and b are the semi-axes of the ellipse, and $\mathbf{k}$ is a unit vector in $\mathbb{R}^3$. Once the joint transformation is defined, we must compute the Jacobian and its derivative to perform dynamic simulation. From (6.10) and (6.12),

$$\mathbf{S} = [\phi'\mathbf{k},\ e^{-[\hat{\mathbf{k}}]\phi}[a\cos q,\ b\sin q,\ 0]^T]^T, \tag{6.19}$$

$$\frac{d\mathbf{S}}{dq} = [\phi''\mathbf{k},\ e^{-[\hat{\mathbf{k}}]\phi}[(b\phi' - a)\sin q,\ (b - a\phi')\cos q,\ 0]^T]^T. \tag{6.20}$$

Since the desired joint motion is complex, defining such a joint transformation in closed form becomes difficult.

Ideally, it should be easy to compute the derivatives of the joint transformation. Since splines can effectively model complex curves and surfaces, it is natural to apply them in modeling complex joints.

6.3 Formulation of Spline Joints

We now introduce the *spline joint*, a novel class of scleronomic joints whose joint transformation is modeled using splines. Any splines may be used for the joint transformation, as long as the following two requirements are satisfied:

1. The spline joint $\mathbf{G}(\mathbf{q})$ must be C^2-continuous in order to maintain the bounded-

ness of the joint acceleration.

2. The joint Jacobian $\mathbf{S}(\mathbf{q})$ must not vanish anywhere in the domain, because the generalized joint velocity will vanish where $\mathbf{S}(\mathbf{q}) = \mathbf{0}$, even for non-zero joint coordinate velocity $\dot{\mathbf{q}}$.

Conceptually, we create the spline joint by defining the joint transformation matrix in $\mathrm{SE}(3)$ as a function of joint coordinates (an n-dimensional vector) using splines, and subsequently deriving its derivatives up to order two. However, since $\mathrm{SE}(3)$ is a curved space, it is by no means trivial to apply splines to the modeling of joint transformations.

We employ the spline curves of Kim et al. [1995] in the modeling of 1-DOF spline joints. Their spline curves in $\mathrm{SO}(3)$ can be straightforwardly extended to $\mathrm{SE}(3)$ in which our spline curve joints are defined. However, higher-dimensional splines (e.g., spline surfaces) on $\mathrm{SO}(3)$ are not yet known.[1] Hence, to create multi-DOF spline joints, we apply splines defined in Euclidean space to the twist coordinates.

6.3.1 Curve Spline Joint

The general interpolation scheme for $\mathrm{SO}(3)$ developed by Kim et al. [1995] achieves C^{k-2}-continuous rotation curves for k^{th}-order splines by introducing a cumulative form of the splines basis functions and the product of their exponentials. The merit of their method from the perspective of joint modeling is that the derivatives are easy to compute. The extension from the spline curve on $\mathrm{SO}(3)$ to $\mathrm{SE}(3)$ is straightforward and it retains all the important features such as local support and C^{k-2} continuity.

[1]Alexa [2002] defined an $\mathrm{SE}(3)$ curve as an exponential of a spline curve in Euclidean space. Similarly, a spline surface on $\mathrm{SE}(3)$ can be defined as an exponential of a spline surface in Euclidean space, but in general the derivatives of such a surface cannot be computed analytically.

While arbitrary C^2-continuous splines may be used for our purposes, consider B-splines of order $k > 3$ as a concrete example. Let $\mathbf{G}(q) \in \mathrm{SE}(3)$ be a joint transformation parameterized by $q \in \mathbb{R}$. Using the cumulative B-spline basis, $\mathbf{G}(q)$ can be expressed as the product of exponentials of a constant *control twist* $\mathbf{z}$ multiplied by a spline basis function

$$\mathbf{G}(q) = \breve{\mathbf{G}}_0 \prod_{j=1}^{m} e^{\hat{\mathbf{z}}_j \tilde{B}_{j,k}(q)}, \tag{6.21}$$

where $\mathbf{z}_j = \log(\breve{\mathbf{G}}_{j-1}^{-1} \breve{\mathbf{G}}_j)$. The cumulative basis function is defined as the sum of B-spline basis functions: $\tilde{B}_{j,k}(q) = \sum_{\ell=j}^{m} B_{\ell,k}(q)$. The *control frame* $\breve{\mathbf{G}}_j$ corresponds to control point j in the regular B-spline function. Kim et al. [1995] show that the $\tilde{B}_{j,k}(q)$ are non-constant only for $\breve{q}_j < q < \breve{q}_{j+k-1}$:

$$\tilde{B}_{j,k}(q) = \begin{cases} 0 & \text{if } q \le \breve{q}_j \\ \sum_{\ell=j}^{j+k-2} B_{\ell,k}(q) & \text{if } \breve{q}_j < q < \breve{q}_{j+k-1} \\ 1 & \text{if } q \ge \breve{q}_{j+k-1} \end{cases} \tag{6.22}$$

and that their derivatives take the simple form

$$\frac{d}{dq} \tilde{B}_{j,k}(q) = \frac{k-1}{\breve{q}_{j+k-1} - \breve{q}_j} B_{j,k-1}(q). \tag{6.23}$$

From (6.22), $e^{\hat{\mathbf{z}}_1 \tilde{B}_{1,k}(q)} = \breve{\mathbf{G}}_0^{-1} \breve{\mathbf{G}}_1$, ..., $e^{\hat{\mathbf{z}}_{j-k+1} \tilde{B}_{j-k+1,k}(q)} = \breve{\mathbf{G}}_{j-k}^{-1} \breve{\mathbf{G}}_{j-k+1}$, and $e^{\hat{\mathbf{z}}_{j+1} \tilde{B}_{j+1,k}(q)} = \cdots = e^{\hat{\mathbf{z}}_m \tilde{B}_{m,k}(q)} = \mathbf{I}$. This greatly reduces the number of multiplications and we can express the joint transformation as the product of only $k-1$ exponentials:

$$\mathbf{G}(q) = \breve{\mathbf{G}}_{j-k+1} \prod_{\ell=j-k+2}^{j} e^{\hat{\mathbf{z}}_\ell \tilde{B}_{\ell,k}(q)}. \tag{6.24}$$

We can also see that the joint transformation is locally defined by k control frames,

$\breve{\mathbf{G}}_{j-k+1}, \ldots, \breve{\mathbf{G}}_j$. In the case of the cubic B-spline, the joint transformation can be expressed as the product of just three exponentials.

6.3.2 Multi-DOF Spline Joint

Unfortunately, the spline curve on $\mathrm{SO}(3)$ does not naturally extend to the spline surface, let alone the spline surface on $\mathrm{SE}(3)$. An alternative is to model the joint transformation as the product of exponentials of six *basis twists* $\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_6$:

$$\mathbf{G}(\mathbf{q}) = \mathbf{H} \prod_{j=1}^{6} e^{\hat{\mathbf{e}}_j \phi_j(\mathbf{q})}, \tag{6.25}$$

where $\phi_j(\mathbf{q}) : \mathbb{R}^n \mapsto \mathbb{R}$ is an n-DOF spline in Euclidean space. Even though (6.25) is similar in structure to (6.21), the two are quite different; in the curve spline joint, the spline is defined in the $\mathrm{SE}(3)$ world space, whereas in multi-DOF joints the spline is defined in each dimension of $\mathrm{se}(3)$.

It is generally advantageous to use the first three basis twists to represent the translation component of the joint transformation, while the last three represent the rotation. This is equivalent to representing the rotation using Euler angles, which suffers from singularities in certain configurations. Therefore, this approach cannot cover the whole of $\mathrm{SO}(3)$. However, in many cases we can choose suitable Euler angles to avoid singularities in the presence of joint limits.

6.3.3 Derivatives of the Spline Joint Transformation

For both the curve spline joint and the multi-DOF spline joint, the joint transformation $\mathbf{G}(q)$ takes the form of a product of exponentials. Having defined the joint transformation, we must then compute its Jacobian and Hessian for the purposes of dynamics simulation. An important feature of spline joints is that the twists are constant, which enables us to compute the analytic joint Jacobian and its derivatives efficiently as follows. The n-DOF spline joint ($n \geq 1$) has the following form:

$$\mathbf{G}(\mathbf{q}) = \mathbf{H} \prod_{j=1}^{m} e^{\hat{\mathbf{g}}_j \phi_j(\mathbf{q})}, \quad \mathbf{q} \in \mathbb{R}^n, \tag{6.26}$$

where $\phi_j \in \mathbb{R}$ is some function of the joint coordinates $\mathbf{q}$ and where $\hat{\mathbf{g}}_j \in \mathrm{se}(3)$, a constant, is a control twist for the curve spline joint or a basis twist for the n-DOF joint. Given the joint transformation, the joint Jacobian consists of n twists; i.e., $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$, where

$$\mathbf{s}_i = \left(\mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial q_i} \right)^{\vee} = \sum_{k=1}^{m} \mathrm{Ad}^{-1}_{e^{\hat{\mathbf{g}}_{k+1}\phi_{k+1}} \cdots e^{\hat{\mathbf{g}}_m \phi_m}} \mathbf{g}_k \frac{\partial \phi_k}{\partial q_i}. \tag{6.27}$$

Eq. (6.27) can be computed efficiently in a recursive manner:

$$\mathbf{s}_i = \mu^i_m, \tag{6.28}$$

where $\mu^i_0 = 0$ and

$$\mu^i_k = \mathbf{g}_k \frac{\partial \phi_k}{\partial q_i} + \mathrm{Ad}^{-1}_{e^{\hat{\mathbf{g}}_k \phi_k}} \mu^i_{k-1} \quad \text{for } 1 \leq k \leq m. \tag{6.29}$$

Its derivatives are also expressed recursively:

$$\frac{\partial \mathbf{s}_i}{\partial q_j} = \frac{\partial \mu^i_m}{\partial q_j}, \tag{6.30}$$

where

$$\frac{\partial \mu^i_k}{\partial q_j} = \mathbf{g}_k \frac{\partial^2 \phi_k}{\partial q_i \partial q_j} + \mathrm{Ad}^{-1}_{e^{\hat{\mathbf{g}}_k \phi_k}} \left(\frac{\partial \mu^i_{k-1}}{\partial q_j} + \mathrm{ad}_{\mu^i_{k-1}} \mathbf{g}_k \frac{\partial \phi_k}{\partial q_j} \right). \tag{6.31}$$

The pseudo code for computing the joint derivatives is as follows:

```
Require: q
1: $\mathbf{s}_i = \mathbf{g}_1 \frac{\partial \phi_1}{\partial q_i}, \frac{\partial \mathbf{s}_i}{\partial q_j} = \mathbf{g}_1 \frac{\partial^2 \phi_1}{\partial q_i \partial q_j}$ for $i, j = 1, \ldots, n$
2: for $k = 2$ to $m$ do
3:     $\frac{\partial \mathbf{s}_i}{\partial q_j} = \mathbf{g}_k \frac{\partial^2 \phi_k}{\partial q_i \partial q_j} + \mathrm{Ad}^{-1}_{e^{\mathbf{g}_k \phi_k}} \left( \frac{\partial \mathbf{s}_i}{\partial q_j} + \mathrm{ad}_{\mathbf{s}_i} \mathbf{g}_k \frac{\partial \phi_k}{\partial q_j} \right)$
4:     $\mathbf{s}_i = \mathbf{g}_k \frac{\partial \phi_k}{\partial q_i} + \mathrm{Ad}^{-1}_{e^{\mathbf{g}_k \phi_k}} \mathbf{s}_i$
```

The complexity of this algorithm is $\mathrm{O}(n^2 m)$, where n is the number of DOFs of the joint and m is the number of exponentials. In most cases, m is small (e.g., $m = 3$ for the 1-DOF cubic spline joint) and $n \leq 2$, so this algorithm is efficient.

6.4 Designing Curve Spline Joints

A spline joint can be designed either by directly specifying the control frames or by providing a series of joint transformations for the spline joint to interpolate. For example, to model a biological joint as a spline joint, one can measure the relative transformations of the two bones in sample configurations and model a spline joint that optimally interpolates between the estimated transformations. We will present a geometric data fitting algorithm to determine the control frames in this scenario. We will focus on the 1-DOF spline curve joint.

6.4.1 Natural Parameterization

Given a set of control frames, we want to construct the spline joint by assigning knot values $\breve{q}_0, \ldots, \breve{q}_m$ to each control frame. While we can assign arbitrary ascending numbers to the knot values, a more intuitive choice would be setting the knot values in such a way that the distance between two frames equals the distance in the joint coordinate space; i.e., $||\delta \mathbf{G}|| = ||\delta q||$ or, equivalently, $||\mathbf{s}(q)|| = 1$. Note that since a twist contains both linear and angular motions, one should define a suitable metric for a particular application. While it is difficult to ensure $||\mathbf{s}(q)|| = 1$ for all q, it is easy to get approximate results by setting $\breve{q}_j = \breve{q}_{j-1} + ||\log(\mathbf{G}_{j-1}^{-1}\mathbf{G}_j)||$.[2]

6.4.2 Data Fitting Algorithm

Given a series of desired joint transformations $\mathbf{G}_d^k$ at q^k for $k = 1, \ldots, N$, the goal of the data fitting process is to compute optimal control frames $\breve{\mathbf{G}}_0, \ldots, \breve{\mathbf{G}}_m$ such that

$$\underset{\breve{\mathbf{G}}_0 \ldots \breve{\mathbf{G}}_m}{\operatorname{argmin}} \sum_{k=1}^{N} d(\mathbf{G}_d^k, \mathbf{G}(q^k)), \tag{6.32}$$

where $d(\cdot,\cdot)$ is a metric on $\mathrm{SE}(3)$. We can solve this nonlinear optimization problem by iteratively updating control frames such that the value of the objective function decreases. Imagine that the control frames $\breve{\mathbf{G}}_j$ (and spline) are moving in space and that at each iteration we update $\breve{\mathbf{G}}_j$ using the body velocity $\breve{\mathbf{u}}_j = \breve{\mathbf{G}}_j^{-1}\dot{\breve{\mathbf{G}}}_j$, so that $\mathbf{G}(q^k)$ approaches $\mathbf{G}_d^k$. To this end, we need the mapping from the velocities of the control frames to the velocities of the joint transformations at q^k.

For simplicity, let us consider the cubic B-spline case: $\mathbf{G}(q) = \breve{\mathbf{G}}_{j-3} e^{\gamma} e^{\beta} e^{\alpha}$, where $\gamma = \tilde{B}_{j-2}\hat{\mathbf{z}}_{j-2}$, $\beta = \tilde{B}_{j-1}\hat{\mathbf{z}}_{j-1}$, and $\alpha = \tilde{B}_j \hat{\mathbf{z}}_j$ for $\breve{q}_j \leq q < \breve{q}_{j+1}$. From the relation

[2]Here we employ the "unweighted" metric, but any reasonable metric can be used. For example, Kaufman et al. [2005] used the inertia-weighted metric for dynamics simulation. The unweighted metric is in some sense a geometrically reasonable one and it may give a better sense of distance to users, while the inertia-weighted metric may be better for modeling controllers.

$\dot{\mathbf{z}}_j = \breve{\mathbf{u}}_j - \mathrm{Ad}^{-1}_{e^{\mathbf{z}_j}} \breve{\mathbf{u}}_{j-1}$, we make the following approximation:

$$\left(e^{-\alpha} \frac{d}{dt} e^{\alpha} \right)^{\vee} \approx \tilde{B}_j \left(\breve{\mathbf{u}}_j - \mathrm{Ad}^{-1}_{e^{\alpha}} \breve{\mathbf{u}}_{j-1} \right). \tag{6.33}$$

Then, the velocity of the joint transformation at q can be expressed neatly as the sum of the velocities of control frames weighted by the (regular) B-spline basis functions:

$$\begin{aligned} \mathbf{u}(q) \approx B_{j-3} \mathrm{Ad}^{-1}_{e^{\gamma} e^{\beta} e^{\alpha}} \breve{\mathbf{u}}_{j-3} + B_{j-2} \mathrm{Ad}^{-1}_{e^{\beta} e^{\alpha}} \breve{\mathbf{u}}_{j-2} \\ + B_{j-1} \mathrm{Ad}^{-1}_{e^{\alpha}} \breve{\mathbf{u}}_{j-1} + B_j \breve{\mathbf{u}}_j. \end{aligned} \tag{6.34}$$

Using (6.34), we construct a matrix that relates the $\breve{\mathbf{u}}_j$ to the $\mathbf{u}(q^k)$:

$$\mathbf{M} \breve{\mathbf{u}}_c = \mathbf{u}_d, \tag{6.35}$$

where $\mathbf{M}$ is a banded matrix, $\breve{\mathbf{u}}_c = (\breve{\mathbf{u}}_0^T, \ldots, \breve{\mathbf{u}}_m^T)^T \in \mathbb{R}^{6(m+1)}$, and the desired velocities $\mathbf{u}_d = (\mathbf{u}(q^1)^T, \ldots, \mathbf{u}(q^N)^T)^T \in \mathbb{R}^{6N}$.

For $\mathbf{G}(q^k)$ to approach $\mathbf{G}_d^k$, we construct $\mathbf{M}$, specify $\mathbf{u}_d$ by setting

$$\mathbf{u}(q^k) = \log(\mathbf{G}(q^k)^{-1} \mathbf{G}_d^k), \tag{6.36}$$

solve the linear system (6.35) for $\breve{\mathbf{u}}_c$, and update $\breve{\mathbf{G}}_j \leftarrow \breve{\mathbf{G}}_j e^{\breve{\mathbf{u}}_j}$, for $j = 0, \ldots, m$, thus decreasing the value of the objective function in (6.32). We iterate these steps until convergence in order to solve the optimization problem.

6.4.3 Smoothing Algorithm

When the desired joint transformations are noisy, the resulting spline joint can produce unnatural motion. In this case, we need to smooth the spline curve in order to improve the quality of motion. An approach defining the smoothness of the spline joint is through the joint Hessian: If it is zero, the joint axis is fixed and produces simple motion like that of the lower pair joints. As in the data fitting algorithm, we compute the relationship between the rate of change of the joint Hessian at each point and the velocities of the control frames, and then we update the control frames in such a way that the joint Hessians approach zero.

We give a slightly different approximation to (6.33) for simplicity:[3]

$$\left(e^{-\alpha}\frac{d}{dt}e^{\alpha}\right)^{\vee} \approx \dot{\alpha} = \tilde{B}_j\left(\breve{\mathbf{u}}_j - \mathrm{Ad}^{-1}_{e^{\mathbf{z}_i}}\breve{\mathbf{u}}_{j-1}\right). \tag{6.37}$$

Using (6.37), we can write the time derivative of the joint Hessian (a vector) as

$$\begin{aligned}\dot{\mathbf{h}} \approx &\left(-\mathbf{N}'_{j-2}\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{z}}_{j-2}}}\right)\breve{\mathbf{u}}_{j-3} + \left(\mathbf{N}'_{j-2} - \mathbf{N}'_{j-1}\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{z}}_{j-1}}}\right)\breve{\mathbf{u}}_{j-2} \\ &+ \left(\mathbf{N}'_{j-1} - \mathbf{N}'_{j}\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{z}}_{j}}}\right)\breve{\mathbf{u}}_{j-1} + \mathbf{N}'_j\breve{\mathbf{u}}_j,\end{aligned} \tag{6.38}$$

where

$$\mathbf{N}_{j-2} = \tilde{B}'_{j-2}\mathrm{Ad}^{-1}_{e^{\beta}e^{\alpha}}, \tag{6.39}$$

$$\mathbf{N}_{j-1} = \tilde{B}'_{j-1}\mathrm{Ad}^{-1}_{e^{\alpha}} + \tilde{B}_{j-1}\mathrm{Ad}^{-1}_{e^{\beta}e^{\alpha}}\mathrm{ad}_{\gamma'}, \tag{6.40}$$

$$\mathbf{N}_j = \tilde{B}'_j\mathbf{I} + \tilde{B}_j\mathrm{Ad}^{-1}_{e^{\alpha}}\mathrm{ad}_{(\beta' + \mathrm{Ad}^{-1}_{e^{\beta}}\gamma')}. \tag{6.41}$$

One can derive the analytic derivatives of the $\mathbf{N}_j$, but numerical differentiation with

[3]Note that $e^{-\alpha}\frac{de^{\alpha}}{dt} = \dot{\alpha} + \frac{1}{2!}[\,\dot{\alpha},\alpha\,] + \frac{1}{3!}[\,[\,\dot{\alpha},\alpha\,],\alpha\,]\cdots$, where the Lie bracket $[\,\alpha,\dot{\alpha}\,] := \alpha\dot{\alpha} - \dot{\alpha}\alpha$. Therefore, the approximation approaches the exact solution when $\dot{\alpha}$ is either small or parallel to α.

respect to q is also simple and effective.

As in the data fitting case, we construct a matrix that transforms the velocities of the control frame to the rate of change of the joint Hessian:

$$\mathbf{N}\breve{\mathbf{u}}_c = \dot{\mathbf{h}}_d. \tag{6.42}$$

We can incorporate the smoothness criterion into the data fitting process by minimizing $(1-w)||\mathbf{u}_d - \mathbf{M}\breve{\mathbf{u}}_c||^2 + w||\dot{\mathbf{h}}_d - \mathbf{N}\breve{\mathbf{u}}_c||^2$, which can be accomplished by solving for $\breve{\mathbf{u}}_c$ in

$$\left((1-w)\mathbf{M}^T\mathbf{M} + w\mathbf{N}^T\mathbf{N}\right)\breve{\mathbf{u}}_c = (1-w)\mathbf{M}^T\mathbf{u}_d + w\mathbf{N}^T\dot{\mathbf{h}}_d,$$

where w is a weight and $\dot{\mathbf{h}}_d$ can simply be set to $-c\mathbf{h}$ for $0 < c \leq 1$.

The smoothing algorithm can also be used to smooth the spline curve while interpolating desired joint transformations by providing more control frames than are necessary for interpolation; i.e., since the number of control frames exceeds the minimum number required for interpolation, the extra control frames provide additional degrees of freedom to achieve smoothness.

6.5 Experiments and Results

We will now present example biological joints and some complex yet interesting mechanisms modeled with spline joints.

Figure 6.2 shows the femorotibial joint modeled as a curve spline joint. We created the desired joint transformations by posing the tibia by eye in a commercial modeling package, then applying the data fitting and smoothing algorithms to generate the con-

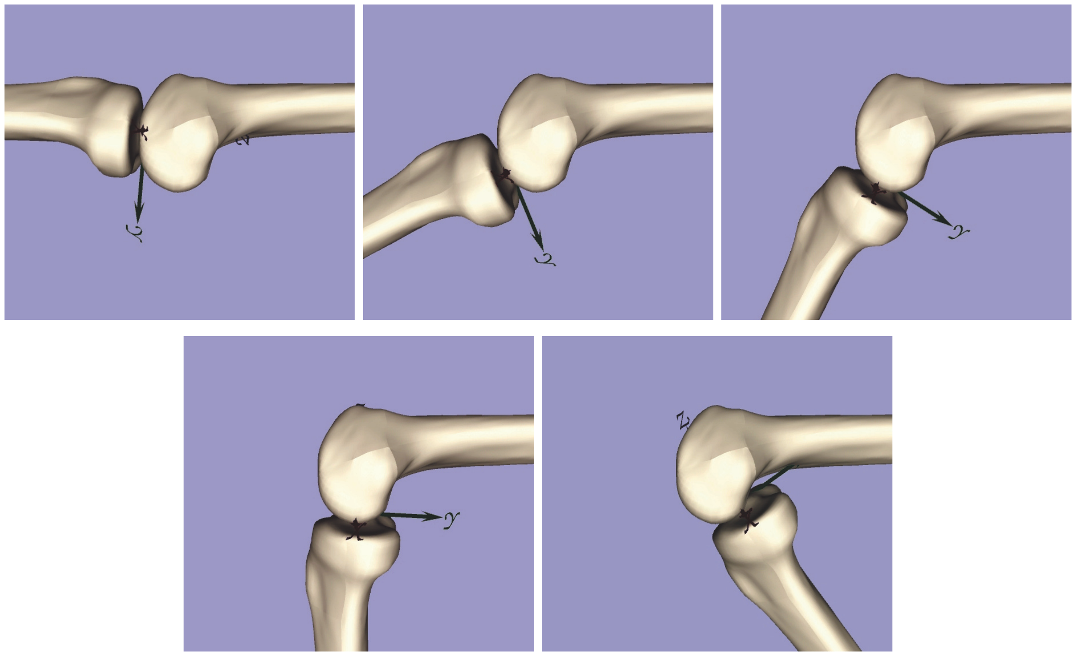


Figure 6.2: The rotation axis of the tibia moves as the joint is flexed and extended. The knee mesh data was created by Marco Viceconti and is available from www.tecno.ior.it/VRLAB.

trol frames. As the accompanying video shows, data-fitting without smoothing creates a somewhat unnatural motion. We achieve a more natural motion after applying the smoothing algorithm.

Figure 6.3 demonstrates that inverse kinematics algorithms can be applied to the spline joint. Given a target position and orientation of the foot, an iterative inverse kinematics algorithm uses the analytic Jacobian to compute the joint coordinates of the leg required to reach the target configuration. The hip and the ankle are modeled as revolute joints.

The scapulothoracic joint in the shoulder is a notorious joint to model in biomechanics. Figure 6.4 demonstrates that it can be modeled as a spline surface joint. The scapula is connected to the rib cage via a 2-DOF spline surface joint and to the clavicle via damped springs. Our approach is contrary to conventional methods in which the clavi-

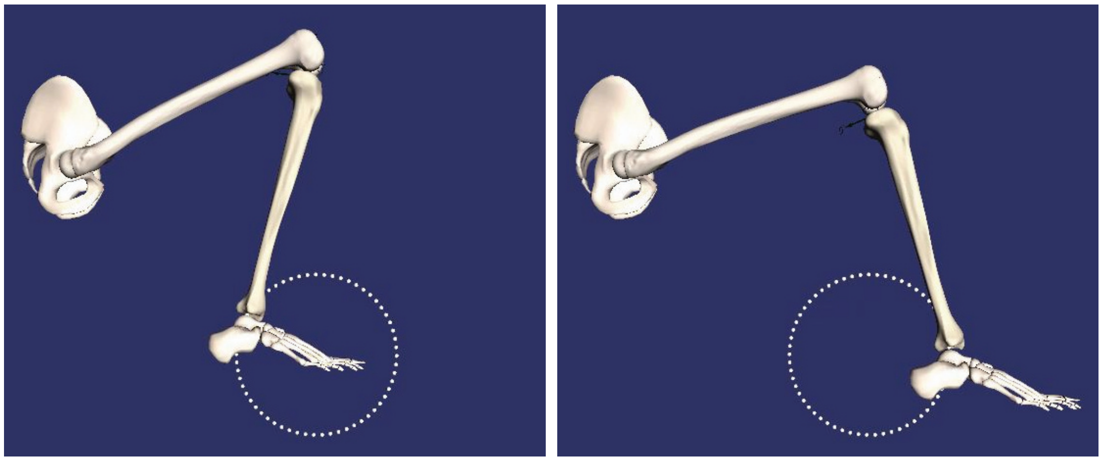


Figure 6.3: Knee spline joint inverse kinematics example.

cle and the scapula form a kinematic hierarchy via a ball joint and auxiliary constraints enforce the contact between the scapula and the thorax. Since we model the spline surface such that the pose of the scapula satisfies the constraint between the scapula and the clavicle (i.e., the distance between the acromion process and the sternoclavicular joint is approximately the same as the length of the clavicle), it is easy to enforce the constraint between the scapula and the clavicle using springs. The gray surface between the scapula and the rib cage in Figure 6.4 is the surface swept by the reference frame of the scapula. The orientations of the frame at sample positions are drawn on the surface. The red box on the surface represents the current position and orientation.

Figure 6.5 illustrates some of our experiments in creating interesting mechanisms using spline joints. In Figure 6.5 (top), each of the three beads is connected to the parent link through a flower-shaped spline joint. In this example, the beads slide along the spinning wire frame in a physically realistic manner in gravity (we can easily enable them to spin freely about the curve by inserting a simple rotational joint between the bead and the spline joint). Similarly, Figure 6.5 (middle) illustrates a mechanism in which toy airplanes slide along the continental coastlines using spline joints. Note that

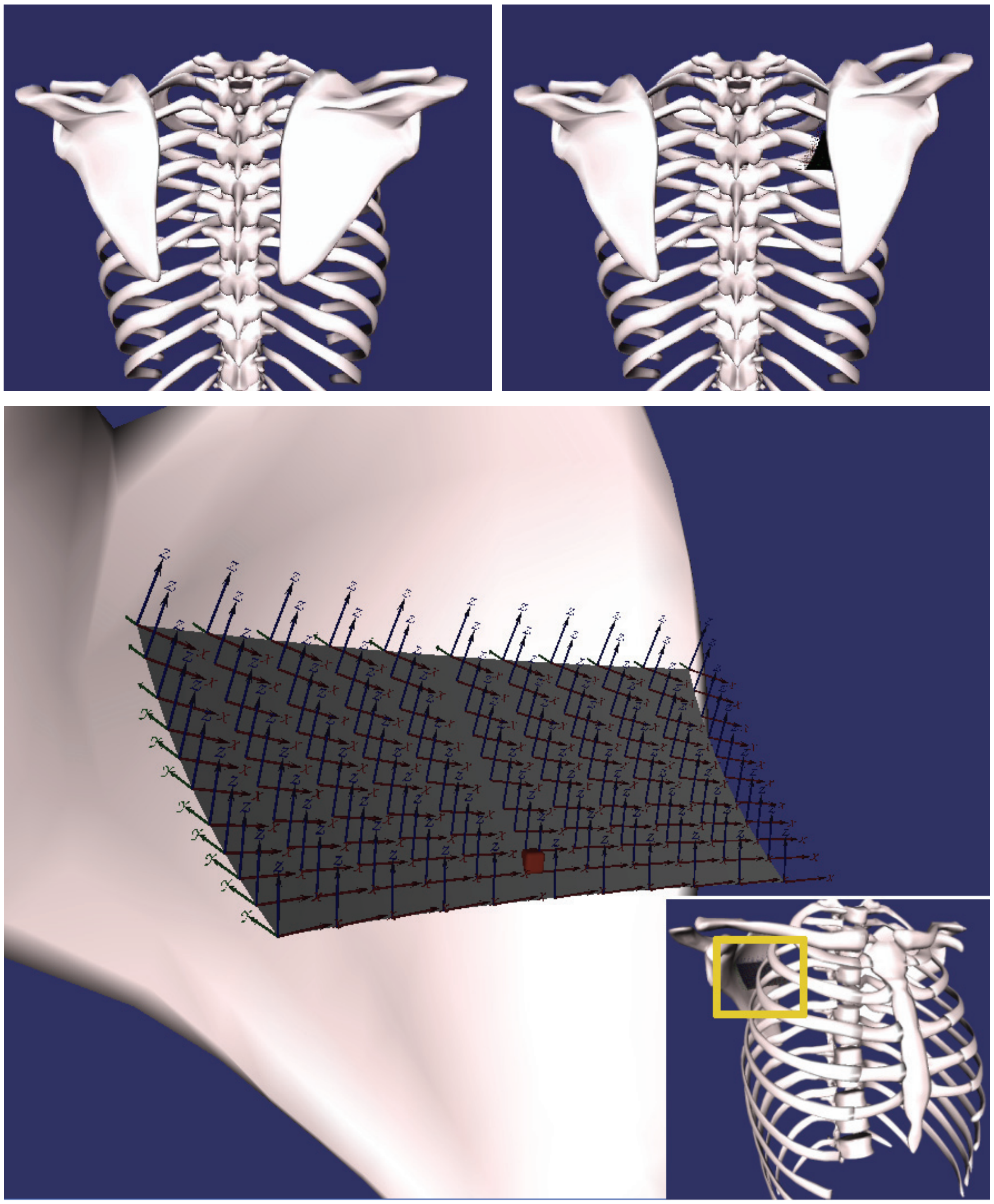


Figure 6.4: Sample poses (top) and closeup internal view (bottom) of the scapulothoracic joint modeled as a spline surface joint.

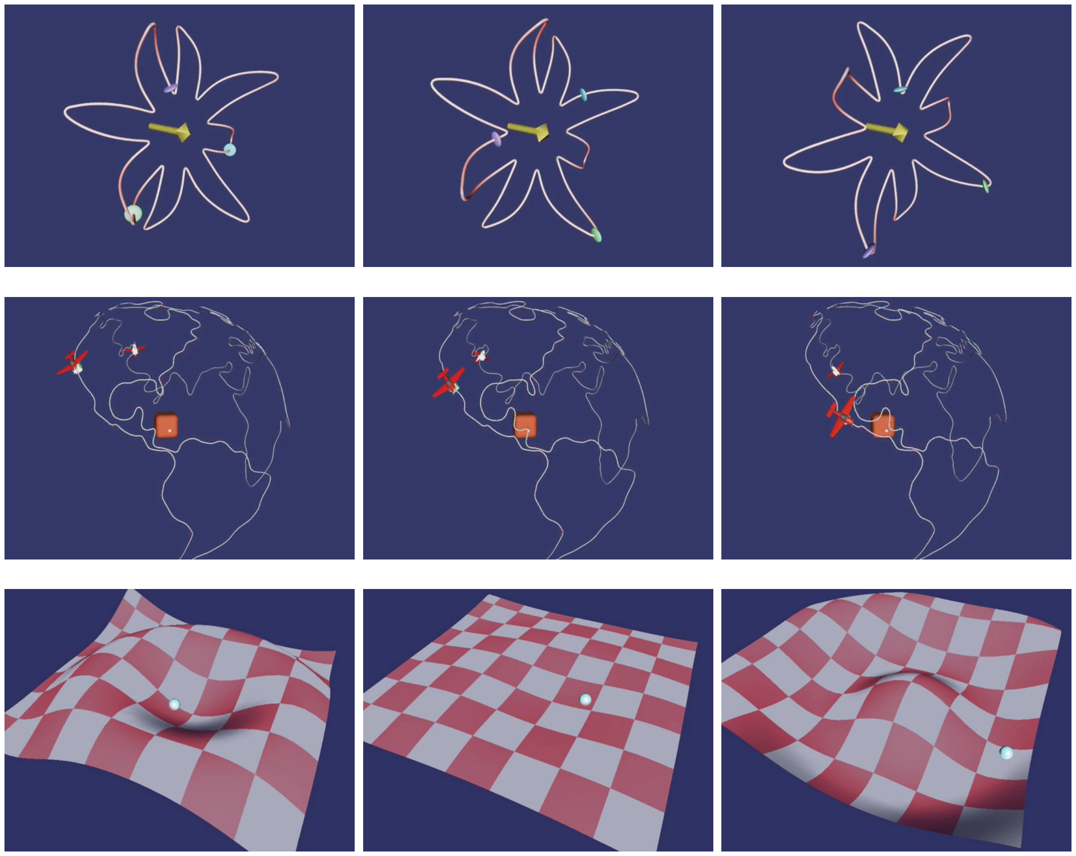


Figure 6.5: Example mechanisms modeled with spline joints. Beads-on-a-Wire (top), Globe (middle), and Deforming Spline Surface Joint (bottom).

for the "Globe" example, each control frame is computed such that two of its axes are in the tangent plane of the globe and one of those two axes is also tangent to the coastal curve. The "Beads-on-a-Wire" example is created in a similar manner. These examples demonstrate that we can create spline joints of arbitrary shape.

Figure 6.6 is a snapshot from the "SIGGRAPH Joint 1" demo. In this example, we use spline joints to constrain each letter to the surface of the parent letter. The chain of letters, which is constrained at the top of the letter S, free-falls in gravity. As we modeled only a single spline joint between a pair of letters, the contact point of one of the two links is constant while that of the other is changing. In the second example, “SIGGRAPH Joint 2” (Figure 6.7), we allow sliding on both surfaces by modeling two spline curve joints between the letters. The first curve is created from the tangent frames while the second is created from the inverse of the original control frames.

Figure 6.5 (bottom) is an example of a “Deforming Spline Surface Joint”. The cyan sphere is constrained to slide along a 2-DOF spline surface joint in a physically realistic manner, subject to gravity. At the same time, the spline surface joint deforms kinematically.

We performed our experiments on a 2.6 GHz Intel Core 2 CPU system. The spline joint permits large numerical time steps using the forward Euler time integration method. In our experiments, the maximum time step ranges from 20 msec (“Globe”) to 100 msec (“Beads-on-a-Wire”). Also, the experiments confirm that the spline joint does not raise the complexity of the dynamics formulation beyond $O(n)$. With our unoptimized implementation, the compute time per time step is 6 msec for the 1-DOF system “Knee” and 49 msec for the 8-DOF system “SIGGRAPH Joint 1”. Since we use local-support basis splines, the number of control points does not adversely affect the computational complexity.

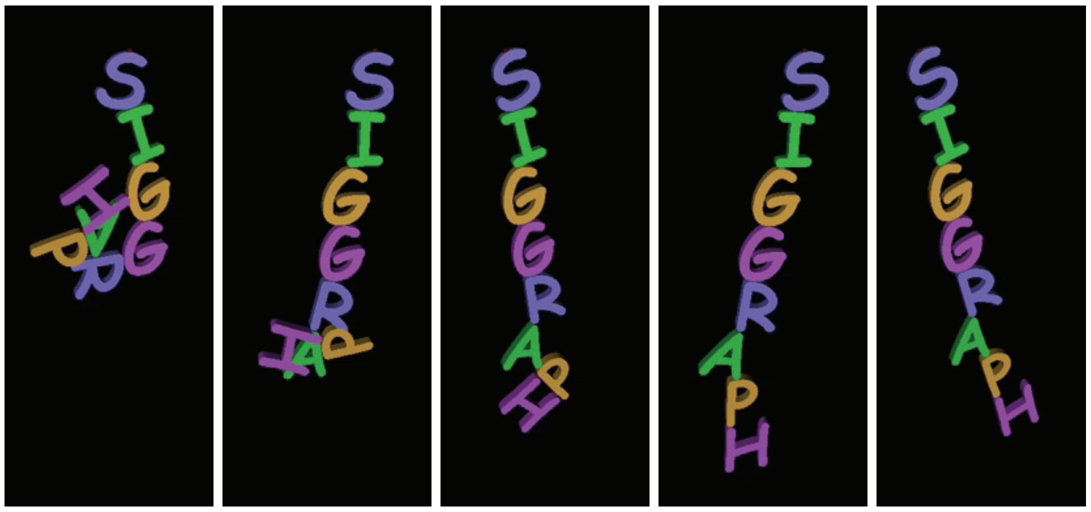


Figure 6.6: “SIGGRAPH Joint 1”. 1-DOF spline joint constrains each letter to slide along the surface of its parent letter.

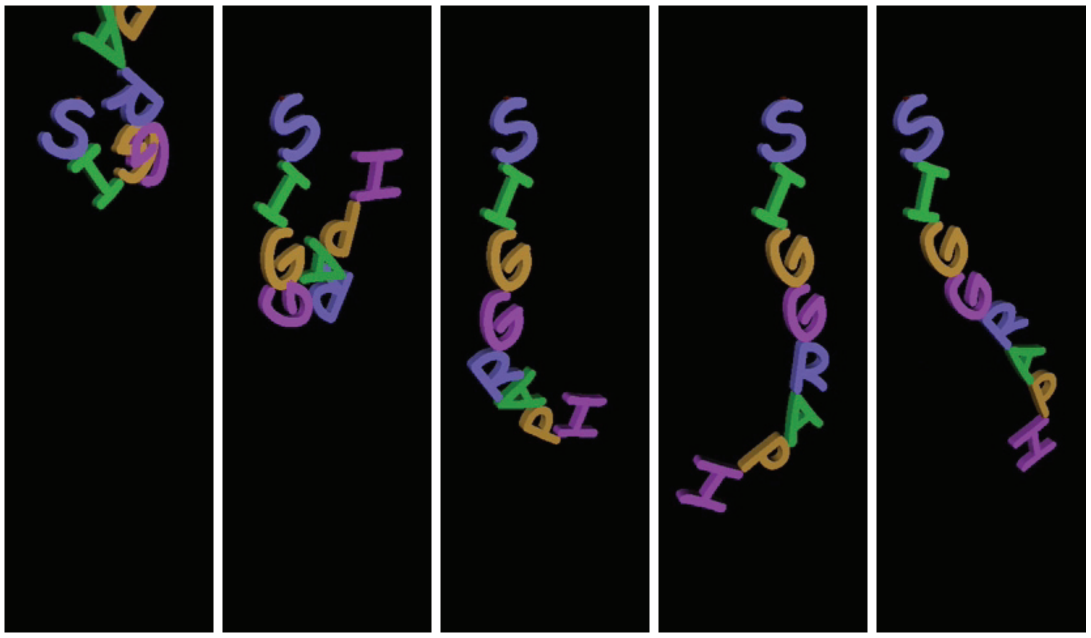


Figure 6.7: “SIGGRAPH Joint 2”. Unlike “SIGGRAPH Joint 1”, two 1-DOF joints inserted between a pair of letters allow sliding of both letters in a pair, as can be clearly seen from the letters S-I and G-R pairs.

CHAPTER 7

Conclusion

7.1 Summary

In this dissertation, we have investigated the biomechanical modeling and control of the human body for the purposes of computer animation.

First, we focused on the neck-head-face complex. Emulating the relevant anatomy, we developed a head-neck model that is characterized by kinematic redundancy (7 cervical vertebrae coupled by 3-DOF joints), as well as muscle actuator redundancy (72 neck muscles arranged in 3 muscle layers). To control the biomechanical model in order to animate the natural motions of the human head, we developed a hierarchical neuromuscular control model that mimics the relevant biological motor control mechanisms. Incorporating a low-level reflex sub-controller, an intermediate-level voluntary sub-controller, and a high-level head motion controller, our novel head-neck control system not only provides inputs to the numerous muscle actuators, but also affords control over muscle tone, which determines the stiffness of the craniocervical multibody system independently of head pose and movement. We showed that it is possible to train the neural networks in our neuromuscular controller offline so that they can efficiently generate the online pose and tone control signals that are required to produce a variety of natural head movements for the autonomous, behavioral animation of the human head and face.

Second, we expanded our interest to the entire body. We introduced a highly-detailed, biomechanical model of the human body that comprises a dynamic, articulated skeleton (75 bones), numerous Hill-type muscle actuators (846 muscle forces), and a finite element simulation of soft tissues. We were able to achieve reasonably fast performance in soft tissue simulation by decoupling the visualization geometry from the simulation geometry, using an embedded model. To tackle the complexity of controlling the skeletal model in the presence of both active and passive joints, we developed a hybrid system dynamics algorithm in the feedforward controller. Additionally, we presented an improved method to compute muscle activation levels by explicitly computing agonist and antagonist muscle forces.

Third, we focused our attention on the accurate modeling of biological joints. To this end, we introduced the spline joint technique which opens up a range of possibilities in modeling dynamic structures and nicely avoids the slower maximal-coordinates-based approach that, in any case, would not simplify the representation of scleronomic constraints. We also demonstrated that spline joints can be used to create interestingly intricate mechanisms for computer graphics.

7.2 Comparison Against Competing Approaches

Biomechanical musculoskeletal simulation governed by neuromuscular and behavioral control layers seems to be the scientifically principled approach to building self-animating, lifelike characters. In particular, our head-neck model and the comprehensive body model aspire to be significantly more biomimetic than simpler joint-torque-driven articulated models inspired by robotics [Neff and Fiume 2002; Faloutsos et al.

2001; Hodgins et al. 1995]. At least for the time being, we believe that it addresses the modeling challenge at the right level of detail. Our work has made progress toward a complete and fully integrated human body simulation in anticipation of an inevitable biomechanical/functional emulation of the whole human body for the purposes of computer animation.

The salient details of human movement cannot easily be mimicked using conventional joint-actuated skeletal models. In particular, the moment-generating capacity of each joint varies—it is determined by the geometry and capacities of the associated muscles. The muscle itself cannot simply be replaced with a PD-servo—it has nontrivial passive dynamic and force-generating properties, as approximated by the Hill model. Our controllers compute the activation level of each muscle, and this provides a natural approach to simulating local skin deformation due to underlying muscle contraction and bulging.

Despite the rich history of biomechanical modeling and control research, we have yet to see a truly integrated muscle-controlled human animation system (except perhaps in the case of facial animation), where volume-preserving 3D soft-tissue muscles actuate hard tissues (bones) and deform the surrounding skin. Our detailed musculoskeletal and soft tissue model represents an important stride toward this challenging long-term objective. Since our soft tissue model approximates the actual deformation of each muscle, instead of approximating muscle forces using line segment models, we can directly compute the moment arm of each muscle from the soft tissue simulator. Moreover, by avoiding the urge to lump all the inertial properties of the surrounding tissues into the bones, the soft tissues can retain their inertial properties, enabling a more accurate dynamics simulation.

Many complex biological joints of animals (including humans) can be more accurately modeled using our spline joint technique. For example, we demonstrated that the femorotibial joint and the scapulothoracic joint can be modeled more accurately using spline joints. We modeled these knee and shoulder examples by eye; more accurate modeling should be possible using medical imaging techniques. In the case of human joints other than the knee and scapula, the condyloid joint types (e.g., the wrist) and saddle joint types (e.g., the thumb) should benefit the most from our technique. Also, as demonstrated in the "SIGGRAPH Joint 2" (Example 6.7), surface-to-surface contact can be simulated by multiplying one spline surface joint with a second "inverted" spline surface joint. Moreover, it is possible to enforce additional constraints, such as a no-slip constraint, by computing constraint forces using Lagrange multipliers.

In our work, we used static optimization to compute optimal muscle activations that generate desired joint space torques. By contrast, some motion control schemes employ more costly dynamic optimizations to solve for optimal actuator input temporal functions that generate desired output motions. Our static optimization yields satisfactory results. Actually, the difference between static optimization and dynamic optimization may not normally be very significant; in the context of normal human gait, Anderson and Pandy [2001] argue that static optimization and dynamic optimization solutions are virtually equivalent. However, the difference may become significant for more vigorous motions such as throwing a ball.

7.3 Future Work

In view of the complexity of the human body, our biomechanical model is inevitably incomplete. An important objective for future work would be to extend our modeling framework to encompass the entire human musculoskeletal system, employ spline joints throughout, and integrate fully compatible, biomechanical models of the face, hands and feet, as well as to generalize our neuromuscular control paradigm for the neck, including control learning, so that it can operate this complete simulated body. It would surely be a significant control challenge to enable such a comprehensive biomechanical model to locomote autonomously while maintaining dynamic balance in gravity. This has been a quagmire for physics-based human simulation. Despite considerable efforts with comparatively simple bipedal characters in simulated physical environments [Hodgins et al. 1995; Faloutsos et al. 2001; Yin et al. 2007], control paradigms that can produce bipedal locomotions with realistic transitions not subject to extraneous constraints remain to be achieved, particularly for a realistic, muscle-actuated biomechanical model like ours.

In our work to date, we have treated each muscle force as an independent actuator and computed its activation level. However, during normal human movements, muscles show highly correlated patterns of activity. We have applied principal component analysis (PCA) to a set of sample activation levels of the 422 torso muscles and satisfied ourselves that only 50 to 100 basis vectors can approximate the sample activations reasonably well. This suggests that an immediate opportunity for future work would be to develop a more efficient muscle controller for the entire body that utilizes a lower-dimensional control space through dimensionality reduction and, ultimately, a machine learning approach which generalizes the one that we demonstrated for the neck (Section 4.3).

Note that our pose controller for the neck-head complex assumes that the global orientation of the musculoskeletal system (i.e., the orientation of the base link) is upright. In other words, it would not output the proper feedforward signal if the system is oriented horizontally. Since neck functionality in arbitrary orientations must be addressed when developing a model of the complete body, one should introduce the global orientation of the system as an additional input to the pose controller. This will require suitably augmented neural networks and re-training on augmented data incorporating global orientation.

Our comprehensive body examples employed a quasistatic time-integration scheme for soft tissue simulation, lacking proper resolution of inertial motion effects. This was a choice motivated by the lower computational cost of a quasistatic simulation. Nevertheness, using an implicit backward Euler scheme or a semi-implicit Newmark integrator is readily supported within our framework. In our future work, we expect to leverage the performance offered by parallel and multi-core platforms to compensate for the cost of dynamic integration schemes.

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

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

Regarding the modeling of joints, since we modeled the scapula as rigidly attached to the clavicle, the shoulder complex is restricted to a moderate range of motion, which also limits the richness of the resulting soft tissue deformation. In future work we will aim to improve the modeling of this region using an advanced spline joint modeling technique such as shown in Figure 6.4

With regard to our spline joint technique per se, since we make use of twice differentiable splines, our technique can best be used for modeling “smooth” joints. However, we believe that sharp corners can be reasonably approximated in most cases by smooth splines using multiple control frames near the discontinuities. If one wants to model sharp corners or cusps using non-C^2-continuous splines, it would be necessary to compute and apply an appropriate impulse at the discontinuities. We did not consider the data fitting problem for multi-DOF joints. A naive approach would be to apply least-squares fitting to each of 6 splines because, unlike the 1-DOF case, each spline is defined in Euclidean space. Presumably this coordinate-wise fitting will give reasonable results; however, it may not be optimal in terms of the given metric on SE(3). A good problem for future work would be to develop an optimal data fitting algorithm for multi-DOF spline joints that respects the metric on SE(3). In the deforming spline surface joint example (Figure 6.5(d)), we deformed the surface spline kinematically and the surface itself did not participate in the dynamic simulation. In future work, however, there is no reason why we cannot employ dynamic NURBS [Terzopoulos and Qin 1994] to create a physically accurate D-NURBS surface joint which would deform elastically under the weight of the ball or any other applied forces.

As the demonstrations in Figures 4.7 and 4.8 suggest, a further developed version of our biomechanical model with refined neuromuscular controllers and expanded behavioral repertoire shows promise as an essential component of future autonomous, intelligent virtual humans. To this end, we need to focus more effort on the face. The face model that we used in our neck-head-face system is only an intermediate solution. It beckons for improvement, probably using the approach in [Sifakis 2007], whose modeling and simulation is in certain ways more compatible with the rest of our body model. Of course, this would tighten the coupling between the biomechanical neck and face models. Currently, the dynamics of the neck do not adequately propagate to the face or vice versa. A proper coupling would yield more interesting dynamic animations of the face, including facial soft tissue deformations when the head is moved vigorously.

Finally, additional interesting venues of future work would be to develop associated algorithms for creating person-specific biomechanical models for use in, say, surgery simulation, as well as adapting and extending our framework to the modeling of non-human primates and other lower animals.

APPENDIX A

CE Contribution to the Stiffness

From $\mathbf{P}(\mathbf{q}) = (\partial \mathbf{l}/\partial \mathbf{q})^T$ and $\frac{1}{2}\delta\mathbf{q}^T\mathbf{K}_J\delta\mathbf{q} = \frac{1}{2}\delta\mathbf{l}^T\mathbf{K}_M\delta\mathbf{l}$, where $\mathbf{K}_M = \mathrm{d}iag(k_1, \ldots, k_m)$ and k_i is the stiffness of muscle i, we obtain the joint space representation of muscle stiffness $\mathbf{K}_J = \mathbf{P}(\mathbf{q})\mathbf{K}_M\mathbf{P}(\mathbf{q})^T$. Since k_i is always positive, $\mathbf{K}_J$ is a positive definite matrix, thus increasing the overall stability of the system. Consider the stiffness of a muscle due to its contractile element k_C in our muscle model. From (5.1),

$$k_C = \frac{\partial f_C}{\partial l} \propto k_{\max}a + \frac{\partial a}{\partial l}F_l.$$

Here, $k_{\max}a$ is the intrinsic stiffness of a muscle, which is effective regardless of the frequency of a perturbation. The reflexive stiffness due to the reflex control is $(\partial a/\partial l)F_l \propto k_pF_l$. Note that, unlike the intrinsic stiffness, the reflexive stiffness is effective only for slower perturbations, since there is a time lag for a reflexive response due to the low speed of neural information delivery. Coactivating muscles increases intrinsic stiffness; hence it is more effective for suppressing quicker perturbations than reflex control.

APPENDIX B

The Recursive Forward Dynamics Algorithm

For use with our spline joints, we derive a Lie Group theoretic recursive dynamics formulation that extends the one in [Park et al. 1995]. Note that the algorithm is essentially the same as the articulated body method for multiple DOF joints developed in [Featherstone 1987]. Our recursive forward dynamics algorithm is $\mathrm{O}(n)$ for tree-type articulated structures.

The articulated inertia $\tilde{\mathbf{J}}_i$ and its associated bias force $\mathbf{b}_i$ are defined such that they satisfy the following relation:

$$\mathbf{f}_i = \tilde{\mathbf{J}}_i \dot{\mathbf{v}}_i + \mathbf{b}_i. \tag{B.1}$$

Substituting (6.14) into (B.1) and pre-multiplying (B.1) with $\mathbf{S}_i^T$, we can decompose $\ddot{\mathbf{q}}_i$ into two parts, one induced by τ_i and the other induced by neighboring joints, as follows:

$$\ddot{\mathbf{q}}_i = \Omega_i^{-1} \left(\tau_i - \mathbf{S}_i^T \tilde{\mathbf{J}}_i ({}^{i}\dot{\mathbf{v}}_{i-1} + \mathbf{c}_i) - \mathbf{S}_i^T \mathbf{b}_i \right), \tag{B.2}$$

where $\mathbf{c}_i = \dot{\mathbf{q}}_i^T \nabla \mathbf{S}_i \dot{\mathbf{q}}_i + \mathrm{ad}_{\mathbf{v}_i} \mathbf{u}_i$ and $\Omega_i = \mathbf{S}_i^T \tilde{\mathbf{J}}_i \mathbf{S}_i$. We want to derive recursive equations for $\tilde{\mathbf{J}}_i$ and $\mathbf{b}_i$. This is accomplished by replacing $\mathbf{f}_{i+1}$ in (6.15) with (B.1) and substituting (6.14) for $\dot{\mathbf{v}}_{i+1}$. Hence, we derive the recursive forward dynamics algorithm for general scleronomic joints as follows:

- Given τ_i and $\mathbf{f}_{e,i}$

- Update $\mathbf{G}_i$, $\mathbf{S}_i$, and $\nabla\mathbf{S}_i$.
- Forward recursion: Update $\mathbf{v}_i$ and $\mathbf{c}_i$.
- Backward recursion:

$$\begin{aligned}
\tilde{\mathbf{J}}_i &= \mathbf{J}_i + \mathrm{Ad}^*_{\mathbf{G}^{-1}_{i+1}} \tilde{\mathbf{J}}_{i+1} \mathrm{Ad}_{\mathbf{G}^{-1}_{i+1}} \\
&\quad - \mathrm{Ad}^*_{\mathbf{G}^{-1}_{i+1}} \tilde{\mathbf{J}}_{i+1} \mathbf{S}_{i+1} \Omega^{-1}_{i+1} \mathbf{S}^T_{i+1} \tilde{\mathbf{J}}_{i+1} \mathrm{Ad}_{\mathbf{G}^{-1}_{i+1}} \\
\mathbf{b}_i &= -\mathrm{ad}^*_{\mathbf{v}_i} \mathbf{J}_i \mathbf{v}_i - \mathbf{f}_{e,i} + \mathrm{Ad}^*_{\mathbf{G}^{-1}_{i+1}} (\tilde{\mathbf{J}}_{i+1} \mathbf{c}_{i+1} + \mathbf{b}_{i+1}) \\
&\quad + \mathrm{Ad}^*_{\mathbf{G}^{-1}_{i+1}} \tilde{\mathbf{J}}_{i+1} \mathbf{S}_{i+1} \Omega^{-1}_{i+1} \left(\tau_{i+1} - \mathbf{S}^T_{i+1} (\tilde{\mathbf{J}}_{i+1} \mathbf{c}_{i+1} + \mathbf{b}_{i+1})\right) \\
\Omega_i &= \mathbf{S}^T_i \tilde{\mathbf{J}}_i \mathbf{S}_i
\end{aligned}$$

- Forward recursion:

$$\begin{aligned}
\ddot{\mathbf{q}}_i &= \Omega^{-1}_i \left(\tau_i - \mathbf{S}^T_i \tilde{\mathbf{J}}_i ({}^i\dot{\mathbf{v}}_{i-1} + \mathbf{c}_i) - \mathbf{S}^T_i \mathbf{b}_i\right) \\
\dot{\mathbf{v}}_i &= {}^i\dot{\mathbf{v}}_{i-1} + \mathbf{c}_i + \mathbf{S}_i \ddot{\mathbf{q}}_i
\end{aligned}$$

APPENDIX C

Parameters of the Musculoskeletal Model

This appendix tabulates the parameters of the bones and the muscles that are modeled in Chapter 5. The parameters of the bones include the parent bone, the position and the axis of the joint, and relevant physical parameters, such as mass, rotational inertia, and the center of mass of each bone. Relevant muscle parameters are the physiological cross sectional area (PCSA), the rest length, the attachment points, and the via points, if any, to the bones. The parameters are measured from the geometry data of the bones and muscles of the purely geometric Ultimate Human Model (www.cgcharacter.com/ultimatehuman.html). The data in the table are given with respect to the default pose of the body, which is shown in Figure 5.3. For bones and muscles that exist in pairs (e.g., left and right arms), we tabulate the parameters of the tissues on the left; the bones and muscles on the right are symmetrical to their corresponding tissues on the left. We use a Cartesian coordinate system in which the *x* axis is oriented from right to left, the *y* axis from front to back, and *z* axis from bottom up. Table C.1 lists the symbols that are used in the subsequent tables.

Cs: Clavicle-scapula	Cc: Costal cartilage	C1∼7: Cervical vertebrae
Fe: Femor	Fo: Foot	Ha: Hand
Hu: Humerus	Hyo: Hyoid	L1∼5: Lumbar vertebrae
Man: Mandible	Pel: Pelvis	Rad: Radius
Rb1∼10: Rib	Sku: Skull	Ste: Sternum
T1∼12: Thoracic vertebrae	Tfp: Tibia-Fibula-Patella	Thy: Thyroid
Ul: Ulnar		
maj.: major	min.: minor	superf.: superficialis

Table C.1: Symbols used in the muscle parameter tables.

Table C.2: Parameters of the bones modeled in Chapter 5. The parameters of the bones in the right half of the body are omitted as they are symmetrical to those of the bones in the left. Joint pos: default position of the joint in world coordinates (*mm*). Type: joint type and axis. The units of mass and rotational inertia are *kg* and $kg \cdot mm^2$, respectively. COM: center of mass with respect to the joint position (*mm*). Rev: Revolute, Univ: Universal, Cla: Clavicle, Sca: Scapula, A:(0.82, 0.57, 0.09), B:(0.75, 0.64, 0.13), C:(0.76, 0.65, -0.07), D:(0.67, 0.74, -0.07), E:(0.66, 0.72, -0.21), F:(0.76, 0.61, -0.21), G:(-0.14, 0.04, -0.99), H:(0.99, 0.04, -0.06), I:(0.99, -0.03, 0.08)

Name	Parent	Joint pos	Type	Mass	Rot. inertia (Ixx Ixy Ixz Iyy Iyz Izz)	COM
Pel		0 -19 1139		16.899	339991 0 0 388019 43252 179678	0 19 -95
L5	Pel	0 -12 1130	Ball	1.351	5730 1 14 3489 -424 6623	0 -33 16
L4	L5	0 -28 1168	Ball	1.369	6311 -5 -5 3521 502 7954	0 -40 19
L3	L4	0 -33 1207	Ball	1.143	3594 0 0 3103 624 5272	0 -14 21
L2	L3	0 -32 1242	Ball	0.948	2591 0 0 2240 389 3681	0 5 19
L1	L2	0 -27 1275	Ball	0.585	1486 0 0 955 324 1683	0 2 18
T12	L1	0 -20 1306	Ball	0.944	3420 -7 6 3760 884 5496	0 34 -6
T11	T12	0 -9 1336	Ball	1.382	6759 0 0 15331 496 14551	0 31 -34
T10	T11	0 0 1367	Ball	0.155	484 9 4 99 16 510	-1 50 1
T9	T10	0 8 1397	Ball	0.168	560 0 0 120 16 614	054 0
T8	T9	0 12 1426	Ball	0.186	763 0 0 143 -9 815	0 59 3
T7	T8	0 14 1458	Ball	0.177	774 0 0 131 -13 776	0 61 4
T6	T7	0 17 1488	Ball	0.115	524 0 0 60 36 535	0 63 -2
T5	T6	0 17 1516	Ball	0.160	622 0 0 106 4 645	0 56 4
T4	T5	0 12 1541	Ball	0.142	511 0 0 127 -36 528	0 53 10
T3	T4	0 10 1567	Ball	0.275	1221 0 0 291 -120 1208	0 56 13
T2	T3	0 6 1592	Ball	0.213	689 15 8 189 -104 699	-1 42 16
T1	T2	0 0 1618	Ball	0.226	609 -7 1 355 -143 641	0 30 21
C7	T1	0 -6 1641	Ball	0.284	598 0 0 392 -94 814	0 26 14

Continued on next page

Table C.2 – continued from previous page

Name	Parent	Joint pos	Type	Mass	Rot. inertia (Ixx Ixy Ixz Iyy Iyz Izz)	COM
C6	C7	0 -10 1661	Ball	0.186	215 0 0 184 -44 313	0 16 13
C5	C6	0 -16 1685	Ball	0.301	549 1 5 423 -54 867	-1 30 9
C4	C5	0 -16 1706	Ball	0.230	315 -1 0 332 -37 549	1 23 10
C3	C4	0 -14 1727	Ball	0.222	291 0 0 256 -8 452	0 23 8
C2	C3	0 -11 1748	Ball	0.257	412 0 0 230 26 536	0 23 6
C1	C2	0 -3 1770	Ball	0.160	108 0 0 178 20 241	0 5 6
Sku	C1	0 -7 1795	Ball	4.502	36930 0 0 30623 2094 19398	0-11 57
Man	Sku	0 -28 1809	Rev X	0.514	2681 0 0 2547 -1137 2053	0 -37 -49
Hyo	C4	0 34 1727	Rev X	0.097	174 0 0 48 26 176	0 -37 9
Thy	C6	0 36 1685	Rev X	0.168	280 0 0 212 -73 242	0 -29 -14
Rb1	T1	0 -7 1635	Rev X	0.399	865 0 0 2228 -346 2518	0 -5 -6
Cla	Rb1	25 -68 1594	Univ ZY	0.305	1161 -1597 -532 3482 -327 4310	167 85 35
Sca	Cla	196 16 1617	Fixed	1.979	19309 8395 -8644 24206 7277 21370	-120 101 -105
Rb2	T2	15 4 1622	Rev A	0.421	2432 1213 1152 3881 -963 4874	80 -26 -27
Rb3	T3	14 12 1595	Rev B	0.492	4384 2983 2417 7389 -1767 9241	106 -53 -43
Rb4	T4	14 17 1569	Rev B	0.572	6703 4343 3926 11258 -2824 13538	120 -61 -55
Rb5	T5	14 19 1543	Rev B	0.518	6726 3321 4044 10743 -2812 12227	119 -51 -62
Rb6	T6	14 23 1519	Rev C	0.430	4686 1329 3812 10443 -1633 10301	131 -26 -66
Rb7	T7	17 24 1490	Rev D	0.465	4709 559 3638 9506 -1306 9231	116 -6 -61
Rb8	T8	17 25 1458	Rev E	0.430	4699 696 3321 8069 -1354 7481	107 -9 -67
Rb9	T9	18 25 1426	Rev F	0.350	2260 -189 1915 4803 -186 4635	94 17 -50
Rb10	T10	17 18 1397	Rev F	0.549	6905 1286 4608 10204 -2062 7647	95 -16 -81
Hu	Sca	202 25 1569	Ball	2.986	10194 -17858 3498 63782 1198 69667	116 35 -10
Ul	Hu	473 115 1554	Rev G	0.634	296 -121 14 13722 -6 13754	125 4 -1
Rad	Ul	477 110 1576	Rev H	0.611	407 106 -295 11549 -59 11654	111 -9 1
Ha	Rad	770 113 1569	Univ	0.387	311 146 -144 1418 51 1311	48 -8 6
Fe	Pel	93 -37 1017	Ball	9.119	659801 21 67486 663894 -7692 36006	32 0 -230
Tfp	Fe	129 -55 538	Rev I	3.270	183773 -2183 18835 182548 20719 9189	25 29 -204
Fo	Tfp	158 -27 60	Rev X	1.090	10392 908 517 2886 -3242 8577	8 -53 -39
Ste	Rb1	0 -64 1598	Univ XY	0.567	15415 -26 -33 13903 -4214 1717	-1 -48 -136
Cc2	Ste	12 -103 1547	Ball	0.049	14 8 -4 36 1 35	22 -7 4
Cc3	Ste	11 -125 1509	Ball	0.093	31 1 -11 130 3 123	31 0 3
Cc4	Ste	15 -139 1471	Ball	0.097	33 -4 -14 130 4 128	30 3 4
Cc5	Ste	19 -143 1446	Ball	0.075	21 -9 18 128 3 125	34 3 -5
Cc6	Ste	14 -142 1423	Ball	1.218	19909 -1500 6814 22316 3570 4632	45 22 -108

Table C.3: Parameters of the muscles in the neck-head. PCSA: physiological cross sectional area (mm^2); ℓ_0: rest length (mm); attachment points: bone (position of the attached (or via) points in mm). The first and the last points are attachment points and the rest are via points. Some of the spinal muscles in the neck are listed in Table C.4.

Name	PCSA	ℓ_0	Attachment (and via) points
Masseter	233	63	Sku(69, -75, 1808) Man(53, -57, 1750)
Iliocostalis	9	193	C5(37, 6, 1682) Rb2(57, 47, 1628) Rb4(64, 63, 1566) Rb6(70, 67, 1509)
cervicis	6	178	C5(29, -7, 1695) Rb2(59, 45, 1631) Rb4(69, 59, 1563) Rb5 (74, 62, 1543)
	6	151	C5(30, -8, 1695) Rb2(61, 45, 1628) Rb4(73, 55, 1570)
	9	133	C4(32, -10, 1714) Rb2(64, 43, 1630) Rb3(69, 48, 1602)
	7	123	Rb6(78, 72, 1500) Rb8(74, 77, 1439) Rb10(78, 58, 1380)
Rhomboid min.	334	94	Cs(75, 92, 1586) C7(4, 57, 1637)
Longus	4	102	Sku(6, -22, 1807) C1(11, -24, 1789) C3(16, -21, 1748) C5(21, -19, 1706)
capitis	2	130	Sku(4, -23, 1807) C2(12, -24, 1762) C4(17, -23, 1726) C6(23, -16, 1679)
	3	89	Sku(8, -22, 1808) C2(17, -23, 1761) C4(24, -18, 1720)
	2	70	Sku(10, -21, 1809) C1(14, -23, 1790) C3(25, -17, 1741)
Longissimus	10	81	C5(30, -8, 1692) C7(44, 9, 1672) T2(39, 32, 1627)
cervicis	23	131	C4(33, -11, 1712) C6(45, 5, 1682) T1(52, 25, 1653) T3(34, 46, 1601)
	8	231	C2(35, -8, 1751) C5(47, -4, 1711) C7(55, 10, 1673) T2(59, 40, 1633)
			T5(34, 47, 1539)
	18	184	C3(34, -10, 1733) C5(43, -4, 1711) C7(53, 9, 1673) T2(54, 41, 1631)
			T4(35, 47, 1568)
	3	33	C6(34, 2, 1678) T1(44, 15, 1649)
Longissimus	5	101	Sku(65, 2, 1786) C3(43, 12, 1751) C5(33, 10, 1693)
capitis	7	170	Sku(63, 16, 1788) C2(39, 22, 1743) C5(40, 35, 1707) T2(38, 32, 1627)
	1	258	T5(33, 47, 1543) C5(35, 43, 1692) C2(37, 27, 1746) Sku(61, 26, 1793)
	2	228	T4(33, 45, 1569) C6(38, 46, 1672) C4(33, 36, 1715)
			C2(36, 27, 1745) Sku(60, 25, 1789)
	5	199	T3(32, 45, 1603) C7(40, 46, 1643) C5(37, 38, 1705)
			C3(40, 23, 1752) Sku(63, 21, 1792)
	5	149	Sku(66, 11, 1789) C2(40, 20, 1744) C4(39, 21, 1738) T1(41, 16, 1649)
	7	132	Sku(65, 7, 1786) C2(41, 18, 1744) C4(40, 20, 1738) C7(38, 6, 1663)
	6	118	Sku(65, 5, 1784) C3(43, 14, 1750) C6(31, 7, 1675)
Semispinalis	32	104	C3(3, 27, 1721) C5(16, 34, 1696) C7(32, 37, 1650) T2(37, 33, 1625)
cervicis	7	174	C5(6, 33, 1684) C7(13, 43, 1650) T2(22, 50, 1608) T5(30, 52, 1538)
			T6(29, 49, 1514)
	12	117	C5(5, 31, 1688) C7(18, 42, 1650) T2(26, 46, 1622) T4(34, 45, 1577)
	13	143	C5(5, 32, 1687) C7(15, 43, 1652) T3(27, 50, 1597) T5(31, 45, 1549)

Continued on next page

Table C.3 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
	24	104	C4(3, 28, 1705) C6(16, 38, 1673) T1(26, 41, 1642) T3(31, 43, 1607)
	15	103	C2(6, 31, 1739) C4(15, 31, 1714) C6(37, 29, 1665) T1(43, 17, 1647)
Splenius capitis	11	178	T1(3, 54, 1630) C6(21, 55, 1657) C4(36, 38, 1717) C2(37, 31, 1744) Sku(55, 36, 1794)
	10	263	T4(2, 66, 1545) T2(18, 68, 1587) C7(33, 58, 1634) C5(52, 16, 1721) C3(53, 9, 1748) Sku(66, 9, 1787)
	9	231	T3(4, 65, 1578) T1(21, 65, 1621) C6(33, 53, 1654) C2(48, 18, 1742) Sku(65, 18, 1790)
	10	208	T2(3, 59, 1601) C7(20, 61, 1638) C5(36, 48, 1679) C3(44, 24, 1749) Sku(61, 27, 1792)
	7	152	Sku(50, 43, 1795) C2(33, 35, 1745) C4(31, 43, 1710) C6(4, 47, 1653)
Splenius cervicis	7	278	T5(2, 65, 1516) T1(48, 40, 1652) C4(48, 11, 1736) C1(54, 1, 1778)
	8	210	T3(2, 66, 1567) C6(46, 41, 1678) C2(35, -6, 1752)
	9	231	T4(3, 66, 1543) C7(46, 42, 1656) C2(35, -7, 1751)
	8	306	T6(2, 70, 1489) T2(44, 46, 1624) C7(51, 36, 1656) C3(49, 5, 1751) C1(56, 1, 1778)
Omohyoid	15	236	Hyo(10, -65, 1724) Cs(61, -41, 1621) Cs(143, 41, 1596)
Sternocleido-mastoid	136	204	Sku(63, 22, 1795) C4(61, -22, 1703) Cs(58, -66, 1611)
	60	234	Sku(65, 20, 1797) C5(41, -30, 1694) Ster(17, -80, 1591)
Semispinalis capitis	18	177	T2(40, 34, 1626) C5(27, 42, 1689) C3(25, 36, 1731) Sku(35, 55, 1797)
	15	251	T5(34, 47, 1545) T3(26, 58, 1597) T1(22, 55, 1624) C6(17, 45, 1670) C4(15, 40, 1711) C2(14, 39, 1745) Sku(18, 55, 1789)
	3	112	C5(31, 7, 1697) Sku(45, 50, 1799)
	6	128	C6(30, 7, 1679) C2(30, 35, 1746) Sku(43, 51, 1798)
	8	140	C7(37, 7, 1661) C3(28, 36, 1731) Sku(39, 50, 1793)
	7	159	T1(45, 16, 1648) C4(27, 38, 1712) C2(28, 35, 1745) Sku(38, 53, 1797)
	19	291	T6(33, 49, 1518) T4(29, 59, 1562) T2(21, 60, 1600) C7(16, 51, 1644) C5(13, 44, 1689) C3(11, 41, 1728) Sku(15, 65, 1800)
	19	193	T3(33, 43, 1604) C6(26, 45, 1672) C4(23, 39, 1710) C2(23, 37, 1736) Sku(30, 54, 1793)
	30	226	T4(36, 46, 1571) T2(27, 55, 1624) C7(24, 52, 1642) C5(20, 44, 1684) C3(19, 39, 1733) Sku(25, 56, 1791)
Longus colli	6	53	C4(13, -23, 1713) C5(16, -22, 1698) C6(17, -19, 1678) C7(14, -16, 1661)
	3	83	C6(24, -15, 1673) C7(20, -11, 1652) T1(15, -7, 1621) T2 (12, -4, 1595) T3(12, -4, 1592)
	6	59	C6(22, -15, 1674) C7(19, -11, 1653) T1(13, -8, 1621) T2(12, -8, 1616)
	2	66	C5(24, -18, 1698) C6(21, -15, 1674) C7(17, -12, 1654) T1(12, -13, 1634)

Continued on next page

Table C.3 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
	5	133	C2(9, -22, 1764) C3(11, -22, 1750) C4(15, -21, 1726) C5(20, -20, 1700) C6(20, -17, 1675) C7(14, -15, 1641) T1(11, -15, 1633)
	4	110	C2(11, -22, 1744) C3(12, -23, 1734) C4(14, -23, 1724) C5(18, -20, 1698) C6(18, -17, 1675) C7(13, -16, 1642) T1(11, -16, 1636)
	4	22	C5(12, -21, 1699) C6(14, -19, 1677)
Obliquus capitis	55	62	C2 (6 , 27 , 1747) C1 (52 , -2 , 1777)
	37	36	C1 (53 , 1 , 1781) Sku (42 , 34 , 1791)
Rectus capitis	105	46	C2 (3 , 32 , 1750) Sku (28 , 37 , 1788)
	72	17	C1 (5 , 30 , 1772) Sku (12 , 38 , 1786)
	26	21	C1 (23 , -19 , 1788) Sku (9 , -18 , 1804)
	44	14	C1 (41 , -13 , 1785) Sku (42 , -7 , 1797)
Geniohyoid	16	31	Man (2 , -98 , 1727) Hyo (4 , -68 , 1726)
Scalenes	33	22	Rb1 (54 , 6 , 1641) C7 (42 , 3 , 1659)
	44	117	Rb1 (62 , -55 , 1589) C5 (27 , -17 , 1694)
	50	133	Rb1 (64 , -53 , 1590) C4 (30 , -17 , 1714)
	34	152	Rb1 (67 , -51 , 1591) C3 (31 , -15 , 1734)
	27	99	Rb1 (60 , -57 , 1589) C6 (26 , -14 , 1672)
	62	123	Rb2 (107 , 10 , 1598) C5 (32 , -9 , 1694)
	24	140	Rb2 (110 , 4 , 1594) C4 (34 , -12 , 1711)
	34	105	Rb2 (103 , 16 , 1603) C6 (33 , -3 , 1679)
	28	139	Rb1 (83 , -3 , 1622) C2 (34 , -10 , 1752)
	45	118	Rb1 (80 , -1 , 1624) C3 (33 , -12 , 1732)
	77	95	Rb1 (76 , 1 , 1627) C4 (33 , -13 , 1710)
	85	72	Rb1 (69 , 5 , 1633) C5 (32 , -11 , 1692)
	70	51	Rb1 (61 , 6 , 1638) C6 (32 , -4 , 1678)
	12	152	Cs (67 , -52 , 1592) C3 (30 , -15 , 1734)
Sternohyoid	35	123	Hyo (5 , -67 , 1722) Cs (18 , -59 , 1600)
Sternothyroid	84	96	Ster (16 , -65 , 1591) Thy (18 , -53 , 1686)
Stylohyoid	6	91	Hyo (13 , -61 , 1730) Sku (48 , -12 , 1798)
Thyrohyoid	20	37	Hyo (12 , -62 , 1724) Thy (16 , -54 , 1688)
Mylohyoid	120	16	Hyo (6 , -67 , 1725) Man (18 , -77 , 1730)
Levator scapulae	60	145	C4 (31 , -8 , 1716) Cs (86 , 79 , 1614)
	45	141	Cs (89 , 76 , 1615) C4 (34 , -12 , 1711)
	77	153	Cs (95 , 71 , 1619) C3 (34 , -10 , 1734)
	74	163	Cs (100 , 66 , 1622) C2 (34 , -6 , 1752)
	43	175	Cs (106 , 63 , 1623) C1 (55 , 0 , 1778)
	106	161	Cs (101 , 65 , 1623) C2 (33 , -6 , 1751)

Continued on next page

Table C.3 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
	94	154	Cs (93 , 73 , 1619) C3 (34 , -9 , 1735)
	49	174	Cs (110 , 58 , 1623) C1 (51 , -1 , 1776)

Table C.4: Parameters of the muscles in the trunk.

Name	PCSA	ℓ_0	Attachment (and via) points
Rectus abdominus	210	396	Ster (5 , -145 , 1392) Pel (16 , -103 , 999)
	210	445	Cc5(35,-147,1440) Cc6(34,-153,1406) Cc6(32,-156,1370) Pel(16,-103,999)
	210	446	Cc5(70,-146,1437) Cc6(65,-154,1403) Cc6(57,-154,1333) Pel(16,-103,999)
Iliocostalis lumborum	24	169	T11(81, 47, 1327) L1(66, 34, 1277) L3(62, 27, 1228) L5(61, 34, 1185) Pel (61 , 40 , 1164)
	18	121	T12 (65 , 32 , 1291) Pel (68 , 33 , 1170)
	21	49	L4 (46 , -6 , 1200) Pel (66 , 28 , 1171)
	38	79	L3 (42 , -6 , 1225) Pel (55 , 38 , 1160)
	49	107	L2 (41 , 6 , 1254) Pel (52 , 44 , 1155)
	24	175	Pel (44 , 69 , 1124) L1 (41 , 11 , 1289)
	16	414	Rb5(108, 60, 1523) Rb7(100, 76, 1465) Rb9(88, 76, 1405) T11(74, 66, 1335) L2 (60 , 50 , 1250) L4 (58 , 52 , 1220) Pel (50 , 72 , 1120)
	14	358	Rb6(112, 64, 1491) Rb8(101, 75, 1429) Rb10(88, 68, 1366) L1(67, 51, 1278) L3 (61 , 46 , 1233) L5 (53 , 63 , 1152) Pel (51 , 65 , 1143)
	15	273	Rb9(105, 60, 1395) T11(84, 53, 1328) L1(70, 37, 1270) L3(63, 34, 1227) L5 (53 , 54 , 1150) Pel (56 , 48 , 1166)
	12	204	Rb10(98, 54, 1361) T12(72, 40, 1286) L2(65, 30, 1247) L4(61, 33, 1207) Pel (58 , 44 , 1166)
	12	333	Rb7(115, 68, 1461) Rb9(100, 69, 1398) T11(83, 61, 1336) L2(63, 42, 1246) L4 (61 , 44 , 1218) Pel (50 , 63 , 1141)
	15	298	Rb8(109, 68, 1428) Rb10(93, 62, 1363) L1(70, 42, 1272) L3(63, 38, 1225) L5 (52 , 56 , 1150) Pel (50 , 59 , 1142)
Iliocostalis thoracis	3	200	Rb6 (76 , 71 , 1499) Rb9 (69 , 71 , 1416) T12 (61 , 34 , 1306)
	2	217	C7(40, 5, 1663) Rb2(84, 35, 1620) Rb4(95, 60, 1566) Rb6(94, 72, 1504) Rb7 (89 , 73 , 1479)
	5	164	Rb6(76, 71, 1501) Rb8(72, 76, 1441) Rb10(70, 59, 1367) T11(72, 49, 1341)
	7	180	Rb1(64, 14, 1638) Rb3(92, 54, 1584) Rb5(94, 68, 1534) Rb7(89, 73, 1479)
	11	156	Rb2(70, 37, 1624) Rb4(91, 62, 1567) Rb6(91, 73, 1505) Rb7(88, 73, 1478)
	8	121	Rb3 (77 , 51 , 1594) Rb5 (90 , 69 , 1532) Rb7 (86 , 73 , 1477)

Continued on next page

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
	8	95	Rb4 (79 , 57 , 1567) Rb6 (88 , 73 , 1506) Rb7 (85 , 73 , 1475)
	7	67	Rb5 (81 , 63 , 1541) Rb7 (84 , 73 , 1475)
	4	59	Rb6 (81 , 69 , 1505) Rb8 (84 , 74 , 1446)
	8	96	Rb6 (79 , 69 , 1505) Rb8 (78 , 77 , 1444) Rb9 (81 , 65 , 1412)
Semispinalis thoracis	10	133	C7(8, 49, 1640) T2(16, 56, 1601) T5(27, 56, 1536) T6(29, 49, 1511)
	8	159	T4(2, 61, 1539) T6(14, 68, 1488) T8(21, 63, 1440) T10(32, 42, 1388)
	17	157	T3(2, 64, 1563) T5(13, 67, 1514) T8(24, 62, 1447) T9(30, 52, 1411)
	21	147	T2(3, 56, 1595) T4(14, 65, 1544) T7(25, 62, 1486) T8(32, 57, 1452)
	19	137	T1(5, 50, 1623) T3(16, 61, 1571) T5(23, 62, 1538) T7(28, 58, 1489)
	8	160	C6(3, 38, 1665) T1(15, 50, 1630) T4(26, 56, 1565) T6(29, 49, 1511)
Spinalis thoracis	4	354	T2 (4 , 59 , 1593) T4 (14 , 70 , 1545) T6 (18 , 72 , 1487)
			T9 (18 , 59 , 1394) T11 (11 , 44 , 1323) L1 (4 , 30 , 1260) L2 (2 , 25 , 1247)
	5	386	T1 (5 , 52 , 1622) T3 (14 , 66 , 1570) T5 (19 , 71 , 1508) T8 (21 , 67 , 1446)
			T10 (16 , 50 , 1358) T12 (10 , 38 , 1287) L2 (2 , 25 , 1245)
	6	272	T4 (2 , 63 , 1537) T6 (14 , 73 , 1485) T8 (15 , 66 , 1424) T10 (11 , 52 , 1358)
			T12 (6 , 35 , 1288) L1 (2 , 28 , 1274)
	5	299	T3 (1 , 65 , 1562) T5 (14 , 72 , 1514) T7 (17 , 70 , 1456) T9 (16 , 59 , 1393)
			T11 (10 , 44 , 1325) L1 (3 , 27 , 1272)
	7	190	T6 (3 , 69 , 1483) T8 (9 , 68 , 1424) T10 (7 , 54 , 1360) T12 (2 , 32 , 1298)
	9	220	T5 (3 , 63 , 1509) T7 (12 , 71 , 1455) T9 (11 , 60 , 1389) T11 (8 , 45 , 1323)
			T12 (2 , 32 , 1297)
	7	125	T7 (1 , 67 , 1453) T9 (6 , 62 , 1389) T11 (2 , 40 , 1332)
Trapezius	202	342	L1 (3 , 37 , 1288) Cs (66 , 121 , 1493) Cs (125 , 101 , 1585)
	202	304	T10 (2 , 60 , 1371) Cs (68 , 125 , 1538) Cs (149 , 84 , 1604)
	202	255	T8 (2 , 77 , 1465) Cs (66 , 112 , 1585) Cs (170 , 75 , 1614)
	202	232	T5 (0 , 81 , 1542) Cs (67 , 103 , 1610) Cs (192 , 53 , 1614)
	202	196	Rb3 (2 , 64 , 1607) Cs (71 , 88 , 1634) Cs (183 , 52 , 1627)
	202	145	C6 (3 , 44 , 1675) Cs (143 , 61 , 1640)
	202	212	Cs (175 , 22 , 1628) Cs (77 , 63 , 1667) C3 (7 , 42 , 1733)
Serratus posterior inferior	40	97	T12 (82 , 27 , 1284) L2 (2 , 35 , 1229)
	48	115	Rb10 (105 , 52 , 1348) T12 (2 , 39 , 1299)
	53	104	T11 (95 , 38 , 1308) L1 (2 , 35 , 1262)
	67	117	Rb9 (107 , 64 , 1383) T11 (2 , 48 , 1335)
	3	91	T8 (2 , 64 , 1420) T10 (6 , 53 , 1357) T11 (2 , 40 , 1333)
Longissimus thoracis	10	403	Rb7(66, 71, 1469) Rb9(55, 73, 1410) T11(49, 63, 1346) L1(42, 50, 1263)
			L3(37, 50, 1214) L5(31, 63, 1161) Pel(15, 87, 1077)
	8	492	Rb4(55, 50, 1570) Rb6(46, 68, 1516) Rb8(36, 75, 1447) Rb10(31, 65, 1386)

Continued on next page

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
			T12(28, 52, 1311) L2(23, 44, 1236) L4(18, 49, 1188) Pel(4, 83, 1093)
	11	428	Rb6(62, 70, 1494) Rb8(53, 73, 1448) Rb10(44, 68, 1384) T12(39, 55, 1311)
			L2(31, 48, 1230) L4(27, 53, 1187) Pel(9, 86, 1077)
	10	476	Rb5(62, 59, 1541) Rb7(50, 71, 1489) Rb9(38, 71, 1409) T11(34, 62, 1350)
			L1(29, 47, 1261) L3(25, 46, 1210) L5(19, 59, 1167) Pel(5, 85, 1078)
	6	272	T11(24, 34, 1353) L1(28, 32, 1293) L3(21, 31, 1216) Pel(4, 81, 1092)
	47	33	L5(46, -3, 1169) Pel(59, 27, 1166)
	27	55	L4(42, -7, 1198) Pel(60, 33, 1166)
	9	300	T10(31, 44, 1384) T12(32, 39, 1321) L2(22, 31, 1233) L4(18, 42, 1183)
			Pel(3, 80, 1096)
	11	302	T9(30, 50, 1408) T11(33, 44, 1344) L1(22, 31, 1256) L3(18, 34, 1215)
			L5(13, 51, 1161) Pel(5, 72, 1117)
	37	98	L3(36, -8, 1226) Pel(46, 47, 1145)
	14	342	T8(30, 54, 1445) T10(36, 51, 1382) T12(30, 41, 1318) L2(19, 34, 1229)
			L4(14, 43, 1182) Pel(3, 72, 1113)
	36	125	Pel(43, 54, 1142) L4(35, 19, 1218) L2(35, 0, 1255)
	14	381	T7(33, 59, 1485) T9(38, 60, 1408) T11(31, 49, 1350) L1(21, 35, 1262)
			L3(15, 37, 1213) L5(7, 51, 1160) Pel(2, 70, 1114)
	27	157	Pel(40, 58, 1139) L5(40, 54, 1152) L3(36, 24, 1245) L1(36, 10, 1288)
	14	380	T6 (32, 50, 1512) T8(38, 68, 1453) T10(32, 55, 1382) T12(27, 43, 1318)
			L2(15, 36, 1232) L4(8, 41, 1183) Pel(2, 56, 1142)
	11	184	T12(49, 34, 1295) L3(43, 38, 1238) L5(47, 59, 1166) Pel(44, 74, 1117)
	13	380	T5 (37, 48, 1539) T7(39, 66, 1491) T9(32, 62, 1411) T11(26, 49, 1346)
			L1(17, 35, 1261) L3(9, 38, 1206) Pel(3, 42, 1167)
	7	395	Rb2(42, 32, 1627) Rb4 (41, 53, 1573) Rb6 (34, 65, 1519) T8(26, 72, 1450)
			T10(20, 52, 1358) T12(13, 37, 1287) L2(3, 34, 1244)
	11	234	T11(57, 46, 1344) L1(47, 42, 1287) L3(46, 41, 1241) L5(43, 60, 1168)
			Pel(39, 76, 1116)
	8	262	Rb10(60, 54, 1374) T12(54, 48, 1301) L2(51, 45, 1254) L5(43, 66, 1154)
			Pel(37, 77, 1117)
	8	332	Pel(29, 79, 1117) L4(40, 60, 1185) L2(47, 52, 1255) T11(56, 60, 1335)
			Rb10(61, 68, 1386) Rb8(67, 71, 1442)
	4	238	T12(27, 27, 1321) L2(23, 26, 1259) L4(21, 37, 1187) Pel(6, 83, 1095)
	12	382	T4 (37, 45, 1565) T6 (36, 56, 1525) T8(33, 72, 1451) T10(27, 57, 1382)
			L1(15, 37, 1261) L3(6, 36, 1208) L4(3, 33, 1191)
	12	385	T3 (35, 44, 1596) T5 (39, 54, 1544) T7(33, 71, 1488) T9(27, 65, 1409)
			T11(19, 43, 1320)L1(11, 35, 1260) L3(3, 31, 1220)

Continued on next page

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
Serratus anterior	178	147	Rb1 (88 , -52 , 1586) Rb2 (121 , -19 , 1578) Cs (93 , 69 , 1614)
	119	184	Rb3 (127 , -61 , 1533) Rb3 (141 , 1 , 1543) Cs (73 , 90 , 1585)
	149	214	Cs (69 , 103 , 1545) Rb4 (155 , -21 , 1504) Rb4 (137 , -75 , 1494)
	148	242	Cs(69, 109, 1503) Rb5(159, -3, 1475) Rb5(159, -61, 1463) Rb5(142, -92, 1455)
	112	254	Rb6(146, -98, 1412) Rb6(163, -58, 1429) Rb6(153, 27, 1456) Cs(69, 108, 1471)
	104	241	Rb7(150, -82, 1379) Rb7(161, -37, 1404) Rb7(146, 39, 1441) Cs(72, 106, 1462)
	97	212	Cs(74, 106, 1457) Rb8(155, -12, 1383) Rb8(151, -55, 1356)
	83	194	Cs(73, 107, 1456) Rb9(147, -30, 1340)
Pectoralis minor	77	164	Cs (171 , -2 , 1593) Rb3 (77 , -114 , 1519)
	145	188	Cs (175 , -4 , 1592) Rb4 (90 , -130 , 1481)
	84	209	Cs (178 , -7 , 1590) Rb5 (101 , -135 , 1443)
Quadratus lumborum	50	141	L5 (93 , 5 , 1180) T12 (42 , 25 , 1310)
	21	119	Pel (82 , 14 , 1178) L1 (41 , 9 , 1289)
	23	47	L4 (46 , -8 , 1200) Pel (69 , 24 , 1173)
	26	87	L2 (42 , 5 , 1255) Pel (78 , 17 , 1177)
	35	66	L3 (43 , -6 , 1227) Pel (73 , 21 , 1175)
Rhomboid maj.	635	91	T3 (3 , 68 , 1556) Cs (71 , 110 , 1512)
Subclavius	29	97	Cs (118 , -15 , 1607) Rb1 (46 , -77 , 1583)
Internal intercostal	254	43	Rb1 (83 , -54 , 1584) Rb2 (114 , -25 , 1577)
	383	39	Rb2 (103 , -65 , 1557) Rb3 (129 , -43 , 1538)
	388	41	Rb3 (125 , -75 , 1521) Rb4 (144 , -42 , 1504)
	353	41	Rb4 (135 , -83 , 1485) Rb5 (151 , -48 , 1470)
	375	45	Rb5 (147 , -74 , 1453) Rb6 (158 , -32 , 1440)
	439	34	Rb6 (157 , -52 , 1423) Rb7 (156 , -20 , 1408)
	397	29	Rb7 (152 , -50 , 1382) Rb8 (151 , -22 , 1376)
	320	23	Rb8 (147 , -38 , 1354) Rb9 (145 , -15 , 1354)
	357	22	Rb9 (145 , -37 , 1327) Rb10 (137 , -17 , 1320)
	113	48	T11 (119 , 0 , 1280) Rb10 (138 , -42 , 1294)
External/ internal obliques	$*$[1]	129	Pel (110 , -2 , 1177) Rb10 (133 , -79 , 1278)
		116	Pel (70 , 33 , 1170) T11 (114 , 1 , 1273)
		142	Pel (142 , -44 , 1171) Rb9 (118 , -116 , 1292)
		135	Pel (-114 , 3 , 1179) T12 (-59 , 33 , 1299)
		155	Pel (-152 , -61 , 1158) T11 (-114 , 17 , 1286)
		201	Pel (-135 , -95 , 1136) Rb10 (-136 , -6 , 1316)
		203	Pel (-99 , -123 , 1109) Rb10 (-146 , -53 , 1294)

Continued on next page

[1] $*$: PCSAs of internal/external obliques are not available because their geometries are defined as surfaces in the source data. We used 500 for these muscles.

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
		299	Pel (68 , -110 , 1058) Rb10 (-139 , -98 , 1274)
		304	Pel (113 , -106 , 1088) Rb9 (-115 , -116 , 1290)
		334	Pel (132 , -90 , 1124) Rb7 (-104 , -137 , 1356)
		353	Pel (145 , -66 , 1153) Rb6 (-97 , -142 , 1398)
External intercostal	151	36	Rb2 (113 , -36 , 1575) Rb1 (96 , -19 , 1602)
	268	49	Rb2 (123 , -16 , 1576) Rb3 (131 , -46 , 1539)
	179	36	Rb4 (145 , -51 , 1503) Rb3 (139 , -32 , 1532)
	285	31	Rb5 (154 , -39 , 1475) Rb4 (150 , -24 , 1501)
	303	35	Rb6 (160 , -42 , 1438) Rb5 (155 , -25 , 1468)
	300	43	Rb7 (156 , -52 , 1393) Rb6 (161 , -32 , 1431)
	309	30	Rb8 (153 , -52 , 1362) Rb7 (155 , -37 , 1388)
	240	29	Rb9 (148 , -59 , 1327) Rb8 (151 , -46 , 1353)
	202	20	Rb10 (140 , -44 , 1307) Rb9 (145 , -39 , 1326)
	100	44	Rb10 (127 , 15 , 1324) T11 (120 , 1 , 1283)
Multifidus	14	105	T7 (2 , 64 , 1451) T9 (14 , 57 , 1397) T11 (22 , 35 , 1354)
	30	70	C5 (5 , 29 , 1688) C7 (19 , 30 , 1660) T2 (35 , 32 , 1625)
	15	72	C2 (6 , 28 , 1738) C4 (13 , 22 , 1714) C6 (28 , 8 , 1673)
	9	64	C3 (5 , 25 , 1720) C5 (27 , 17 , 1682) C7 (35 , 8 , 1667)
	22	68	C4 (5 , 26 , 1706) C6 (30 , 21 , 1665) T1 (39 , 15 , 1649)
	10	109	T11 (10 , 35 , 1323) L1 (10 , 25 , 1292) L3 (37 , -8 , 1228)
	12	118	T11 (8 , 37 , 1322) L2 (18 , 21 , 1256) L4 (31 , 2 , 1211)
	13	83	T12 (3 , 26 , 1288) L2 (14 , 18 , 1253) L4 (30 , 1 , 1214)
	11	90	L1 (4 , 23 , 1262) L3 (14 , 11 , 1214) L5 (35 , -1 , 1182)
	14	101	L2 (1 , 22 , 1235) L4 (10 , 24 , 1194) Pel (39 , 20 , 1143)
	19	125	L2 (1 , 23 , 1234) L4 (12 , 30 , 1191) Pel (39 , 59 , 1121)
	25	132	L2 (2 , 27 , 1230) L4 (9 , 35 , 1196) Pel (37 , 71 , 1112)
	8	105	T8 (2 , 61 , 1421) T10 (13 , 47 , 1367) T12 (25 , 29 , 1324)
	10	111	T9 (2 , 56 , 1385) T11 (24 , 33 , 1322) L1 (33 , 10 , 1290)
	17	112	T10 (6 , 43 , 1357) T12 (19 , 34 , 1318) L2 (39 , 6 , 1257)
	14	143	T10 (4 , 45 , 1356) T12 (20 , 28 , 1292) L3 (40 , -6 , 1228)
	14	82	C7 (5 , 45 , 1646) T2 (17 , 50 , 1606) T4 (32 , 44 , 1570)
	16	89	T1 (5 , 48 , 1624) T3 (17 , 51 , 1579) T5 (32 , 46 , 1540)
	21	87	T2 (4 , 53 , 1595) T4 (17 , 57 , 1555) T6 (26 , 47 , 1512)
	15	83	T3 (2 , 62 , 1563) T5 (14 , 61 , 1525) T7 (25 , 55 , 1484)
	12	95	T4 (2 , 59 , 1540) T6 (13 , 63 , 1490) T8 (29 , 56 , 1450)
	14	100	T5 (3 , 61 , 1508) T7 (13 , 65 , 1465) T9 (29 , 52 , 1413)
	13	104	T6 (3 , 68 , 1484) T8 (13 , 60 , 1435) T10 (30 , 43 , 1386)

Continued on next page

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
	22	135	L1 (3, 27, 1260) L2 (8, 28, 1246) L4 (16, 31, 1196) Pel (41, 49, 1134)
	15	151	T12 (2, 30, 1286) L2 (12, 26, 1245) L4 (24, 24, 1186) Pel (42, 22, 1142)
	15	129	L1 (3, 25, 1261) L3 (12, 22, 1213) L5 (34, 25, 1150) Pel (40, 23, 1139)
	13	44	C7 (34, 6, 1661) C5 (5, 24, 1689)
	20	56	T1 (41, 17, 1646) C5 (5, 26, 1689)
	11	45	C4 (31, 5, 1709) C2 (9, 23, 1744)
	17	58	C5 (28, 10, 1692) C2 (7, 25, 1744)
	5	40	C5 (24, 10, 1693) C3 (5, 20, 1727)
	11	60	C6 (25, 8, 1672) C3 (4, 22, 1726)
	7	39	C6 (25, 8, 1674) C4 (5, 21, 1706)
	11	55	C7 (36, 6, 1663) C4 (5, 23, 1705)
	14	43	T1 (38, 16, 1645) C6 (3, 28, 1666)
	25	51	T2 (31, 30, 1623) C6 (2, 31, 1666)
	13	70	T3 (29, 43, 1601) C6 (2, 35, 1665)
	18	38	T2 (28, 29, 1620) C7 (5, 41, 1648)
	13	52	T3 (27, 42, 1600) C7 (5, 43, 1647)
	9	79	T10 (7, 42, 1357) L1 (31, 10, 1289)
	11	78	T11 (12, 33, 1324) L2 (38, 4, 1257)
	10	76	T12 (3, 24, 1289) L3 (34, -9, 1228)
	13	58	L1 (4, 21, 1262) L4 (29, 3, 1213)
	13	63	L2 (4, 13, 1235) L5 (36, 1, 1181)
	12	116	T12 (2, 28, 1288) L5 (36, 0, 1180)
	23	164	L3 (3, 21, 1206) Pel (38, 75, 1055)
	15	176	L3 (3, 23, 1205) Pel (32, 76, 1040)
	19	183	L3 (3, 25, 1205) Pel (28, 75, 1030)
	11	77	L3 (4, 14, 1209) Pel (37, 22, 1140)
	23	97	L4 (4, 25, 1178) Pel (29, 73, 1097)
	22	131	L4 (4, 27, 1178) Pel (31, 75, 1059)
	24	157	L4 (4, 28, 1178) Pel (22, 76, 1029)
	24	157	L4 (4, 28, 1178) Pel (22, 76, 1029)
	17	68	L4 (5, 19, 1178) Pel (26, 49, 1121)
	18	24	L5 (8, 14, 1161) Pel (10, 24, 1139)
	12	34	T3 (25, 41, 1598) T1 (6, 44, 1626)
	15	63	T4 (30, 43, 1567) T1 (6, 46, 1625)
	8	43	T12 (24, 27, 1319) T10 (9, 39, 1358)
	6	44	L1 (30, 10, 1288) T11 (14, 30, 1324)
	10	51	L2 (37, 2, 1257) T12 (4, 24, 1289)

Continued on next page

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
	11	38	T4 (26, 41, 1565) T2 (5, 48, 1596)
	17	62	T5 (29, 45, 1539) T2 (4, 50, 1596)
	12	39	T5 (25, 43, 1537) T3 (3, 58, 1565)
	12	58	T6 (23, 46, 1512) T3 (3, 60, 1564)
	9	38	T6 (21, 45, 1511) T4 (4, 56, 1543)
	12	62	T7 (22, 54, 1482) T4 (3, 57, 1542)
	13	67	T8 (26, 54, 1449) T5 (3, 59, 1511)
	9	36	T7 (20, 52, 1481) T6 (4, 57, 1513)
	9	45	T8 (23, 52, 1448) T6 (3, 64, 1486)
	13	77	T9 (26, 50, 1413) T6 (3, 66, 1485)
	16	76	T10 (27, 43, 1385) T7 (3, 62, 1455)
	7	49	T9 (23, 48, 1413) T7 (4, 60, 1456)
	13	46	T10 (25, 42, 1385) T8 (4, 57, 1423)
	7	74	T11 (19, 35, 1353) T8 (3, 58, 1421)
	7	42	T11 (17, 35, 1352) T9 (2, 52, 1388)
	12	73	T12 (24, 29, 1322) T9 (2, 54, 1387)
Rotatores	23	20	T4 (31, 43, 1572) T3 (11, 40, 1577)
	20	23	T5 (29, 45, 1543) T4 (10, 48, 1554)
	4	33	C4 (32, 3, 1711) C3 (9, 14, 1731)
	13	22	T6 (28, 44, 1518) T5 (7, 49, 1524)
	4	23	C5 (27, 10, 1697) C4 (9, 16, 1710)
	19	17	T7 (23, 54, 1486) T6 (9, 51, 1496)
	4	30	L1 (9, 12, 1266) L2 (35, 0, 1258)
	20	20	T8 (25, 52, 1451) T7 (9, 55, 1462)
	4	29	C7 (29, 15, 1653) C6 (10, 12, 1674)
	20	17	T9 (23, 45, 1417) T8 (11, 50, 1429)
	8	30	T1 (35, 16, 1643) C7 (10, 27, 1654)
	5	21	L1 (29, 12, 1290) T12 (10, 20, 1295)
	5	31	C6 (31, 8, 1674) C5 (8, 18, 1692)
	6	16	T12 (20, 21, 1321) T11 (12, 32, 1328)
	9	13	T11 (17, 30, 1354) T10 (11, 39, 1363)
	14	24	T10 (27, 40, 1387) T9 (10, 49, 1401)
	23	16	T3 (28, 40, 1604) T2 (12, 38, 1605)
	3	32	L3 (15, 5, 1212) L4 (41, -9, 1202)
	12	24	T2 (35, 30, 1627) T1 (13, 29, 1635)
	4	30	L2 (9, 7, 1234) L3 (35, -9, 1230)
Inter-	76	16	T2 (0, 46, 1607) T1 (0, 44, 1623)

Continued on next page

Table C.4 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
spinalis	54	13	T1 (0, 43, 1635) C7 (0, 38, 1647)
	24	23	T12 (0, 28, 1303) T11 (0, 34, 1326)
	53	10	C7 (0, 33, 1657) C6 (0, 28, 1665)
	33	13	C4 (0, 24, 1711) C3 (0, 22, 1724)
	52	12	L2 (0, 16, 1250) L1 (0, 15, 1262)
	52	12	L1 (0, 23, 1277) T12 (0, 22, 1288)
	53	10	L3 (0, 14, 1207) L4 (0, 16, 1197)
	32	9	L3 (0, 15, 1224) L2 (0, 13, 1233)
	23	12	L5 (0, 29, 1169) L4 (0, 20, 1176)
	32	10	C3 (0, 24, 1733) C2 (0, 24, 1743)
	58	15	C6 (0, 27, 1674) C5 (0, 24, 1689)
	37	10	C5 (0, 24, 1695) C4 (0, 22, 1705)
Inter-transversi	4	29	T12 (28, 19, 1320) T11 (25, 27, 1348)
	14	25	L3 (36, -11, 1229) L2 (37, -2, 1253)
	3	28	L1 (36, 7, 1291) T12 (29, 18, 1316)
	4	24	L3 (36, -9, 1227) L4 (40, -9, 1203)
	24	15	T1 (42, 12, 1648) C7 (40, 5, 1661)
	18	17	C7 (37, 1, 1661) C6 (28, -5, 1673)
	3	30	T10 (32, 42, 1383) T11 (26, 33, 1354)
	23	18	C6 (27, -6, 1674) C5 (29, -9, 1693)
	18	17	C5 (31, -10, 1695) C4 (31, -11, 1712)
	6	25	L5 (51, -8, 1177) L4 (38, -8, 1197)
	17	18	C4 (32, -11, 1716) C3 (32, -10, 1734)
	11	30	L2 (37, -1, 1257) L1 (37, 7, 1286)
	14	15	C3 (33, -10, 1736) C2 (32, -7, 1751)
	11	33	C2 (34, -6, 1753) C1 (55, 0, 1777)

Table C.5: Parameters of the arm muscles.

Name	PCSA	ℓ_0	Attachment (and via) points
Latissimus dorsi	160	329	Hu(291, 41, 1564) Hu(265, 41, 1546) Rb6(111, 117, 1496) T6(1, 75, 1477)
	160	339	T8(0, 70, 1420) Rb8(113, 114, 1467) Hu(239, 54, 1537) Hu(281, 37, 1564)
	160	369	T10(1, 52, 1353) Rb8(116, 107, 1444) Hu(255, 31, 1547) Hu(269, 33, 1565)
	160	403	T12(1, 33, 1289) Rb9(79, 90, 1394) Hu(250, 29, 1547) Hu(263, 30, 1566)
	160	446	L2(0, 35, 1229) T11(60, 82, 1346) Rb9(137, 91, 1414)

Continued on next page

Table C.5 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
			Hu(244, 25, 1548) Hu(251, 24, 1567)
	160	478	L4(4, 48, 1181) T11(80, 82, 1338) Rb10(138, 84, 1392)
			Hu(241, 24, 1548) Hu(245, 22, 1571)
	160	517	Pel(3, 75, 1126) T11(104, 75, 1332) Rb9(150, 79, 1387)
			Hu(234, 20, 1549) Hu(239, 19, 1572)
Pectoralis	165	248	Cs(67, -63, 1599) Cs(182, -26, 1597) Hu(289, 39, 1570)
major	468	271	Ster(2, -100, 1564) Ster(21, -118, 1570) Hu(227, 13, 1584)
	468	296	Ster(2, -131, 1508) Ster(30, -144, 1520) Hu(230, 15, 1581)
	468	323	Ster(0, -146, 1463) Ster(36, -155, 1475) Hu(237, 17, 1581)
	468	308	Ster(41, -156, 1428) Hu(244, 23, 1575)
	468	283	Rb6(101, -145, 1407) Hu(251, 24, 1576)
Extensor	100	334	Hu (469, 114, 1589) Rad (755, 130, 1591) Ha (801, 126, 1581)
carpi	99	366	Hu(440, 115, 1581) Hu(465, 113, 1598) Ha(761, 122, 1597)
			Ha(800, 120, 1596)
	99	336	Hu(465, 122, 1584) Ul(508, 132, 1585) Ul(739, 133, 1554)
			Ha(795, 121, 1542)
Flexor	130	362	Hu(461, 120, 1518) Ul(528, 87, 1541) Rad(754, 97, 1585) Ha(804, 111, 1578)
carpi	261	312	Ha (773, 108, 1543) Hu (463, 126, 1516)
Flexor	59	228	Rad (561, 107, 1578) Rad (741, 97, 1572) Ha (788, 99, 1570)
digitorum	105	300	Ul (492, 122, 1545) Ul (741, 99, 1565) Ha (790, 101, 1565)
superf.	38	182	Rad (624, 109, 1589) Rad (743, 97, 1577) Ha (805, 103, 1577)
	119	333	Hu (464, 123, 1517) Rad (743, 100, 1558) Ha (793, 102, 1561)
Palmaris longus	72	320	Hu (463, 121, 1517) Ul (742, 93, 1569) Ha (777, 92, 1568)
Pronator	182	34	Ul (716, 120, 1551) Ul (720, 117, 1557) Rad (729, 112, 1581)
	226	180	Hu (474, 106, 1527) Rad (583, 102, 1576) Rad (640, 111, 1595)
	226	58	Ul (500, 128, 1551) Rad (530, 102, 1565) Rad (539, 102, 1579)
Supinator	194	29	Ul (516, 124, 1571) Rad (533, 116, 1582) Rad (539, 113, 1583)
Supraspinatus	682	96	Cs (123, 74, 1601) Cs (168, 40, 1610) Hu (192, 13, 1594)
Teres maj.	385	197	Cs (97, 100, 1480) Hu (247, 24, 1554) Hu (259, 26, 1560)
Deltoid	987	229	Cs (137, -4, 1619) Cs (183, -31, 1610) Hu(301, 23, 1587) Hu(322, 54, 1564)
	987	160	Cs (215, 19, 1619) Hu (341, 49, 1582) Hu (348, 64, 1563)
	987	212	Cs(134, 90, 1591) Cs(178, 106, 1575) Hu(306, 74, 1591) Hu(324, 53, 1579)
Infraspinatus	1025	134	Cs (102, 90, 1533) Hu (197, 49, 1585) Hu (201, 33, 1593)
Biceps	634	362	Hu (177, 20, 1579) Hu (214, 4, 1580) Rad (520, 104, 1570)
brachii	850	355	Cs (185, -6, 1585) Hu (321, 29, 1566) Rad (519, 108, 1568)
Brachioradialis	218	348	Hu (408, 102, 1576) Hu (463, 96, 1592) Rad (753, 112, 1597)

Continued on next page

Table C.5 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
Brachialis	1118	133	Hu (378, 83, 1565) Hu (445, 87, 1547) Ul (498, 121, 1554)
Coracobrachialis	70	197	Cs (183, -8, 1586) Cs (208, -7, 1565) Hu (349, 76, 1554)
Triceps	775	183	Hu(303, 67, 1571) Hu(353, 101, 1589) Ul(447, 133, 1557) Ul(462, 131, 1556)
brachii	2526	111	Hu (366, 91, 1560) Hu (411, 130, 1556) Ul (462, 131, 1556)
	912	307	Cs (172, 45, 1552) Hu(391, 130, 1548) Ul(439, 134, 1557) Ul(462, 131, 1556)

Table C.6: Parameters of the leg muscles.

Name	PCSA	ℓ_0	Attachment (and via) points
Gluteus maximus	6478	672	Pel (39 73 1073) Fe (186 -33 942) Tfp (170 , -67 , 495)
	2825	145	Pel (102 1 1126) Fe (157 -24 994)
	698	109	Pel (113 , -25 , 1089) Fe (160 , -43 , 993)
Gemellus	28	79	Pel (79 , 25 , 967) Fe (143 , -20 , 984)
	43	105	Pel (56 , 33 , 1013) Fe (141 , -23 , 990)
Iliopsoas	23	293	L4 (30 , -31 , 1172) Pel (66 , -80 , 1019) Fe (126 , -7 , 932)
	23	266	L5 (30 , -31 , 1142) Pel (66 , -80 , 1019) Fe (126 , -7 , 932)
	117	327	L3 (27 , -34 , 1209) Pel (70 , -82 , 1015) Fe (123 , -8 , 932)
	248	359	L2 (25 , -33 , 1244) Pel (71 , -83 , 1024) Fe (122 , -10 , 931)
	203	392	L1 (22 , -26 , 1276) Pel (75 , -86 , 1023) Fe (121 , -13 , 930)
	101	412	T12 (23 , -18 , 1309) Pel (76 , -80 , 1024) Fe (117 , -19 , 935)
Obturator	518	101	Pel (43 , -41 , 971) Fe (140 , -15 , 973)
	501	150	Pel (54 , -24 , 1009) Pel (67 , 30 , 989) Fe (141 , -22 , 987)
Pectineus	507	172	Pel (63 , -74 , 1017) Fe (118 , -33 , 893) Fe (141 , -27 , 875)
Piriformis	357	155	Pel (39 , 30 , 1076) Pel (85 , 23 , 1037) Fe (155 , -33 , 1006)
Quadratus femoris	190	86	Pel (70 , -2 , 962) Fe (154 , -12 , 967)
Extensor digitorum	440	297	Tfp (170 , -39 , 358) Fo (172 , -77 , 63)
Extensor hallucis	284	232	Tfp (167 , -37 , 297) Fo (173 , -76 , 68)
Flexor digitorum	588	265	Tfp (145 , -40 , 316) Fo (136 , -10 , 53)
Flexor hallucis	1135	178	Tfp (171 , -22 , 223) Tfp (154 , -2 , 153) Fo (154 , -2 , 50)
Gastrocnemius	781	516	Fe (170 , -36 , 549) Tfp (170 , 18 , 302) Fo (170 , 28 , 40)
	1099	508	Fe (115 , -36 , 550) Tfp (140 , 18 , 292) Fo (157 , 18 , 49)
Peroneus	106	149	Tfp (175 , -27 , 213) Fo (186 , -8 , 65)
	513	341	Tfp (173 , -38 , 398) Fo (184 , -1 , 59)
	50	126	Tfp (175 , -29 , 186) Fo (178 , -68 , 66)

Continued on next page

Table C.6 – continued from previous page

Name	PCSA	ℓ_0	Attachment (and via) points
Plantaris	15	519	Fe (167 , -25 , 544) Tfp (121 , 1 , 357) Fo (145 , 35 , 36)
Popliteus	154	76	Fe (173 , -43 , 523) Tfp (125 , -41 , 464)
Soleus	2415	390	Tfp (168 , -35 , 438) Tfp (156 , 11 , 201) Fo (156 , 11 , 53)
Tibialis anterior	553	322	Tfp (156 , -56 , 352) Tfp (162 , -60 , 127) Fo (135 , -91 , 40)
Adductor	674	187	Pel (17 , -76 , 968) Fe (142 , -31 , 836)
	555	251	Pel (21 , -96 , 982) Fe (132 , -43 , 763)
	598	371	Pel (27 , -49 , 951) Fe (107 , -43 , 589)
	2193	162	Pel (49 , -11 , 939) Fe (142 , -34 , 808)
Biceps	730	255	Fe (144 , -43 , 739) Fe (177 , -39 , 564) Tfp (179 , -51 , 488)
femoris	726	489	Pel (77 , 28 , 965) Tfp (179 , -41 , 492)
Gracilis	248	534	Pel (15 , -68 , 957) Fe (96 , -12 , 533) Tfp (124 , -63 , 453)
Rectus femoris	1216	511	Pel (109 , -79 , 1062) Tfp (128 , -99 , 552)
Sartorius	360	652	Pel (116 , -96 , 1086) Fe (87 , -24 , 539) Tfp (126 , -66 , 458)
Semimembranosus	1021	483	Pel (83 , 11 , 968) Fe (115 , -19 , 596) Tfp (105 , -34 , 489)
Semitendinosus	495	545	Pel (75 , 30 , 961) Tfp (102 , -36 , 465) Tfp (124 , -59 , 435)
Tensor fascia latae	298	625	Pel (128 , -91 , 1111) Fe (190 , -64 , 863) Tfp (170 , -67 , 495)
Vastus	3787	285	Fe (130 , -86 , 846) Tfp (129 , -92 , 561)
	3658	162	Fe (143 , -47 , 694) Fe (182 , -75 , 626) Tfp (141 , -97 , 562)
	4584	245	Fe (127 , -39 , 776) Fe (70 , -53 , 668) Tfp (117 , -94 , 563)

REFERENCES

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

ALEXA, M. 2002. Linear combination of transformations. *ACM Transactions on Graphics 21*, 3 (July), 380–387.

ALLEN, B., CHU, D., SHAPIRO, A., AND FALOUTSOS, P. 2007. On the beat! Timing and tension for dynamic characters. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, ACM, New York, NY, USA, 239–247.

ANDERSON, F., AND PANDY, M. 2001. Static and dynamic optimization solutions for gait are practically equivalent. *Journal of Biomechanics 34*, 153–161.

ARMSTRONG, W. W., AND GREEN, M. W. 1985. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer 1*, 4 (Dec.), 231–240.

BARAFF, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH Comput. Graph. 23*, 3, 223–232.

BARR, A. H., CURRIN, B., GABRIEL, S., AND HUGHES, J. F. 1992. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 313–320.

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

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

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 97–104.

CARPENTER, R. 1988. *Movements of the Eyes*, 2nd ed. Pion, London.

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

CROWNINSHIELD, R. 1978. Use of optimization techniques to predict muscle forces. In *Journal of Biomech. Eng.*, 88–92.

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

DELP, S. L., LOAN, J. P., HOY, M. G., ZAJAC, F. E., TOPP, E. L., AND ROSEN, J. M. 1990. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *IEEE Transactions on Biomedical Engineering 37*, 8 (Aug.), 757–767.

DELP, S. L., ANDERSON, F. C., ARNOLD, A. S., LOAN, P., HABIB, A., JOHN, C. T., GUENDELMAN, E., AND THELEN, D. G. 2007. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering 54*, 1940–1950.

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

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

FEATHERSTONE, R. 1987. *Robot Dynamics Algorithms*. Kluwer Adademic Publishers, Boston.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 181–188.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 29–38.

GABRIEL, S., AND KAJIYA, J. 1985. Spline interpolation in curved space. *In SIGGRAPH 85 Course Notes for "State of the Art in Image Synthesis"*.

GRZESZCZUK, R., AND TERZOPOULOS, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of ACM SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 63–70.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proc. of ACM SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 9–20.

HAY, J., AND REID, J. 1988. *Anatomy, Mechanics, and Human Motion*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ.

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

HOGAN, N. 1984. Adaptive control of mechanical impedance by coactivation of antagonist muscles. *IEEE Transactions on Automatic Control AC-29* (Aug.), 681–690.

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

KANDEL, E., SCHWARTZ, J., AND JESSELL, T. 2000. *Principles of Neural Science*, 4th ed. McGraw Hill, New York.

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

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. *ACM Transactions on Graphics 24*, 3 (Aug.), 946–956.

KAWATO, M., FURUKAWA, K., AND SUZUKI, R. 1987. A hierarchical neural network model for control and learning of voluntary movement. *Biological Cybernetics 57*, 169–185.

KIM, J., AND HEMAMI, H. 1998. Coordinated three-dimensional motion of the head and torso by dynamic neural networks. *IEEE Trans. on Systems, Man and Cybernetics. B 5*, 653–666.

KIM, M.-J., SHIN, S. Y., AND KIM, M.-S. 1995. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of*

SIGGRAPH 95, Computer Graphics Proceedings, Annual Conference Series, 369–376.

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

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

KRY, P. G., AND PAI, D. K. 2003. Continuous contact simulation for smooth surfaces. In *ACM Transactions on Graphics*, vol. 22, 106–129.

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

LEE, S.-H., AND TERZOPOULOS, D. 2008a. Learning neuromuscular control for the biomechanical simulation of the neck-head-face complex. In *Proc. of the Learning Workshop 2008*, 157–159.

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

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics*. In press.

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

MACIEL, A., NEDEL, L. P., AND FREITAS, C. M. D. S. 2002. Anatomy-based joint models for virtual human skeletons. *Proceedings of the Computer Animation 2002 Conference*, 220–224.

MCGUIGAN, M., 2006. Graphics turing test. arXiv:cs/0603132v1 [cs.GR].

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

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

MOON, J. T., WALTER, B., AND MARSCHNER, S. 2008. Efficient multiple scattering in hair using spherical harmonics. *ACM Transactions on Graphics 27*, 3 (Aug.), 31:1–31:7.

MORI, M. 1970. The uncanny valley. *Energy 7*, 4, 33–35.

MURRAY, R., LI, Z., AND SASTRY, S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

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

NEFF, M., AND FIUME, E. 2002. Modeling tension and relaxation for computer animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA'02)*, 81–88.

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

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

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

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

PARK, F. C., AND RAVANI, B. 1997. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics 16*, 3 (July), 277–295.

PARK, F. C., BOBROW, J. E., AND PLOEN, S. R. 1995. A lie group formulation of robot dynamics. *The International Journal of Robotics Research 14*, 6, 609–618.

RAMAMOORTHI, R., AND BARR, A. H. 1997. Fast construction of accurate quaternion splines. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 287–292.

REULEAUX, F. 1876. *Kinematics of Machinery: Outlines of a Theory of Machines*. MacMillan and Co, London.

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

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

SHAO, W., AND NG-THOW-HING, V. 2003. A general joint component framework for realistic articulation in human characters. In *Proceedings of the 2003 ACM Symposium on Interactive 3D Graphics*, 11–18.

SHAPIRO, A., KALLMANN, M., AND FALOUTSOS, P. 2007. Interactive motion correction and object manipulation. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, 137–144.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, 245–254.

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

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

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

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

SPELLUCCI, P. Donlp2. www.netlib.org/ampl/solvers/donlp2/.

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

TÄNDL, M., AND KECSKEMÉTHY, A. 2007. A comparison of B-spline curves and Pythagorean hodograph curves for multibody dynamics simulation. *Proceedings of Twelfth World Congress in Mechanism and Machine Science* (June), 380–387.

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

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*.

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

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 269–278.

TERZOPOULOS, D., AND LEE, Y. 2004. Behavioral animation of faces. In *Facial Modeling and Animation*, J. Haber and D. Terzopoulos, Eds., vol. 60 of *ACM SIGGRAPH 2004 Course Notes*. ACM SIGGRAPH, Aug., 119–128.

TERZOPOULOS, D., AND QIN, H. 1994. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics 13*, 2 (Apr.), 103–136.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 205–214.

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

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

TSANG, W., SINGH, K., AND FIUME, E. 2005. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 319–328.

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of ACM SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, 43–50.

VASAVADA, A., LI, S., AND DELP, S. 1998. Influence of muscle morphometry and moment arms on the moment-generating capacity of human neck muscles. *Spine 23*, 412–422.

WARFEL, J. 1985. *The Head, Neck, and Trunk*, 5 ed. Lea & Febiger, Philadelphia.

WATERS, K. 1987. A muscle model for animating three-dimensional facial expression. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 17–24.

WEYRICH, T., MATUSIK, W., PFISTER, H., BICKEL, B., DONNER, C., TU, C., MCANDLESS, J., LEE, J., NGAN, A., JENSEN, H. W., AND GROSS, M. 2006. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics 25*, 3 (July), 1013–1024.

WILHELMS, J., AND BARSKY, B. A. 1985. Using dynamic analysis to animate articulated bodies such as humans and robots. In *Graphics Interface '85*, 97–104.

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

WINTERS, J., AND CRAGO, P., Eds. 2000. *Biomechanics and Neural Control of Posture and Movement*. Springer-Verlag, New York.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 159–168.

WITKIN, A., AND POPOVIC, Z. 1995. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 105–108.

YAMAZAKI, Y., OHKUWA, T., ITOH, H., AND SUZUKI, M. 1994. Reciprocal activation and coactivation in antagonistic muscles during rapid goal-directed movements. *Brain Research Bulletin 34*, 587–593.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: simple biped locomotion control. *ACM Trans. Graph. 26*, 3, 105.

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

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