ANATOMICALLY-BASED MODELS FOR PHYSICAL AND GEOMETRIC RECONSTRUCTION OF HUMANS AND OTHER ANIMALS

by

Victor Ng-Thow-Hing

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 2001 by Victor Ng-Thow-Hing

Abstract

Anatomically-based models for physical and geometric reconstruction of humans and other animals

Victor Ng-Thow-Hing
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2001

The relationship between the design elements of form and function is fundamental in producing new insights and understanding of the objects we encounter every day. Of particular interest is the study of human and animal anatomy, and how the elegance of shape and form we take for granted must simultaneously serve a practical role to locomote the body and perform other essential tasks for survival. We demonstrate how an integrated mathematical model, the B-spline solid, can be used to successfully capture geometric aspects of musculotendons as well as their physical characteristics. The B-spline solid is flexible enough to specify large insertion and origin attachment areas for musculotendons to the skeleton. Furthermore, it is scalable to facilitate detailed three-dimensional fibre reconstruction of internal muscle architecture. From these geometric reconstructions, we can embed different physical models to simulate phenomena such as volume preservation, active muscle contraction, and contact collisions. A software framework is developed to allow these musculotendon models to co-exist with other anatomic tissues, such as bones, ligaments, fat, and skin. The construction of these musculotendon models and the existence of a software environment for their utilization in different scenarios promises to enable detailed studies of the interdependencies in the body design of humans and other animals.

Dedication

Dedicated to Xiaohuan, Mom and Dad

Acknowledgements

The work presented in this volume would not have been possible without the occurrence of many happy and lucky accidents. When I first decided to explore anatomically-based or muscle-based animation, articulated figures were chiefly being simulated with pure torque motors at the joints. There is some question as to why one would want to model redundant and complex force actuators that are difficult to control. Well, that is the reality of how our muscles work. I owe a sincere and neverending debt of gratitude to my supervisor, Eugene Fiume, for giving me the freedom to take risks and explore my ideas, while tolerating the missteps along the way. He has instilled within me a research style that can best be explained as a pure, love of science and exploration, mixed with the pragmatism and industriousness needed to achieve research goals.

I would like to thank the other members of my thesis committee: Scott Delp, James Stewart, Michiel van de Panne, and Demetri Terzopoulos. They all asked critical questions and gave me valuable insights at different stages of my research that enabled me to strengthen the final content of my thesis. Each member had a unique perspective on issues that allowed the research to have relevance to more than one type of audience.

During the process of developing techniques described in the thesis, I spoke to many people and eagerly obtained their opinions on muscle modelling from many interesting perspectives: computer graphics (Petros Faloutsos, Rudy Ziegler, Joe Laszlo, Michiel van de Panne, Demetri Terzopoulos, Caleb Howard), sculpture (Alexander Moyle), biomechanics (Kevin Ball, Douglas Richards), and anatomy (Anne Agur, Nancy McKee, Bernie Liebcott). I would especially like to thank my closest direct collaborators, Anne Agur and Petros Faloutsos for helping to share the mental and physical labour necessary to tackle the monumental problem of muscle architectural modelling and building a software architecture capable of physical simulation of complex anatomical systems. All my collaborators have directly helped to make this thesis much stronger than what I could have done on my own. I feel truly lucky for spending my graduate school years at the Dynamic Graphics Project Laboratory, for the unique culture of individuals who were able to share with me their own research dreams as well as personal interests - thank you all.

The challenge of designing and developing software was made much easier by the incredible phenomenon of the Internet and the accompanying open source movement. Different components of this thesis utilized important software that the authors have kindly decided to make available (Tcl/Tk, RAPID, GLUT, Mesa, VCOLLIDE, Minpack, Meschach). The transferring and combining of this digital knowledge to build powerful software solutions is one of the great achievements of our field of Computer Science.

My parents, Guy and Ah-Kim, have given me the opportunity to pursue my interest in research with numerous personal sacrifices and hardship. I thank them both for letting me choose my direction in life, without judgement or conditions. They managed to give me critical experiences in my youth that played a large part in how I view life today, and the tools to cope with unforseen problems that will arise.

Finally, I would like to thank my wife, Xiaohuan, for her patience and loving support during the long years of graduate school. She gave me critical boosts of confidence when I most needed it, as well as the strong discipline I was lacking when I was in danger of veering from my path. Her invaluable lessons and advice continue to be one of the most precious gifts I have received from sharing my life with her.

It's nice to be done. See you later.

Victor Ng-Thow-Hing
March 2001

Contents

List of Figures

Chapter 1

Introduction

> What causes those depressions, corregations, indentations, bulges, pockets, hollows, dimples, creases, crannies, ridges, furrows, knobs, folds, curves, grooves, tucks, knots, clefts, pits, dents and bumps?
>
> **John Cody, M.D. in *Visualizing Muscles: A New Ecorche Approach to Surface Anatomy, 1990***

1.1 Motivation

The complex assembly of skeleton, muscle, tendon and ligament in animals is a testament to the creative forces of nature. They create a mechanism that is capable of producing a versatile array of locomotion, from flying to playing the piano. In addition to the functional role of these tissues, it is their interesting shapes and underlying change of form during motion that gives them a visual beauty. The great Renaissance artist and scientist Leonardo da Vinci studied the elegant relationship between form and function of muscle which played a critical role in the success of his sculptures[20]. The early Disney artists studied comparative anatomy to perfect their animation techniques when drawing animals such as those seen in *Bambi*[117].

This thesis presents a technique of modelling the underlying tissues that comprise the animal form using knowledge from the disciplines of computer graphics, numerical methods, biomechanics and anatomy. The result is an anatomic modelling system that can capture both the geometric and physical nature of the *musculotendon* unit - the fusion of muscle and tendon tissues which are the primary motivators of motion in animals. Several disciplines of study focus on different characteristics of muscle and tendon for their own unique purposes. An appealing challenge is the development of a model that can serve a wide array of applications, and yet be customizable for the special needs of these areas. To identify these needs, we will explore different perspectives of anatomic modelling from three application domains: anatomy, biomechanics, and computer graphics.

1.1.1 Anatomy Perspective

Anatomists are interested in exploring the relationship between form and function of tissues in the body. In addition to basic limb motion, muscles are important as an expressive method

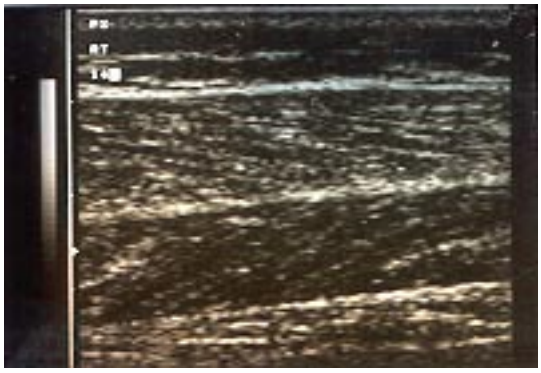


Figure 1.1: Ultrasound imagery of contracted human *soleus* muscle. Photo courtesy of Roger Leekam[77].

of communication. They actively contribute to the creation of facial expressions and body language, such as when tense muscles convey fear or stress. In the case of muscle, anatomists need to examine the arrangement of muscle fibres, tendons, and ligaments in determining their functional role in maintaining body posture or creating limb movement. This highlights the need for a musculotendon model that contains a solid component that can have volumetric representation to visualize internal fibre arrangements and their influence on the overall shape of the musculotendon.

Anatomists have studied muscle by examining cadavers and recording measurements of muscle fibre length and orientation. Because the fibre tissue is inactive, it is impossible to observe active muscle contraction. The best clinical tool available for observing *in vivo* contractions is ultrasound imagery. This technology is subject to noise and other restrictions on the view of the process from a few fixed vantage points (Figure 1.1).

The fibre structure or architecture of muscle is not a simple case of a group of muscle fibres running together in parallel from end to end. A single muscle can have several distinct regions of fibres, each with their own orientations. The overall shape of muscle can take on a diversity of forms from fusiform, triangular, parallel, and bipennate shapes (Figure 2.3)[42]. A shape primitive must be capable of representing these forms. To complicate matters, the tendon attachment region to the muscle fibres can occur in different configurations from local areas to wide regions of tissue known as *aponeurosis*).

Muscles have been reconstructed from medical images by segmenting the regions from transverse sections of specimens taken from the Visible Human dataset. A boundary surface representation of the muscle is created, but the internal fibre architecture between adjacent slices is not captured. With only superficial representations of the muscle, functional studies of contraction are impossible because we do not know the direction of fibre shortening.

A functional model of muscle and tendon allows closer study of the relationship between fibre orientation and the magnitude and direction of forces generated. This has significance in providing insights of how various muscles work within the body and the roles they provide. For example, the soleus muscle is an antigravity muscle that is important for balance and posture control, yet may not actively contribute to limb motion. Surgical procedures such as reconstructive plastic surgery can be simulated virtually. Palaeontologists can attempt to reconstruct the musculature of fossilized skeletons in order to gain insight in the appearance

of extinct animals who were never seen in the history of our species. These reconstructions can then be potentially simulated to test hypotheses of limb motion and joint articulation by creating the necessary muscle forces directly from the models to achieve these motions.

1.1.2 Biomechanics Perspective

Biomechanists are concerned with the role of muscles and tendons as force actuators that drive animal locomotion. They seek to answer questions such as how our nervous system coordinates muscles activations to produce limb motion. With this information, new training techniques can be developed to improve athletic performance. Through study, methods can be devised to prevent injury or heal damaged muscles, ligaments and tendon through rehabilitation.

Although there are biomechanists who have studied non-human locomotion, a large amount of work has focused on the particularities of the human body. A complete analysis of the human body would entail a physical system consisting of over 200 bones and 600 muscles. As the largest physical simulations of articulated figures currently contain about ninety degrees of freedom[80], it is currently not computationally feasible to create a complete simulation of the human body. Consequently, previous work involves the creation of isolated musculoskeletal models that target subsystems such as lower leg or shoulder assemblies.

To maximize the number of musculotendon units that can be simulated, muscles are often approximated as piecewise line segments whose lines of action have zero curvature. Although optimization studies have been able to produce qualitative correlations between muscle and recorded muscle activations called *electromyograms* (EMGs), there are several deficiencies.

- **Lines of action.** The lines of action of muscle must be computed properly in order to calculate the correct torques on the joints that the muscles act on. The situation is complicated by the fact that the moment arm changes depending on joint configuration and that provisions must be made to prevent interpenetration of the lines of action into the bone. Current models assume zero curvature in the line of action but observation shows there is a definite non-zero curvature in the action lines.

- **Pennation effects.** A given muscle has a distinct fibre architecture where the orientations of the fibres directly influence the direction of the generated force vector. These fibre patterns often resemble that of a feather where the muscle fibres converge to a middle axis of tendon material (Figure 2.3). For some motions, a lumped-sum parameter for the fibre orientation may be adequate, but various anti-gravity muscles that are used for postural stability and fine control may need detailed functional study that examines the interplay between distinct regions of fibre orientation. The control signals from the nervous system to muscle are sent to *motor neurons* which innervate groups of fibres. To model the phenomenon of motor recruitment of fibre, the ability to partially activate selected regions of muscle are necessary.

- **Global shape effects.** The majority of physical studies of muscle have concentrated on the resulting force at the attachment site to the skeleton. We will show that some phenomena of muscle contraction may need to consider global muscle constraints such as volume conservation. For example, we will present evidence that the observed changes of muscle pennation angles could be influenced by volume preservation forces. The pathological

condition of *compartmental syndrome* requires the study of the internal muscle pressures as they exert forces on each other and the surrounding *fascia* tissue of each muscle fibre bundle.

- **Insertion and origin sites.** The area over which the tendon attaches itself to the bones of a skeleton directly influence the magnitude and direction of musculotendon forces on the bone. These areas can span over large areas of boney surface. Models that represent muscles as line segments treat the attachment site as single points. In order to simulate large areas of attachment such as the sternocostal portion of the *pectoralis*, many line segments must be used. The single generation model we represent allows a continuous representation of these attachment areas and a unified way of enumerating points in the area.

A detailed muscle model that incorporates both the physical and geometric characteristics of muscle allow functional studies to be conducted in greater detail than has been performed before. By using a compact representation for muscle, computational overhead can be minimzed and multiple muscle simulations are possible even with the larger set of parameters for each muscle. The model can open up new directions for study such as simulation of compartmental syndrome and motor control theories that can be tested through simulation. Clinicians can visualize which muscles are being used the most as skeletons undergo various motions. Reconstructive surgery can also be planned by simulating various scenarios and observing the resultant changes in range of motion or flagging potential areas of excessive strain that could result in the modified musculoskeletal system.

1.1.3 Computer Graphics Perspective

The ability to include humans and other animals in computer animation has allowed a greater freedom of storytelling by populating digital realms with interesting characters. As in classical animation, to maintain the “illusion of life”, animators can decompose all movement in living characters to a layered collection of motions. There is the base positioning of limbs as the figure transitions from pose to pose. The projectile trajectory of the centre of mass can be seen in jumping, walking and running. There is a global change of shape in the body to account for the volume preserving effects of stretch and squash as deformable material reacts to various external forces. Finally, there is the secondary motion in the form of visible muscle and skin deformations that occur due to muscles driving the motions in the limbs.

Being able to model synthetic animals for film or interactive entertainment has many advantages. Computer generated animals can be directly controlled by a human operator without the unpredictability associated with real animals. The inherent dangers of working with wild animals and potential hardships on the animal’s health are eliminated. With knowledge of existing anatomical configurations in animals, fictional creatures with similar body structures can be designed and animated with potentially greater realism. In the area of entertainment, more emphasis is placed on visual accuracy rather than physical correctness or muscle architecture. Nevertheless, an anatomically-inspired approach has several advantages over other methods for attempting to represent the anatomical features of animals.

Previous models have focused on simulating skin deformations automatically from muscles with shape changes being based on deformation techniques that neither accurately depict the

correct anatomical configuration of the body nor adequately capture the dynamic solid tissue nature of muscle. While this approach may be satisfactory for an initial visual approximation for animation, there are several phenomena which may be excluded:

1. **Tendons and ligaments define shape** When a skeletal muscle contracts to create motion, the contractile force is transmitted through to the tendons. These tendons can become stretched taut and can visibly protrude out of the skin. This is apparent when you clench a fist or straighten out your fingers.

2. **Isometric contractions and negative work** In animal motion, muscles often actively contract while exhibiting no limb motion. Visually, muscles appear to tense and sometime bulge, conveying to the viewer the anticipation or exertion of effort during a motion. Many schemes that simulate muscle deformations correlate the deforming of muscle to joint movement. Since isometric contractions do not result in any joint motion, no noticeable deformations will be generated with these schemes. In reality, as a muscle contracts, there is compliance in the attached tendon. This compliance allows a muscle to change shape while the affected limbs are held stationary.

3. **Inertial effects** Many muscle models feature deformations that are exclusively a function of the kinematic state without consideration of time derivatives or mass properties. As a result, stress waves[23] that propagate through muscle or local inertial effects that can cause muscles to demonstrate damped oscillations will not be reproduced with pure geometric models. This phenomenon can be observed as "jiggles" that occur in the muscles after a sudden stop of a quick motion.

4. **Collision and interpenetration** Nature has designed muscles to be compactly placed on the skeleton of animals. As a result, numerous muscles often press against each other, surrounded by an envelope of fat tissue and skin. Consequently, shape changes in one muscle will result in pressure changes on surrounding muscles. Biomechanists have realized the importance of this phenomenon when they study compartmental syndrome (the injuries in muscle that are caused by the built up pressures of muscle tightly packed against each other). These geometric and physical consequences have not been modelled. Previously, muscle volumes have been allowed to intersect with each other.

To consistently apply these visual phenomena to computer-generated characters, a method based on the underlying anatomical structure can create life-like motions. With the goal of visualizing these features and simulating the mechanics of musculotendon systems, this thesis develops a methodology and useful set of physical and geometric primitives that can be applied in the fields of anatomy, biomechanics and entertainment. The key challenge will be the creation of a muscle model that can represent a large array of shapes and levels of complexity. Furthermore, anatomy and biomechanics require a higher standard of accuracy than applications for entertainment purposes.

1.2 Problem Statement

Given these considerations, our problem can be stated as:

Given a skeleton of an animal consisting of bone geometry, produce an additive process that allows the layering and attachment of muscles, tendons and ligaments to the underlying skeleton. For physical simulation, these elements can be made to exert force on the skeleton and undergo shape changes consistent with the behaviour of the material they are meant to model. This includes shape changes due to external forces or internal potential energies. In the case of muscle tissue, these forces are generated internally and we wish to be able to accurately model the muscle architecture (fibre orientation) as well as its volume preservation characteristics. The muscles produced should be controllable to the extent that a force model allows a control input variable to be specified that directly affects the active contractile force of muscle.

1.2.1 Restrictions

To model the entire anatomy of an animal would be an extremely complex task. For our purposes, we are concerned only with modelling the elements of anatomy that are the causal agents for motion and body shape definition: skeletal muscle, tendons and ligaments. Less consideration will be given to fat and skin. Smooth (stomach lining) and cardiac muscle (the heart) will not be considered. Specialized models that account for the unique fibre arrangements of these types of muscle and their interactions with fluids have been developed[97].

The stiffness of bone is measured by its Young's Modulus, which is about $10^{11} dyn/cm^2$[2]. Tendon and muscle can be up to several orders of magnitude more compliant than bone. Tendon has a strength of around $10^{10} dyn/cm^2$ and active muscle tissue in frogs experience a maximum stress of $2 \times 10^6 dyn/cm^2$[36]. Consequently, bones will be treated as rigid objects and their viscoelastic and plastic effects under high stress will be ignored.

Regarding skeletal muscle, we will not directly look at the molecular mechanisms that occur during the contraction of muscle, such as those described by the Cross-bridge theory [61]. Computer simulations of the hypothesis of cross-bridge links have been created in [127]. Muscles have also been known to have history-dependent effects and can be affected by factors such as age, exercise, pre-stretching, immobility and fatigue[131]. We will not attempt to model muscles dependent on these parameters.

1.3 Contributions and Results

This thesis presents a solution to the stated problem in the design and implementation of an anatomically-based modelling system. A fundamental concern is to develop an integrated physical and geometric model for muscle. Muscles are the primary actuators of locomotive force in animals and also define the shape of the animal's body. Previous work has only focused on one side of this dual nature of muscle. Biomechanists have constructed force models for linear muscle model without 3-D volume. In contrast, computer graphics has only considered modelling gross geometric effects of muscle deformation on skin with little concern for physical simulation. An important first step to modelling both aspects simultaneously is to choose and define equations of motion for the parameters of a geometric muscle model while keeping in mind that the muscle must be able to collide and maintain non-penetration con-

straints with other objects. By focusing on the the low-level building blocks that combine to create the form and function of humans and other animals, these elements can be re-configured to create novel forms adhering to anatomical constraints. The major contributions of this work can be listed as follows.

1.3.1 Muscle shape primitive

We chose to model muscles as solids, instead of a boundary surface representation. All points that consist of the solid can be enumerated with a parameterization of a B-spline triple tensor product formulation. This representation has allowed us to model the internal fibre arrangements of actual muscle, providing an important visualization tool and link for validation of our muscle with actual measurements from cadaveric specimens. In order to create initial shapes for these solids, we designed a procedure using several template guides to fit to the B-spline solid model.

1.3.2 Lagrangian Physics formulation

As the model's shape can be represented uniquely by a set of state variables, we developed a set of equations of motion derived from Lagrangian dynamics. This has provided a convenient way of decomposing the various forces into separate parts that can be adjusted. Volume-preserving forces can co-exist with external forces such as gravity and spring forces. Constraints can be applied to portions of the solid material to create point on plane or fixed point constraints. We have developed an assortment of physical force models that can be selectively added or removed from the muscle model depending on the targetted application.

1.3.3 Biomechanical Muscle Model

The physical parameters of our solid primitive allow us to extend the range of applicability of the model from muscle to tendon to ligaments. This model for soft tissue can be used in a dynamic anatomical system. The well-known force-length-velocity relationship of the contractile element of muscle will be integrated into our three-dimensional volume model for muscle. Similarly, we can choose to endow our tendon and ligament models with elastic effects.

1.3.4 Development of System

In order to build a framework to test our models, we have developed a software architecture that can allow the integration of all the bone, muscle and other soft tissue elements to create a simulated dynamical system. Various joint constraints can be created and the skeleton subsequently manipulated interactively. A system was designed using object-oriented techniques and dynamic loading of object code to provide a flexible environment to construct our dynamic, anatomical systems. The system was co-designed with Petros Faloutsos and is referred to as *DANCE* (Dynamic Animation and Control Environment)[86].

1.3.5 Skin Deformation

The final stage is to envelope the bone and muscle constructs with a skin model. The skin is deformed by the underlying muscle motion and rigid motion of the limbs of the skeleton. The effects of the fat layer that lie underneath the epidermal and dermal layers of skin and over the musculature are accounted for.

1.4 Overview

We have explained the relevance of the work in the context of anatomy, biomechanics and computer graphics. Given the needs of these three areas, a problem statement was presented that motivated the creation of an anatomically-based modeller and its component soft tissue models. The contributions of this modeller and its parts were listed. The structure of this thesis consists of nine chapters: 1) Introduction, 2) Background, 3) B-spline Solids, 4) The Geometric Musculotendon Model, 5) The Physical Musculotendon Model, 6) Bones, Ligaments, and Skin, 7) System Design, 8) Results, and 9) Conclusion and Future Work. Chapter 1 motivates and introduces the need for an anatomically-based modeller. Chapter 2 reviews the related previous work in biomechanics and computer graphics as well as giving a primer on the related anatomical material we will use in this work. Chapter 3 will present the B-spline solid model and its mathematical properties. Chapter 4 will describe how initial shapes can be defined for the B-spline solid from different data sources. Chapter 5 covers the physically-based modelling aspects of the model for creating dynamic musculotendons. Chapter 6 will describe the modelling of other soft tissues. Chapter 7 details the design and implementation of the system architecture used in creating the modeller followed by several results illustrated in Chapter 8. Finally, Chapter 9 concludes with discussion and possible future work.

Chapter 2

Background

> Understand the freedom from the conformity of styles. Free yourself by observing closely what you normally practice.
>
> **Bruce Lee in *Tao of Jeet Kune Do, 1975***

It is important to examine some of the key ideas and prevailing methods from each of the fields of study - anatomy, biomechanics and computer graphics - with which we are intersecting. This chapter has a major section for each discipline, covering important issues relevant to muscle modelling that are specific to each area. Although some readers will be familiar with certain material, for others the same concepts will be new.

2.1 Anatomy

If you are already familiar with the anatomy of muscles, tendons, ligaments and bone, you can skip this entire section.

The biology of animals is a large area of study. Accordingly, we will be focusing on a set of tissues that are important for defining body form and motion: bone, ligaments, tendon and muscle. It is primarily these structures which create the complex machinery that allows animals to locomote and that will directly influence the changes of features in superficial body shape on the skin. These materials are categorized into passive and active structures. The passive structures have inherent physical properties, while only muscle is an active structure that can generate forces on its own.

2.1.1 The living machine

All these structures work together as a cohesive, efficient organic machine. The bones create a skeleton that provides structural support for the body as well as protection of the internal organs. Mechanically, in conjunction with musculotendon units, they make up a complex lever system that enables locomotion by applying forces on the external environment to manipulate objects or push off the ground. The ligaments provide stability at the joints to prevent adjacent bone segments from slipping away from each other. They act to guide motion as the joints articulate. Muscles are the primary force actuators of motion and are attached to tendons,

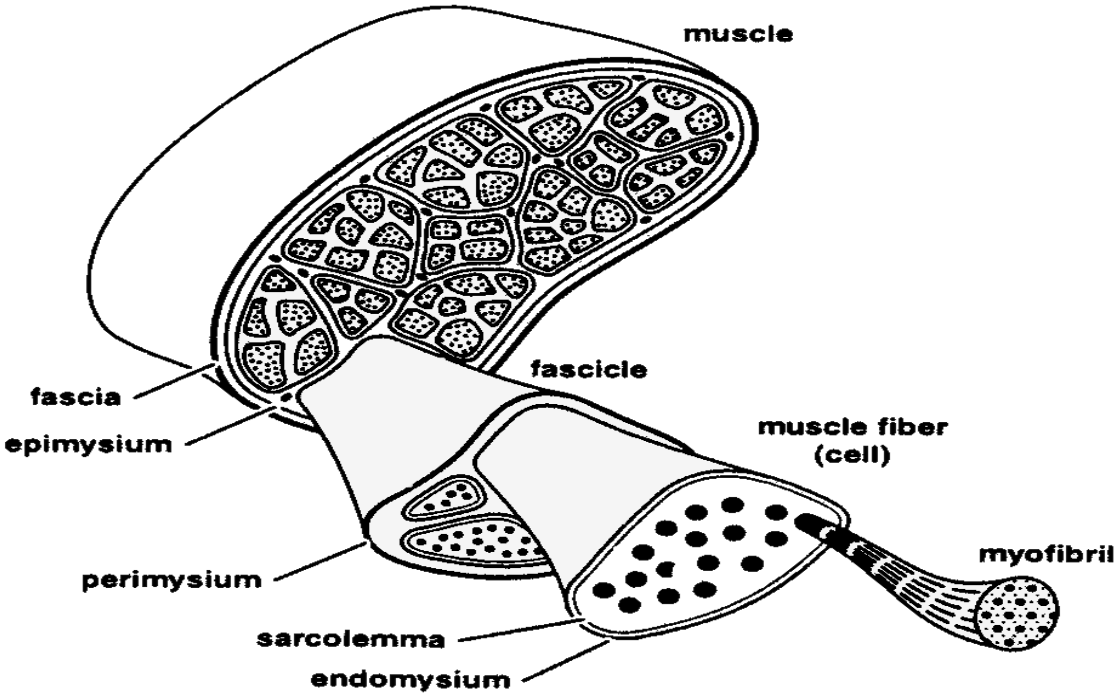


Figure 2.1: The hierarchical structure of muscle. From [52], reproduced with permission.

which transmit forces from the muscles to the bones that they are attached to. We will discuss each of these tissues in closer detail.

2.1.2 Active structures

Muscle

Muscles are considered active structures because they are capable of generating forces on their own. Muscles have a hierarchical structure (Figure 2.1) that at their lowest level consist of muscle fibre cells that contain contractile units called *sarcomeres*. Within these sarcomeres, there is a parallel arrangement of thick and thin filaments that according to the prevailing *cross-bridge theory*, form chemical bonds that generate the force in muscle[52]. The muscle fibres group themselves into bundles known as *fascicles* which are in turn enveloped by a material known as *fascia*. Each fascia group represents different heads of a muscle. For example, the gastrocnemius muscle consists of a medial and lateral head (Figure 2.2).

Within the fascia, the muscle fibres can have an orientation characterized by the angle they make with respect to the tendons they attach to. This fibre arrangement is known as the muscle's *pennation*. Figure 2.3 illustrates several types of pennation patterns observed in skeletal muscle. These arrangements directly determine the direction of forces produced.

Muscles can attach to tendon material either at a narrow site or over wide sheets of tendon known as *aponeurosis*. As both muscle and tendon work closely together to create a functional unit of force generation and transmission, we will collectively refer to this structure as a *musculotendon unit*. Tendon is several orders of magnitude stiffer than muscle. Therefore, the areas

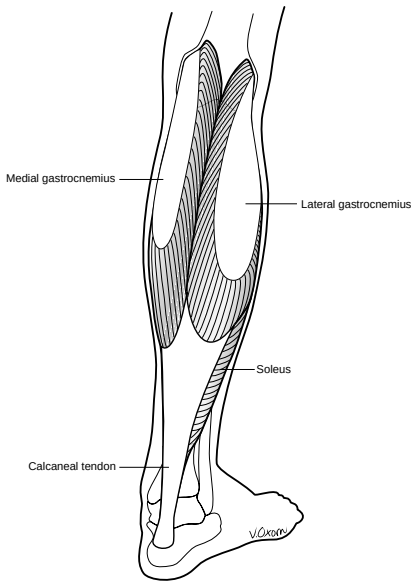


Figure 2.2: The gastrocnemius muscle in the lower leg. From Valerie Oxorn 1997, reproduced with permission.

where tendon is in contact with muscle have restricted movement compared to the uncovered portions of muscle[42]. The tendon portion of the musculotendon unit attaches to the skeleton at two different sites known as the *origin* and *insertion* . By convention, the origin is the more proximal site to the body's centre of mass and due to the greater mass located at the centre, is usually the less mobile of the two sites (Figure 2.4).

All these mentioned factors have a direct influence on the shape changes within the musculotendon units as contraction occurs. The musculotendon and its relationship to the skeleton also directly affects the resultant torque generated at a joint. A skeleton posed in two different configurations will be capable of exerting two entirely different sets of forces as both the magnitude and direction of forces are dependent on the length constraints set by the underlying skeleton. To capture these aspects of muscle, our strategy will focus on the development of a unified geometric and physical model for the musculotendon unit that can be used as building blocks to create a diverse set of musculoskeletal systems.

2.1.3 Passive structures

Passive structures consist of material that does not actively generate force on its own. Rather, these materials exhibit tension when strained by other external forces.

2.1.4 Bones

Bones make up the skeletal framework to which musculotendon and ligaments are attached. There are 206 bones in the human skeleton[87]. For our purposes, we will treat bones as rigid objects to simplify the dynamics by reducing the number of degrees of freedom in their equations of motion. In actuality, bone material is several times stiffer than muscle, with an

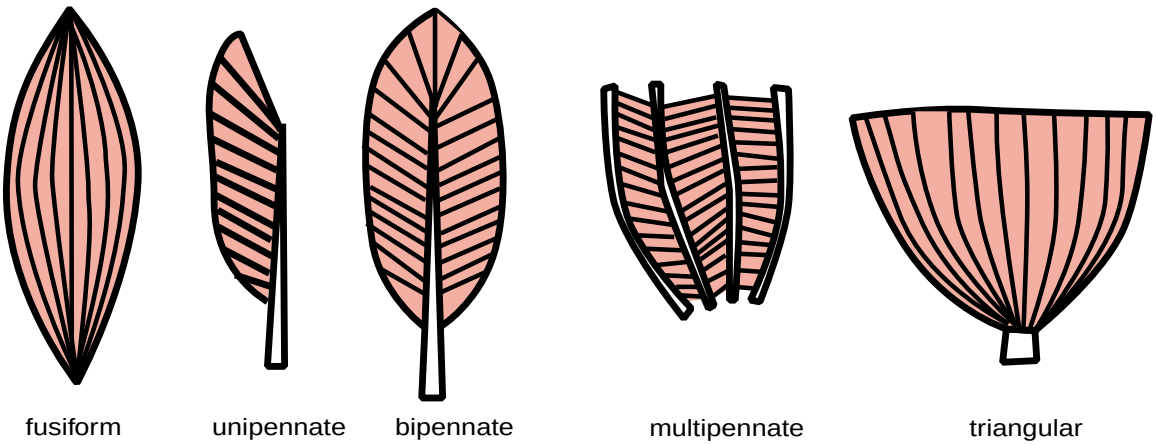


Figure 2.3: Different patterns of muscle pennation.

elastic modulus of $10^{11} dyn/cm^2$[2]. There are stress limitations of bone which if exceeded will cause fractures. We will make the assumption that in our model, the bones will be operating under nominal conditions to avoid accounting for the deformation characteristics of bone.

2.1.5 Tendon

Tendon is the second important component of the musculotendon unit. It transmits the force generated by the attached muscle to bone. The tendon material has different shapes depending on its attached material. The *external tendon* connects muscle to bone whereas the *internal tendon* or aponeurosis provides an attachment area for muscle and resembles a thin, sheet-like material.

Relative to muscle, tendon is much stiffer and stronger in tension when pulled. As tendon is made up of parallel-aligned *collagen* fibres, its Young's modulus is about $10^{10} dyn/cm^2$[2]. Due to the alignment of the fibres, there is tension along its longitudinal length, but there is still enough compliance to allow tendon to bend around joints. This allows forces generated by a muscle to be transmitted and applied to a different area where the muscle is located. For example, the hands have long tendons along the fingers that are attached to muscles in the forearm. Under normal operation, tendon is rarely stretched to more than 5% strain[53]. At about $8 - 10\%$ strain, tendon begins to experience irreversible strain and can experience damage. Tendon can also serve a spring-like role for storing elastic energy, as when the *Achilles tendon* alternatively stores and releases energy during running[3].

2.1.6 Ligaments

Ligaments attach adjacent bones to one another across joints. They help to guide joint movement and maintain stability at the joints during movement. As they are made up of *collagen*, they have similar physical properties to tendon. Ligaments can also be more elastic than tendon. *Elastic ligaments* such as the ligamentum nuchae located on the back of a horse's neck[2], have twice as much elastin than collagen[35]. They help to store elastic strain energy and require less energy to hold postures than active muscle contraction.

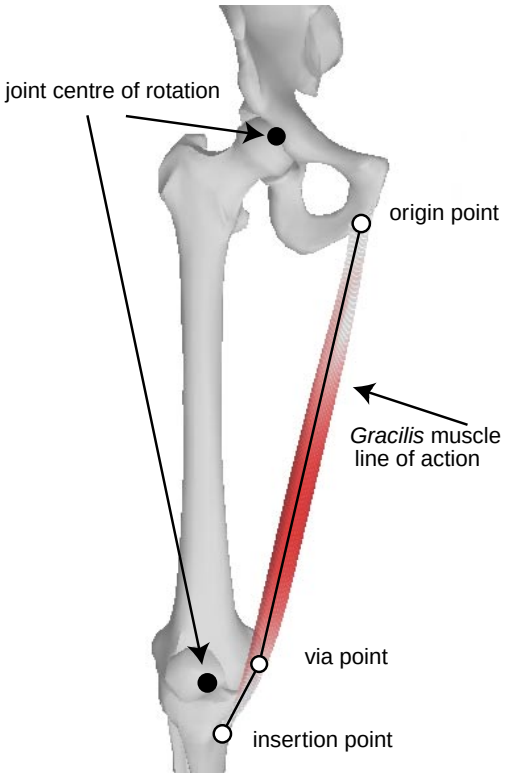


Figure 2.4: A musculoskeletal system configuration. The *gracilis* muscle is biarticular because it spans two joints. The line of action is in the direction of the point of origin to the point of insertion. In [24], a via point is added to show changing lines of action around bony protrusions.

2.2 Biomechanics

Having introduced the roles of the different anatomic elements we wish to study, we will need to quantify their behaviour by reviewing existing models for calculating forces. The majority of this work is from the field of biomechanics.

2.2.1 Modelling musculotendon forces

If we look at muscle as force actuators in a mechanical system, they have several unique characteristics compared to standard torque motors used as joints in robotic systems. Musculotendons can span over several joints. A biarticular muscle that can contribute motion directly to two joints plays an important role in determining the resulting motion. Simulations have shown that vertical biarticular jumping using the gastrocnemius produces greater heights than monoarticular muscle action[125]. The intrinsic properties of muscle and tendon can act as stabilizers for highly-explosive movements, potentially allowing more robust control. The built-in compliance in tendons smoothes out perturbations introduced to a motion[124]. If prescribed torques were applied to joints, any perturbations would most likely contribute to instability in the original motion.

In order to simulate the effects of the musculoskeletal system, biomechanists have created mathematical models for muscle force. The majority of models have focused on the skeletal

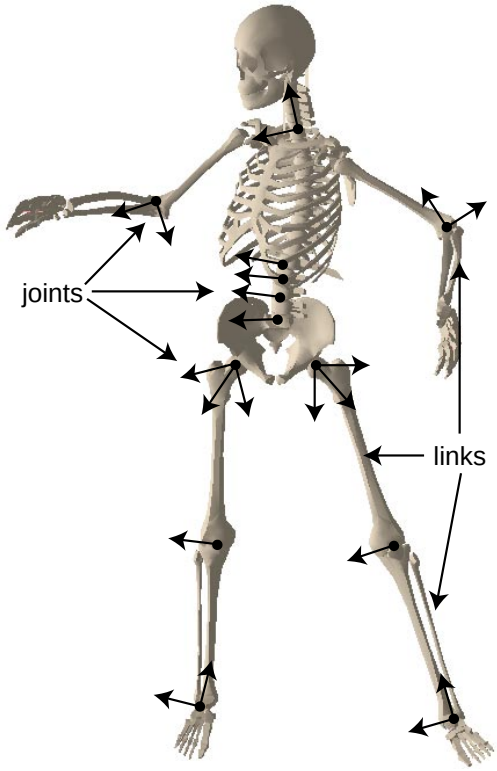


Figure 2.5: Articulated figures consist of rigid links that have various joint constraints between them.

muscle itself, as these forces are actively generated through voluntary neural control signals. Several models for representing the force generated by muscles have been created from the fields of *physically-based computer animation (PBCA)* and biomechanics simulation. In reality, these fields have the same mathematical framework for modelling the physical system. The distinction is that the models created in biomechanics have more parameters that relate to actual physiological measurements of musculotendons, making them more accurate. On the other hand, with PBCA, the models are simpler and easier to control, at the expense of accuracy.

2.2.2 Musculoskeletal system

Before discussing the various force models used for musculotendons, we will introduce the typical physical system arrangement for representing skeletons and the forces acting on them. The majority of studies simulating animal movement have represented skeletons as a set of rigid links that are held together by joint constraints. This assembly is termed an *articulated figure* (Figure 2.5). Depending on the formulations used to describe the equations of motion governing how these links move in response to external and internal forces on the system, the joints are modelled either as constraint forces that are maintained between adjacent links or using the principle of virtual work, equations of motion can be created that captures only the independent degrees of freedom with all the joint kinematic constraints built into the formulation. Both methods allow external and internal forces to be applied to the system. Internal

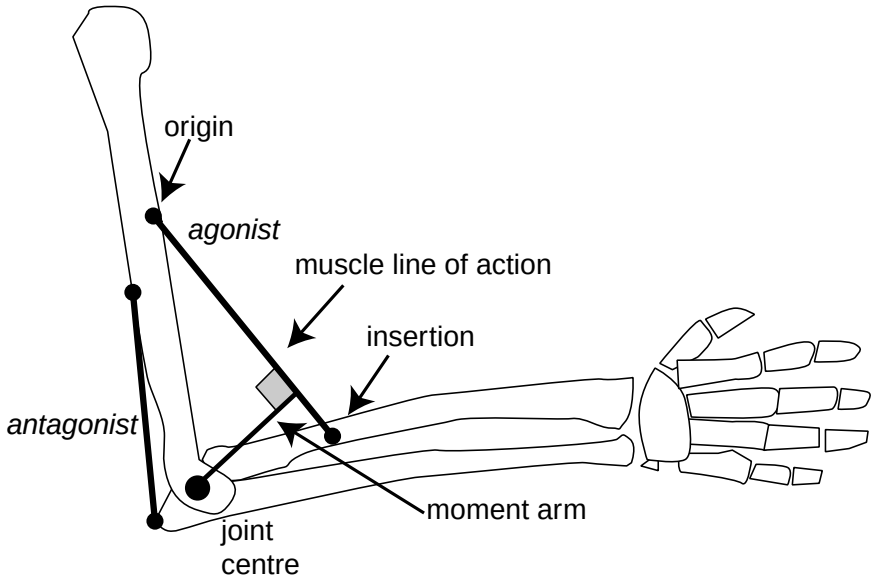


Figure 2.6: The muscle-derived torque is proportional to the muscle force along the line of action and its moment arm about the joint. Muscles act in agonist and antagonistic groups at a joint to pull in opposing directions for bidirectional joint movement.

forces are generated by actuators. For example, a robotic arm may have motor actuators at its joints to move the links while animals have muscles which generate motion.

2.2.3 Pure torque models

Instead of modelling each of the musculotendon actuators contributing to the movement of a joint, the resultant torque can be applied directly at the joints. This eliminates the need for a musculotendon model because we have a one-to-one relationship between joints and actuators. In a biological joint, muscles come in antagonistic pairs that pull in opposite directions, creating opposing torques (moments) in a joint (Figure 2.6). Consequently, there are many possible combinations of forces that can be applied between opposing muscles to create the same resultant torque at a joint. Applying control signals to torque motors directly eliminates the redundancy inherent in muscle actuators. Working with pure torques, we can use inverse dynamics to determine the required forces needed to obtain a prescribed motion in a skeleton. Isaacs and Cohen[63] used this method to embed a keyframing animation system within a dynamics framework.

The majority of work in controller synthesis attempts to solve controlling joint torques to create various motions. Hodgins developed control laws for torques generated at different phases of motion[55] and van de Panne has used dynamic programming[95] and stochastic search techniques[94] for synthesizing controllers that use torque-based actuators. These methods use *proportional-derivative (P-D) control* for their low-level torque generation. Given a desired target angle θ_d for a joint, the torque is computed as:

$$\tau = k_p(\theta - \theta_d) - k_d\dot{\theta}, \tag{2.1}$$

where θ is the current angle and $\dot{\theta}$ is its time derivative. The coefficients k_p and k_d are stiffness and damping parameters to control the elasticity and viscosity effects in the joint. By having appropriate combinations of k_p and k_d, underdamped, and critically-damped motion can be achieved as the joint is drawn to θ_d.

The drawbacks of using torque motors are the lack of intrinsic characteristics associated with muscles. Muscle geometry for deforming skin must be created separately. Muscle-generated forces are nonlinear functions of muscle length, shortening velocity and neural activation. All of these parameters create the dynamic viscoelastic characteristics of the muscle actuator. In contrast, techniques that use P-D control usually keep the viscoelastic coefficients constant throughout a simulation. Following this approach, these coefficients must be hand-tuned until reasonable motions are achieved. As muscle forces are influenced by their moment arms, joint changes during limb motion will affect the range of forces produced. Under a pure torque scheme, dynamic limits of torque for each body part must be computed as in [74]. Treating joints independently with their own torques does not take into account the dependencies that occur in biarticular muscles (muscles that directly contribute to motion in two or more joints). Biarticularity can directly affect the performance of motions such as jumping[125]. Optimization studies for human motion have shown that using muscle models instead of pure torque models in the objective function will produce physiologically reasonable forces[22].

2.2.4 Linear Spring-Damper Muscle Actuators

Rather than model forces indirectly through torques, the forces can be generated as linear actuators whose direction is determined by a line segment connecting two limbs at an origin and insertion point. McNeill Alexander refers to muscles as being spring-like in various situations, particularly when the length of the tendon is relatively long compared to the muscle, such as in the leg muscles of a kangaroo[3]. Raibert was inspired by this observation to create "pogo-stick" hopping robots that use pneumatic air springs which are essentially spring-damper based actuators[102]. In computer graphics, these linear forces were modelled as spring-damper systems (Figure 2.7). Linear springs and dampers have been used successfully to animate fish[119], and snakes[83]. The general form of these force models is:

$$f^m = k_p(l^m - l_o^m) - k_d\dot{l}^m. \tag{2.2}$$

By choosing appropriate stiffness (k_p) and damping (k_d) coefficients, the natural frequencies of motion can be adjusted. The muscle's spring restlength l_o^m can be changed during a motion to provide a variable target length for changing pose configurations. This representation has worked well with periodic motions that are inherently sinusoidal, but they may not be suitable for a larger class of motions such as quick-start motions that require sudden changes of velocity and may require more nonlinear elasticity and damping in the dynamics. Biomechanists require a model that has parameters corresponding to actual muscle and tendon measurements to study real musculotendon function. Such a model can allow empirical measurements to be used to create a model that can be parameterized to capture the features of any skeletal muscle in a body. The same actuators can then be used in a wider range of motions without needing to re-tune the parameters for new motions. To provide these capabilities, biomechanists use a more sophisticated linear actuator consisting of three elements that can be individually modelled. Hence, these actuators are collectively referred to as the *three-element Hill-based models*.

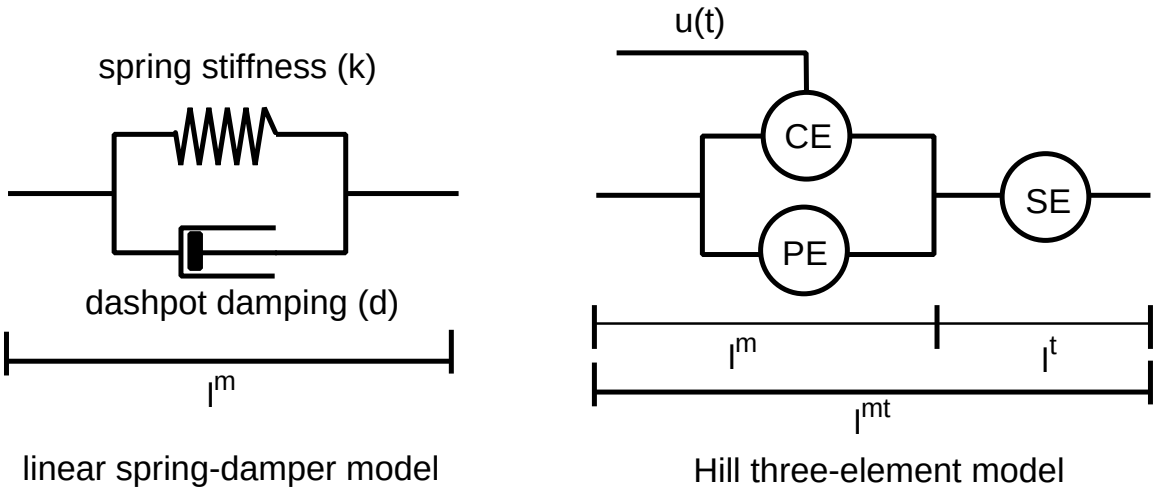


Figure 2.7: Viscoelastic models for linear actuators. **Left**: a standard linear spring-damper system **Right**: the Hill three-element model consisting of the contractile element (CE), parallel element (PE), and series element (SE). The CE and PE act over the muscle length (l^m) and the SE is dependent on the tendon length (l^t). The musculotendon length is l^{mt}

2.2.5 Three-Element Hill-based models

If you are familiar with the three-element Hill-based model for muscle force generation and do not care about alternative formulations for the different elements, you can skip over this section and the sections on the series, parallel and contractile elements.

> Hill's model is simple if the contractile element is entirely stress free and freely distensible in the resting state, and is described exactly by Hill's equation when the muscle is activated, if the series and parallel elements are elastic, and if the whole muscle is a simple combination of identical sarcomeres in series and in parallel. Unfortunately, none of these is true.
>
> **Y.C. Fung, *Biomechanics: Mechanical Properties of Living Tissues, 1981***

The quotation above serves to remind us that although the three-component Hill-based model (Figure 2.7) has been used extensively in the literature[138, 93], it is still not complete in capturing all observed phenomena in muscle. However, its predictive power for estimating qualitative patterns of muscle activation is well-supported in several different motion studies[121, 92]. It is a phenomenologically-based, lumped-parameter model based on a series of controlled experiments on muscle, namely the force-length and force-velocity dependencies observed in active muscle[138]. The model has three major components: the series element (SE), the parallel element (PE) and the important contractile element (CE). Subsequent work has sought to learn more about each of these mechanisms and practitioners generally use different forms of the equation based on desired characteristics of muscle on which they wish to focus. It is not uncommon to remove one of these elements if they do not have a significant effect on forces generated. Reviewing the physiological motivation behind the design of each element is important for deciding how to use the three-element model in a combined geometric and physical 3-D musculotendon context.

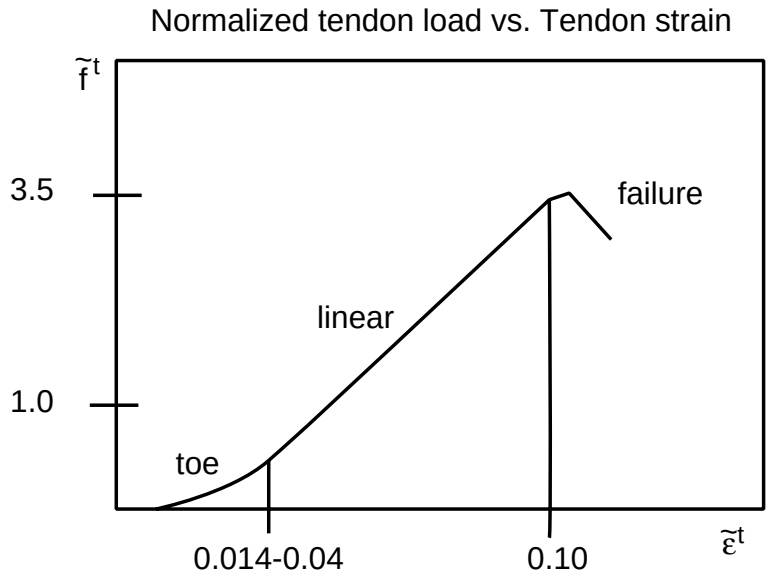


Figure 2.8: The normalized version of the tendon force-strain curve showing the toe, linear, and failure regions. Based on data from [138].

2.2.6 Series Element (SE)

The series element (SE) (Figure 2.7) lumps together the effects of several biological materials in musculotendon. This element represents mainly the elastic effects of tendon and intrinsic elasticity of structures within the sarcomere[36], isolated by quick release experiments of contracting muscle[7]. Although important at the sarcomere level, the elastic effects of the sarcomere can usually be omitted for whole muscle studies because the tendon elasticity dominates[138].

It is claimed in [138] that the aponeurosis (internal tendon) experiences the same strain at the external tendon. Tendon also has viscous properties. Fung has described hysteresis properties during cyclic loading and unloading of ligaments[36], which are made of the same material as tendon: collagen and elastin. This viscous effect can also be removed, as Bahler[7] and Hatze[50] has shown that the damping component is negligible.

Given these simplifications, we can focus on the stress-strain properties of tendon. A typical curve is shown in Figure 2.8. The x-axis refers to the SE strain, which is calculated as

$$\epsilon^{SE} = \frac{l^{SE} - l_o^{SE}}{l_o^{SE}}. \tag{2.3}$$

In whole muscle simulation, the descriptor SE can be interchanged with t for tendon. The value ϵ^{SE} refers to the elongation of the SE. The initial toe region has a curved nonlinear relationship to tendon strain and is hypothesized to be due to the way collagen fibres of tendon straighten out from their previous slack state as they are stretched. At around $1.4 - 4\%$ strain[138], the curve becomes linear until excessive strain of around 10% causes material failure.

Several different functions have been used to model this tension-strain function. Bahler[6] has used cubic polynomial relationships based on observations, but the coefficients used have

no physical interpretation. Hatze[50] and Fung[36] use an exponential function:

$$f^{SE} = k_s e^{\epsilon^{SE}-1} + d_{SE}\dot{\epsilon}^{SE}, \tag{2.4}$$

which is the solution of a first order differential equation of tendon behaviour. This is probably the most accurate model, but suffers from the expense of computing the exponential function. For simulation purposes, the shape characteristics of the exponential function can be substituted for a simpler quadratic function with conditional branches[122]:

$$f^{SE}(l^{SE}) = \begin{cases} 0 & \text{if } l^{SE} < l^{SE}_{slack}. \\ k(l^{SE} - l^{SE}_{slack})^2 & \text{if } l^{SE} \geq l^{SE}_{slack}. \end{cases} \tag{2.5}$$

The top branch recognizes that no tension is developed if the tendon length does not exceed the tendon slack length. This is comparatively different from normal linear springs which experience restorative forces when they are compressed. The tendons only experience these forces when they are stretched.

2.2.7 Parallel Element (PE)

This element of the Hill model (Figure 2.7) represents the passive elastic properties of muscle, disregarding the active contractile machinery of the muscle. The parallel element (PE) represents the connective tissue sheaths (endomysia, perimysia, epimysia, fascia) of the muscle[36], with the fascia dominating these elastic effects. Due to the material properties of these sheaths, tension is only produced when the PE is actively strained beyond its rest length. As with tendon, we can define the elongation of the PE as a unitless ratio:

$$\epsilon^{PE} = \frac{l^{PE} - l^{PE}_o}{l^{PE}_o}. \tag{2.6}$$

The term l^{PE} is usually equivalent to the length of the muscle portion (l^m) of the musculotendon unit. The passive portion of muscle force is illustrated in the graph in Figure2.9. To model this curve, Hatze[50] used a least-squares fit to an exponential:

$$f^{PE} = k(\exp(c\epsilon^{PE}) - 1). \tag{2.7}$$

The constants k and c can be adjusted to match the behaviour of different muscles. This model can be expanded to more precisely capture the individual structural influences of the different connective sheathes in muscle[51]:

$$f^{PE} = \sum_{i=1}^{n} k_i(\exp(c_i\epsilon^{PE}) - 1), \tag{2.8}$$

with each term corresponding to a different connective tissue. For simulation purposes, a simple parabola can be used for quicker evaluation[122]:

$$f^{PE} = k(\epsilon^{PE})^2. \tag{2.9}$$

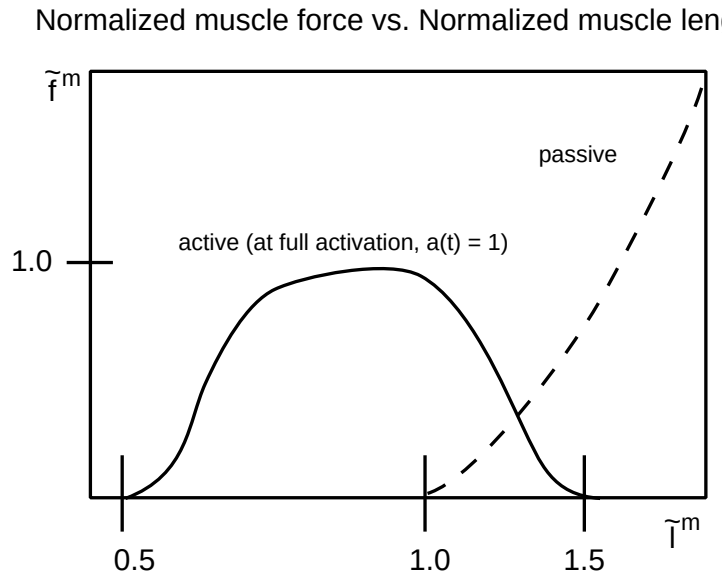


Figure 2.9: The normalized version of the muscle's force-length curve. The force response for fully-activated muscle has limited range. Passive force occurs when muscle is stretched beyond its rest length. Based on data from [138].

In addition to elastic effects, passive muscle has also demonstrated hysteresis effects during cyclic loading and unloading[36]. As muscles are made primarily of water, this viscous behaviour is expected. Hatze added a damping term[51]:

$$f^{PE}_{damped} = f^{PE} + d\dot{\epsilon}^{PE}. \tag{2.10}$$

If muscle can be shown to operate below its passive rest length, the PE element can be removed from the Hill force model as it will never play a role[6]. However, there are many motions where muscle is being actively lengthened, creating passive stretching of muscles while they are being actively stimulated. The active range of a muscle appears to be dependent on the animal species. Human leg muscles tend to operate at distances less than l^m_o, while frog muscles can operate in regions where the passive PE is a factor[52].

2.2.8 Contractile Element (CE)

The contractile element is the most interesting component of the three-element model. It contains the source of active force generation. This force is under voluntary control in skeletal muscle. A special parameter $u(t)$ is introduced to represent the time-varying neural control signal to muscle (Figure 2.7). The neural signal originates from the central nervous system and leads to motor neurons in the muscle. Each motor neuron innervates a set of muscle fibres which will contract when stimulated. This entire functional unit is termed the *motor unit*[138]. Further details of the chemical processes that occur during muscle contraction can be found in [52].

Experiments have been conducted to learn about the behaviour of the contractile machinery. It is through these examinations that the important *force-length* and *force-velocity* properties of

muscle have been discovered. Although these relationships are widely recognized, one should be aware that the measurements are made in controlled conditions, such as with fully-activated muscle or constant loading. An actual muscle *in vivo* undergoes dynamic loads and activations. An interesting use of simulation is to apply these force curves in controlled conditions to validate the predictive power of these models.

Force-length property

As early as 1894, Blix described that muscle force depends on its length[52]. It was not until 1966[52] that the exact nature of the force curve in relation to lengths of sarcomeres was revealed. By working with fully-activated muscle (in its *tetanus* state), the recruitment of all the muscle fibres is ensured[81]. The muscle length is constrained to be constant (*isometric*) and the generated force is measured. If this process is performed for several different lengths, the resulting curve appears in normalized form as shown in Figure 2.9.

This curve has previously been modelled with quadratic functions[124], piecewise cubic splines[19] and piecewise line segments[138]. The use of piecewise line segment is the least expensive to compute and conveniently attributes each line segment to a specific stage in the hypothesized cross-bridge contractile process within the sarcomeres[52]. In practice, the force-length function is smoother in shape, and the quadratic form can be used[122]:

$$f^{CE}(l^m) = f_o^m \left[1 - \left(\frac{l^m - l_o^m}{W} \right)^2 \right]. \tag{2.11}$$

Note that the muscle length is considered to be the same as the length of the contractile element, so that $l^{CE} = l^m$. The values l_o^m refers to the isometric length where the maximum isometric force, f_o^m is generated (Figure 2.9). The parameter W adjusts the width of the parabolic curve. Figure 2.9 clearly shows there is a limited operational range for active force generation. The lower range of the curve starts at 0.5 and ends at 1.5[138]. For sub-maximal activation, the curve is usually linearly scaled by the activation[122, 138].

Force-velocity property

In addition to the dependency of muscle force on its length, the force is also influenced by the muscle's velocity of shortening (Figure 2.10). In order to quantify the relationship, fully-activated muscle was clamped isometrically and suddenly released to allow shortening against an external load. The resulting relationship was first described by Fenn and Marsh[32]. However, the model most frequently used was the hyperbolic equation that Hill formulated[54]:

$$f^m = \frac{f_o^m b - a v^m}{b + v^m}, \tag{2.12}$$

where v^m is the velocity of *shortening* of the muscle. This relationship is invertible so that velocity can be calculated from the measured tension:

$$v^m = b\frac{f_o^m - f^m}{f^m + a}, v_o^m = b\frac{f_o^m}{a}. \tag{2.13}$$

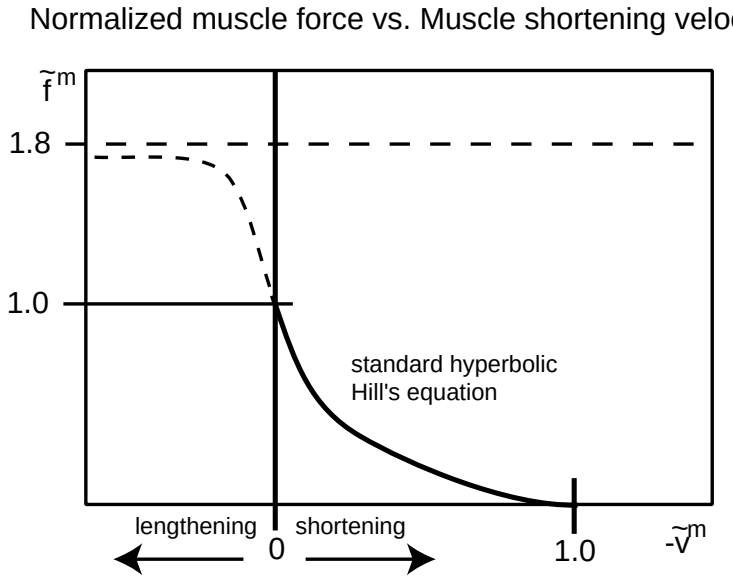


Figure 2.10: The normalized version of the muscle force-velocity curve. As the muscle shortens, its behaviour can be described by the hyperbolic Hill's equation. Muscle can lengthen during periods of negative work, where the muscle is being stretched while activated. Based on data from [138].

The parameter v_o^m is termed the maximum velocity of shortening and is experienced when there is no load on the muscle. From Equation 2.12, when the muscle is isometric ($v^m = 0$), the force developed will be the maximum isometric force (f_o^m). The coefficients a and b are termed *Hill coefficients* and roughly represent the effects of temperature and type of muscle respectively. By the type of muscle, we are referring to slow, fast and mixed twitch fibres that can contract at different speeds[52]. For mammalian muscle, slow fibres contract at maximum speeds of $6 \cdot l_o^m/s$ and fast fibres at $16 \cdot l_o^m/s$[52]. Mixed fibres can be modelled as a weighted combination of slow and fast fibres.

It is important to realize that since Hill's equation was created under maximum activation and constant loading, its accuracy in typical operating conditions is limited. Studies measuring shortening velocity under changing load conditions have shown deviations from the hyperbolic form[64]. Many motions in the body may be initiated from submaximal activations, where Hill's equation may not apply. Modifications have been made to Hill's model to account for these effects, mainly by paying closer attention to activities of the contractile element in the sarcomeres during muscle shortening[51]. Others have modified Hill's equation to account for the effects of shortening and its influence on depressing force production[82]. Nevertheless, Hill's original equation is used extensively in biomechanics simulation[122].

Activation/contraction dynamics

The previously-mentioned force-length and force-velocity relationships have been measured with fully-activated muscle. As neural stimulation to muscle occurs as a train of pulses, the frequency will determine whether tetanus (full activation) can occur. At lower frequencies, each pulse is followed by a single *twitch*. As the twitches occur closer together in time as fre-

quency increases, the twitches will fuse into a steady state of force production[36]. Typically, there can be several milliseconds of delay before the initial stimulation is given and the muscle begins to contract[98].

Zajac has accounted for this delay by creating a first-order differential equation relating the activation state of the muscle, $a(t)$, to the neural excitation input signal, $u(t)$[138]:

$$\dot{a}(t) + \frac{1}{\tau_{act}}(\beta + (1-\beta)u(t))a(t) = \frac{1}{\tau_{act}}u(t). \tag{2.14}$$

The rate τ_{act} is the time-constant when the muscle is fully excited ($u(t) = 1$). Similarly, we can define a time-constant, $\tau_{deact} = \frac{\tau_{act}}{\beta}$, when muscle is deactivated ($u(t) = 0$). The constant β is the ratio of the time-constants for muscle activation to muscle deactivation and consequently is bounded between zero and one. A consequence of this model is that active contractile force will rise faster during excitation than relaxation, a property that is stated to be well-observed in [132]. For biomechanical simulation, Bogert describes a simple version, assuming the rate of activation and relaxation are the same ($\tau = \tau_{act} = \tau_{deact}$)[122]:

$$\dot{a}(t) = \frac{1}{\tau}(u(t) - a(t)). \tag{2.15}$$

At the other extreme, Hatze has developed a detailed model that factors in the various biochemical processes that occur in the contractile element during the activation and force generation process[51].

2.2.9 Putting it all together

Now that we have described each of the elements of the Hill model, we can combine their effects together into a model for musculotendon. From Figure 2.7, we have the relationship:

$$f^t = f^{SE} \tag{2.16}$$
$$f^m = f^{PE} + f^{CE} \tag{2.17}$$
$$f^{mt} = f^m = f^t, \tag{2.18}$$

where f^{mt} is the force generated from the musculotendon unit. Since $l^t = l^{mt} - l^m$, these force quantities can be expressed as functions of only muscle length, muscle velocity and activation: $f^t(l^m), f^m(\dot{l^m}, l^m, a(t))$. This relation among l^m, $\dot{l^m}$, and $a(t)$ forms a volume where each isosurface created at a fixed activation corresponds to a force-length-velocity constraint surface (Figure 2.11)[51, 131]. It is this volumetric function which we are modelling in the contractile element to capture the active effects of muscle.

2.2.10 Normalized version of the three-element Hill model

To encourage a reusable musculotendon model, it is convenient to abstract the basic nature of the force-length and force-velocity relationships from the specific values produced in different muscles. Creating a parameterized version of Hill's three-element model allows muscles to be configured with a set of standard muscle measurements. The standard force curves of the

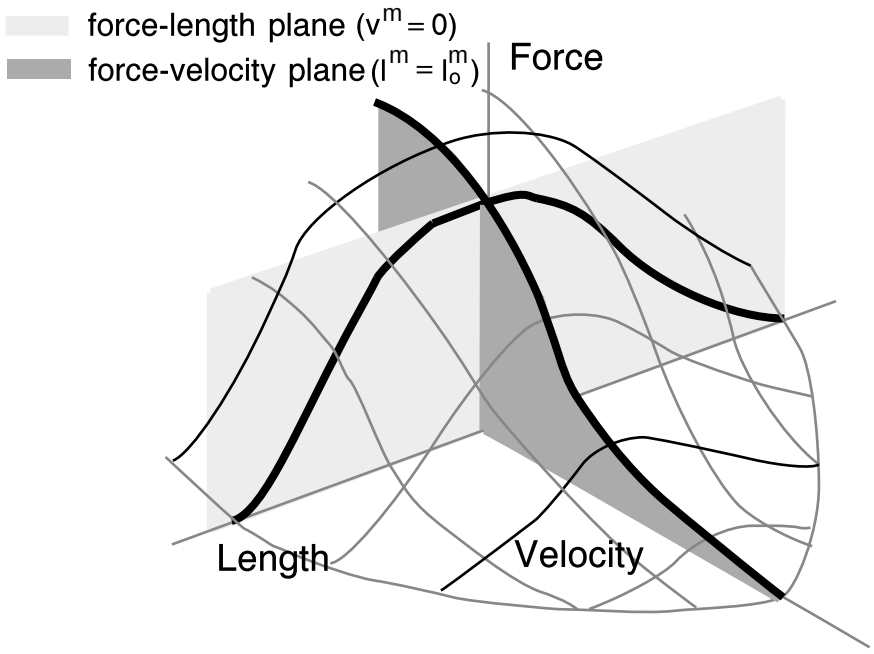


Figure 2.11: The force-length-velocity volume.

elements can be normalized with respect to parameters such as *maximum isometric length* (l_o^m), *maximum isometric force* (f_o^m) and *tendon slack length* (l_{sl}^t). The normalized curves can be scaled with these parameters to represent any skeletal musculotendon unit, as long as we can provide estimates for these values. The resulting physical quantities become unitless ratios. Zajac formalized this model and termed it the *dimensionless musculotendon actuator* in his classic review article in 1989[138]. Normalized musculotendon ratios are calculated as follows:

$$\tilde{f}^m = \frac{f^m}{f_o^m}, \quad \tilde{l}^m = \frac{l^m}{l_o^m}, \quad \tilde{l}^t = \frac{l^t}{l_{sl}^t}. \tag{2.19}$$

Here, f_o^m is the maximum isometric force which is experienced at the maximum isometric length, l_o^m. It is often estimated using the physiological cross-sectional area (PCSA):

$$f_o^m = 25N/cm^2 \cdot PCSA. \tag{2.20}$$

The PCSA is defined in [137] to be the volume of the muscle divided by its "gross muscle length or its fibre length with or without accounting for pennation". The interpretation of "gross muscle length" often depends on using simplified geometrical representations for muscle[2] (Figure 2.12):

$$PCSA = \begin{cases} m/(\rho l) & (fusiform) \\ m/(2\rho t)\sin(2\alpha) & (unipennate) \end{cases} \tag{2.21}$$

where m, ρ, and l are the muscle mass, density and length. The pennation effects are handled by t, which is the layer thickness and α, which is the pennation angle.

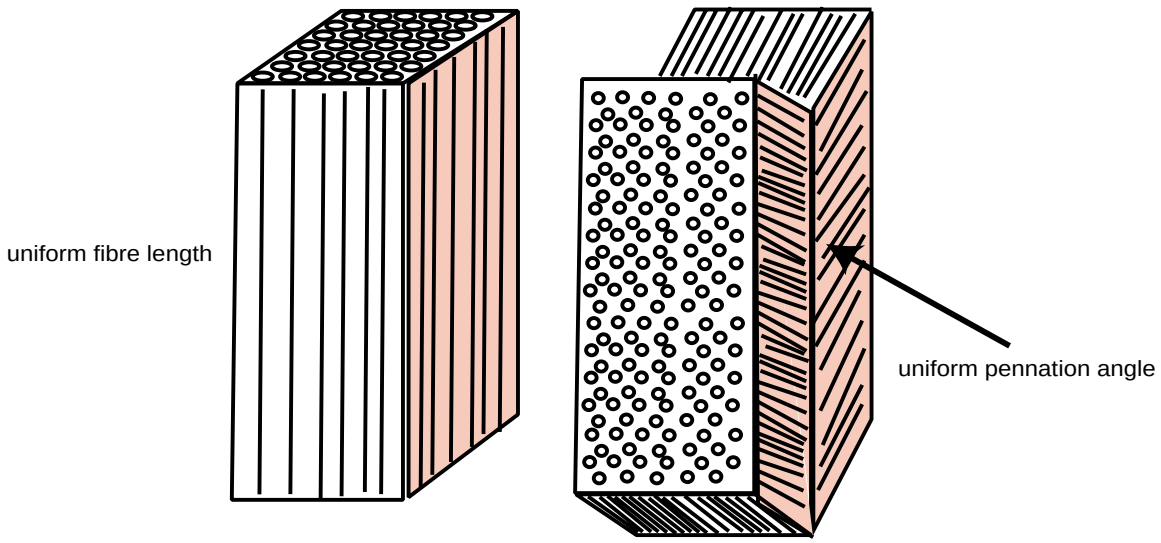


Figure 2.12: The fibre arrangements for calculating the physiological cross-sectional area (PCSA) are often simplified to have uniform fibre length and pennation angle. Adapted from [2].

Almost all subsequent work in simulation of biomechanical musculoskeletal systems have used a form of this dimensionless model to parameterize it for different muscles[93, 19, 122]. Although the basic Hill model cannot reproduce phenomena such as yielding during muscle lengthening or force enhancement after stretch[131], the simplicity of Hill's three-element model and its relatively few parameters has allowed the standardization of reported measurements in the literature to allow reuse of data in different motions. Pandy uses the same muscle parameters to successfully simulate walking and jumping motions[91]. Yamaguchi has created collections of these parameter values for several muscles[137]. Simulations have shown that the Hill model can be used to produce results that agree with observations measured from actual muscles[91].

2.2.11 Muscle line of action

Hill's three-element model provides the magnitude of the forces, but the muscle's geometry determines the direction or *line of action* of the force vector. There are two main methods for representing the line of action. Delp uses piecewise line segments[25] (Figure 2.13) for musculotendons attached to a skeleton. Intermediate *via points* (Figure 2.4) are introduced at different joint angles to constrain the muscle's line of action when it wraps around bone or is forced to go through tendon sheathes called *retinacula*. Others[65] use centroid curves which are constructed by interpolating a curve through estimated centroids from several transverse sections throughout the muscle. Although centroid curves appear to visually follow the muscle's shape more accurately, the line segment approach may be acceptable for physical modelling. The requirement that muscles create *couples* or sets of forces that sum to zero indicate that if a muscle was to contact a bone at a finite number of positions, the net sum of those contact forces would be zero and hence the line of action can be simplified with piecewise line segments. The challenge still remains to find the correct directional vectors of these line

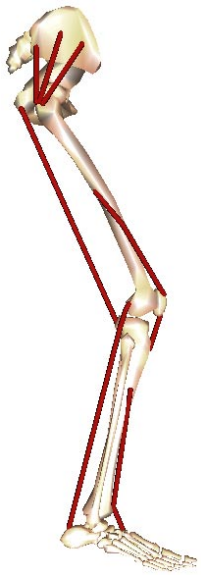


Figure 2.13: The SIMM system[24] uses line segments to represent musculotendons. Reproduced with permission.

segments.

However, there are several sources for potential inaccuracy when using line segments for muscle line of action:

1. **Insertion and origin points** As the musculotendon is represented with line segments, they necessarily only have a single point of attachment at both the insertion and origin ends. Single points of attachment of musculotendon to a skeleton may be suitable if the tendons attach over a small surface area. However, where muscles attach over a large area, the lines of action distributed over this area will substantially differ from that of a single point. This can be corrected by approximating the musculotendon units with several lines of action. Van der Helm has shown that six lines of action can be used without losing significant accuracy compared to a more extensive set of two hundred line segments[123]. Furthermore, we can sum the forces and moment vectors generated by these points to get a single force and moment vector, allowing us to apply these loads only once per body rather than separately in succession.

2. **Lack of inter-muscle effects** The large assemblies of line segments that can result in these simulations neglect to consider the inter-muscle collision forces that will occur as they exert pressure on each other.

3. **Architectural modelling** These inaccuracies have led researchers to incorporate more architectural features into the model, primarily targetting the gastrocnemius of muscle. There have been several attempts[136, 68, 141] to capture the geometrical configuration of aponeurosis and pennation of muscle fibres. The models are constrained to preserve volume, but use simplified shapes to approximate the musculotendon's shape. Furthermore, the muscle was only studied in isolation.

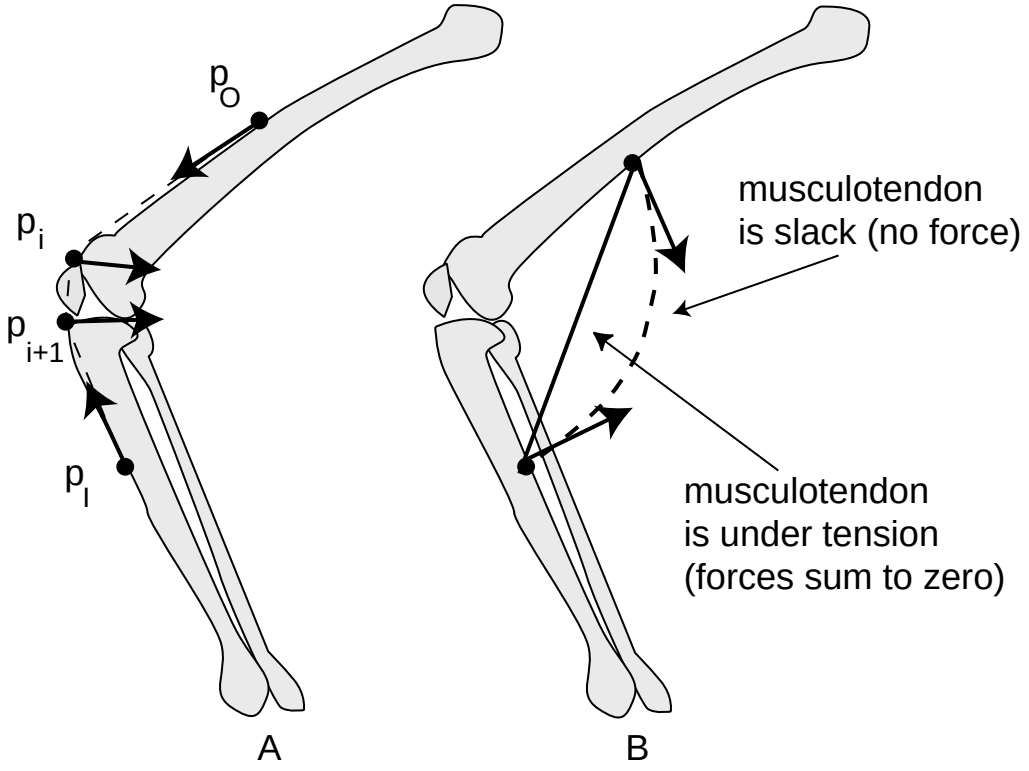


Figure 2.14: Different internal muscle force configurations in an articulated figure.

4. **Pennation** The orientation of muscle fibres to the tendon tissue are generally accounted for by a single angle of pennation. This angle is used to compute the component of force being applied by the muscle to tendon. Detailed architectural modelling can capture varying pennation within a muscle.

2.2.12 Internal muscle forces

Muscles create *internal forces* in the musculoskeletal system. One part of the system (musculotendon) exerts a force on another part (the bone). Consequently, the total momentum of the system must be conserved in the absence of external forces. This implies that the centre of mass of the system should experience no acceleration after all external forces are removed. The individual momentum of the links can change, but the total sum is constant.

Regarding muscles, we must be careful not to introduce force components that violate this principle. This can be achieved by ensuring the sum of all internal forces acting on the bones of the skeleton sum to zero to result in no momentum changes. This example is illustrated in Figure 2.14. In diagram A, we have two resultant force vectors applied at the origin, p_O, and insertion, p_I, attachment ends of the muscle. This results in a net force component that will pull the system to the left. The balancing force required to make the vector sum to zero can be applied at points of contact p_i and p_{i+1}. Alternatively, if we assume equal tension along the muscle, forces in the direction of p_i to p_{i+1} can be applied in both directions and the forces at the attachment areas are ignored because their effects on the bone are cancelled out. Another

interesting situation is diagram B. The force vectors along the dashed line at the attachments do not sum to zero, yet there is a lack of contact along the muscle length to provide application of a balancing force to zero. This apparently violates the conservation of momentum. How can we explain this situation physically? The musculotendon unit can only generate force if tension is achieved with the muscle. As the muscle fibres contract, as long as they are able to shorten without resistance, there will be negligible tension experienced at the bone. The fibres will be in a slack state. When the fibres begin to experience resistance to the shortening, muscle force will begin to generate. This can happen as the muscle's line of action begins to straighten out (solid line) or when the contracting muscle begins to exert pressure on interior muscles.

The net result of musculotendon units on the skeleton is that the translational forces sum to zero and only a pure torque remains. This does not imply that we can ignore musculotendon geometry altogether and apply the torque directly. Torque is a function of the moment arm of the muscle about the joints it spans. The moment arm is directly determined by the geometry of the muscle and its lines of action.

2.2.13 Calculation of muscle-derived torques

As the joints undergo rotation, the moment arm may change as the line of action moves closer or farther away from the joint centre. For regions where the musculotendon can wrap around bony surfaces or have a curved path from origin to insertion, intermediate points can be introduced to define a piecewise line segment to represent the musculotendon's line of action over the underlying tissues[24]. This allows a better estimate of the moment arm of the musculotendon than simply using a two point line segment. In [24], the intermediate points stay fixed to their underlying bones during joint motion and can be made active only within certain joint ranges to model changing musculotendon kinematic conditions. It is not clear if these points should stay fixed when they are active, as there may be cases where the musculotendon's dynamics cause the points to shift relative to their underlying bones. For example, in sets of tightly, packed muscle, as underlying muscles contract, they would push out more superficial muscles, resulting in changing moment arms for the latter muscle. Furthermore, the matter in the muscle can have its own inertial motion where the centroid axis can move independently of joint angle.

The moment arm can be calculated geometrically as the perpendicular distance from the joint centre to the muscle's line of action. However, this method does not work when the joint centre is not fixed[122], such as in the knee[88]. For cases where the joint kinematics are not restricted to a single rotation axis and fixed joint centre, we can invoke the principle of virtual work[72] to compute the moment arm for a generalized coordinate, q_i, to produce a generalized force, Q_i on that degree of freedom. The generalized force due to a particular musculotendon can be calculated as:

$$Q_i = -f^{mt}\frac{\delta l^{mt}}{\delta q_i}. \tag{2.22}$$

The partial derivative of l^{mt} with respect to q_i is the musculotendon's moment arm for that degree of freedom. Delp and Loan[24] apply this method for their piecewise line segments and call it the *partial velocity technique*. If q is a single joint angle, then Q is the torque about that joint's single axis.

2.2.14 Joint modelling

We define a *joint* as a constraint between two rigid objects. The proximal object is considered to be the parent link while the distal object is the child link (the one furthest from the torso or centre of mass of the figure). Mathematically, the joint is a transformation expressing the child link's translation and orientation with respect to the parent's frame of reference. Usually the transformation is constrained in some way, such as rotation about a single fixed axis. The most general case of a joint is a six degree of freedom joint where the child can take arbitrary translation and orientations relative to its parent.

In practice, computer animation applications use one to three degrees of freedom rotational joints in articulated figures. Keeping the joints simple allows the articulated figures to be represented with a minimal set of *generalized coordinates* that describe the configuration state of the figure. Smaller state vectors increase the speed of algorithms such as those used in controller synthesis[95] and inverse kinematics[40].

Biomechanical systems require higher fidelity of joint description. The SIMM (Software for Interactive Musculoskeletal Modelling)[24] system allows kinematic functions to be described that specify joint with changing joint centers and axes. This is done by interpolating a cubic spline through several key joints that constrain the joint centers and axes at several different joint angles. For complex joint interactions in the shoulder, Badler et al. uses a spherical surface to represent the rotational centre of the humerus[5], while Maurel et al. use an ellipsoid as a contact surface between the scapula and thorax[79].

This concludes a review of methods for computing the magnitude of musculotendon forces as well as the geometric issues for determining the force vectors and moment arms of these musculotendons. Very rarely do these biomechanical models account for the 3-D shape and volume of muscle and its interaction with other muscles. In contrast, geometric models in computer graphics and deformable object techniques from physically-based computer animation have been developed to capture the visual and motion effects of 3-D objects.

2.3 Computer Graphics

We now will examine the contributions that computer graphics has made on providing tools and techniques for visualizing and animating shapes. These techniques can be applied to represent muscle geometry, animate muscle contraction, and deform shape in response to interaction with other musculotendons and bones. We first will survey the major geometric primitives, followed by techniques for modulating or deforming shape characteristics. Finally, we discuss the use of physically-based methods for creating realistic motions by specifying the time-varying properties of these shape parameters.

2.3.1 Geometric shape primitives

Our pursuit of modelling muscle shape requires that a suitable mathematical model can be found to define the geometry and parameters to modulate shape. Several methods have been developed to model *deformable objects* that not only translate and rotate, but can be stretched, squashed or bent. The prevailing primitives are parameteric surfaces, implicit objects, polygons

and subdivision surfaces. In the following discussion, we will assume these primitives reside in a 3-dimensional cartesian space, $\Re^3$.

2.3.2 Notation for geometry and related coordinate systems

Let $O \subseteq \Re^3$ be the set of points belonging to a given object. If there exists a function $\mathbf{p}$ and domain $U \subseteq \Re^3$ such that $O = \{\mathbf{p}(\mathbf{u}) | \mathbf{u} \in U\}$, then $\mathbf{p}$ is a parameterisation of O. The tuples in $\mathbf{u}$ can be called the *material coordinates* or *parameters* of O. On the other hand, if there exists an algebraic function $\forall \mathbf{x} \in O$, such that $f(\mathbf{x}) - L = 0$, then O defines an *implicit* object. L is a constant used to specify a particular iso-surface boundary for the object. Implicit objects do not use material coordinates.

Finally, a geometric object can have a set of values $\mathbf{q}$ which can modify its overall shape. For example, a sphere can have a vector $\mathbf{q} = \{(x, y, z, r)$ where x,y,z is the centre of the sphere and r is the radius. A particular set of parameters in $\mathbf{q}$ will map to a shape configuration. However, it is possible for two sets of parameters to represent the same shape. These parameters $\mathbf{q}$ are referred to as *generalized coordinates* or *degrees of freedom* of the shape.

Parametric shapes

Formally, a parametric shape $\mathbf{S_p}$ is a vector function:

$$\mathbf{x}(\mathbf{u}) = \mathbf{S_p}(\mathbf{u}, \mathbf{q}), \mathbf{x} \in \Re^3, \tag{2.23}$$

where $\mathbf{u}$ is a vector that represents the *material coordinates*[118]. The material coordinates enumerate points within the shape's domain. If $\mathbf{u} \in \Re^2$, then $\mathbf{S_p}$ can describe a surface, where if $\mathbf{u} \in \Re^3$, $\mathbf{S_p}$ is a volumetric shape. The material coordinates and the generalized coordinates together are used in a vector function to produce the spatial coordinates $\mathbf{x}$ that are in the 3-D world.

In computer graphics and computer-aided design, tensor product parametric functions that use the B-spline, Bezier and Non-uniform rational B-spline basis functions are primarily used due to their local shape control properties.

Polygonal shapes

With scan conversion-based graphics hardware and 3-D application programming interfaces (API), a rendering pipeline scan converts polygons, mainly in the form of triangles or quadrilaterals. For interactive rendering of geometry, parametric shapes need to be tessellated into polygons or other shapes are modelled directly with polygons. The vertices and their edges connecting them form the boundary surface of the shape. Due to the ostensibly planar nature of polygons, a large number of polygons are needed to accurately capture regions of high curvature. The polygonal shapes can be changed by animating the vertices.

Subdivision surfaces

One of the major deficiencies of parametric shapes are their lack of topological flexibility. In the case of parametric surfaces, they are the topological equivalent of a rectangular or triangular sheet. Surface patches must be pasted together with continuity constraints at the boundaries

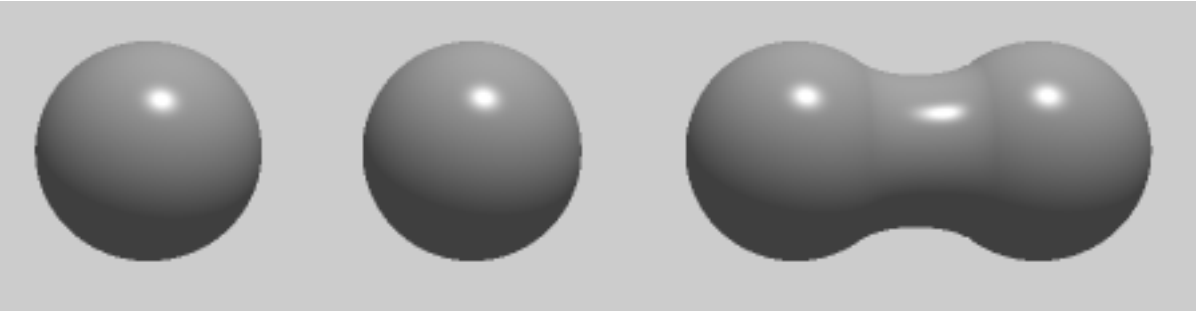


Figure 2.15: The two implicitly-defined spheres on the left are blended on the right because the strength of their overlapping fields matches the iso-level value corresponding to the blended surface.

to create surface areas with protruding branches. In contrast, a polygon mesh allows shapes of arbitrary topology to be defined. *Subdivision surfaces* combine the smooth characteristics of parametric surfaces with the topological flexibility of polygonal meshes. The surfaces can either be created by repetitive subdivision rules that create successively smoother meshes[15] or evaluated directly[110]. The ability to selectively refine a region of a mesh and create protrusions in surface shape make subdivision surfaces an appealing choice for modelling the skin of animals and humans[27]. However, finding useful techniques for animating these surfaces due to the dense set of mesh points are continuing open problems.

Implicit shapes

Parametric shapes explicitly evaluate points on a surface. By contrast, the points $\mathbf{x}$ of an implicit shape must be found as a solution to a root-finding problem:

$$f(\mathbf{x}) - L = 0, \tag{2.24}$$

for a given constant L. Whenever the scalar function f evaluates to an iso-level L, a corresponding iso-surface which forms the boundary of the shape is created. For shape control, the choice of f often takes the form of the sum of local field functions where each field has a characteristic decay with the single maximum located at the source of the radial force. Blinn used summations of exponentials centred at specific points to build his implicit shapes (he termed them "blobbies")[11]. Convolution surfaces use a curve or area as the field source rather than single points[13]. As the field source has a geometric interpretation, the Euclidean distance of the point $\mathbf{x}$ to the field source is usually calculated and used as the independent variable to determine field strength.

Where two or more field functions f overlap in influence, a smooth blending effect can be created (Figure 2.15) between each of them. These blending properties have allowed implicit objects to be used to model body shapes, but the general quality of the model seems to be an unnatural global smoothness as if the surface details have been low-pass filtered. Although texture mapping is possible[96], the texture coordinates for the iso-surface are not directly computable from the implicit function.

2.3.3 Animation and deformation of shape

Given the geometric representation using one of these four shape representations, the shapes can be adjusted by changing the values of various parameters, $\mathbf{q}$, that influence shape. In many cases, the size of $\mathbf{q}$ can be large, making it infeasible to specify these values manually. Techniques have been developed to animate these parameters either by creating high level functions that completely specify the values for the more numerous parameters or creating automatic methods that evolve $\mathbf{q}$ to animate a shape according to desired criteria. We will review two methods of choosing these parameter values: deformations and Lagrangian dynamics.

2.3.4 Deformation techniques

An advantage of being able to specify a finite set of parameters $\mathbf{q}$ for shape configuration is that it is easier than defining the position of every point in $\mathbf{x}$ individually. In cases where $\mathbf{x}$ is a continuous domain of the object, parameter modification can be used to specify shape modification. However, there may still be too many parameters in $\mathbf{q}$ for their manual modification to be feasible.

Deformations allow a higher level of manipulation handles to be used to specify and modify values in $\mathbf{q}$. These handles are usually represented by fewer parameters or a new set that may be intuitively easier to understand for a shape designer. The use of fewer parameters implies that dependencies are assumed between the elements of the original parameters. Various spatial or continuity constraints are maintained through these dependencies.

A deformable function and its inverse can be generally defined as follows:

$$D_{\Gamma}(\mu) = \mathbf{q} \tag{2.25}$$

$$D_{\Gamma}^{-1}(\mathbf{q}) = \mu, \tag{2.26}$$

where Γ denotes a set of animatable parameters which define the deformation, and the μ represent the local coordinate representation of $\mathbf{q}$ in the deformation space.

The inverse procedure is usually done once as a precomputation step for each $\mathbf{q}$ to produce a corresponding μ. Subsequent deformation operations involve changing the higher level set of parameters Γ, while repeatedly invoking D. For example, $D_{\Gamma(t)}(\mu) = \mathbf{q}$ illustrates the time-varying nature of $\mathbf{\Gamma}$, while $\mathbf{q}$ denotes a new configuration for the object's shape. The two deformation methods that are used in computer graphics are *volume-based* deformation and *feature-based* deformations. The former results in global changes of shape while the latter tends to localize shape changes around certain visible features of the object.

2.3.5 Volume-based deformation

In volume-based deformation, the geometry is usually embedded within a parameterically-define finite volume. The vector $\mathbf{q}$ usually consists of a set of 3-D points in the Cartesian coordinate system (vertices or control points). We can concatenate all these points into the monolithic vector $\mathbf{q}$:

$$\mathbf{q} = \{q_i | i = 0, \cdots, 3|\mathbf{q}| - 1\}. \tag{2.27}$$

The n^{th} point will have $(q_{3n}, q_{3n+1}, q_{3n+2})$ as its Cartesian coordinates. We apply the same indexing convention to the material coordinates $\mathbf{u}$. As the enclosing volume is deformed, the

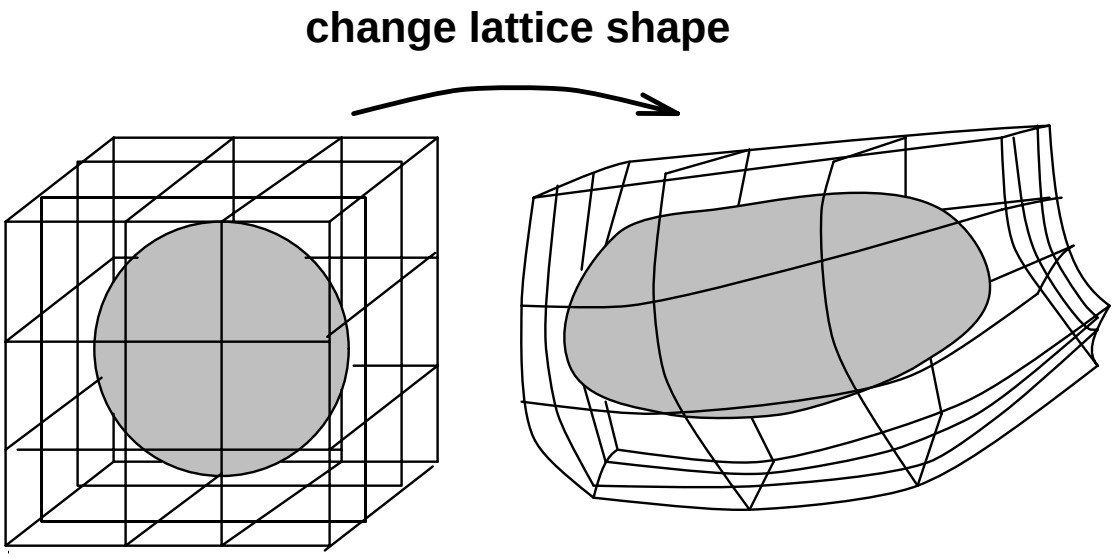


Figure 2.16: Free-form deformations are used to globally deform geometry embedded within the control lattice.

parameters $\mathbf{q}$ are modified with the same deformations, and results in a global change in shape configuration (Figure 2.16).

A common parametric volume deformation is the use of triple tensor product volumes:

$$D_{\mathbf{\Gamma}}(\mu) = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} B_i^u(u) B_j^v(v) B_k^w(w) \mathbf{\Gamma}_{ijk}, \tag{2.28}$$

where B are basis functions which create the tensor product, $(u, v, w) \in \mu$, and $\mathbf{\Gamma}_{ijk}$ is the equivalent of the point $\mathbf{\Gamma}_{(i|m|+j)|n|+k)}$. The points of $\mathbf{\Gamma}$ represents a a set of points that define a *control lattice* that influences the overall volume shape.

Sederberg and Perry introduced the term *free-form deformation* (FFD) to describe these parametric volume deformations. They used the Bernstein polynomial basis functions in Equation 2.28[106]. In order to implement the inverse procedure D^{-1}, which they termed "freezing", the two coordinate systems of $\mathbf{\Gamma}$ and μ were constrained to be axis-aligned to allow the inverse to be found by using simple translation offset calculations. Using a Bernstein volume requires a control lattice of $4 \times 4 \times 4 \ = 64$ points. The deformations possible were often too coarse to allow local refinements of the underlying geometry shape. Subsequent extensions such as extended free-form deformations[21] allowed several lattices to be connected together with continuity constraints and permitted lattices to be pre-deformed to more closely overlap the underlying shape, giving more control. Root-finding techniques were needed to evaluate the inverse procedure D^{-1}.

To avoid explicitly setting constraint conditions, B-spline volumes [48] and NURBs volumes [71] have been used to allow lattices to be extended and refined independently in each of its three dimensions. Unfortunately, to refine the points in one dimension required adding an entire new level 2-D grid of points into the lattice plane. MacCracken and Joy introduced subdivision FFDs based on the Catmull-Clark subdivision rules[78] to allow local refine of points within the volume.

The existence of all the control lattice points to define the solid, especially for subdivision solids which involve nested groups of points raises the problem of how to let an animator specify how their positions will change over time. Hsu devised a direct manipulation technique[59] to solve for the control points given a a set of equality constraints for points on the embedded geometry to be at particular spatial points.

2.3.6 Feature-based deformations

Geometry embedded in a deforming volume will be subject to global changes of shape. In other situations, a more local deformation is required that is related to defined features visible on an object. The first of these originated in image-based morphing[9] where directed vectors coinciding with image features were animated, locally deforming the image space in each vector's proximity.

The main characteristics of feature-based deformation are the influence each feature has in its local neighbourhood and allowing a point in an image or 3-D object to be influenced by several features, depending on a weight roughly inversely proportional to the distance of the point to each feature. Singh and Fiume use 3-D space curves termed "wires" to shape surfaces, adding creases and folds in the surface that follow the shape of the curves[109]. Later, we will describe anatomically-based methods which use muscle shapes as features to deform surrounding skin geometry.

2.3.7 Physically-based Models

The previous discussion on deformation techniques presents a method of specifying changes in shape through parameter adjustment. For physically-based muscles, the time-varying properties of these parameters need to be modelled. There have been several techniques for embedding physical, dynamic properties into geometry.

Viscoelastic networks

The first attempts to model deformable objects consisted of representing objects with a distributed set of mass points that were connected to each other by links that had viscoelastic bonds between them (Figure 2.17). Since the mass is concentrated at a single point for each particle, the equations of motion are relatively easy to determine and the resultant forces are simply the vector sum of all forces acting on that point. Reaction forces for various constraints can be determined explicitly by analysing the dynamics at each point[99]. Viscoelastic networks using springs with variable rest lengths to control muscle contraction have been used to simulate snake locomotion[83] and fish[119]. Damping provides dissipative forces that simulate energy loss, preventing perpetual oscillations in these physical systems. These models also included frictional and hydrostatic forces to simulate external environmental forces necessary for locomotion.

The disadvantage of constrained particles are directly due to their representation. For finer resolution or detail, a larger number of particles and springs may be required. Furthermore, global deformation constraints, such as volume preservation cannot be directly modelled as particles are only influenced by their nearest neighbours. The empty space between adjacent

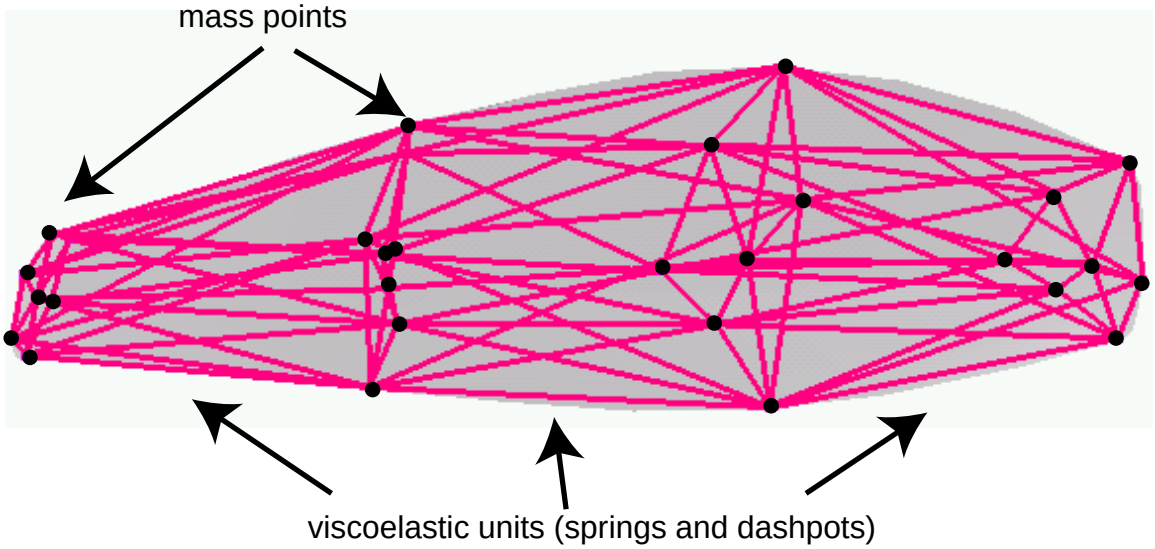


Figure 2.17: A network of mass points interconnected with viscoelastic units can be used to model deformable objects. Many connections are necessary for structural stability.

mass points can cause the structure to collapse on itself. To counteract these effects, cross-springs or other supplementary forces[14] can be applied to the mass points to reduce shearing or to preserve volume locally.

Using numerical explicit integration methods, very stiff objects would require extremely small timesteps to allow forces initiated in one region of the object to propagate to the other regions without causing numerical instability. Recently, implicit integration techniques have been proposed that allow larger timesteps to be utilized at the expense of accuracy because they effectively introduce artificial damping to the system to improve stability[28].

Lagrangian dynamics

If you are familiar with Lagrangian dynamics, you can skip over this discussion and derivation of the equations of motion.

In viscoelastic networks, all forces are directly applied to the mass points in the spatial domain. Constraint forces must be explicitly applied to the system. For many parametrically-defined objects, relatively fewer parameters are needed to completely describe an object's shape configuration. the method of formulating Lagrangian equations of motion allows differential equations to be described for each of the time-varying parameters, without considering the implicit constraints. The major advantage of this formulation is that there is increased freedom to work with other coordinate systems outside of Cartesian space.

The Lagrangian equations of motion follow from D'Alembert's Principle that allowed dynamics to be viewed as equilibrium problems with the introduction of an *inertial force*[72]. A key quantity is the *Lagrangian function*, which is the excess of kinetic energy (T) over potential energy (V):

$$L = T - V. \tag{2.29}$$

Hamilton's principle states that the motion of an arbitrary system occurs when

$$\delta \int_{t_0}^{t_1} L dt = 0, \tag{2.30}$$

where δ denotes a variation of the definite integral of the motion from time t_0 to t_1. For equation 2.30 to hold true, the following differential equations must be satisfied:

$$\frac{d}{dt}\frac{\delta L}{\delta \dot{q}_i} - \frac{\delta L}{\delta q_i} = 0, i = 0, \cdots, n-1. \tag{2.31}$$

The q_i are the n generalized coordinates that specify the system configuration. These differential equations create the *Lagrangian equations of motion*. Equation 2.31 can be written out explicitly in terms of the energies as:

$$\frac{d}{dt}\frac{\delta T}{\delta \dot{q}_i} - \frac{\delta T}{\delta q_i} - \frac{d}{dt}\frac{\delta V}{\delta \dot{q}_i} + \frac{\delta V}{\delta q_i} = 0. \tag{2.32}$$

In many cases the potential energy is a function of only the q_i, allowing us to remove the $\frac{\delta V}{\delta \dot{q}_i}$ term. In this case, the conservative forces due to the potential energy is simply the gradient of V with respect to the $\mathbf{q}$. Non-conservative external forces can be added to these equations as a result of D'Alembert's principle:

$$\frac{d}{dt}\frac{\delta T}{\delta \dot{q}_i} - \frac{\delta T}{\delta q_i} = Q_i - \frac{\delta V}{\delta q_i}, i = 0, \cdots, n-1. \tag{2.33}$$

In particular, the methods that use Lagrangian equations of motion model various potential energy functions to generate shape restorative forces to a rest shape configuration when deformation occurs. By choosing a good set of generalized coordinates that implicitly maintains constraints, we greatly simplify the equations of motion.

Applications of Lagrangian dynamics to computer animation

As deformable objects often need to interact with other objects in the scene, the values that the generalized coordinates take on may have to be constrained. We need to evaluate the constraint forces needed to maintain the kinematic dependencies among the generalized coordinates. Recall the use of a generalized force Q_i in equation 2.33. Rather than include the constraint forces in Q_i, we can partition Q_i into its unconstrained forces, which we will keep in Q_i and the forces of constraint which we can store in Q_i'[133]. The addition of m constraints restricts the n free parameters q_i of the system. One strategy is to reformulate the system to have only $n-m$ generalized coordinates, but this can be tedious or difficult to do algebraically. A widely used strategy is the method of *Lagrange multipliers* which introduces m *new* variables for each constraint. The physical system is transformed into a a set of $n+m$ free variables with $n+m$ equations:

$$\bar{L} = L - (\lambda_1 f_1 + \cdots \lambda_m f_m). \tag{2.34}$$

The modified Lagrangian equations of motion are:

$$\frac{d}{dt}\frac{\delta T}{\delta \dot{q}_i} - \frac{\delta T}{\delta q_i} = Q_i + \sum_{j=1}^{m} \lambda_j \frac{\delta C_j}{\delta q_i}. \tag{2.35}$$

If the constraints are all linear with respect to the q_i, the Lagrange multipliers λ_j can be found as the solution to a linear system involving the Jacobian matrix of the constraints $J_C = \frac{\delta \mathbf{C}}{\delta \mathbf{q}}$. The multipliers are then used to evaluate the constraint forces:

$$Q_i' = \sum_{j=1}^{m} \lambda_j \frac{\delta C_j}{\delta q_i}. \tag{2.36}$$

This method has been used for modelling attachment and point-to-plane constraints[133]. For nonlinear constraints, Platt and Barr introduced the method of augmented Lagrangian constraints[99] which provide estimates for the Lagrangian multipliers by integrating a differential equation, $\dot{\lambda}_i = C_i(\mathbf{q})$, to iteratively update λ_i in addition to the Lagrangian equations of motion. They use this method to maintain volume preservation and simulate plastic deformations.

Applications to parameterized geometry

Due to the generality of Lagrangian dynamics, the method has been applied to superquadrics [113], NURBs [115] and FFD lattices[30, 140]. In addition, spring-mass systems can be cast as a Lagrangian mechanics problem by simply treating the particle coordinates as the generalized coordinates q_i of the system. Terzopoulos et al. introduced the idea of modelling elastic deformations with potential energies. Various metrics for solids, surfaces and curves could be defined that are invariant to rigid body motion, but which change value for deformations[114]. With these tensors, they were able to create deformation energies and subsequent restorative forces.

Deformations which are linear with respect to generalized coordinates can be used to simplify greatly the related matrices needed to solve these equations of motion. Witkin and Welch used deformations that depended linearly on the q_i[135]. Their formulation allows precomputation of terms in the equations of motion to allow interactive animation. A useful class of linear deformations are tensor product shapes whose points can be represented as a linear combination of control points with local basis functions. By assigning masses to a distributed set of particles with these shapes, the Lagrangian equations of motion can be created.

2.3.8 Finite element approaches for physical modelling of muscles

Engineers have approximated solid materials by representing them as a composite of simpler, finite elements. The idea is that objects with complex shapes or nonuniform physical conditions can be simulated by a piecewise representation of smaller elements. Each element has a finite domain that has a value of zero outside of the domain. The dynamics for the whole solid material is then the solution of many smaller problems that are easier to solve. Relating finite element methods to Lagrangian dynamics, each finite element can be formulated with its own Lagrangian equations of motion for its smaller set of degrees of freedom.

A common finite element technique is to use 3-D brick elements made up of a different number of nodes (Figure 2.18). Zhu et al.[139] use 8-node 3-D brick elements (Figure 2.18) to represent muscle as a collection of voxel elements. They use low-resolution voxel meshes to capture gross shape changes, but the visual representations appear coarse as a result. To

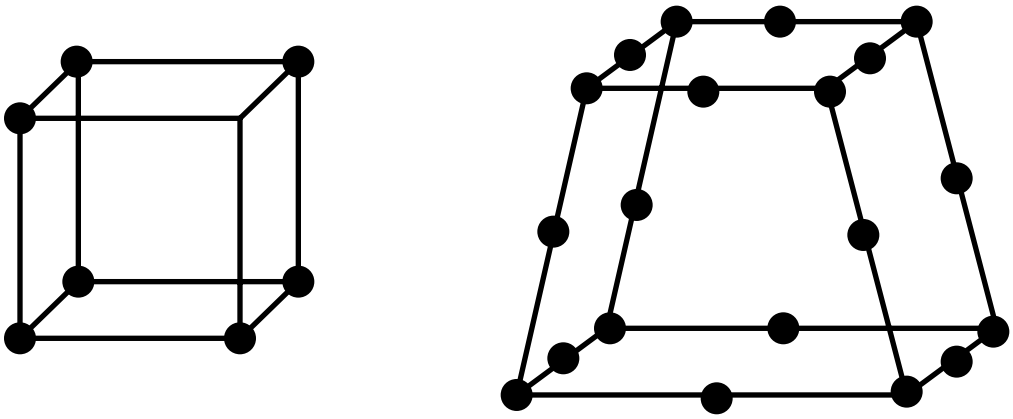


Figure 2.18: 3-D brick finite elements used to model muscle tissue. Left: the 8-node element used in [139]. Right: the 20-node element used in [19].

get smoother shapes, the number of voxels must be increased dramatically and computational requirements subsequently become prohibitive. In addition, the muscle force model they use is only valid under very idealized conditions such as parallel fibre arrangements, maximum activation and uniarticulate joint action. Chen and Zeltzer[19] used 20-node 3-D brick elements (Figure 2.18) to model musculotendon material. In contrast to Zhu et al.[139], they use the more accurate Zajac's dimensionless version of the three-element Hill model[138] to simulate contractile forces. They used this approach to simulate gastrocnemius muscle under relaxed and active conditions. Reaction constraints[99] were used for attachment to bone. Their work, while effective, only demonstrated single muscles working in isolation. This may be due to large computational requirements needed to simulate the many brick elements that would be needed for a multiple muscle situation.

Overall, the brick-element representation may be suitable for fusiform-shaped muscle where muscle fibres run in parallel from end to end and tendon material is located only at the endpoints. However, although the boundary shape may be adequately deformed, little insight is gained on the internal mechanisms of muscle tissue. There are many other skeletal muscles that have more complex fibre architecture and tendon regions that cannot be adequately represented with brick elements. The ability to capture internal fibre arrangements and contraction mechanisms in addition to boundary shape deformation is important for functional study of muscle tissue. We will demonstrate this capability with the B-spline solid muscle developed in this thesis. The use of B-spline basis functions can be considered a finite element method as these basis functions have a finite domain. In contrast to brick elements, B-spline finite elements implicitly maintain continuity conditions between elements by overlapping the domains of adjacent elements. By assuming implicit smoothness conditions, this choice of finite elements can allow a smaller number of elements to be used, thus decreasing simulation times.

2.3.9 Layered approaches to modelling

Given the techniques of deformations and Lagrangian dynamics, they can be employed together to model deformable objects. Motivated by the existence of skeletons in vertebrate animals, animation systems have used articulated bodies to represent jointed animals, including humans.

This representation has been successfully used to configure body positions as well as a compact representation for the figures by using the joint angles of each link as generalized coordinates. This has allowed motion to be specified as a time-varying vector of joint angles. Kinematic techniques that specify these joint trajectories include keyframe interpolation, motion-captured data and spacetime optimization[134]. Equations of motion have also been formulated for articulated figures for physically-based animation[4]. The superficial body geometry of animals can be modelled directly or digitized from physical models, but it is tedious to specify the location of each point on the skin manually for each frame of animation. Consequently, animation techniques employ strategies that use higher level representations which allow fewer parameters to be adjusted.

Skin over skeleton layer

To deform skin with skeletal articulation, one can embed an articulated figure skeleton within the geometry and associate each part of the skin geometry with one or more joint frames of reference[73]. When a point has more than one frame associated with it, the frame's influence on the joint is weighted so that the sum of weights is one. This method is used in many interactive games in the late 1990's to create whole mesh deformations. The weights must be manually tweaked to get reasonable motions. Others have tried to create descriptive parameterizations of skin movement during joint motion to describe muscle-based deformations while the skin mesh is locally anchored to underlying bones. The work in [112] creates a linear space of skin vertex transformations based on several key poses of an arm. The arm deforms realistically as transitions are made from pose to pose. These poses must be defined manually and increase exponentially as the number of parameters increase. For limbs having many superficial muscle features, this can lead to prohibitively large data requirements. Since these methods feature skin deformation depending only on the joint kinematics, the skin will lack any dynamic effects such as vibrations due to the inertial properties of muscle or other soft tissues that make up the limb.

Skin over deformable layer

To address this shortcoming, several methods added a deformable layer on top of the articulated skeleton. The skin was either directly generated from the deformable layer or associated with the deformable layer. Change in the deformation layer would propagate to the skin either physically or via a direct feature-based deformation method.

Gourret et al. simulated human skin deformation in the hands during grasping tasks by creating a finite element layer of parametric volumes[47]. They solved a state equilibrium problem to recalculate the final deformation, but dynamical effects were ignored. Dynamic effects become significant for areas of skin covering large muscular regions because the mass is significant enough for the inertial oscillations to be visible. Lee et al. use muscle vectors between the skull and the skin to create dynamicaly realistic facial expressions[75]. The layered structure of skin is a triple layer model of springs with different properties.

For whole body deformations, Chadwick et al. introduced layered construction of characters by using a layer of FFD's over the skeleton and animating the associated lattices by parameterized them deformation as functions of joint angles[16]. They were able to obtain

secondary stretch and squash motions as the body changed from pose to pose. Dynamic effects were obtained by treating the FFD lattice nodes as mass points interacting with each other through viscously damped Hookean springs. Gascuel used a deformation layer consisting of radial springs attached to a rigid skeleton to deform a spline-based skin surface[38].

To model smooth skin geometry, the blending properties and smooth profiles of implicit objects have been used to generate skin. Wilhelms used ellipsoidal primitives as the deformation layer for muscle and tendon. The skin was generated by finding the boundary of the shape by extracting the isosurface of the combined volume of ellipsoidal and skeleton geometry[129]. Shen and Thalmann interpolate NURBs surfaces through the isosurface of an internal layer of implicit objects that form the body shape[108]. Implicit objects can be used for physically-based modelling. In addition to shape representation, Gascuel has used the implicit functions as potential fields to compute forces that can be used for collision modelling between implicit shapes[37].

Anatomically-based deformable layers

Traditionally, sculptors and animators have studied anatomy in order to correctly capture the subtle contours of muscle and fat on the skin in both static and dynamic motions[42]. Several methods have been developed to model the deformable layers in an anatomically-based manner. Wilhelms and van Gelder used muscle-approximating polyhedral elements attached to skeletal geometry[130]. These polyhedral elements could change shape and used to deform the vertices of a skin mesh. Scheepers et al. used ellipsoids and bicubic patches as muscle models that can be layered over the skeleton[105]. They generated an isosurface from the musculoskeletal assembly to interpolate bicubic patches over the muscle and bone. In both of these examples, dynamics were absent. The force-generating properties of muscle were not modelled, making these techniques only suitable for geometric representation and deformations. As many of these models deform the skin only as a result of changing joint angles, the effects of isometric muscle contraction cannot be observed. The exception is in Scheepers et al. [105], where they provide a separate tension parameter to provide independent control of isometric muscle bulging.

There have been some attempts of physically-based layered models. Lee and Terzopoulos [75] use viscoelastic layers to represent different layers of skin over attached to muscles that contract to create facial expressions. Turner and Gobbetti use a three-part layered elastic model, consisting of a stick figure skeleton, covered by a deformable layer of spheres for simple muscle approximation and a Hookean-spring layer for fat[120]. Skin is the outermost layer and is modelled as an elastic surface. The area of physically-based layered models for creating humans and other animals is where this thesis continues.

2.4 Thesis context

We briefly review how the thesis relates to the background material and previous work presented in this chapter.

Anatomic context: By presenting a continuous solid model for muscle, we allow virtual reconstructions of fibre architecture to be captured and examined from novel viewpoints.

The procedure for fitting a solid model from medical images and digitized 3-D points allows a static reconstruction to be transformed into a deformable, simulated muscle model, aiding in functional studies. By utilizing a 3-D parameterized model, we can apply Lagrangian equations of motion for the muscle model as well as obtain measurements automatically that were previously tedious or difficult to obtain.

Biomechanical context: By providing better shape reconstruction and physical modelling capabilities, these geometric and physical muscle models can be used in musculoskeletal systems for biomechanical analysis. The capability of combining the standard Hill 3-element model with a solid model that has large attachment areas and internal fibre arrangements permits detailed study of force generation and the resultant motion on the underlying skeleton. Conversely, evaluating the effects of detailed models can give biomechanists more confidence on the suitability of using simplifications in other muscle models.

Computer graphics context: Finally, we extend layered construction techniques for character creation by providing an anatomically-based deformation layer that can model a more diverse set of muscle shapes. These muscles are capable of exhibiting inertial oscillations and volume preservation during contraction. Since their dynamics can be self-generated, isometric contractions are possible because shape changes are not exclusively generated by skeletal joint motion. By attempting to create a muscle model that can be used in anatomy, biomechanics and computer graphics, we have created a framework suitable for animal reconstruction on a visual and physical level.

Chapter 3

B-spline Solids

> Art is the expression of the invisible by means of the visible.
>
> **Fromentin**

For everyday objects in our physical world, we use mathematics to develop geometric models of their shapes, and computer graphics to visualize these models. Mathematical boundary representations are primarily used to represent these objects. This is usually an adequate representation since there is often no need to generate the internal contents of an object. However, when modelling muscle architecture, it is important to visualize internal features such as fibre arrangements. We introduce the use of a triple tensor product B-spline solid as a mathematical model for musculotendon. The characteristics of B-spline functions make them ideal for representing the smooth, geometric component of muscles. The three dimensional parameterization of the solid allows us to reference any point within the volume of the muscle.

In computer graphics, tensor-product solids have been used as free-form deformation lattices (see Section 2.3.5 for more details). The lattice defines a three dimensional space which becomes warped as the control points of the lattice are moved. Typically, geometry such as spline surfaces or polygons are embedded in the space so that the vertices or control points undergo a transformation that can roughly follow the deformation of the warped space. Surprisingly, there has been little exploration of the actual rendering of these lattice solids. In [58], tensor-product Bezier and B-spline solids and their mathematical properties are described in detail. We will briefly overview some of the important mathematical properties of B-spline solids which are important in the development of our musculotendon model.

Rappoport et al. developed an algorithm for preserving the volume of a Bezier solid used for free form deformations[103]. We extend this original volume formulation to B-spline solids. Although Rappoport et al. describe how the volume can be computed using only boundary control points, we will provide a constructive proof that clarifies the conditions over which this computational shortcut can be taken. We introduce how to set up a constrained optimization problem to conserve B-spline solid volumes in conjunction with other constraints. This technique does not use physically-based methods, but Chapter 5 will introduce volume-preservation in a physically-based context.

Given the B-spline solid formulation, we can display the solids interactively and display various isosurfaces throughout the solid. We define two sets of generalized coordinates that

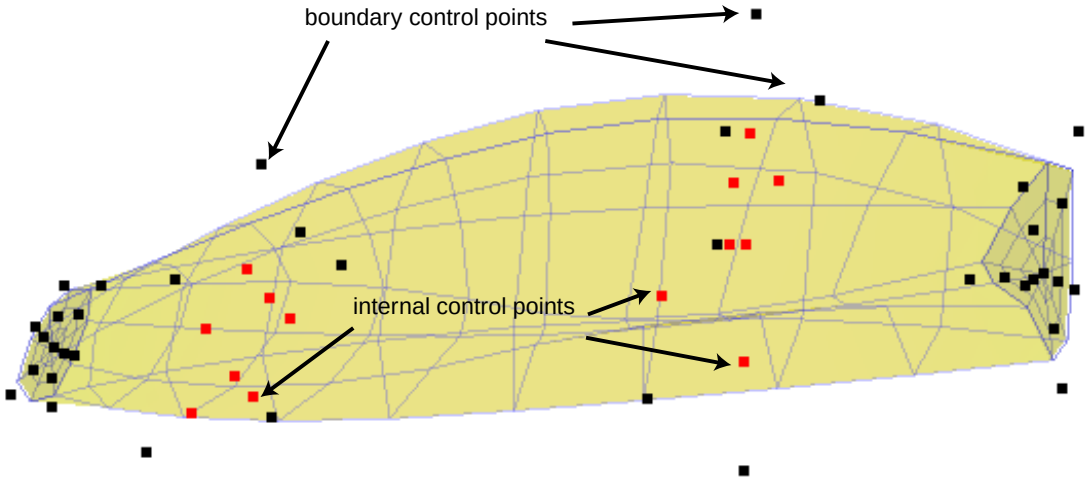


Figure 3.1: A B-spline solid with boundary and internal control points.

can be used to manipulate the solid's shape: control points and spatial points. The former is the traditional method of modulating the shape of tensor-product geometry. However, the latter allows direct manipulation of shape. For spatial points, we will describe how to assign material coordinates for reference to the solid's parameterization, including finding material coordinates for arbitrary spatial points. In the case where a point may not belong to a solid, we explain how to find the closest solid point to the external point. These mathematical properties and operations will be used for developing methodology in subsequent chapters.

3.1 B-spline solid mathematical formulation

A B-spline solid is represented by the following triple tensor-product parametric function:

$$\mathbf{x}(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} B_i^u(u)B_j^v(v)B_k^w(w)\mathbf{c}_{ijk}, \tag{3.1}$$

where the $\mathbf{c}_{ijk}$ are points in space indexed by i, j and k. These set of points in

$$\mathbf{c} = \{\mathbf{c}_{ijk} \in \mathcal{R}^3 | i = 0, \cdots, l, j = 0, \cdots, m, k = 0, \cdots, n\} \tag{3.2}$$

form a control point lattice which will influence the shape of the B-spline solid (Figure 3.1). The vector $\mathbf{x}$ represents the evaluated point computed from the triple tensor-product of these B-spline basis functions (in this case, the scalar polynomials: $B_i^u(u), B_j^v(v), B_k^w(w)$) with the control points in $\mathbf{c}$. The ordered triple (u, v, w) represents the material coordinates that map to corresponding spatial points in the volume of the solid. We can represent this triple as the material coordinate $\mathbf{u}$, as described in Section 2.3.2.

3.2 Properties of B-spline basis functions

The B-spline basis functions are piecewise polynomial functions that have a finite domain of interest, over which they take on non-negative values. Although B-spline solids represent a triple tensor-product of these functions, we will focus on a single, univariate basis function for clarity. We define a sequence of $n+1$, K^{th} order basis functions of the parameter u as:

$$B_{0,K}(u), \ldots, B_{n,K}(u). \tag{3.3}$$

The domain of this family of basis functions spans the *knot vector* U which is defined as

$$U = (u_0, u_1, \ldots, u_{n+K})^T. \tag{3.4}$$

In particular, basis function $B_{i,K}(u)$ is a piecewise polynomial defined in the *support interval* $[u_i, u_{i+K}]$. The B-spline basis functions obey the partition of unity rule:

$$\sum_{i=0}^{n} B_{i,K}(u) = 1, u \in [u_{K-1}, u_{n+1}]. \tag{3.5}$$

There are $n+1$ control points ($\mathbf{c}_i, i = 0, \ldots, n$) that correspond to the $n+1$ basis functions. The *knots* u_i of the knot vector U form a non-decreasing sequence. If the interval $[u_i, u_{i+1}]$ is of the same length for all i, we have a *uniform* knot vector (Figure 3.2A). Otherwise, it is termed a *non-uniform* knot vector (Figure 3.2C). Periodic knot vectors are used to create closed curves, surfaces, and solids. The knots of a periodic knot vector represent one period of the infinitely, repeating sequence (Figure 3.2B). The periodicity can be simulated with a nonperiodic knot vector by extending the knot vector with additional knots at the beginning and end of the sequence. This will define basis functions that will maintain continuity at the parameter boundaries of the period.

Multiple knots can be added to the terminal ends of a knot vector to create lower orders of continuity at these knot values. In particular, the order of continuity will decrease by i at a knot of multiplicity $i+1$. We can use this property to interpolate the first and last control points of a B-spline curve by repeating the first and last knots in a knot sequence K times (as done in Figure 3.2C). This control of continuity is an inherent part of the formulation of B-splines. In contrast, stitching together Bezier curves requires explicitly setting constraints between adjacent curves or surface patches to maintain continuity at the boundaries. Finally, for shape refinement, techniques can be applied to insert knots into the knot vector without changing the shape of the curve[128].

3.2.1 Evaluation of B-spline basis functions

We explicitly define the B-spline basis function. The valid domain of a parameter u, with knot vector U is $[u_{K-1}, u_{n+1}]$. The basis functions can be evaluated using the *Cox de Boor* recurrence relation[58]:

$$B_{i,K}(u) = \frac{u - u_i}{u_{i+K-1} - u_i} B_{i,K-1}(u) + \frac{u_{i+K} - u}{u_{i+K} - u_{i+1}} B_{i+1,K-1}(u). \tag{3.6}$$

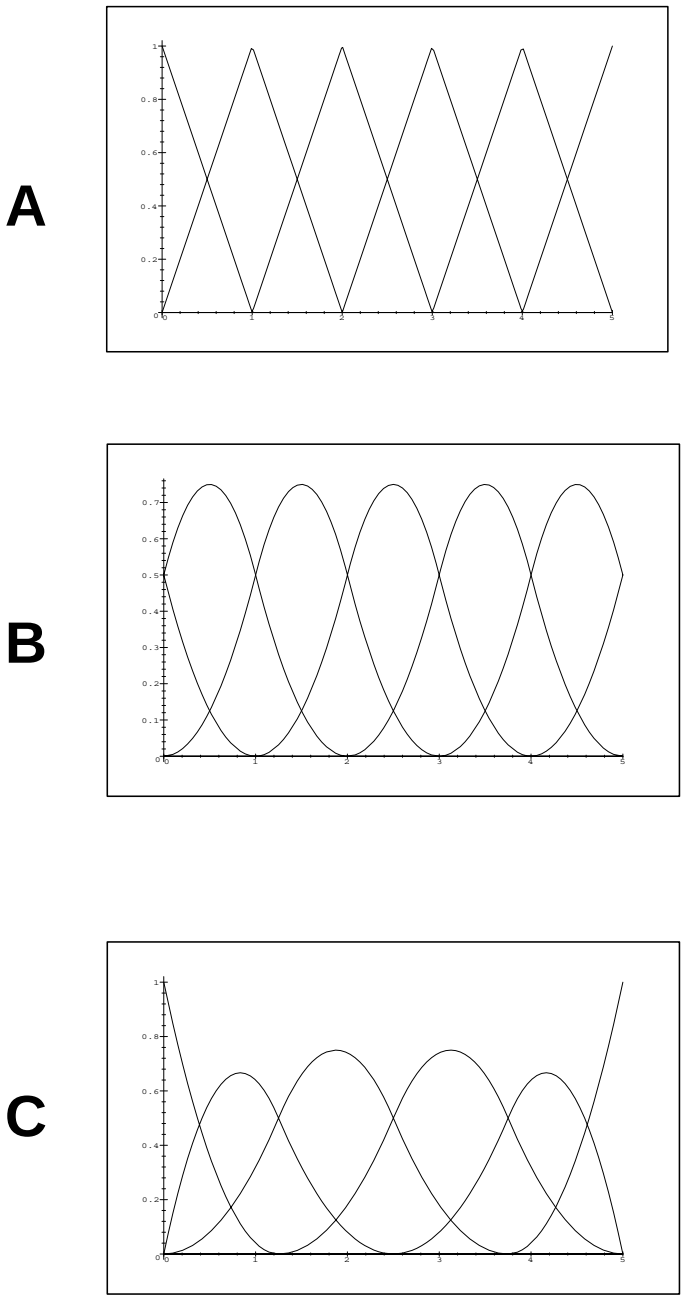


Figure 3.2: B-spline basis functions. A: Degree 1 uniform B-spline basis functions. B: Degree 2, periodic, uniform B-spline basis functions. C: Degree 3, non-uniform B-spline basis functions.

We assign:

$$B_{i,1}(u) = \begin{cases} 1 & u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The first subscript of $B_{i,K}$ represents the i^{th} basis function whose *support* consists of the knot interval $[u_i, u_{i+K}]$. The second subscript is the order K of the basis function. Each i^{th} basis function has a non-negative value in the domain $[u_i, u_{i+K}]$ and is zero everywhere else, making the basis function have local influence. Notice that the K^{th} order basis function is represented as a weighted sum of two branches of $(K-1)^{th}$ order basis functions. Since each branch has non-zero values over a smaller domain, comparing the parameter value u to be evaluated with each branch's domain, it is possible to cull out entire branches of computation simply by replacing a branch's value with zero whenever u is outside the branch's domain.

3.2.2 Evaluation of first derivatives

The first derivatives of a B-spline basis function of order K can be expressed in terms of B-spline basis functions of lower order[43]:

$$\frac{dB_{i,K}(u)}{du} = (K-1)\left(\frac{B_{i,K-1}(u)}{u_{i+K-1} - u_i} - \frac{B_{i+1,K-1}(u)}{u_{i+K} - u_{i+1}}\right). \tag{3.8}$$

By implementing the Cox de Boor algorithm as a recursive function, we can reuse the same code to compute the first derivatives to evaluate partial derivatives of the triple tensor-product B-spline solid for calculating gradient and normal vectors. Alternatively, for computing derivatives of B-spline curves, there are computationally efficient methods, such as the *de Boor Algorithm*[58] or Lee's B-spline computation routines[76]. These methods use divided difference tables that can evaluate a B-spline curve and its derivatives from just the control points without referencing explicitly the basis functions. However, extending these techniques to surfaces and volumes would be complicated by multiple divided difference tables to handle the extra dimensionality.

3.2.3 Solid and boundary domain relationships

As we are working with B-spline solids, the control points $\mathbf{c}$ form a lattice that will define the overall shape. Since the control points are weighted with basis functions, the knot vector determines the shape of the basis functions and the influence the control points have on the overall solid shape.

For a family of B-spline basis functions of order K, a non-periodic knot vector with K multiple knots at the boundaries will cause the first and last basis functions to evaluate to one on the boundaries of the parameter domain (Figure 3.2C). We can partition the control points $\mathbf{c}$ of a B-spline solid into boundary points and internal points (Figure 3.1). A boundary control point $\mathbf{c}_{ijk}$ is defined as any point such that its corresponding tensor product $B_i^u(u)B_j^v(v)B_k^w(w)$ contains any of the first or last B-spline basis functions derived from a non-periodic knot sequence.

For the non-periodic knot sequence, if the first or last B-spline basis functions of any parametric domain evaluate to one at the boundaries, the *partition of unity* property

$$\sum_{i=0}^{n} B_{i,K}(u) = 1, \forall u \in [u_{K-1}, u_{n+1}], \tag{3.9}$$

implies that internal control points have no influence on the boundary shape (their corresponding tensor product terms evaluate to zero on the boundary). Therefore, the boundary control points of a B-spline solid can be interpreted as the control points of a regular B-spline surface that forms the solid's boundary. This fact allows B-spline solids to be easily integrated into standard 3-D modelling packages that use B-spline surfaces for geometric representation. The boundary is determined exclusively by boundary control points. We will later use this property to simplify volume computations of B-spline solids.

3.2.4 Interpretation of lattice indices

By using different combinations of periodic and non-periodic knots for the three sequences of B-spline basis functions that define the solid, we can achieve diverse classes of shapes. Setting dependency constraints amongst the control points allows us to create spherical, toroidal and cylindrical shapes in addition to cubical and tubular shapes. Table 3.3 shows the various shapes we have implemented. For the periodicity assignments, we have arbitrarily assigned periodic or non-periodic properties to each of the knot sequences for u, v, and w respectively. These periodic properties can be rearranged amongst the three parameters in any combination to get the same shape, as long as the corresponding control point constraints are also rearranged.

3.2.5 B-spline solids as supersets for curves and surfaces

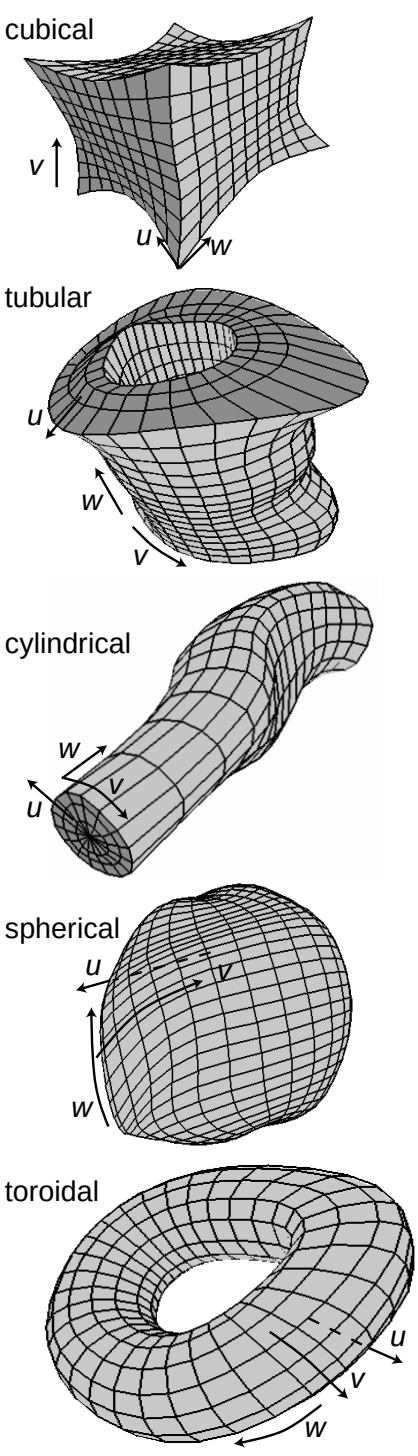
Curves and surfaces can be created from Equation 3.1 by holding one or two parameters constant to define iso-surfaces and iso-curves respectively (Figure 3.4). This can be used to generate streamlines within a solid or visualize a boundary surface. For muscle modelling, streamlines can be used to visualize individual muscle fibres and boundary surfaces can represent aponeurosis regions of thin tendon material attached to the muscle's surfaces.

3.3 Interactive Display of B-spline Solids

Repetitive calls of Equation 3.6 can be expensive when evaluating many points to tessellate the outer surface of a B-spline solid for display purposes. Fortunately, we can evaluate the tensor product portion of the B-spline solid at a predetermined set of material coordinates and store the values in a table to avoid recomputation[104].

We illustrate this for a one-dimensional example. Suppose we want to plot ten uniform sample points on a curve with the material coordinates of u sampled as follows: $s_0, s_1, s_2, \ldots, s_9$ on the domain $[u_{K-1}, u_{n+1}]$ so $s_i = u_{K-1} + i\frac{u_{n+1}-u_{K-1}}{9}$. The actual point evaluated on the curve will be:

$$\mathbf{p}_i = \sum_{j=0}^{n} B_j(s_i)\mathbf{c}_j. \tag{3.10}$$



	Periodicity assignments	**Control point constraints**
cubical	u,v,w non-periodic	none
tubular	u,w non-periodic v periodic	none
cylindrical	u,w non-periodic v periodic	for k = 0 ... n, $c_{0jk} = c_{0j'k}$
spherical	u,w non-periodic v periodic	for i = 0 ... l, $c_{ijn} = c_{ij'n}$ $c_{ij0} = c_{ij'0}$ $c_{0jk} = c_{0j'k'}$
toroidal	u non-periodic, v,w periodic	for k = 0 ... n, $c_{0jk} = c_{0j'k}$

Figure 3.3: B-spline solid shapes and their corresponding periodicities and control point constraints. If an index is primed, it refers to a different index value from the corresponding unindexed one. For example, in the cylindrical solid for a given value of k, $\mathbf{c}_{0jk} = \mathbf{c}_{0j'k}\ \forall j | j \neq j'$.

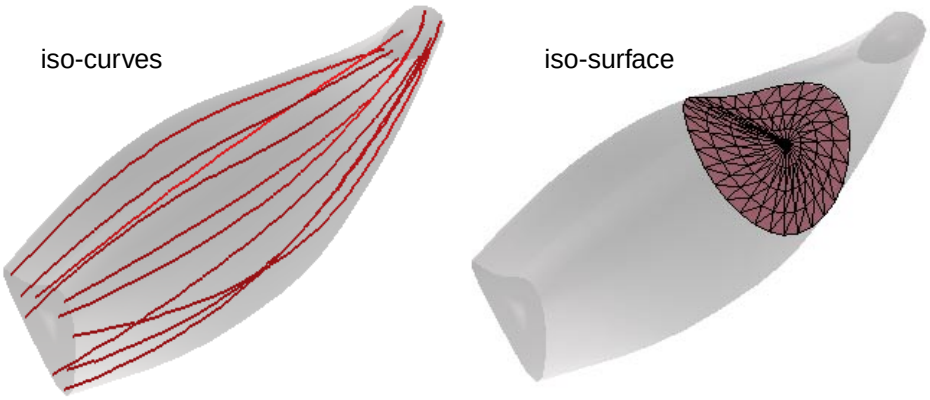


Figure 3.4: Iso-curves and iso-surfaces extracted from a B-spline solid.

If the sample points s_i are fixed, we can evaluate all the $B_j(s_i)$ in advance for all valid i and j[104]. Since there are at most K non-zero B-spline basis function values for any material coordinate, s_i, we can save space and computation time. To render the curve, these sampled curve points can be joined together to get a piecewise linear approximation of the curve.

For the purposes of visualization, we usually only want to display the boundary surfaces of a solid. This can be achieved by partitioning the boundary into a set of B-spline surfaces. Each of these surfaces corresponds to a particular iso-surface (see Section 3.2.5).

We presample a grid created from sampled material coordinates over the surface of the solid for the purposes of tesselation and display. This strategy is adequate as long as the sampling density is high enough to capture the curvature of the surface and other shape features. By changing the sampling interval of the material coordinates in each iso-surface, we can create different levels of detail in our tessellation. This is useful in scenes where we may have many muscles at various distances from the viewer. Figure 3.5 depicts the same solid tessellated with different sampling densities.

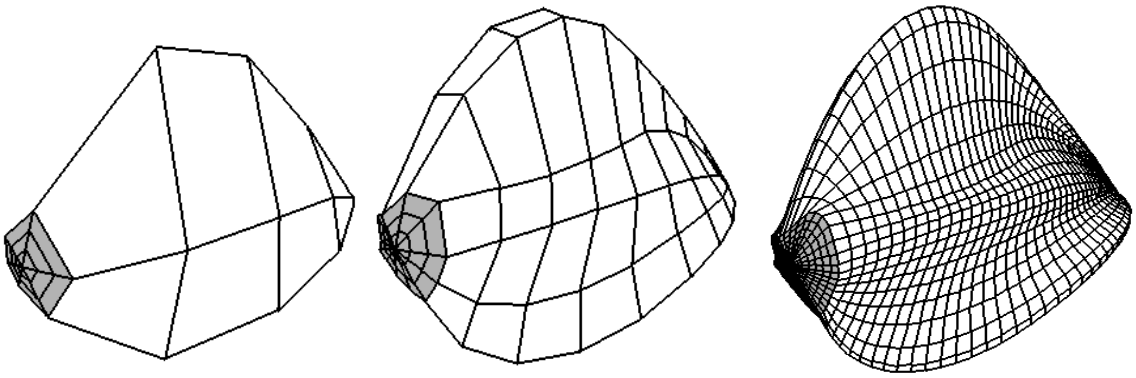


Figure 3.5: Different levels-of-detail for display of the same B-spline solid.

As the control points change position, we can update a new tessellation point position $\mathbf{p}$ for material coordinate, (u_p, v_p, w_p) with a fast vectorized dot product calculation, assuming the

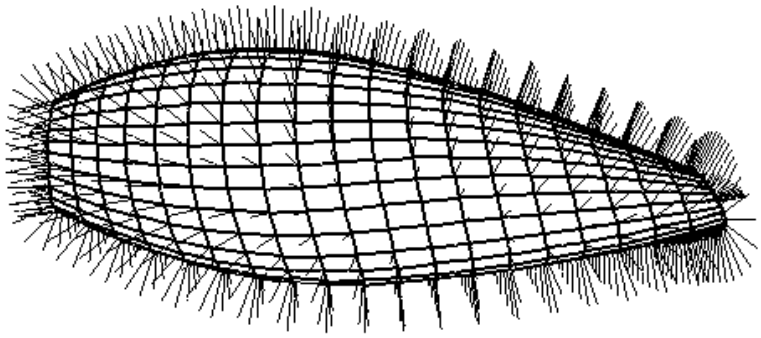


Figure 3.6: Normals can be generated at the tessellation points of the B-spline solid's surface for shading calculations.

knot sequence does not change:

$$\mathbf{p}(u_p, v_p, w_p) = \mathbf{B}_{(u_p,v_p,w_p)} \cdot \mathbf{c}. \tag{3.11}$$

$\mathbf{B}_{(u_p,v_p,w_p)}$ and $\mathbf{c}$ are vectors where the corresponding $[(i(m+1)+j)(n+1)+k]^{th}$ entry positions[1] are defined by $(B_i^u(u_p) \cdot B_j^v(v_p) \cdot B_k^w(w_p))$ and control point terms ($\mathbf{c}_{ijk}$). In the course of interactive editing of control points, only the entries of $\mathbf{c}$ will change and need to be recomputed. Since $\mathbf{B}$ is sparse and its entries are precomputed, incremental updates of the surface tessellation points using Equation 3.11 are quick.

3.3.1 Normal calculations for shading

For real-time shading calculations with graphics hardware, we require the normal vectors for each tessellation point on the surface (see Figure 3.6). The normals can be calculated analytically using Equation 3.1. For example, an iso-surface formed by holding $u = u_b$, would have normals:

$$\mathbf{n}(u_b, v, w) = \frac{\frac{\partial \mathbf{x}(u_b,v,w)}{\partial v} \times \frac{\partial \mathbf{x}(u_b,v,w)}{\partial w}}{\| \frac{\partial \mathbf{x}(u_b,v,w)}{\partial v} \times \frac{\partial \mathbf{x}(u_b,v,w)}{\partial w} \|}. \tag{3.12}$$

The partial derivatives need only be applied to the basis function portion of Equation 3.1 and they can be pre-computed for each tessellation point. Using table look-up, incremental updates of the normal vectors on a surface can be performed as the solid is interactively deformed. For proper lighting effects, it is important that the normals are outward facing. This is equivalent to maintaining a positive surface orientation. If the normals are pointing inwards, the parameterization of v or w must be reversed so that tangent vectors are generated in the opposite direction to change the orientation of the surface.

[1]The quantities $m+1$ and $n+1$ are the number of basis functions defined in the v and w knot sequences respectively.

3.4 Calculating and conserving volumes of B-spline solids

There is an efficient closed-form expression for the volume of a B-spline solid. The solution is a dot product of two vectors whose values correspond to control points and basis function values. We first derive an explicit formula for the volume of a B-spline solid, following closely the work by Rappoport et al.[103], who derived a similar formula for Bezier solids. We prove how we can accelerate volume computation by showing that the volume only depends on the boundary control points that define the outer surface bounding the solid.

For a parameterized solid defined by a tritensor B-spline with n_u, n_v, and n_w basis functions in each parameter (and therefore of these respective degrees), the volume is defined as:

$$\mathcal{V} = \int_{u_{K-1}}^{u_{n_u}} \int_{v_{K-1}}^{v_{n_v}} \int_{w_{K-1}}^{w_{n_w}} \det J_{\mathbf{x}} dw dv du, \tag{3.13}$$

where $\det J_{\mathbf{x}}$ refers to the determinant of the Jacobian matrix of the volume function,

$$\mathbf{x}(u,v,w) = \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} \sum_{k=0}^{n_w-1} B_i^u(u) B_j^v(v) B_k^w(w) \mathbf{c}_{ijk}. \tag{3.14}$$

with respect to the three parameters u, v, and w. The integrals are taken over the valid domain of u,v, and w. The indices of integration are for three non-periodic knot vectors. For periodic domains, the indices are adjusted to be the boundaries of one period. In the following discussion, we will collapse all summation and integration symbols to one symbol for clarity, such as:

$$\begin{aligned} \sum_{ijk} &= \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} \sum_{k=0}^{n_w-1} \\ \int &= \int\int\int. \end{aligned}$$

By looking at the subscripts and differentials, we can determine the scope of integration or summation from this simplified notation. If no indices of integration are shown, then integration is assumed to occur over the entire parameter domain. In addition, we use the following notational shorthand from [103]:

$$\begin{aligned} \underline{u}_{ijk} &= \frac{d}{du} B_i^u(u) B_j^v(v) B_k^w(w) \\ \underline{v}_{ijk} &= B_i^u(u) \frac{d}{dv} B_j^v(v) B_k^w(w) \\ \underline{w}_{ijk} &= B_i^u(u) B_j^v(v) \frac{d}{dw} B_k^w(w). \end{aligned}$$

Now we expand the Jacobian $\det J_{\mathbf{x}}$ to get:

$$\det J_{\mathbf{x}} = \begin{vmatrix} \sum_{ijk} c_{ijk}^x \underline{u}_{ijk} & \sum_{ijk} c_{ijk}^x \underline{v}_{ijk} & \sum_{ijk} c_{ijk}^x \underline{w}_{ijk} \\ \sum_{ijk} c_{ijk}^y \underline{u}_{ijk} & \sum_{ijk} c_{ijk}^y \underline{v}_{ijk} & \sum_{ijk} c_{ijk}^y \underline{w}_{ijk} \\ \sum_{ijk} c_{ijk}^z \underline{u}_{ijk} & \sum_{ijk} c_{ijk}^z \underline{v}_{ijk} & \sum_{ijk} c_{ijk}^z \underline{w}_{ijk} \end{vmatrix}, \tag{3.15}$$

where $\mathbf{c}_{ijk} = (c^x_{ijk}, c^y_{ijk}, c^z_{ijk})^T$. By expanding this determinant and collecting terms, we can divide out the control point terms from the basis function terms and rewrite the basis function components in terms of another determinant:

$$\det J_{\mathbf{x}} = \sum_{ijklmnopq} c^x_{ijk} c^y_{lmn} c^z_{opq} \begin{vmatrix} \underline{u}_{ijk} & \underline{u}_{lmn} & \underline{u}_{opq} \\ \underline{v}_{ijk} & \underline{v}_{lmn} & \underline{v}_{opq} \\ \underline{w}_{ijk} & \underline{w}_{lmn} & \underline{w}_{opq} \end{vmatrix}. \tag{3.16}$$

Notice that the summation in Equation 3.16 has undergone some reindexing to produce a collection of nine summations. This is not as daunting as it would seem since we can combine the nine indices into a common index $I = ijklmnopq$. Using this new index, we replace the determinant on the right side of equation 3.16 with the notation: $\det_I(u, v, w)$.

Now using Equation 3.13, we apply the triple integral only to the basis function part of each term:

$$\begin{aligned} \mathcal{V} &= \int \|J_{\mathbf{V}}\| dw dv du \\ &= \int \sum_{ijklmnopq} c^x_{ijk} c^y_{lmn} c^z_{opq} \det_I(u, v, w) dw dv du \\ &= \sum_{ijklmnopq} c^x_{ijk} c^y_{lmn} c^z_{opq} \int \det_I(u, v, w) dw dv du. \end{aligned} \tag{3.17}$$

If we place the elements $c^x_{ijk} c^y_{lmn} c^z_{opq}$ in a vector $\mathcal{C}$ indexed by $I = ijklmnopq$, and place the elements $\int \det_I(u, v, w) dw dv du$ in a vector $\mathcal{B}$ indexed by I, we can replace equation 3.17 with a simple dot product of two vectors indexed by I:

$$\mathcal{V} = \mathcal{C}^T \cdot \mathcal{B}. \tag{3.18}$$

Since our knot vectors are predefined and the integrals are over the entire parameteric domain, we can precompute the elements of $\mathcal{B}$ for quick retrieval. When we edit control points, we can selectively update the terms in $\mathcal{C}$ to quickly recompute the volume.

3.4.1 Computation of the elements of $\mathcal{B}$

Rappoport et al.[103] used gaussian quadrature to compute the integrals in $\mathcal{B}$. This is a sensible choice when the integrand is a product of Bezier basis functions since all the functions are defined on the same domain and gaussian quadrature can compute the exact integrals for these polynomial forms[100]. However, B-spline basis functions each have different local domains where anything outside the local support of the basis function is evaluated to zero. If the gaussian quadrature is not aware of the local properties of B-splines, the method could result in numerous useless function evaluations because many combinations of basis functions could result in a product of zero. We will attempt to make use of our knowledge of the local domains of basis functions to restrict the indices of integration to domains that contain non-zero integrand values.

In subsequent equations, we will remove the indices of integration for clarity. If we expand each element in $\mathcal{B}$, we produce:

$$\begin{aligned}
&\int \det_I(u,v,w)dwdvdu \\
&= \int \underline{u}_{ijk}\underline{v}_{lmn}\underline{w}_{opq}dwdvdu + \int \underline{u}_{lmn}\underline{v}_{opq}\underline{w}_{ijk}dwdvdu + \int \underline{u}_{opq}\underline{v}_{ijk}\underline{w}_{lmn}dwdvdu \\
&- \int \underline{u}_{opq}\underline{v}_{lmn}\underline{w}_{ijk}dwdvdu - \int \underline{u}_{lmn}\underline{v}_{ijk}\underline{w}_{opq}dwdvdu - \int \underline{u}_{ijk}\underline{v}_{opq}\underline{w}_{lmn}dwdvdu.
\end{aligned} \tag{3.19}$$

Focusing on the first term in (3.19), we can rearrange the integrand so that the triple integral is transformed into the product of three single integration operations:

$$\begin{aligned}
&\int \underline{u}_{ijk}\underline{v}_{lmn}\underline{w}_{opq}dwdvdu \\
&= \int \frac{d}{du}B_i^u(u)B_j^v(v)B_k^w(w)B_l^u(u)\frac{d}{dv}B_m^v(v)B_n^w(w)B_o^u(u)B_p^v(v)\frac{d}{dw}B_q^w(w)dwdvdu \\
&= \int \frac{d}{du}B_i^u(u)B_l^u(u)B_o^u(u)du \cdot \int \frac{d}{dv}B_m^v(v)B_j^v(v)B_p^v(v)dv \cdot \int \frac{d}{dw}B_q^w(w)B_k^w(w)B_n^w(w)dw.
\end{aligned} \tag{3.20}$$

Notice that each single integral contains an integrand which is a function of one variable only. This allows us to separate the triple integral into three decomposable single integrals. We can precompute the values of each of these single integrals separately for all possible combinations. This technique is also applicable to the other terms of (3.19). Since the integrands have a polynomial structure that is easy to integrate analytically, we computed the integrals in symbolic form using **Maple V**[18], a symbolic math package. Symbolic expressions allow us to use subexpression collection to reduce the number of mathematical operations executed during the integration. Care must be taken to divide the range of integration into smaller intervals as each basis function is actually a piecewise polynomial. However, by knowing the domains of each basis function, we can decrease the number of subintervals to integrate because the valid range of integration will be restricted to the intersection of the non-zero domains of all basis functions involved in the integrand.

Now that we have detailed how to compute the components of $\mathcal{B}$ and $\mathcal{C}$, the size of these vectors is potentially very large at $n_u^3 n_v^3 n_w^3$. It turns out that we can apply the Divergence Theorem to Equation 3.13, implying that we can compute the volume by integrating over only the outer surface of the solid. We prove in Appendix A that components of $\mathcal{B}$ evaluate to zero whenever the integrand contains only basis functions whose domains lie entirely within the solid. This allows us to reduce the size of $\mathcal{B}$ and $\mathcal{C}$ by an order of magnitude. In addition, the vector $\mathcal{B}$ will be sparse due to the local support of the B-spline basis functions, allowing us to collect the non-zero entries in a compact data structure illustrated in Figure 3.7. For each unique non-zero value in $\mathcal{B}$, there is an associated list of the matching indices in $\mathcal{C}$.

It is important that the boundary surface of the B-spline solid have a positive orientation (see Section 3.3.1), with the normals pointing outwards. Otherwise, the volume will be a negative value (although the absolute value is still correct). Similarly, if the B-spline solid self-intersects, the portions that are intersecting will produce negative terms that will decrease the overall computed volume.

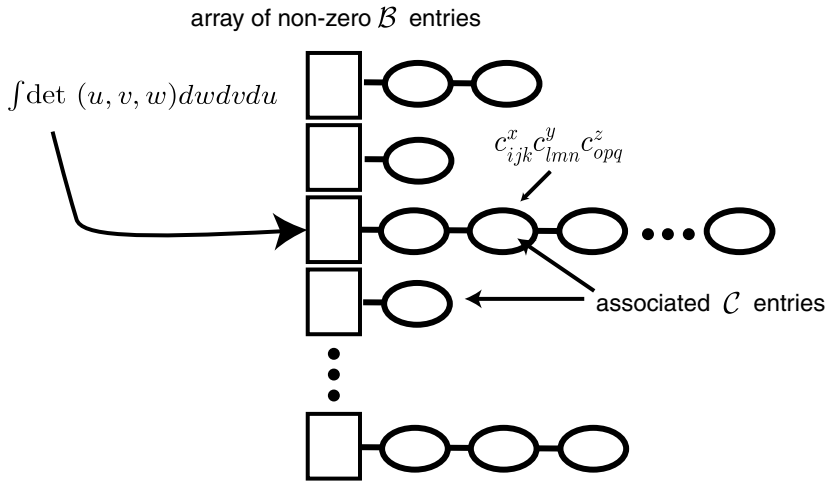


Figure 3.7: The vector $\mathcal{B}$ can be compactly represented by only storing the unique non-zero values and associating these entries to their corresponding indices in $\mathcal{C}$.

3.4.2 Gradient of Volume Formula

Once we compute the components of $\mathcal{B}$, it is fairly simple to compute the gradient vector $\nabla\mathcal{V}$ for Equation 3.18 with respect to the control point components in $\mathbf{c}$. We can reindex $\mathbf{c}$ to concatenate the components of the N control points into a single monolithic vector of size $3N$, where the i^{th} control point's three components are c_{3i}, c_{3i+1} and c_{3i+2} with $i = 0, \ldots, N-1$. In subsequent discussion, c_i refers to an individual component of $\mathbf{c}$, while $\mathbf{c}_{ijk}$ is a specific 3-D control point. The gradient vector for the volume equation, $\nabla\mathcal{V}$, will also have $3N$ components as each entry is $\frac{\partial\mathcal{V}}{\partial c_i}$, $i = 0 \ldots 3N-1$. These entries can be computed quickly by traversing the data structure in Figure 3.7 and accumulating separate sums for the entries of the gradient vector, $\nabla\mathcal{V}$. For each non-zero entry in $\mathcal{B}$, it is multipled with all non-zero partials of its linked entries in $\mathcal{C}$ and the resulting terms are added to sums stored in $\nabla\mathcal{V}$.

The volume formula (3.13) and its gradient vector $\nabla\mathcal{V}$ provide us with the necessary information to use the volume constraint in any constrained optimization technique. The volume gradient, $\nabla\mathcal{V}$, is especially useful as it is used to accelerate the convergence process of many iterative optimization methods by helping to provide a better estimate of the search direction for the optimal solution. To show their dependence on $\mathbf{c}$, we will use $\mathcal{V}(\mathbf{c})$ to indicate the scalar volume function of $\mathbf{c}$ and $\nabla\mathcal{V}(\mathbf{c})$ to show the gradient vector function of $\mathbf{c}$.

3.4.3 Deformation of volume-conserving B-spline solids

In the absence of dynamics, we can solve the volume preservation problem as a static, constrained optimization problem. The designer can manipulate control points, and the rest of the control points are subsequently adjusted to conserve volume. Volume preserving deformations are described as being *isochloric*[118]. We will discuss volume-preserving motions in a dynamics context in Chapter 5.

We use a constrained optimization technique to deform the B-spline solid while conserving volume and fixing selected control points. We can use *sequential quadratic program-*

ming (SQP)[39] to perform optimization of a non-linear objective function with non-linear constraints. A sequence of quadratic sub-problems are solved which produce an improving sequence of estimates of the solution vector (the control points, $\mathbf{c}$). For each sub-problem, the quadratic objective function is minimized while constraining the search to a given constraint set. If the constraints are nonlinear, they are locally approximated with linear constraints. Further details of the SQP algorithm can be found in [39].

We formulate the objective function as the sum of the least-squares deviation between the original control points and the current estimate,

$$f(\mathbf{c}) = \sum_{i=0}^{3n-3} \frac{w_i}{2}(c_i - o_i)^2, \tag{3.21}$$

where there are n control points with their coordinates collected in $\mathbf{c}$. The values o_i represent the components of the original control point configuration. By adding weights, w_i, we can bias a coordinate to change more or less than others.

Volume conservation is achieved using the scalar volume function, $\mathcal{V}(\mathbf{c})$(Equation 3.18) and its gradient vector $\nabla\mathcal{V}(\mathbf{c})$ to satisfy the *Kuhn Tucker* necessary conditions for constrained optimality[39]:

$$\nabla f(\mathbf{c}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\mathbf{c}^*) = 0 \tag{3.22}$$

$$\lambda_i^* g_i(\mathbf{c}^*) = 0, i = 1, \ldots, m \tag{3.23}$$

$$\lambda_i^* \geq 0, i = m_e + 1, \ldots, m. \tag{3.24}$$

The vector $\mathbf{c}^*$ is the optimal configuration of control points for a problem with m constraints (with m_e equality constraints and $m - m_e$ inequality constraints). Condition 3.22 describes how the gradients of the objective function, f, and of the weighted active constraints, g_i, sum to zero at a solution point $\mathbf{c}^*$. λ represents a vector of *Lagrangian Multipliers* for each constraint that roughly weights each constraint's influence on the objective function value. Conditions 3.23 and 3.24 claim that at a solution point $\mathbf{c}^*$, the constraints will either be inactive ($\lambda_i^* = 0$) or active ($g_i(\mathbf{x}^*) = 0, \lambda_i > 0$). The inactive condition only applies if the optimization contains inequality constraints, $g_i(\mathbf{c}) \leq 0, i = m_e + 1, \ldots, m$.

For our optimization problem, we only have equality constraints. In particular, the constraint for volume-preservation is

$$g(\mathbf{c}) = \mathcal{V}(\mathbf{c}) - \mathcal{V}_o = 0, \tag{3.25}$$

where $\mathcal{V}_o$ is the desired target volume. The optimization problem can be stated as:

$$\text{Minimize } f(\mathbf{c}) \text{ subject to } g(\mathbf{c}) = 0. \tag{3.26}$$

We also can constrain selected control points to be fixed. Rather than formulating explicit constraint equations for each fixed control points (three equations per control point), we simply set the corresponding entries in $\nabla\mathcal{V}$ and in $\nabla f(\mathbf{c})$ to zero. During the optimization algorithm, a zero entry in the gradient vectors implies that modifying the corresponding control point component will not affect the objective function or violate constraints. Consequently, the point will

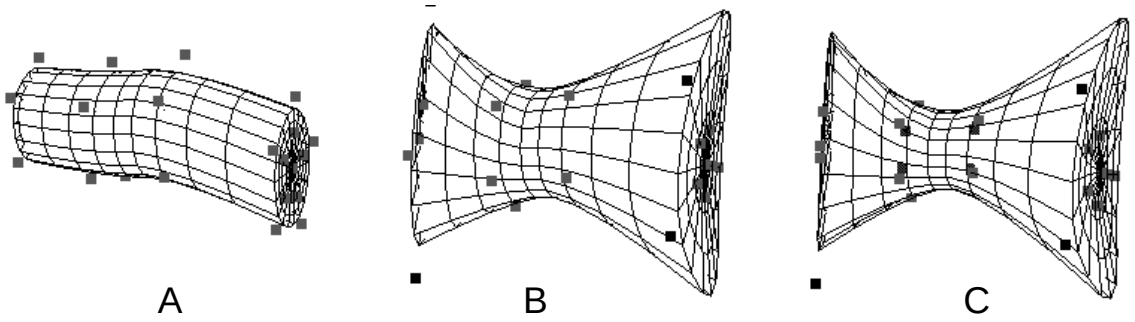


Figure 3.8: After modifying A to produce an initial configuration B, a volume-preserving static optimization process is applied to produce C with the same volume as A. Several control points are constrained not to move during this process to prevent the optimization process from reverting back to its original form, A.

not be adjusted. The parts of the solid under the local influence of the fixed control point will be subject to less degrees of freedom, but still be modifiable by neighbouring free control points. Figure 3.8 depicts an initially deformed B-spline solid and the new control point configuration after volume preservation with selected fixed points.

3.5 Generalized coordinates for B-spline solids

Recall from Section 2.3.2, that the generalized coordinate vector $\mathbf{q}$ defines a shape configuration for an object. We describe control points and spatial points as two choices of generalized coordinates that will influence how we formulate the generalized forces and potential energy functions in the corresponding Lagrangian equations of motion (see Section 2.3.7) for physically-based modelling.

3.5.1 Control points as generalized coordinates

Setting the control points, $\mathbf{c}$, to be the generalized coordinates, $\mathbf{q}$, allows us to use constraints that can be expressed naturally in terms of control points. We have shown that properties such as volume can be computed with a closed-form expression in terms of elements of $\mathbf{c}$. This allows volume-preservation to be directly added as an energy term to the Lagrangian equations of motion. Unfortunately, control points do not necessarily correspond to actual points in a solid and can reside completely outside the solid's spatial domain. Manipulating control points only indirectly modifies the solid's shape. Hsu et al.[59] have developed least-squares techniques for directly manipulating solid shapes, yet it may not be possible to satisfy multiple point constraints exactly.

3.5.2 Spatial points as generalized coordinates

As an alternative, we can define the same number of *spatial points*, $\mathbf{s}$, corresponding to different material coordinates in $\mathbf{u}$, with the distinction that these points are actually part of the solid.

The spatial points, $\mathbf{s}$, can also be treated as generalized coordinates, $\mathbf{q}$. Forces can be applied directly to spatial points without needing to transforming the forces to a different coordinate space. Manipulating these points will allow direct manipulation of the solid, albeit as a finite set of points. We are also guaranteed that these multiple point constraints will be satisfied.

3.5.3 Selecting material coordinates for the spatial points

In order to select spatial points, we need to determine how to choose their corresponding material coordinates in the u,v, and w parameter spaces. We will concentrate on how to sample material coordinates for the u parameter space, and generalize the results to v and w. Assume there are $n+1$ B-spline basis functions for u of order K: $B_{0,K}(u) \ldots B_{n,K}(u)$.

We would like to choose points that have predictable shape changes when we manipulate them. Since control points offer intuitive, local control on the surface shape, we would like to select spatial points that closely emulate this behaviour. By selecting the material coordinate that is maximally influenced by a control point, we can create a strong correlation between the modification of a spatial point with the control point. Forsey and Bartels[33] used the same idea and termed these material coordinates *points of maximal influence*.

In our univariate example, each of the $n+1$ basis functions have a corresponding control point. By choosing parameter values for u that maximize each basis function, we can produce a set of $n+1$ spatial points. That is, parameter $u_{max,i}$ maximizes $B_{i,K}(u)$ for $i = 0 \ldots n$. We have computed solutions for linear, quadratic and cubic B-spline basis functions. If the basis functions are symmetrical in shape, as with uniform knot vectors, the maxima occur trivially at the centre of each basis function. For non-uniform knot vectors, the solutions can be more difficult to find.

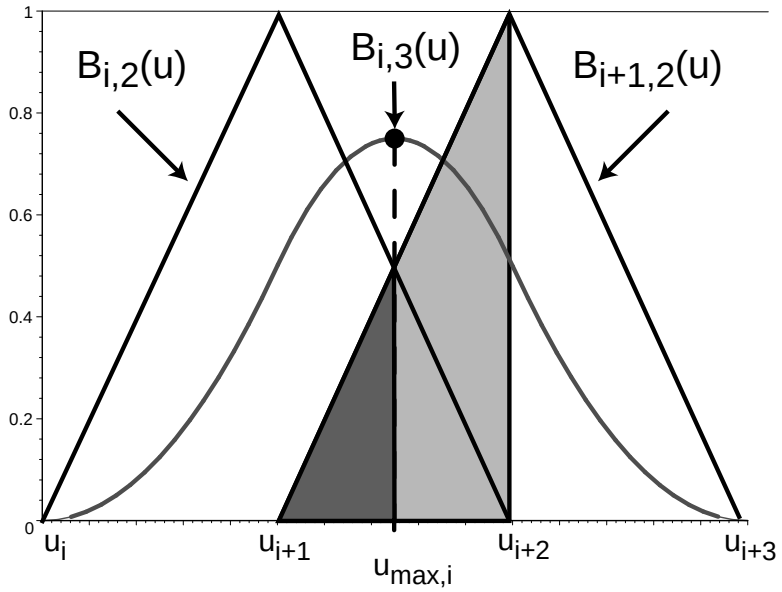


Figure 3.9: Computing the maximum of a quadratic B-spline function.

B-splines of 2nd order (degree 1) are hat-shaped (see Figure 3.2A) and the maxima will occur at the middle knot, $u_{max,i} = u_{i+1}$, in the support $[u_i, u_{i+2}]$. For quadratic B-splines

(degree 2, order 3), we can find the maximum of $B_{i,3}(u)$ by first setting the derivative formula to zero:

$$\frac{dB_{i,3}(u_{max,i})}{du} = 2\left(\frac{B_{i,2}(u_{max,i})}{u_{i+2}-u_i} - \frac{B_{i+1,2}(u_{max,i})}{u_{i+3}-u_{i+1}}\right) = 0 \tag{3.27}$$

$$\Rightarrow \quad \frac{B_{i,2}(u_{max,i})}{B_{i+1,2}(u_{max,i})} = \frac{u_{i+2}-u_i}{u_{i+3}-u_{i+1}}. \tag{3.28}$$

Using the partition of unity property, we can eliminate one of the basis functions since there are a maximum of two B-splines of order 2 that have non-zero values at the maximum:

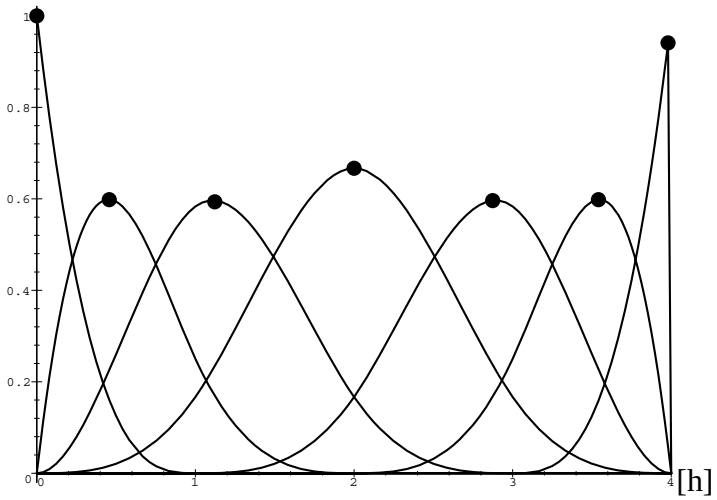
$$B_{i,2}(u_{max,i}) + B_{i+1,2}(u_{max,i}) = 1 \tag{3.29}$$

$$\Rightarrow \frac{u_{i+3}-u_{i+1}}{(u_{i+2}-u_i)+(u_{i+3}-u_{i+1})} = B_{i+1,2}(u_{max,i}). \tag{3.30}$$

Due to the arrangement of the B-spline functions, the maximum of $B_{i,3}(u)$ will occur in the interval $[u_{i+1}, u_{i+2}]$ (see Figure 3.9). Using Equation 3.30 and the geometry of similar triangles (see shaded triangles in Figure 3.9), the location of the maximum occurs at:

$$u_{max,i} = u_{i+1} + (u_{i+2}-u_{i+1})\frac{u_{i+3}-u_{i+1}}{(u_{i+2}-u_i)+(u_{i+3}-u_{i+1})}. \tag{3.31}$$

Finding the maximum of a general, non-uniform cubic B-spline basis function involves analysis of the lower-order basis functions in the cubic B-spline's support interval. The maximum may occur in any interval depending on the knot values chosen. Rather than attempt to derive a closed-form solution, the maxima were found using the **Maple V** symbolic-computation package[62] for all possible uniform cubic B-spline functions with knot intervals of width one and containing up to four multiple knots. These B-spline functions are displayed in Figure 3.10.



[h]

Figure 3.10: All possible cubic B-spline functions from the knot sequence of $[0,0,0,0,1,2,3,4,4,4,4]$. Maxima positions are indicated with solid circles.

Now that we have defined points of maximal influence for degree 1, 2, and 3 B-splines, the corresponding material coordinates for a 3-D parameterization will be n-tuples of all the individual knot maxima:

$$\mathbf{u}_{max,I} = (u_{max,i}, v_{max,j}, w_{max,k}), i = 0 \ldots n_u - 1, j = 0 \ldots n_v - 1, k = 0 \ldots n_w - 1, \quad (3.32)$$

where $I = (in_v + j)n_w + k$ and we have n_u, n_v, and n_w B-spline basis functions in each of the parameter domains of u, v and w respectively.

3.5.4 Material coordinates for arbitrary spatial points

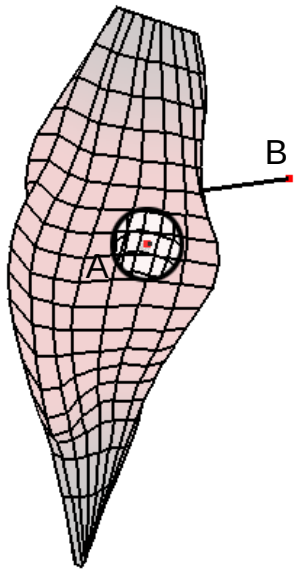


Figure 3.11: For a given point, we can find the corresponding material coordinates (point A is inside the solid) or find the closest point on a B-spline solid to a point B.

As well as using the B-spline solid equation to evaluate spatial points of the solid, the inverse problem of finding the material coordinates for a point is equally useful. Given a point in world coordinates, $\mathbf{p}$, we define a procedure called **closest**$(\mathbf{p})$, which returns the material coordinates that coincides with $\mathbf{p}$. If the point is external to the solid, **closest**$(\mathbf{p})$ finds the material coordinate that produces the closest point on the solid to $\mathbf{p}$. Figure 3.11 illustrates these operations.

The existence of **closest**$(\mathbf{p})$ allows the detection of collision and interpenetration of the solid with other objects. This is necessary to detect interaction of our muscle models with the underlying bones as well as with other musculotendons. Being able to relate external points to the local coordinate system of the solid is an important requirement of deformation techniques for computer animation (Section 2.3.4). B-spline solids can act as general free-form deformation lattices where a designer can pre-shape the lattice to closely bound the internal geometry to be deformed. Skin geometry can be anchored to underlying muscle to deform skin based on musculature movements.

Using Equation 3.13 for evaluating a B-spline solid, we can cast the inverse problem as a non-linear least-squares optimization procedure:

$$\text{Minimize } R(\mathbf{u}) = \|\mathbf{x}(\mathbf{u}) - \mathbf{p}\|_2^2, \tag{3.33}$$

where $R(\mathbf{u})$ is the objective function that minimizes the Euclidean norm of the deviation between $\mathbf{p}$ and the point of the B-spline solid, $\mathbf{x}(\mathbf{u})$. We wish to find the optimal material coordinate $\mathbf{u}^*$ that corresponds to the point $\mathbf{p}$ exactly ($R(\mathbf{u}^*) = 0$)) or that is closest in terms of Euclidean distance to the point.

Using a least-squares approach instead of a nonlinear root-finding problem permits both the point inclusion and proximity problems to be solved with the same technique. Numerically, global convergence can be obtained with a least-squares problem, but not necessarily for a root-finding problem since there may be no real roots[26]. There is a chance that the solution found with a least-squares solver may coincide with a local minimum, making it important to choose an initial estimate close to the true minimum. The initial estimate is determined by choosing among the stored spatial points that we use for the generalized coordinates of a B-spline solid. For each of these spatial points, we keep track of the material coordinates. We choose the closest spatial point to our test point. As the spatial points are distributed throughout the solid, there is a higher probability that the spatial point we choose will be close to the true minimum. In cases where we may be dynamically varying the test point and wish to quickly compute the inverse or closest coordinate, we can use the last known solution as the initial guess for the current problem. Using this method, we have successfully observed convergence after a few iterations.

Least squares problems take advantage of the special form of the objective function and iteratively evolve an initial guess towards the solution. The key process among the different methods is the determination of the step direction and step length for modifying the current iterate towards the solution. We use the *Levenberg-Marquardt* method implemented as a scaled trust region approach by Moré[26] from the MINPACK numerical optimization library[85]. Convergence is global and when $R(\mathbf{x}^*) = 0$ (point inclusion), convergence is quadratic[26].

We modify the Levenberg-Marquardt algorithm to restrict the search for material coordinates to within the valid parameteric domain. After the step length and direction is determined, we check that the new material coordinate iterate does not lie outside the solid, $\mathbf{u} \in \mathbf{u}_{min} \times \mathbf{u}_{max}$. Whenever a component of $\mathbf{u}$ lies outside the respective boundaries, we clamp the value to the boundary value. The iterates are allowed to move along the boundary, but never leave the solid's domain. Convergence is guaranteed by ensuring all derivatives are defined inside the domain. Although technically, the derivatives do not exist at the parametric boundaries due to lack of continuity, we use the one-sided derivatives from the direction of the internal domain to the boundary.

Using **closest**$(\mathbf{p})$, we nest the control points of one B-spline solid inside another, allowing the inner solid to be deformed with the free-form deformation lattice formed by the surrounding solid. This can be used to visualize other biological structures within muscles using arbitrary geometry. Figure 3.12 illustrates nested B-spline solids being deformed.

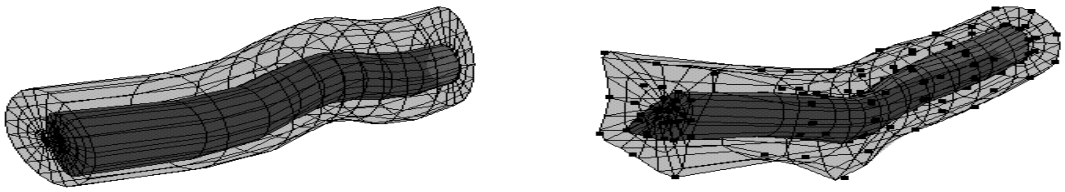


Figure 3.12: B-spline solids can be used to create volumes that tightly bind other geometry for free-form deformations.

3.6 Summary

In this chapter, we have reviewed the important properties of the B-spline basis functions that we wish to utilize for modelling muscles. Using these properties, we have developed a complete set of techniques for interactively displaying and shading B-spline solids. We presented a closed-form expression for the volume of the solid, including data structures for fast evaluation. The volume formula allows volume-preserving, deformations of B-spline solids using optimization techniques.

Control points and spatial points were presented as two different sets of generalized coordinates that can be used to allow different modes of shape manipulation. In addition, we present methods for assigning the material coordinates to spatial points, as well as finding suitable material points for arbitrary spatial points. In the latter case, this technique can be used to create point inclusion and closest point procedures. The next chapter uses these generalized coordinate schemes to develop data-fitting techniques for creating initial shapes of B-spline solids from different data sources.

Chapter 4

The Geometric Musculotendon Model

> The shape of the organ [muscle] is susceptible of an incalculable variety, while the original property of the muscle. The law and line of its contraction remains the same and is simple. Herein the muscular system may be said to bear a perfect resemblance to our works of art.
>
> **Palley's Theology**

In this chapter, we use the notation and techniques developed from Chapter 3 to create a methodology for generating useful initial shapes with B-spline solids from external data. In Section 3.5, we introduced control points and spatial points as generalized coordinates for B-spline solids. We will now develop the solution procedure for solving for a set of control points, given a set of spatial data points. We solve exact systems of n control points for n spatial points as well as under-determined and over-determined systems where we have m spatial points for n control points with $m \neq n$.

We will focus on how to generate suitable spatial points from different sources of data. By creating a common methodology for data-fitting, we show how to apply this technique to several specific examples of muscle fitting. Each case focuses on a different application of muscle modelling and illustrates the versatility and suitability of the B-spline solid model for approximating the geometry of musculotendons.

4.1 Conversion between control points and spatial points

It is important to be able to translate back and forth from control points to spatial points while working with B-spline solids as some operations are easier to work with in one configuration space opposed to the other. We will need to formulate a linear system that relates the control points to the spatial points.

To simplify the systems we are solving, we can flatten out the 3-D indexing of n control and spatial points into a single index. We have n_u, n_v, and n_w basis functions for each parameter, with $n = n_u n_v n_w$. For the control points in $\mathbf{c}$, we reindex the points, $\mathbf{c}_{ijk}, i = 0 \ldots n_u - 1, j = 0 \ldots n_v - 1, k = 0 \ldots n_w - 1$, to create a new index $I = (in_v + j)n_w + k$. Control point, $\mathbf{c}_i$, will have coordinates (c_i^x, c_i^y, c_i^z). We can perform the same re-indexing on the spatial points $\mathbf{s}$ to get $\mathbf{s}_i = (s_i^x, s_i^y, s_i^z)$. Using the same index I, we express the triple tensor-product terms

with:

$$\beta_I(\mathbf{u}) = B_i^u(u)B_j^v(v)B_k^w(w), \text{and } \mathbf{u} = (u, v, w). \tag{4.1}$$

The complete linear system can be described as

$$\begin{pmatrix} \beta_0(\mathbf{u}_{max,0}) & \dots & \beta_{n-1}(\mathbf{u}_{max,0}) \\ \beta_0(\mathbf{u}_{max,1}) & \dots & \beta_{n-1}(\mathbf{u}_{max,1}) \\ & \dots & \\ \beta_0(\mathbf{u}_{max,n-2}) & \dots & \beta_{n-1}(\mathbf{u}_{max,n-2}) \\ \beta_0(\mathbf{u}_{max,n-1}) & \dots & \beta_{n-1}(\mathbf{u}_{max,n-1}) \end{pmatrix} \begin{bmatrix} c_0^x & c_0^y & c_0^z \\ c_1^x & c_1^y & c_1^z \\ & \dots & \\ c_{n-2}^x & c_{n-2}^y & c_{n-2}^z \\ c_{n-1}^x & c_{n-1}^y & c_{n-1}^z \end{bmatrix} = \begin{bmatrix} s_0^x & s_0^y & s_0^z \\ s_1^x & s_1^y & s_1^z \\ & \dots & \\ s_{n-2}^x & s_{n-2}^y & s_{n-2}^z \\ s_{n-1}^x & s_{n-1}^y & s_{n-1}^z \end{bmatrix}, \tag{4.2}$$

or more succinctly as

$$\beta\mathbf{c} = \mathbf{s}, \tag{4.3}$$

where β is the $n \times n$ matrix described in Equation 4.2, and $\mathbf{u}_{max}$ are the material coordinates assigned to the spatial points as described in Section 3.5.3. The matrix is sparse, as each row will have an upper bound of $K \cdot L \cdot M$ non-zero entries, where K, L, and M are the orders of the B-spline basis function sequence for each parameter. In practice, we solve three separate linear systems for each of the column vectors in $\mathbf{c}$ with the corresponding vector for $\mathbf{s}$. The elements of the array β can be pre-computed once and β is the same matrix for all three linear systems. An LU factorization[100] of β can be performed, storing the factors for repeated solution of Equation 4.3 with different $\mathbf{s}$. The combination of sparsity and precomputation of factors permits quick solution computation. This allows interactive editing of the solid's shape and real-time animation performance. The solution of these linear systems is a key phase of the data-fitting process. It allows the interpolation of B-spline solids through external data points, which we represent as spatial points.

4.2 Least-squares data-fitting

In Section 4.1, we described how to fit a B-spline solid consisting of n control points to n spatial points. There may be situations where we may want to fit a B-spline solid of n control points to a data set consisting of m spatial points where $m \neq n$. In these cases, we need to find an appropriate solid shape that can minimize the deviation of all spatial points to the solid in a least-squares sense (Figure 4.1).

Formerly, our system relating control points to spatial points is described as in Equation 4.3. We wish to find the control points, $\mathbf{c}$, that minimizes

$$f(\mathbf{c}) = \|\beta\mathbf{c} - \mathbf{s}\|_2^2, \tag{4.4}$$

which is the Euclidean norm of the residual vector $\beta\mathbf{c} - \mathbf{s}$.

In cases where $m > n$, we use QR-factorization[26] to find a least-squares solution. The matrix β can be factored as $\beta = QR$ where Q is an $m \times m$ orthogonal matrix and R is upper triangular in the first n rows, with the remaining $m - n$ being zero rows. Expanding Equation

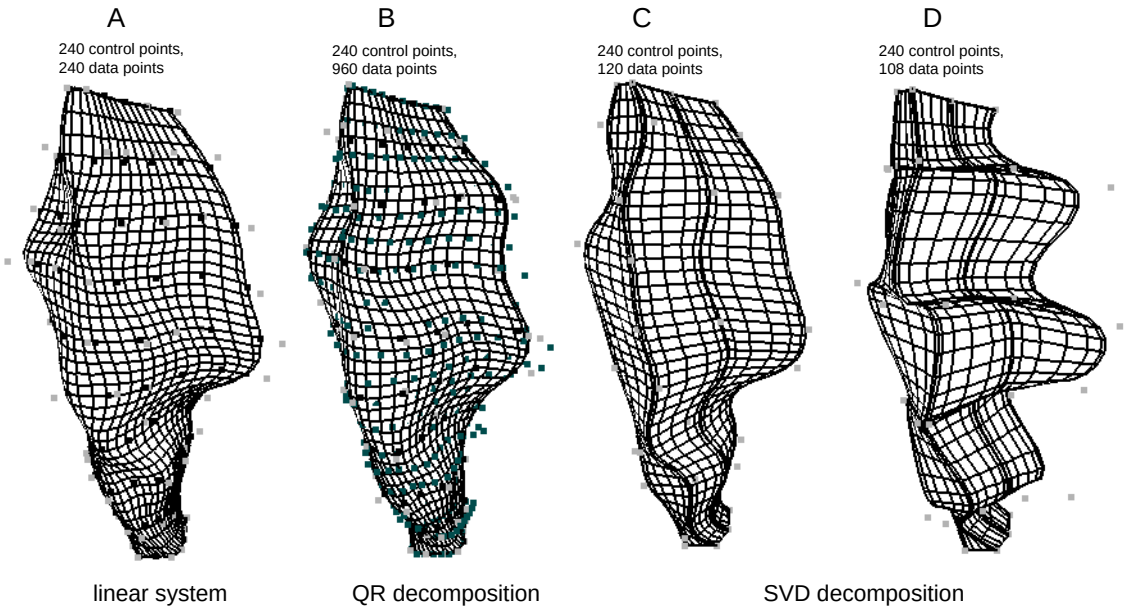


Figure 4.1: Data-fitting techniques for B-spline solids. Data points are in black and control points are in gray. Images A, B, and C represent $m = n$, $m > n$ and $m < n$, where m are the number of data points and n are the number of control points. Image D illustrates the case where an insufficiency of data points can lead to unwanted results.

4.4, we have

$$\begin{aligned} \|\beta\mathbf{c} - \mathbf{s}\|_2^2 &= \|R\mathbf{c} - Q^T\mathbf{s}\|_2^2 \\ &= \left\| \begin{array}{cc} R_1\mathbf{c} & -Q_1^T\mathbf{s}_1 \\ 0 & -Q_2^T\mathbf{s}_2 \end{array} \right\|_2^2 \\ &= \|R_1\mathbf{c} - Q_1^T\mathbf{s}_1\|_2^2 + \|Q_2^T\mathbf{s}_2\|_2^2, \end{aligned} \tag{4.5}$$

where R_1 is the $n \times n$ upper triangular portion of R and Q_1 contains the first n rows of Q with the remaining $m - n$ rows in Q_2. The spatial points, $\mathbf{s}$ are also partitioned into two vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ with the first n entries in $\mathbf{s}_1$ and the remaining $m - n$ in $\mathbf{s}_2$. To minimize Equation 4.5, we need to solve the system $R_1\mathbf{c} = Q_1^T\mathbf{s}$ to reduce the first term in the last line of Equation 4.5 to zero. As R_1 is upper triangular, $\mathbf{c}$ can be found easily with back-substitution.

For the case where $m < n$, QR factorization cannot be used. Since the system is underdetermined, the Singular Value Decomposition (SVD) method can be used to find a least-squares solution amongst the many possible solutions. The SVD factorization of β is UDV^T, where U and V are both orthogonal and $D \in \mathcal{R}^{m\times n}$ contains non-negative values on the diagonal entries,

$$d_{ii} = \sigma_i \geq 0, i = 1, \ldots, \min\{m, n\}, d_{ij} = 0, i \neq j, \tag{4.6}$$

and zero elsewhere[26]. The least-squares solution is found by using the pseudoinverse, $\beta^+ = VD^+U^T$, with

$$D^+ = \left\{ \begin{array}{c} d_{ii}^+ = \\ d_{ij}^+ = 0, \end{array} \begin{array}{c} \left\{ \begin{array}{cc} 1/\sigma_i, & \sigma_i > 0 \\ 0, & \sigma_i = 0 \end{array} \right. \\ i \neq j \end{array} \right., \tag{4.7}$$

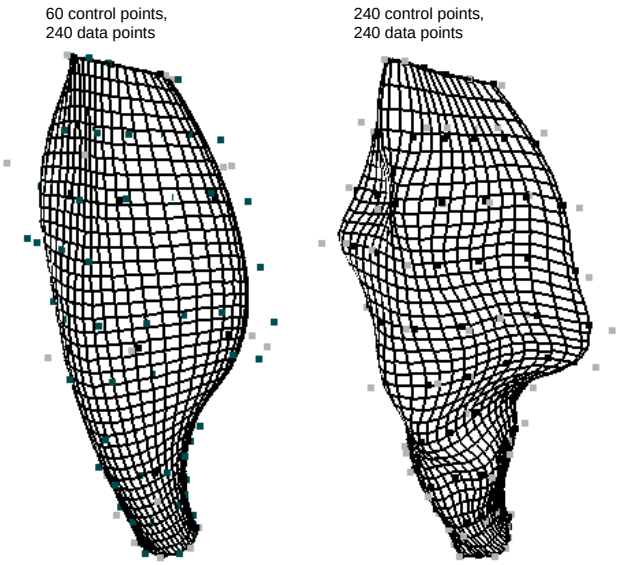


Figure 4.2: Given the same number of data points, but different control points for a B-spline solid, the solid with the fewer control points (left) produces a smoother looking B-spline solid because high frequency details cannot be captured.

and $\mathbf{c} = \beta^{+}\mathbf{s}$.

Although SVD can also be used to solve cases where $m > n$, less computation is required to perform a QR decomposition. Consequently, we use QR decomposition for overdetermined systems and SVD for underdetermined systems. Where the system is underdetermined, the SVD technique tends to concentrate the control points at the data points. Visually, this can lead to problems where m is relatively small compared to n, as oscillations in the shape can appear between data points (Figure 4.1).

The overdetermined systems allow us to approximate a large data-set of points with a compact shape representation of fewer control points. The use of least-squares methods with B-spline solids and their inherent smoothness properties tend to act as a low-pass filter process that produces smooth surface fairing than if the original spatial points were fitted to a B-spline solid with an equal number of control points (Figure 4.2).

Although techniques exist for producing better quality faired surfaces[70, 126], these methods assume that the surface will remain static. For animation of musculotendons, the B-spline solids will be constantly changing shape, requiring a faired surface to be recomputed for every timestep. Not only may this prove computationally prohibitive, there is no guarantee of coherence in control points (or spatial points) from frame to frame. For this reason, we use the linear system defined in Equation 4.2 to quickly recompute the new control point configuration for a changing set of spatial points in an animated B-spline solid.

4.3 Data-fitting using the continuous volume sampling function

In section 4.1, we described how to solve for a set of control points given the same number of spatial points. In order to achieve this, material coordinates had to be assigned to each spatial point. Using this mapping, we can establish a methodology for fitting a B-spline solid to a set of data points that represents the spatial pionts.

The data-fitting process can be described as a series of stages:

1. From various raw data sources, extract 3-D coordinate data that samples the physical object we wish to represent. Samples can occur within the volume of the object as well as on its boundary.

2. Using the 3-D data, construct a *continuous volume sampling function* (CVSF) that constructs a continuous, volumetric parameter space over which we can generate an arbitrary number of samples that span the solid's domain. The CVSF can be described as: $\mathbf{CVSF}(\tilde{u}, \tilde{v}, \tilde{w})$, such that $\tilde{u}, \tilde{v}, \tilde{w} \in [0, 1]$. We normalize the three parameters to allow reuse of the CVSF for solids of varying number of control points.

3. We uniformly sample the parameter space of the CVSF to generate the same number of spatial points as we have control points in our model. Material coordinates are assigned to these spatial points as described in Section 3.5.3.

4. Using the sampled data points as spatial points, $\mathbf{x}$, solve the linear system described in Equation 4.3.

Once the B-spline solid is created with an initial shape based on the data points, we can subsequently manipulate the shape for minor adjustments or animate global changes through physically-based modelling or deformation.

We have developed this procedure for three different sources of data which may arise for different muscle modelling applications. Stages 1 and 2 must be custom-designed for each different data source, while 3 and 4 are independent of the original form of the raw data. We will outline the first two stages for three different examples: a stack of contour curves, a set of digitized muscle fibres, and profile curves for interactive creation of muscle.

4.4 Stack of contour curves

We may have a series of images which represent cross-sectional (axial) slices of various body segments. For example, we used portions of the Visible Human data-set[89] in the lower leg region to isolate human soleus muscle (Figure 4.4). Each image was obtained from the Visible Human Male data-set from the anatomical portion of the data set (versus the MRI and CT sections). The resolution of an image was originally 2048x1216 pixels, at 24-bit colour, at 1 mm intervals between axial slices. To reduce storage requirements and to prepare for boundary extraction, the images were scaled down to 800x500 pixels and reduced to 256 gray levels. As the muscle boundaries we were extracting did not vary dramatically from slice to slice, we

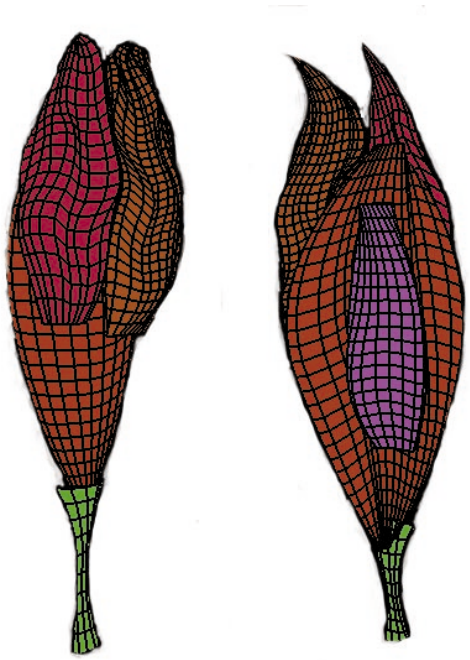


Figure 4.3: Posterior (left) and anterior (right) views of B-spline solids derived from Visible Human soleus and gastrocnemius data.

obtained images at 10.0 mm intervals to decrease the amount of data to process. This also illustrates the benefit of using B-spline solids to smoothly interpolate between an incomplete set of image contours to create a whole muscle (Figure 4.3).

As we could not reliably segment the individual posterior and anterior regions of the soleus by automatic means, an anatomist examined each of the data images and delineated the different regions, outlining the boundaries using a dark thick line with an image paint program. This facilitated the use of active contours ("snakes")[67] to quickly guide a deformable closed contour curve to the marked boundaries. Active contours operate by finding a curve configuration that navigates an energy gradient created from the gray scale image of the anatomical slice. The curve settles in the local minima of the energy landscape. By tracing the boundaries with dark lines, we accentuate these energy troughs, so the active contours quickly become attracted to the contours. Once the curve has reached equilibrium, it is sampled to extract a piecewise line segment representation of the contour. The set of 3-D points that make up these contour curves are used to create a CVSF.

4.4.1 Creating the CVSF

Given a set of n curves extracted from the axial slices: $C_0, \ldots, C_{n-1}$, we index the curves with the vertices that it contains. If curve i has m vertices, we represent each point on the curve as $C_{i,j}$, where $j = 0 \ldots, m-1$. We wish to generate a CVSF that produces a uniform sampling of points throughout the volume spanned by these image contour curves.

Our strategy is to interpolate the contour curves such that traversing the parameters will span the solid in an intuitive manner. We designate $\tilde{u}$ to increase radially outward from a

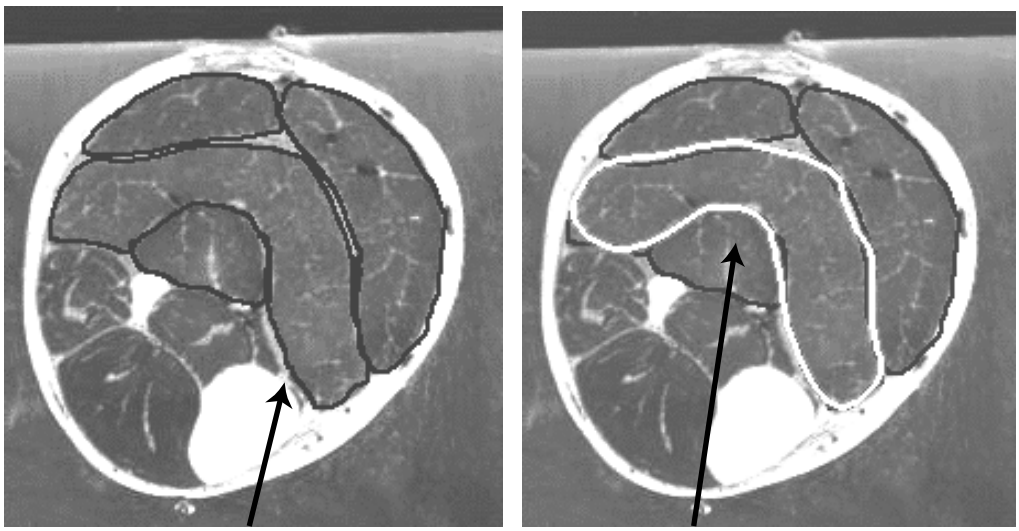


Figure 4.4: The left image is an image slice from the Visible Human dataset, showing the muscle contours outlines in black. On the right, an active contour (in white) has attached itself to the dark contour.

central axis of the solid to its contour boundaries. Parameter $\tilde{v}$ will traverse the perimeter around the axis and $\tilde{w}$ will range from the bottom of the stack to the top (Figure 4.5).

To define a centre curve along the axis of the muscle, we compute the centroid points, $\hat{c}_i$ of each contour and interpolate a B-spline curve through these points. We denote the centre axis as axis$(\tilde{w})$, where we can reparameterize the curve from $\tilde{w} \in [0, 1]$. Interpolating a closed B-spline though the points of each contour curve C_i, produces the curves $C_i(\tilde{v}), i = 0 \ldots, n-1$. Our algorithm for constructing the CVSF from contour curves can now be stated as:

Given $(\tilde{u}_0, \tilde{v}_0, \tilde{w}_0) \in ([0, 1] \times [0, 1] \times [0, 1])$,

1. Construct a B-spline curve, profile$_{\tilde{v}_0}(\tilde{w})$ that interpolates all points $C_i(\tilde{v}_0), i = 0 \ldots, n-1$.
2. Compute $\mathbf{p_0} =$ axis$(\tilde{w}_0)$ and $\mathbf{p_1} =$ profile$_{\tilde{v}_0}(\tilde{w}_0)$.
3. Obtain the final sampled point, $\mathbf{p}$, by linear interpolation: $\mathbf{p} = \mathbf{p_0}(1 - \tilde{u}_0) + \mathbf{p_1}\tilde{u}_0$.

Linear interpolation in step 3 produces a uniform radial sampling from the centre axis to the outer boundary. However, if we were to simply define parameterizations that linearly interpolate between the contour levels , our iso-lines running longitudinally down the solid would be aligned to the planes of the image slices (Figure 4.6C).

The iso-lines should be independent of the original number and orientation of the contour curves. Instead, we parameterize axis$(\tilde{w}_0)$ and profile$_{\tilde{v}_0}(\tilde{w}_0)$ by arclength to produce a sampling that uniformly spans the spatial dimensions of the data. To reduce twisting about the axis, we repeatedly increment the indices j for each curve, $C_{i,j}$, using the update rule,

$$j_{new} = (j_{old} + 1) \bmod m, \tag{4.8}$$

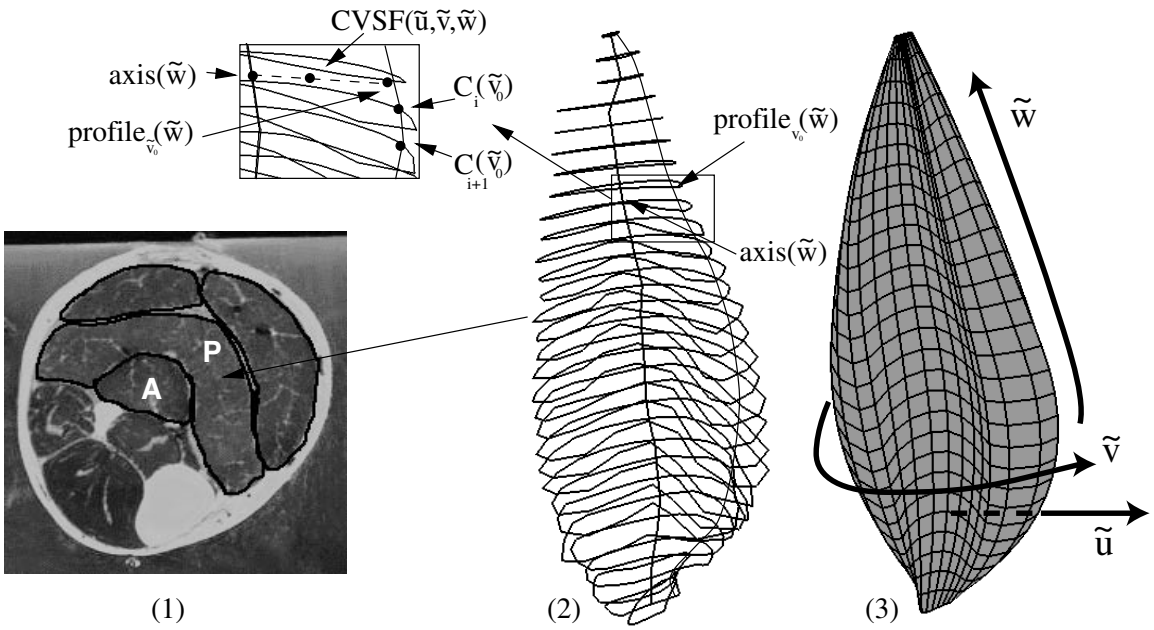


Figure 4.5: Contour curves extracted from the Visible Human data-set are used to create a CVSF to generate a B-spline solid. Parameter directions are displayed for the CVSF.

until the sum of squared distances between points in adjacent contour levels,

$$\text{dist}^2 = \sum_{j=0}^{m-1} (C_{i,j} - C_{i+1,j})^2, \tag{4.9}$$

is minimized. Figure 4.6D shows the effects of solid-fitting with and without twist correction.

With the defined CVSF($\tilde{u}, \tilde{v}, \tilde{w}$), we can freely sample spatial points as follows,

$$\mathbf{s}_{ijk} = CVSF(\frac{i}{N_u - 1}, \frac{j}{N_v}, \frac{k}{N_w - 1}), \tag{4.10}$$

where $i = 0, \ldots, N_u - 1, j = 0, \ldots, N_v - 1, k = 0, \ldots, N_w - 1$. In closed, contour curves, the $\tilde{v}$ parameter is periodic. Consequently, the denominator is set to N_v in Equation 4.10, to reflect the fact that $\tilde{v} = 0$ and $\tilde{v} = 1$ will produce the same point if the other two parameters are the same value.

By transforming the original static images from the Visible Human data-set to a deformable, B-spline solid model, we can subsequently animate its shape, simulate contractions and visualize fibres in its volume. Unfortunately, from an anatomical standpoint, the fibres reconstructed from a contour curve-derived B-spline solid generates iso-curve fibres that do not match muscle architecture observed in real muscle (Figure 4.7). The contour curves inherently contain only boundary information, with no information about the internal structure of muscle. By using fibre set data, we can build a more appropriate model that can capture fibre architecture.

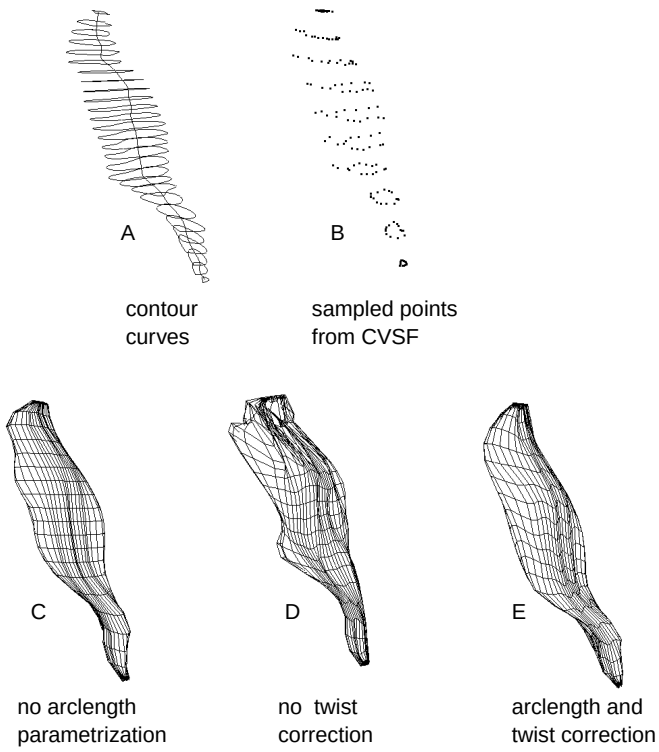


Figure 4.6: Sources of distortion in data-fitting of B-spline solids without arclength parameterization and twist correction.

4.5 Fibre sets

In order to depict the correct arrangement of fibres that make up muscle tissue, it was necessary to look at alternative data collection techniques to traditional medical images of axial sections. It is extremely difficult, if not impossible to track individual fibres between adjacent axial sections. Working with anatomy experts[1], the following procedure was developed for retrieving 3-D coordinate information on muscle fibres. We describe the procedure as performed with human soleus muscle.

4.5.1 Specimen preparation for serial dissection

The human soleus muscle has three architecturally, distinct regions: posterior, marginal and anterior (Figure 4.8). The muscle specimen was serially dissected to obtain fibre arrangements throughout the entire volume of the muscle. The specimen was placed on a calibrated base plate to establish a common reference coordinate system for all specimen layers. The position and orientation of the base plate was calibrated with three cameras using the *direct linear transformation* (DLT)[1] procedure (Figure 4.9).

Using the three camera setup, images were taken at different stages in the serial dissection of the human soleus. In serial dissection, the layers of muscle tissue are removed one at a

[1]Anne Agur and Nancy McKee from the Department of Anatomy and Surgery, University of Toronto.

Figure 4.7: Comparison of generated fibres for the posterior soleus region from different data sources. The image on the left shows fibres incorrectly running from top to bottom in a side view of posterior soleus derived from contour stacks. The right image shows the correct fibre arrangement, running posterior to anterior, in a solid generated from fibres obtained by serial dissections.

time from the most superficial layers to the deeper, interior ones. The human soleus specimens were dissected in the posterior to anterior direction. At each level, the end points of 50-100 representative fibres were pinned with colour-coded beads. The fibres were chosen to be evenly distributed along the length of each muscle layer (Figure 4.10B).

The muscle specimens were attached to the fibula and tibia bones of the lower leg *in situ*. The bones were rigidly attached to the base plate. The position of the colour-coded beads were extracted from each of the three images using a modified form of DLT[8]. These muscle-related coordinates were sampled points taken along representative fibres distributed throughout the muscle specimen (Figure 4.11).

4.5.2 Fibre set CVSF construction

For each architecturally distinct muscle region, each muscle fibre can be represented with an ordered n-tuple, $\mathbf{F} = (\mathbf{p}_0, \ldots, \mathbf{p}_{n-1})$, where n points have been sampled from the fibre with $\mathbf{p}_0$ and $\mathbf{p}_{n-1}$ representing the fibre end points. Each fibre is given two indices to reference its position within the muscle. The fibre $\mathbf{F}_{i,j}$ represents the j^{th} fibre from the i^{th} layer of serial dissection. If a third index is added, $\mathbf{F}_{ijk}$ denotes the point $\mathbf{p}_k$ of the fibre $\mathbf{F}_{ij}$.

Fibres are not constrained to have the same number of samples. This allows flexibility in choosing samples as a minimum of two points can be used for relatively straight fibres, with more points for fibres that are curved. The number of fibres do not have to be the same within each layer, allowing more fibres to be sampled for muscle layers that are larger in area and would require more fibres to maintain the same sampling density per area of muscle.

Using this indexing scheme for fibres and their points, we can design the CVSF as follows.

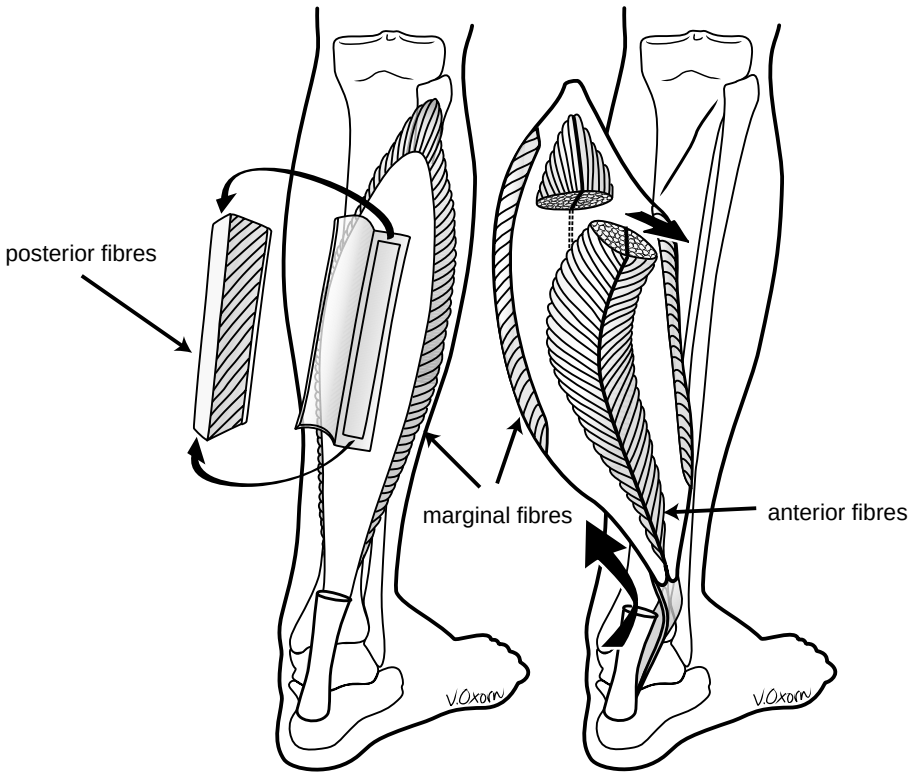


Figure 4.8: There are three architecturally-distinct regions in the human soleus: marginal, posterior and anterior fibres. Images courtesy of Valerie Oxorn, Copyright 1998-1999.

Given $(\tilde{u}_0, \tilde{v}_0, \tilde{w}_0) \in ([0,1] \times [0,1] \times [0,1])$,

1. Let $\tilde{u}_0$ measure the amount of depth within a muscle. Let **numLayers** be the number of layers obtained with serial dissection. The layers are indexed 0-relative. Let **numFibres**$_i$ be the number of fibres in layer i. From $\tilde{u}_0$, we compute the indices using floor functions:

$$i = \lfloor \tilde{u}_0(\textbf{numLayers} - 1) \rfloor, j = \lfloor \tilde{v}_0(\textbf{numFibres}_i - 1) \rfloor. \tag{4.11}$$

2. Create the B-spline curve, bcurve$_{i,j}(\tilde{w})$ that interpolates the points of $\mathbf{F}_{i,j}$. We parameterize the curve by the arclength. For fibres consisting of two endpoints only, we use degree one B-spline curves. For all other fibres, we use degree two curves. Cubic curves introduced unwanted oscillations that may overestimate fibre lengths.
3. Obtain the following four points:

$$\begin{aligned} \text{point}_{00} &= \text{bcurve}_{i,j}(\tilde{w}_0) \\ \text{point}_{01} &= \text{bcurve}_{i,j+1}(\tilde{w}_0) \\ \text{point}_{10} &= \text{bcurve}_{i+1,j}(\tilde{w}_0) \\ \text{point}_{11} &= \text{bcurve}_{i+1,j+1}(\tilde{w}_0). \end{aligned}$$

These four points define the corners of a bilinear space over which we will interpolate to get the final point.

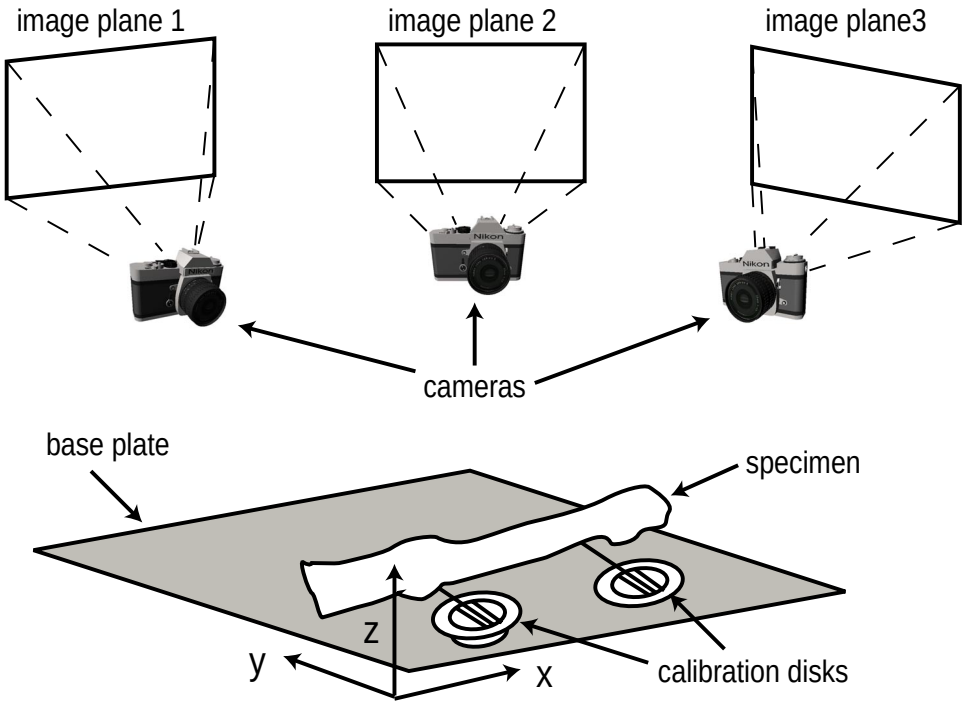


Figure 4.9: Camera setup for digitizing fibres. The muscle specimen is placed *in situ* on top of a base plate. Three cameras record a 2-D image in preparation of the DLT process.

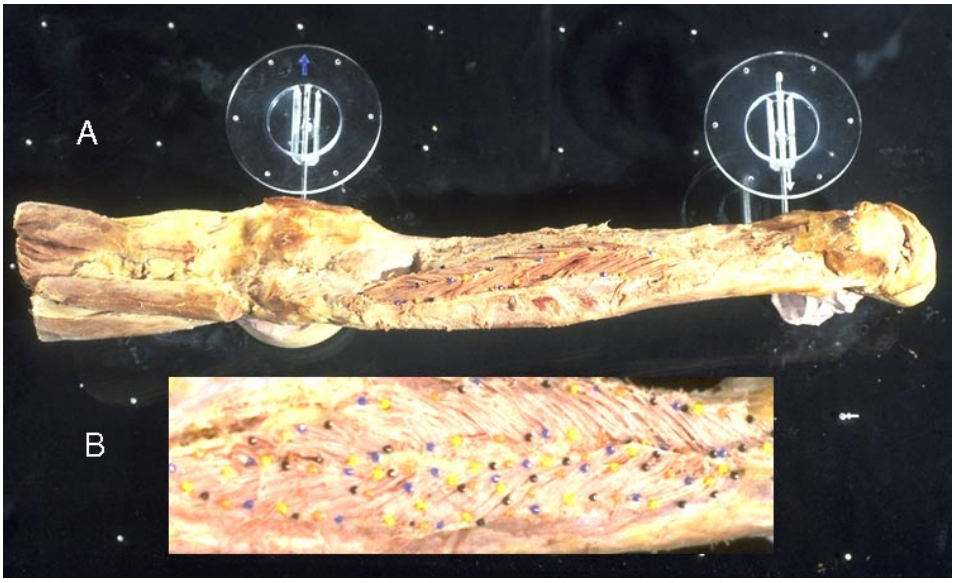


Figure 4.10: Serial dissection of human soleus muscle is conducted by placing the specimen on the base plate (A). The fibres of the soleus are marked with colour-coded beads for identification and 3-D coordinate extraction (B).

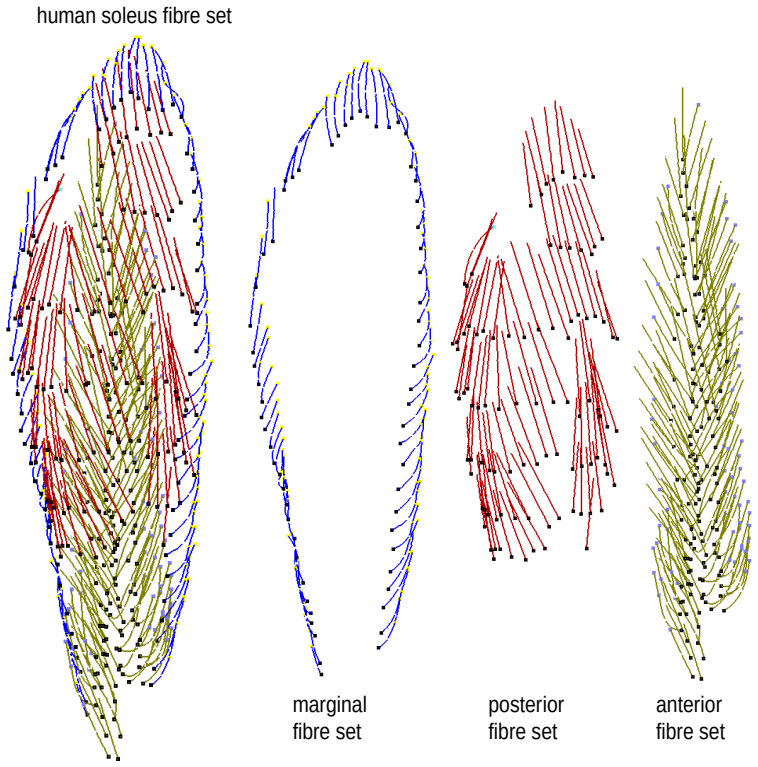


Figure 4.11: Fibre sets for human soleus data. On the left, are all the fibre sets shown together. Each architecturally-distinct fibre set is displayed in isolation on the right.

4. Determine the interpolation parameters:

$$\hat{u} = \tilde{u}_0(\textbf{numLayers} - 1) - i, \hat{v} = \tilde{v}_0(\textbf{numFibres}_i - 1) - j \qquad (4.12)$$

5. Linearly interpolate between the pair of points:

$$\begin{aligned} \text{point}_0 &= \text{point}_{00}(1-\hat{v}) + \text{point}_{01}\hat{v} \\ \text{point}_1 &= \text{point}_{10}(1-\hat{v}) + \text{point}_{11}\hat{v}. \end{aligned}$$

6. Perform the second part of the bilinear interpolation:

$$\textbf{CVSF}(\tilde{u}_0, \tilde{v}_0, \tilde{w}_0) = (1-\hat{u})\text{point}_0 + \hat{u}\text{point}_1. \qquad (4.13)$$

The bilinear interpolation in the procedure occurs in the space spanned by the four B-spline curves computed with parameter $\tilde{w}$. Figure 4.12 illustrates this process.

Using the CVSF, we have built B-spline solids that can capture muscle architectural details of internal fibre arrangements. Fibres can be generated by creating streamlines (refer to Section 3.2.5), where each iso-curve represents a muscle fibre (two parameters are held constant while the third varies). Since the streamlines have an analytical expression, we can compute physical characteristics within the muscle model that would be tedious to perform on a real specimen.

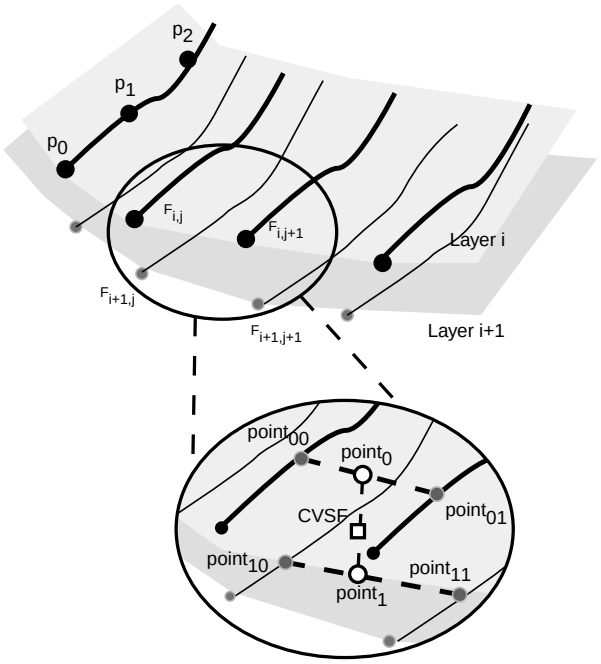


Figure 4.12: CVSF construction for fibre sets. The CVSF sample point is the result of a bilinear interpolation process on the fibre set data.

4.5.3 Fibre lengths

With the streamlines representing fibres, we can compute the arclength to estimate fibre length. The arclength can be computed using numerical integration schemes such as Simpson's rule [100]. Alternatively, a less-accurate, but faster estimate can be computed by chord length approximation (summing the lengths of the line segments joining sampled points on each fibre). As the number of sampled points increase, the length converges to the true arclength. Figure 4.13 illustrates computed lengths of fibre streamlines generated in a model of the posterior soleus region.

4.5.4 Fibre pennation angles

The angle a fibre's orientation has with respect to the tendon is traditionally approximated as a single, lumped average for the entire muscle[138]. Observations taken from the dissected soleus muscle in Section 4.5.1 show that pennation angle can vary significantly within a single muscle. The existence of a 3-D muscle model motivates the possibility of developing a more accurate method to measure pennation angle. Pennation angle is usually measured by examining the fibre angle within a planar cross-section of muscle relative to the tendon attachment area. Mechanically, the pennation angle of the fibre should be measured relative to the local tangent plane of the tendon aponeurosis that it is attached to. This orientation information is not obtainable from a cross-sectional view of the musculotendon.

With a B-spline solid model, we have the muscle represented with a solid continuum, complete with derivatives defined at all its points. We can compute pennation angles directly by

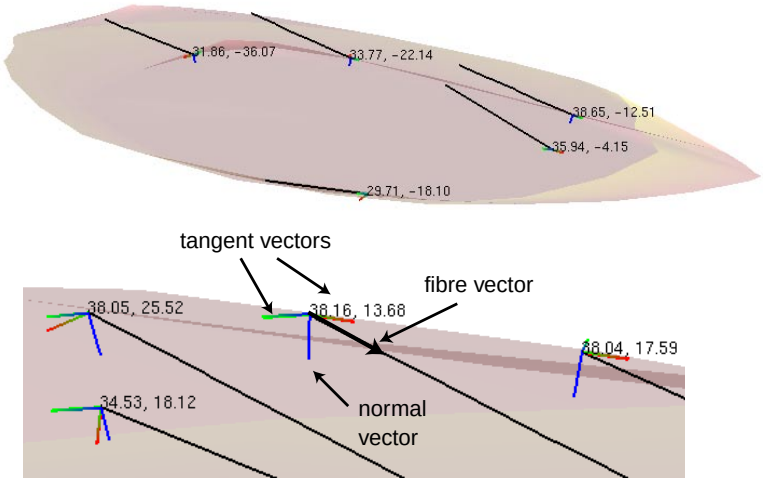


Figure 4.13: Fibre measurements such as fibre length and pennation angle can be computed analytically on the virtual muscle model. In this example, sampled fibres from the posterior soleus are measured for fibre length and pennation angle and the measurements are labelled as a pair, (fibre length, pennation angle), beside each fibre. The bottom image illustrates how the pennation angle can be calculated by measuring the angle the fibre tangent makes with the local tangent plane.

computing the tangent plane of the attachment point of the surface each fibre inserts into. We represent a particular streamline created by holding $u = u_0$ and $v = v_0$ constant and varying w by

$$\mathbf{x}(u_0, v_0, w), w \in [w_{min}, w_{max}] \tag{4.14}$$

where $\mathbf{x}$ is the B-spline solid Equation 3.1. Therefore, the unit tangent vector at the attachment site, along the streamline is

$$\mathbf{t}_{u_0,v_0} = \frac{\frac{\partial \mathbf{x}(u_0,v_0,w)}{\partial w}}{\|\frac{\partial \mathbf{x}(u_0,v_0,w)}{\partial w}\|}. \tag{4.15}$$

The unit normal vector of the tangent plane of the aponeurosis is computed from the cross-product of the tangent vectors making up the plane:

$$\mathbf{n}_{u_0,v_0} = \frac{\frac{\partial \mathbf{x}(u_0,v_0,w)}{\partial u} \times \frac{\partial \mathbf{x}(u_0,v_0,w)}{\partial v}}{\|\frac{\partial \mathbf{x}(u_0,v_0,w)}{\partial u} \times \frac{\partial \mathbf{x}(u_0,v_0,w)}{\partial v}\|}. \tag{4.16}$$

The angle, α, the fibre's tangent vector, $\mathbf{t}_{u_0,v_0}$ makes with the tangent plane is:

$$\alpha = \frac{\Pi}{2} - \cos^{-1}\left(\mathbf{t}_{u_0,v_0} \cdot \mathbf{n}_{u_0,v_0}\right) \textit{ radians} \tag{4.17}$$

This pennation angle equation is closer to the true pennation angle because it takes into account the differences between the local orientation of the surface the fibre is attached to and

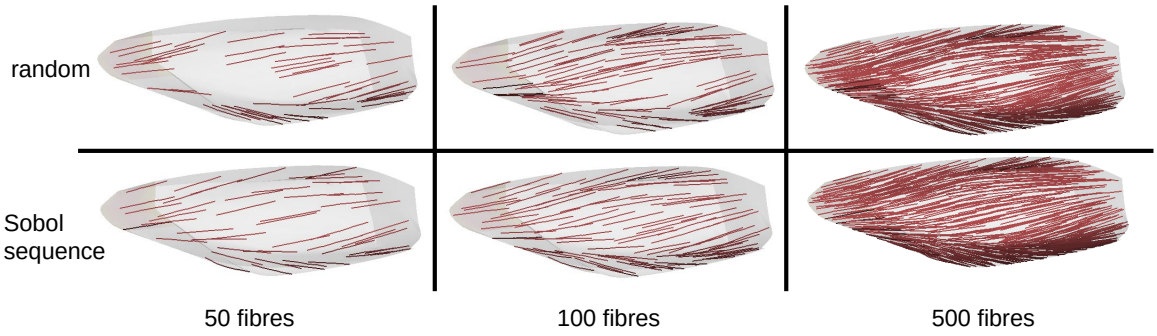


Figure 4.14: Generation of fibres. 50, 100, and 500 fibres were generated in a model of the posterior soleus using uniform pseudorandom number generation and Sobol sequences. The Sobol sequence produced better distribution of fibres. However, as the number of fibres get larger, the differences are less noticeable, making Sobol sequences more valuable for lower sampling densities.

the fibre's local orientation by computing instantaneous derivatives at the point of attachment (Figure 4.13). This improved measure of pennation angle allows muscle force components to be calculated more accurately at a finer resolution than previous lumped-average estimates of fibre pennation.

4.5.5 Visualizing individual fibres

Once the B-spline solid is fit to the fibre set data, we can visualize arbitrary fibres within the solid by generating streamlines. In our solids, one of the parameters will coincide with the longitudinal direction of the fibre. In our examples, we vary the w parameter. We can represent these streamlines as 3-D space curves,

$$\mathbf{streamline}(u_0, v_0, w), \tag{4.18}$$

where $u = u_0, v = v_0$, and $w \in [w_{min}, w_{max}]$.

Sampling using parameter values at uniform intervals produces an unnaturally, regular grid-like distribution of fibres. We can randomly generate fibres, but we may get clumps of fibres where the pseudorandom number generation has selected fibres that are spatially close together. Solids with a cylindrical topology will have the characteristic that the same parameter interval covers a denser spatial region closer to the axis than at the perimeter of the cylindrical boundaries. We can adjust the random sampling to bias the probability of choosing parameters of larger radius to create a spatially even distribution. We have experimented with linear ramps and increasing quadratic functions. Ideally, we would like every sample to be chosen to give an overall non-structured appearance, simultaneously avoiding clumped fibres in regions of the muscle model. We use a two-dimensional Sobol sequence to achieve this[100]. Figure 4.14 compares these different fibre generation methods.

Sobol sequences for fibre generation

A Sobol sequence is considered to be a quasi-random or sub-random sequence where the points are chosen to maximally avoid each other, distributing the fibres evenly across the solid (Figure 4.14). As we increase the sampling density, newer fibres are generated between the space of existing fibres, never generating the same parameter values twice. We use the Sobol sequence implemented in [100], which generates new numbers from the manipulation of binary functions called *direction numbers*. Other equidistributed sequences, poisson-disk sampling, or *stratified sampling techniques*[41] can be used for creating uniformly distributed fibres.

4.6 Profile curves

> If you have drawn the bones of the hand and you wish to draw the muscles covering and joining the bones, then draw threads, though not muscles.
>
> **Leonardo da Vinci**

The first two data sources presented offer a migration path from sources of actual anatomical, muscle data to construct a virtual 3-D B-spline solid model. However, there may be occasions where interactive creation of solid shapes is desirable. For example, for an application involving muscle reconstruction of extinct animals, it would be impossible to obtain soft tissue specimens of real anatomy. A computer artist may wish to design a novel creature with a unique musculoskeletal system that does not exist. Generally, a creature designer will use comparative anatomy to guide where major muscle groups are placed on a fictional animal. These applications provoke the need to be able to design the shapes of B-spline solids interactively, with visual guidance from a human designer.

The attachment of musculotendon onto a skeleton can be characterized by three important curves: origin, insertion and axial. The origin and insertion curves indicate the attachment areas of the tendon portions of a musculotendon to the bony surfaces of the skeleton. The axial curve indicates the direction the muscles takes from one end of the curve to the other - its line of action. We developed a CVSF that constructs the solid muscle using these *profile curves*.

We represent the origin and insertion as two closed, periodic B-spline curves, origin$(\tilde{v})$ and insertion$(\tilde{v})$. We used quadratic (degree 2) B-spline curves, but higher order curves can be used. The axial curve is an open, non-periodic, degree 2 B-spline, axial$(\tilde{w})$. In practice, quadratic curves provide a good compromise between smooth curves and reduced oscillations in the curve. We would like the CVSF to have the following properties that guarantee the solid will interpolate the profile curves on its boundaries:

$$\begin{aligned} \mathbf{CVSF}(1,\tilde{v},0) &= \text{origin}(\tilde{v}), \\ \mathbf{CVSF}(1,\tilde{v},1) &= \text{insertion}(\tilde{v}), \\ \mathbf{CVSF}(0,\tilde{v},\tilde{w}) &= \text{axial}(\tilde{w}), \forall \tilde{v} \in [0,1]. \end{aligned}$$

Note that the last condition implies there is some redundancy where several parameter coordinates with different $\tilde{v}$ map to the same axial point. This occurs because the cylindrical solid we

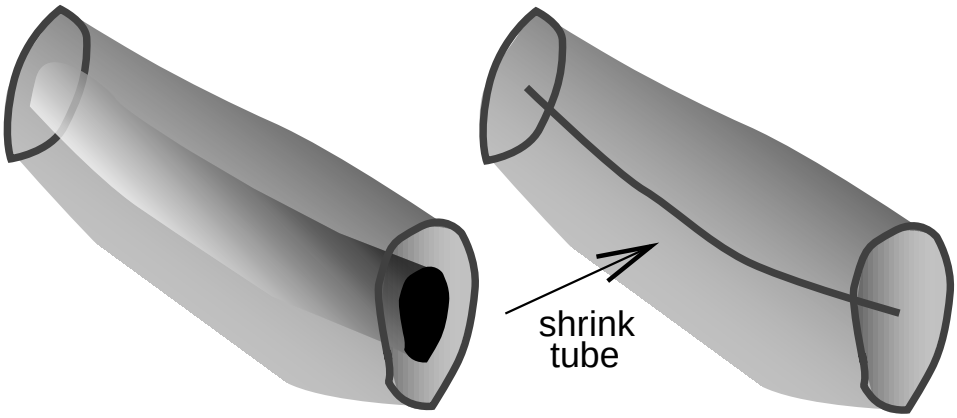


Figure 4.15: Creation of a cylindrical solid. The inner surface of a tubular solid (left) is collapsed to a single axial curve (right).

are using is actually tubular, with the inner $\tilde{u} = 0$ surface collapsing to a single curve (Figure 4.15).

In addition to interpolating these profile curves, we would like to have some control over the midsection of the muscle. If both the origin and insertion curves were small, a swept curve that linearly interpolates the two terminal curves would produce a solid shape with a narrow midsection. This would restrict the number of useful shapes that can be created, especially for muscle where the midsection of muscle is often wider than its ends. To remedy this, we can generate automatically an initial mid-sectional curve, middle$(\tilde{v})$, which can be positioned anywhere along the axis. We chose $\tilde{w} = 0.5$ for a balanced weighting between the two end curves. The middle, along with the end curves, influence the shape of the musculotendon along its length, similar to a swept, lofted model. Rather than linearly interpolate between the sections directly, we developed a nonlinear weighting to account for the observation that the shape does not uniformly change from the midsection to the extremal sections. Furthermore, we wish the sections to be roughly perpendicular to the local orientation of the axial curve in the area. That is, we wish to produce a type of deformable, generalized cylinder[107].

4.6.1 Reference frames for the axial curve

To generate points that deform as the axial curve moves, we need to define a reference frame that can be centred for any $\tilde{w}$ of axial$(\tilde{w})$. There are several candidates.

The *Frenet frame*[31] defines the reference axes from the first and second derivatives of a space curve. Specifically,

$$\begin{aligned} \mathbf{t} &= \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}, \\ \mathbf{m} &= \mathbf{b} \times \mathbf{t}, \\ \mathbf{b} &= \frac{\dot{\mathbf{x}} \times \ddot{\mathbf{x}}}{\|\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\|}, \end{aligned}$$

which are known as the tangent, main normal and binormal vectors respectively. These axes follow the orientation and curvature of the curve. These three vectors can be the columns of

a matrix, $L_{Frenet} = [\mathbf{t}\ \mathbf{m}\ \mathbf{b}]$ that can be used as a rotation matrix to transform points to and from this frame (the transpose of L_{Frenet} is the inverse transformation). Unfortunately, the Frenet frame is not defined for line segments (due to second derivative values of zero) and experiences sudden discontinuities in orientation at inflection points on the curve. This can cause undesirable twisting along the shape's axis.

We use Bloomenthal's rotation minimizing frame[12] to avoid sudden large changes in orientation of the local frames along the axial space curve. This reference frame can be represented as a 3×3 matrix, L_{rotmin}. By ensuring these axes are orthogonal to each other and all unit vectors, L_{rotmin} is orthonormal, and further $L_{rotmin}^{-1} = L_{rotmin}^{T}$. We denote $L_{rotmin,\tilde{w}}$ to be the reference frame with origin at axial$(\tilde{w})$.

We compute $L_{rotmin,\tilde{w}}$ as follows:

$$L_{rotmin,\tilde{w}} = L_{first,0} R_q, \tag{4.19}$$

where $L_{first,0}$ is a reference frame calculated by rotating the world axes of $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ so that the *z*-axis, $(0,0,1)$ aligns with the tangent vector, $\mathbf{t}_0 = (t_0^x, t_0^y, t_0^z)$, at axial$(0)$. As there are many possible orientations that can align these two axes, we choose the rotation that minimizes the angle of rotation, θ_0, for the alignment,

$$\theta_0 = \cos^{-1}(t_0^z), \tag{4.20}$$

about the axis

$$\mathbf{a}_{first} = (-t_0^y, t_0^x, 0). \tag{4.21}$$

The reason why we chose to use L_{first} instead of L_{Frenet} is that the latter reference frame will twist as we manipulate its space curve. L_{first} does not change orientation as drastically and is more resistant to twisting along the axis because it is created relative to a constant world axes. Whereas the Frenet frame does not exist for straight lines, L_{first} can always be defined as we only need to know the line's tangent vector.

R_q is a rotation matrix derived from a quaterion $\mathbf{q}$ which encodes the angle and axis of rotation that transforms the tangent, $\mathbf{t}_0$, at axial(0) to the tangent, $\mathbf{t}_{\tilde{w}}$ at axial$(\tilde{w})$. The tangent of the axial curve is simply the unit vector,

$$\mathbf{t}_{\tilde{w}} = \frac{\dot{\text{axial}}(\tilde{w})}{\|\dot{\text{axial}}(\tilde{w})\|}, \tag{4.22}$$

where $\dot{\text{axial}}(\tilde{w})$ denotes the first derivative of the axial space curve. The angle, θ in radians and axis of rotation, $\mathbf{a}$ is defined as

$$\theta = \cos^{-1}(\mathbf{t}_0 \cdot \mathbf{t}_{\tilde{w}}) \tag{4.23}$$

and

$$\mathbf{a} = \frac{\mathbf{t}_0 \times \mathbf{t}_{\tilde{w}}}{\|\mathbf{t}_0 \times \mathbf{t}_{\tilde{w}}\|}. \tag{4.24}$$

The quaternion $\mathbf{q}$ is computed as:

$$\mathbf{q} = (\cos\frac{\theta}{2}, a^x \sin\frac{\theta}{2}, a^y \sin\frac{\theta}{2}, a^z \sin\frac{\theta}{2}), \tag{4.25}$$

and can be converted to a 3×3 rotation matrix using the algorithm described in [128].

Having developed a local coordinate system based on profile curves, we will describe an interactive user interface for defining these curves.

4.6.2 Profile curve sketching with direct manipulation

Our strategy is to allow the user to select points directly on the visible bone surface geometry through which profile curves should interpolate, allowing direct manipulation and edition of points. A musculotendon shape designer would proceed to select points to sketch out the curves for the origin, insertion and axial profiles of the muscle. In a modelling system, the bones can be represented using polygonal meshes or tensor product surfaces like NURBs or B-splines. If the user is using a 2-D pointing device to pick points on a 3-D image of the skeletal system, the problem is to find the corresponding 3-D points of the picked geometry.

One alternative is to locate the intersection of a projected ray from the picked screen coordinate with the surface geometry. This problem is encountered in ray-tracing applications[128]. The solution is not quick enough for interactive applications because of the computational expense involved in locating which geometry will first intersect this ray, and then mathematically computing the intersection point of the ray with the geometry. Furthermore, the original picked point is derived from a pixel which is already a discrete approximation of the true point on the viewing plane.

To take advantage of current depth buffer-based graphics hardware, we can extract the screen coordinates (x, y, z), where x and y are image coordinates and $z \in [0, 1]$ contains the depth buffer value for that pixel. With these coordinates and using a standard 3-D application programming interface like OpenGL, we can perform the inverse transformations of the modelling, camera and projection matrices in the graphics pipeline to determine the corresponding world coordinates for the picked point on the screen.

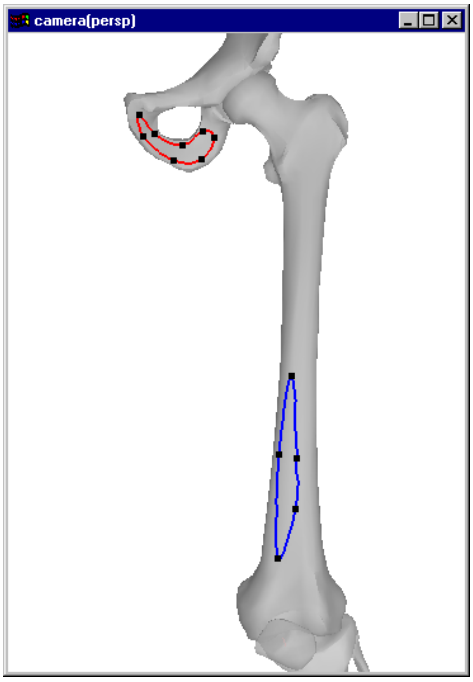
As we are dealing with discretized representations of the geometry, there is always the danger of imprecision. To minimize this potential inaccuracy, we closely bound the geometry with the near and far clipping planes to increase the dynamic range of the depth buffer. Current depth buffers have 16 to 32 bits of precision, providing enough significant digits for finding adequate world coordinates. A nice side-effect of using this method is that if we require more precision, the designer can zoom the camera closer to the bone geometry, so that accuracy is bound to the discretization error of the picked visible pixel. Figure 4.16 illustrates several stages where a designer can sketch a musculotendon directly onto the bone by successive construction of the profile curves.

4.6.3 CVSF construction

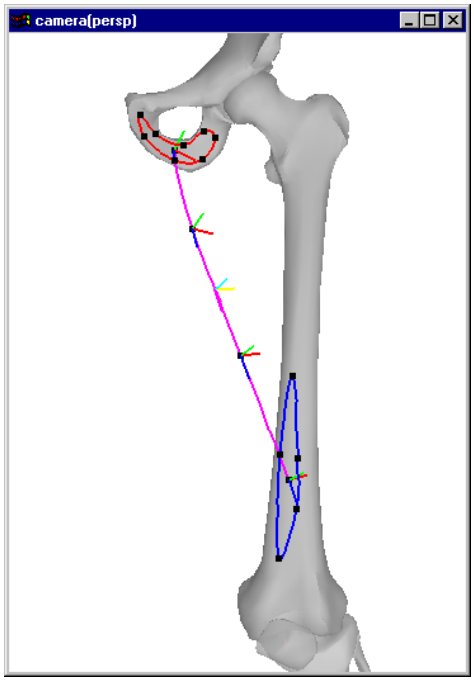
Having created a local reference frame for our axial curve and having obtained point samples for the origin, insertion and mid-section curves, we now describe how to compute $CVSF(\tilde{u}, \tilde{v}, \tilde{w})$.

We are given $(\tilde{u}_0, \tilde{v}_0, \tilde{w}_0) \in ([0, 1] \times [0, 1] \times [0, 1])$.

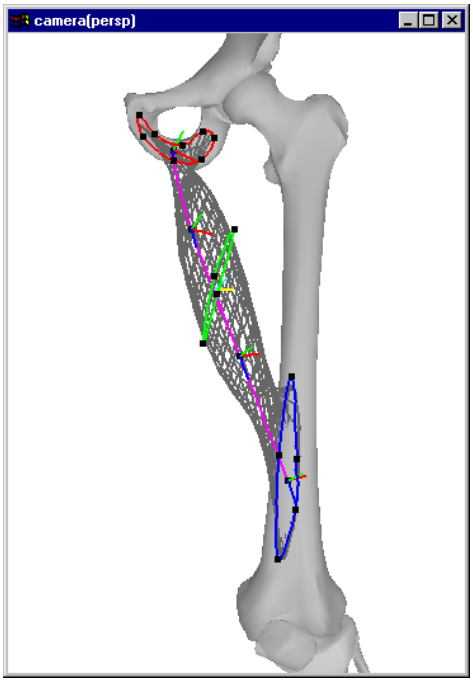
1. Create interpolating curves, origin($\tilde{v}$), insertion($\tilde{v}$), and middle($\tilde{v}$) from the obtained point samples created interactively from the user.
2. Evaluate points $\mathbf{p}_O = \text{origin}(\tilde{v}_0)$, $\mathbf{p}_I = \text{insertion}(\tilde{v}_0)$, and $\mathbf{p}_m = \text{middle}(\tilde{v}_0)$,
3. Express $\mathbf{p}_O$, $\mathbf{p}_I$, $\mathbf{p}_m$ in terms of the axial local frame of references, $L_{rotmin,0}$, $L_{rotmin,1}$, and $L_{rotmin,0.5}$ respectively, to get $\mathbf{p}'_O = \mathbf{p}_O L^T_{rotmin,0}$, $\mathbf{p}'_I = \mathbf{p}_I L^T_{rotmin,1}$, and $\mathbf{p}'_m = \mathbf{p}_m L^T_{rotmin,0.5}$.



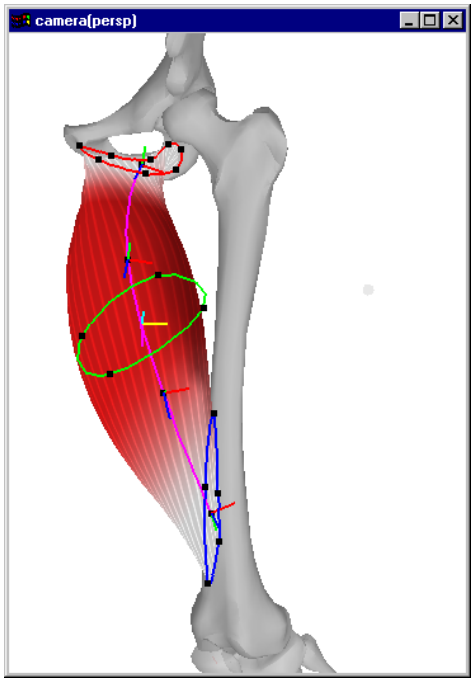
1) Sketch origin and insertion curves



2) Sketch axial curve



3) Build B-spline solid from profiles



4) Edit solid by manipulating profile curves

Figure 4.16: Stages of development of B-spline solids from profile curves. The middle curve is generated automatically in step 3. In the fourth step, the solids can be shaded and textured with striation patterns.

4. Interpolate between $\mathbf{p}'_O$, $\mathbf{p}'_m$, and $\mathbf{p}'_I$. Although we can linearly interpolate between these points, the resulting swept surfaces are not satisfactory and we will get only C^0 continuity on the mid-section curve. Rather, we nonlinearly bias the swept surfaces towards the mid-section curve by using a quadratic function in the following interpolation algorithm:

   $t = -4\tilde{w}_0(\tilde{w}_0 - 1)$
   if $\tilde{w}_0 \in [0, 0.5)$
   // Interpolate from the origin curve to the mid-section curve
   $\mathbf{p}' = \mathbf{p}'_O(1-t) + \mathbf{p}'_m t$
   else $\tilde{w}_0 \in [0.5, 1]$
   // Interpolate from the mid-section curve to the insertion curve
   $\mathbf{p}' = \mathbf{p}'_m t + \mathbf{p}'_I(1-t)$

5. Transform points back to world coordinates and scale by $\tilde{u}$:

$$\mathbf{CVSF}(\tilde{u}_0, \tilde{v}_0, \tilde{w}_0) = \text{axial}(\tilde{w}_0) + \tilde{u}_0 L_{rotmin,\tilde{w}_0}\mathbf{p}'. \qquad (4.26)$$

By allowing a muscle shape designer to interactively modify the origin, insertion and axial curves, we can completely specify a new B-spline solid which interpolates these profile curves. This technique allows the larger number of control points of the solid to be completely specified by a fewer set of points that make up the curve. In Chapter 5, we will show how the CVSF can be used to generate stable physically-based motion as well as dynamical effects not possible with static deformation. In the absence of physical simulation, profile curves can be considered a deformation technique to shape the solids because the points of the solid are referenced and transformed relative to a local coordinate system based on the profile curves.

4.7 Summary

In order to define an initial configuration of control points for a shape, we have developed a general methodology that can create data fits for exactly-, under-, and over-determined systems. The key idea is to generate samples from data-sets by creating a continuous volume sampling function (CVSF). The CVSF can subsequently be used to generate an arbitrary number of samples referenced by a well-defined parameter coordinate system. The samples are used to solve either a linear system or a least-squares problem for the control points that create the B-spline solid that fits the data.

Once an initial shape for the solid has been created, the geometry can be deformed several ways. The CVSF can be modified to create slightly different samples and the data-fitting process repeated to create a modified solid, as in the case with profile curves. However, physically-based methods are attractive techniques for generating automatic, realistic motion, creating shape responses to other objects during collision, and providing an important bridge for functional simulations of muscle. We develop several methods for physically-based B-spline solids in the next chapter.

Chapter 5

The Physical Musculotendon Model

> Muscles constitute the contractile or power system; they produce action by their contraction or shortening. In contraction they are lifted and bulged, while in their relaxed state they are flabby and soft.
>
> **George B. Bridgman in *Constructive Anatomy, 1920***

In Chapter 4, we described data-fitting methods to define initial muscle shapes for B-spline solids. We would like to embed dynamic behaviour into these solids for visualization of muscle contraction, volume-preserving deformations and collision resolution with other parts of the musculoskeletal system.

5.1 System definition

From a simulation perspective, the *system* represents the physical object or collection of objects that we wish to describe the motion of. It must be described completely by a *state vector* that is referred to as a the *degrees of freedom* of the system. The state vector consists of the generalized coordinates, $\mathbf{q}$, that describe the system and their first derivatives, $\dot{\mathbf{q}}$, with respect to time, the *generalized velocities*.

We can have different perspectives of the physical system. When studying musculotendon units in isolation, the system is the B-spline solid defined by its control or spatial points. In cases where a muscle is comprised of several architecturally distinct regions, the system definition expands to include all the B-spline solids that are part of the muscle tissue. For example, the human soleus would form a system of three B-spline solids, one for each of the posterior, anterior and marginal portions (Figure 5.1).

In the context of a musculoskeletal system, the degrees of freedom of the joints of the bones can be added to the state vector. However, we partition the musculotendon and skeleton state variables for modularity and conceptual simplicity. We can isolate the equations of motion for the degrees of freedom of the rigid, articulated skeleton by considering the musculotendons as external force actuators, that apply external forces to the skeletal system.

This separation of the musculotendon state from the skeletal state allows us to work with an isolated set of equations of motion for individual B-spline solids. For example, we can use an implicit integration scheme for the B-spline solid if numerical stiffness exists in the

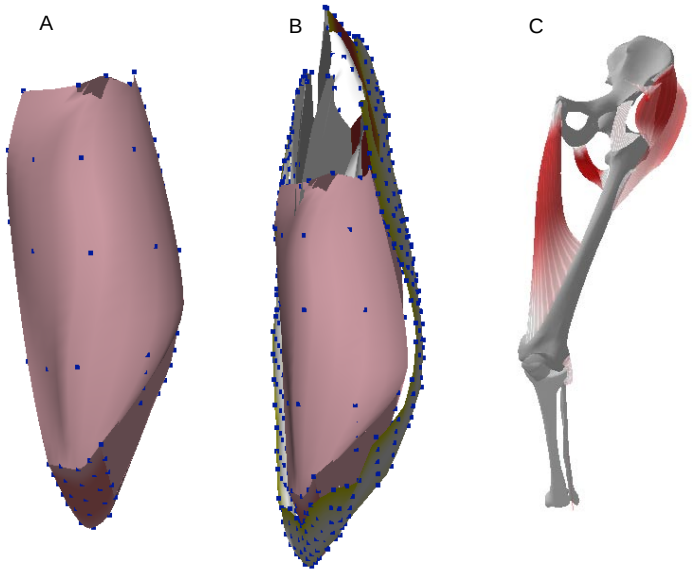


Figure 5.1: Different system perspectives for simulation purposes: A) Single portion of muscle, B) Multiple architecturally-distinct regions of muscle, C) Musculoskeletal system.

musculotendon's equations of motion while using an explict integration scheme on the rest of the skeleton.

5.2 Equations of motion

In Chapter 2, we reviewed the Lagrangian formulation for developing equations of motion for a system. The Lagrangian equations of motion are expressed in terms of the degrees of freedom of a system. The formulation accommodates the incorporation of energy functions, constraints and external forces, providing a complete framework for embedding physical, dynamic behaviour into the geometric B-spline solid model.

5.2.1 Lagrangian formulation for B-spline solids

We restate the Lagrangian equations of motion as:

$$\frac{d}{dt}\frac{\delta T}{\delta \dot{q}_i} - \frac{\delta T}{\delta q_i} = Q_i - \frac{\delta V}{\delta q_i}, i = 0, \ldots, n-1, \tag{5.1}$$

where q_i are the generalized coordinates and are either control or spatial points. T and V are kinetic and potential energy respectively. Q_i is a generalized force that is applied to the generalized coordinates. Various potential functions, V, and external forces Q_i are designed for desired physical behaviour. The vector $\mathbf{q}$ is formed by concatenating the coordinates of n points so that the i^{th} point has its coordinates located at q_{3i}, q_{3i+1}, q_{3i+2}, $i = 0, \ldots, n-1$. For the i^{th} control point, $\mathbf{c}_i$, or spatial point, $\mathbf{s}_i$ we can convert its x, y, and z components to the positions in the monolithic $\mathbf{q}$ vector as follows:

$$\mathbf{c}_i = (c_i^x, c_i^y, c_i^z) = (q_{3i}, q_{3i+1}, q_{3i+2}) \tag{5.2}$$

or

$$\mathbf{s}_i = (s_i^x, s_i^y, s_i^z) = (q_{3i}, q_{3i+1}, q_{3i+2}). \tag{5.3}$$

In subsequent discussion, the subscript of a function or gradient operator denotes the choice of generalized coordinate the function is expressed in, or the gradient operator is in terms of, respectively. For example, ∇_s is the gradient with respect to spatial coordinates, while V_c is the volume function expressed in terms of control points, so that $V_c \equiv V_c(\mathbf{c})$ is implicitly understood. In the absence of any subscript, either generalized coordinate system can be assumed.

5.3 Spatial points as generalized coordinates

Many external forces are easier to conceptualize and define in the spatial domain. To formulate the Lagrangian equations of motion with respect to spatial points, we need to represent all forces and energies as functions of the spatial points and their velocities. External forces can be directly applied to spatial points as they are expressed with the same generalized coordinates.

Suppose we have n spatial points. If we associate a mass m_i with each spatial point, $\mathbf{s}_i$, the total kinetic energy of all the points is:

$$T = \sum_{i=0}^{n-1} \frac{1}{2} m_i \dot{\mathbf{s}}_i^2. \tag{5.4}$$

This is equivalent to the matrix equation,

$$T = \dot{\mathbf{s}}^T M \dot{\mathbf{s}}, \tag{5.5}$$

where $\dot{\mathbf{s}}$ is the vector formed by concatenating all the spatial velocities of each point. The $3n \times 3n$ diagonal matrix M has the following diagonal entries:

$$m_0, m_0, m_0, m_1, m_1, m_1, \ldots, m_{n-1}, m_{n-1}, m_{n-1}. \tag{5.6}$$

Referring to the general form of the Lagrangian equation[72], we can express equations of motion in terms of spatial coordinates as

$$M\ddot{\mathbf{s}} = \mathbf{f}_s - \nabla_{\mathbf{s}} V_s + \mathbf{f}_d, \tag{5.7}$$

where $\mathbf{f}_s$ is a vector of external spatial forces and $\nabla_{\mathbf{s}} V_s$ is the gradient vector of a potential energy or work function V_s with respect to $\mathbf{s}$. A damping force, $\mathbf{f}_d$, proportional to the velocity of the spatial points, $\dot{\mathbf{s}}$, is applied to dampen oscillations and simulate energy loss caused by phenomena such as internal friction in the musculotendons. We will provide examples of $\mathbf{f}_s$ and V_s in subsequent sections. During numerical integration, we solve for the acceleration $\ddot{\mathbf{s}}$, in Equation 5.7. As the mass entries will stay constant during our physical simulations, the entries of the matrix M are constant and the changing values occur only on the right-hand-side of Equation 5.7. Once $\ddot{\mathbf{s}}$ is found, successive integrations produce the updated spatial positions and velocities.

5.4 Control points as generalized coordinates

By examining the B-spline solid equation (Equation 3.1), we can think of the control points, $\mathbf{c}$, as parameters that describe the deformation of all points in the solid. In order to introduce dynamics, we need to embed mass into the solid, associating mass to each point in the solid.

Witkin and Welch[135] showed that by representing the deformations as linear functions of the generalized coordinates $\mathbf{q}$, Lagrangian equations of motion can be formulated that can be computed rapidly. In particular, we can assign mass points to selected material coordinates. For B-spline solids, we already have a convenient set of material coordinates, $\mathbf{u}_i = \mathbf{u}_{max,i}$ (see Section 3.5.3) that correspond to the spatial points $\mathbf{s}$. Therefore, we assign a mass m_i to each of the spatial points $\mathbf{s}_i, i = 0, \ldots, n-1$.

To work with control points, $\mathbf{c}$, as generalized coordinates, we need to convert between the generalized control point velocities and spatial point velocities and between the generalized control points forces and spatial forces. For a given spatial point, $\mathbf{s}_i$, with material coordinates $\mathbf{u}_i$, $\mathbf{s}_i$ can then be treated as a function of the control points, $\mathbf{s}_i(\mathbf{c})$. This function allows us to relate the spatial velocity, $\dot{\mathbf{s}}_i$ to the generalized velocity of the control point, $\dot{\mathbf{c}}$:

$$\dot{\mathbf{s}} = J_{\mathbf{c}}^{\mathbf{x}}(\mathbf{u}_i)\dot{\mathbf{c}}, \tag{5.8}$$

where $\mathbf{x}$ refers to the B-spline solid equation (Equation 3.1) and the entries of the Jacobian matrix, $J_c^x(\mathbf{u}_i)$ are the partial derivatives of $\mathbf{x}$ taken with respect to the control point coordinates, c evaluated at the material coordinates, $\mathbf{u}_i$.

The spatial forces can be converted to a generalized force acting on control point coordinates through the following relation[135, 140]:

$$\mathbf{f}_{\mathbf{c}} = (J_{\mathbf{c}}^{\mathbf{x}}(\mathbf{u}_i))^T \mathbf{f}_{s_i}, \tag{5.9}$$

where $\mathbf{f}_{s_i}$ is the spatial force applied to point $\mathbf{s}_i$ and $\mathbf{f}_{\mathbf{c}}$ is the corresponding generalized force. As $J_{\mathbf{c}}^{\mathbf{x}}$ is a sparse matrix due to the local properties of B-splines, $\mathbf{f}_{\mathbf{c}}$ will have relatively few non-zero values corresponding to the control points that will be affected by the spatial force.

The next sections will describe various potential energy forces and external forces that are modelled or applied to the physically-based B-spline solids. The choice of whether to use spatial points or control points as generalized coordinates depends on which set of coordinates can be used to more naturally express various constraints with minimal computation.

5.5 Volume-preserving potential forces

In Section 3.4, we introduced an expression of the volume of a B-spline solid in terms of control points (Equation 3.17), $\mathcal{V}_c$. A volume-preserving potential function can be created by introducing a rest volume, $\mathcal{V}_0$, that is the desired volume to hold constant:

$$V_{volume}(\mathbf{c}) = \frac{k_{volume}}{2}(\mathcal{V}_c - \mathcal{V}_0)^2, \tag{5.10}$$

where V_{volume} is the potential energy for volume preservation and $\mathcal{V}_c$ is the volume formula expressed in terms of $\mathbf{c}$. Referring to the Lagrangian equation of motion, the corresponding

volume-preserving forces are:

$$\nabla_c V_{volume}(\mathbf{c}) = k_{volume}(\mathcal{V}_c - \mathcal{V}_0)\nabla_c \mathcal{V}_c, \tag{5.11}$$

where ∇_c is the gradient vector of the volume potential function, $V_{volume}(\mathbf{c})$, with respect to the coordinates of $\mathbf{q}$. If we are working with spatial coordinates, $\mathbf{s}$, we can express the volume-preserving forces with respect to $\mathbf{s}$ directly in terms of $\mathcal{V}_c$ and $\nabla_c \mathcal{V}_c$:

$$\nabla_s V_{volume}(\mathbf{s}) = k_{volume}(\mathcal{V}_s - \mathcal{V}_0)\nabla_s \mathcal{V}_s, \tag{5.12}$$

with $\mathcal{V}_s$ computed by performing a coordinate transformation, $\mathbf{c} = \beta^{-1}\mathbf{s}$, and evaluating the volume with $\mathcal{V}_c$, and

$$\nabla_s V_s = (J_s^c)^T \nabla_c V_c, \tag{5.13}$$

where J_s^c is the Jacobian matrix of $\mathbf{c}$ expressed in terms of $\mathbf{s}$. If we were to represent $\mathbf{c} = \beta^{-1}\mathbf{s}$, so that $\mathbf{c}$ and $\mathbf{s}$ are one-dimensional monolithic vectors with all the point coordinates concatenated together, the matrix β^{-1} would correspond exactly to J_s^c.

The matrix β^{-1} is computed one column at a time by reusing the LU factors of β to solve the following n successive systems to find each column of β^{-1}:

$$\beta\gamma_j = \mathbf{i}_j, j = 0, \ldots, n-1, \tag{5.14}$$

where γ_j is the j^{th} column of β^{-1}, $\mathbf{i}_j$ is the i^{th} column of the identity matrix, $I_{n\times n}$, and n is the size of the monolithic vectors, $\mathbf{c}$ and $\mathbf{s}$.

5.5.1 Relationship between volume and deformations

Terzopoulos et al.[114] introduced methods of defining energies of deformation using the symmetric matrix G, where the matrix entries are:

$$G_{ij}^u(\mathbf{u}) = \int \frac{\delta \mathbf{x}}{\delta u_i} \cdot \frac{\delta \mathbf{x}}{\delta u_j} dV, \tag{5.15}$$

where V represents the volumetric domain of the material coordinates. This matrix is known as the *metric tensor* or *first fundamental form*. If two solids have the same metric tensors, they must have the same shape. In contrast, lower-dimensional geometric forms such as surfaces and curves require more conditions for two shapes to be declared identical[114]. Potential functions describing energy deformations use the metric tensor to create scalar functions by approximating norms of the deviation of the metric tensor to a reference shape matrix. The metric tensor can be thought of characterizing the relative deformations of the infinitesimal axes along each material coordinate of the B-spline solid equation, $\mathbf{x}(\mathbf{u})$ (Equation 3.1)[17].

The volume-preserving formula is related to the metric tensor and therefore is closely related to shape deformation characteristics. There is a direct relationship between the volume of a B-spline solid, $\mathcal{V}$, and its metric tensor G[99]:

$$\mathcal{V}^2 = \det(G). \tag{5.16}$$

However, as our volume is only computed over the solid's boundary, we do not have to integrate over the entire volumetric domain, as in the case with the metric tensor. In contrast to using a

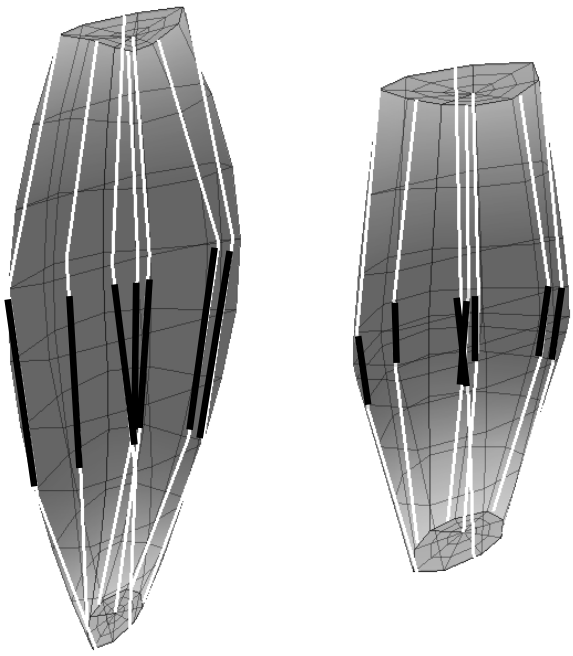


Figure 5.2: Volume-conserving motions: Viscoelastic links embedded in the solid are contracted (black line segments) while simultaneously maintaining constant volume.

rest shape matrix as in [114], our B-spline solid shape is free to change its shape as long as it keeps the same volume. This is important for simulating contracting muscles where the shapes change while preserving volume. Muscles have been shown in classic experiments to preserve volume[81].

Unfortunately, no force will be generated on internal points because the gradient vector of the volume-preserving energy function (Equation 5.11) has zero components corresponding to the internal points. This can be explained by observing that the volume is defined by only its boundary points and is independent of the internal point configuration. Therefore, we must design alternative forces to specify the internal point configurations over time. For the internal points to be volume-preserving,

$$|J_u^x(\mathbf{c})| = 1, \tag{5.17}$$

where $J_u^x(\mathbf{c})$ is the Jacobian matrix of the B-spline solid equation's (Equation 3.1) partial derivatives with respect to the material coordinates, $\mathbf{u}$, evaluated with the control points, $\mathbf{c}$. Equation 5.17 is a nonlinear constraint that the control points, $\mathbf{c}$, must satisfy. Multiple nonlinear constraints for each spatial point inside the solid can be prohibitively expense to solve. The fact that volume is dependent only on the boundary allows other less computationally-expensive methods to determine the internal configuration of points.

5.5.2 Stiffness coefficient for volume forces

The stiffness term k_{volume} controls the strength of the attraction towards the target volume. Larger values of k_{volume} create a more narrow tolerance range for the target volume. A large k_{volume} value or a large component of ∇V may create stiff differential equations requiring the

use of implicit integration techniques[100] or smaller timesteps for simulation. We have implemented a backward Euler integration technique and were able to achieve numerical stability with larger timesteps (several orders of magnitude larger), but with increased computational cost to compute force derivatives with respect to the generalized coordinates and velocities.

In order to use explicit integration techniques, we normalized the gradient vector coordinates by dividing each entry by the maximum absolute value of all components. This prevents excessively large gradient vector components that can cause numerical instability, while still allowing the use of k_{volume} to modulate the potential energy's attractive strength. Although the gradient vector has been scaled, corresponding generalized coordinates are perturbed in the right direction, albeit with smaller step lengths. Using k_{volume}, we can control the tolerance of error from the target volume during simulations.

The use of volume-preserving potential forces, allows us to influence the motion of generalized coordinates so that volume is maintained (Figure 5.2). However, there is no consideration given to other physical properties such as local viscoelastic material characteristics in the solid. Other physical forces need to be modelled in conjunction with the volume potential force to constrain shape displacements of the solid. This motivates the technique of embedding viscoelastic networks in the B-spline solid.

5.6 Viscoelastic networks

A viscoelastic network is a collection of mass points interconnected with links. Equal and opposite viscoelastic forces are applied on the two points connected by a common link (Figure 5.3). The forces within a link can be generated by various elements such as springs, dampers and other nonlinear tension generators such as contractile units used for modelling the active forces developed in muscles. The vector sum of all link forces on a point determine the direction of application of these forces.

Chadwick et al.[16] established a spring-mass network over the control points of free-form deformations (FFDs), allowing the embedded geometry to display propagated physically-based motion. The control point lattices were fairly regular in shape compared to the control point lattices of B-spline solids. As the link lengths are less homogeneous in an arbitrary B-spline solid compared to a rectangular FFD, the use of control points for a viscoelastic network in a B-spline solid shape is not as convenient. A viscoelastic network between spatial points allows links to correspond to spatial constraints between solid points. Control points can be moved arbitrarily from the solid's boundary surface, so that their locations can have little correspondence to spatial relationships.

By allowing a designer to vary force magnitude models in the network, nonuniform physical effects can be embedded in the solid. For example, using a model of human soleus muscle, we can assign links in the top surface to have elastic effects to approximate the aponeurosis tendon material. Within the same solid, links running along fibres can be modelled with springs of different stiffnesses or more sophisticated Hill-based muscle force models. Figure 5.4 illustrates this arrangement.

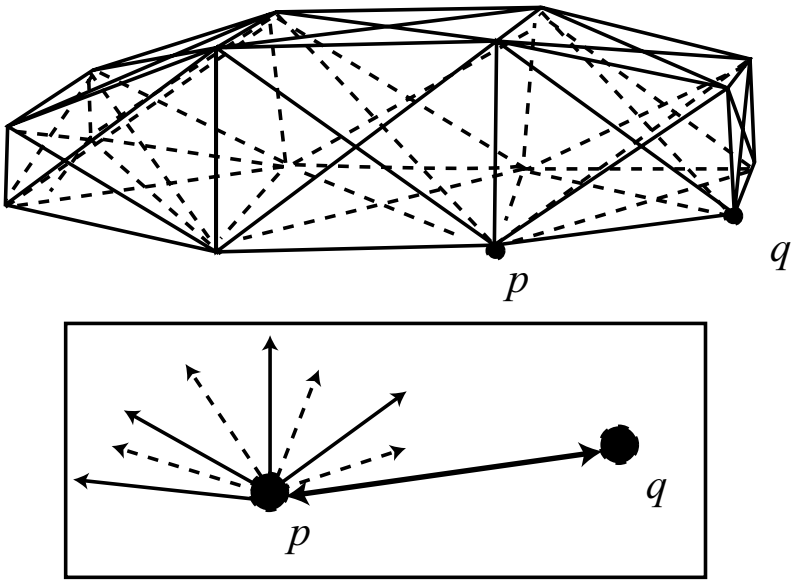


Figure 5.3: Viscoelastic networks: Mass points are interconnected by links that generate equal tension between two points. Inset: The individual forces on each point are summed to apply the resultant force at that point. Points p and q generate equal tension so that the pair of points shared by a link experience equal and opposite forces.

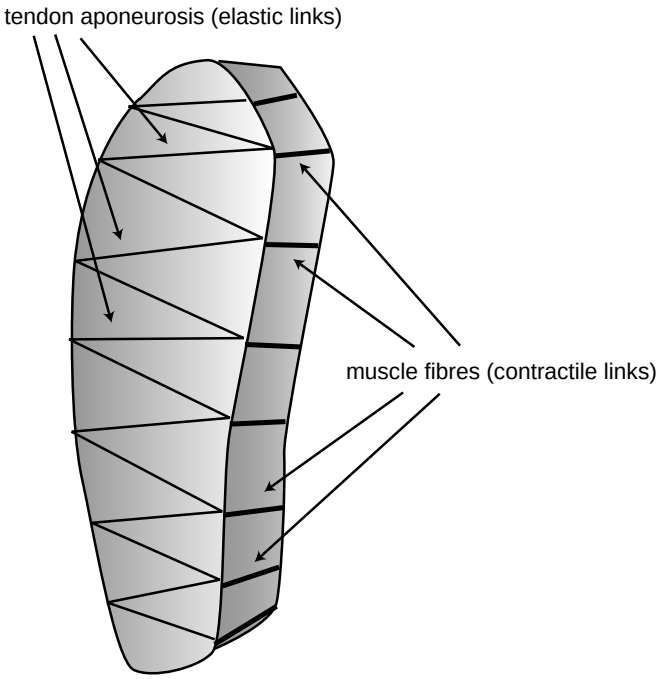


Figure 5.4: Nonuniform physical properties within a musculotendon: The top surface of a solid can have elastic properties while fibres within the solid can demonstrate nonlinear contractile forces.

5.6.1 Link force models and control inputs

The choice of force model for each link depends on the desired application and goals of simulation. If we are given a time series of fibre lengths obtained from experimental measurements, a spring can be used to contract the links to a desired length. The rest length of the spring becomes the control input which would be data-driven. Figure 5.2 shows a series of images produced by contracting links embedded in a human soleus model derived from a fibre set CVSF (see Section 4.5).

If the objective is to simulate fibres using neural excitation signals as the control input, a more suitable model is a Hill-based model where fibre characteristics such as series, parallel and contractile elements can be used to model fibre forces. Refer to Section 2.2.5 for a review of different mathematical models for these elements.

The advantages of using Hill-based models over a linear spring are the ability to perform forward dynamics experiments with a higher degree of accuracy. The published research on skeletal muscle provides parameter values for Hill-based elements that can be matched for the muscle under study[137]. Using Hill-based models, we can further differentiate between linear, elastic tendon fibres and the nonlinear force-generating properties of muscle fibres. The only control input needed for simulation would be a single neural signal. The same Hill parameters can then be used with a variety of different motions instead of being tuned for just one special case.

We have experimented with creating deformable models from solids constructed from Visible Human data sets and fibre sets using viscoelastic units. Solids extracted from the Visible Human data set are already in a deformed state because the muscles were captured *in situ*, packed against muscles, bones and other soft tissues. Using a spatial viscoelastic network, we averaged the radial and longitudinal lengths of links, setting the average link length to be the rest length. The resulting solid with the embedded viscoelastic network produces a muscle model with radial symmetry along its axis, for an approximation of an isolated, undeformed muscle rest shape (Figure 5.5).

The estimate of the true rest shape from Visible Human data may only be helpful for gross shape modelling, as the true rest shape is probably a function of the fibre architecture. In Chapter 4, we stated that the Visible Human data lacks architectural information, and therefore detailed fibre set data must be used for more accurate physical modelling.

5.6.2 Composition of forces

Using the Lagrangian dynamics framework, the viscoelastic network forces and volume potential forces can act together on the same system.

Figure 5.6 illustrates how we have simulated a contracting soleus muscle with and without volume conservation. An interesting observation was that the fibres changed their orientation as they shortened when both volume preservation and contracting forces were present. Without the volume-preservative forces, these fibres kept the same orientation.

Ultrasound imagery studies[77] of contracting human soleus *in vivo* confirm that the fibres do change their pennation in a direction similar to the virtual experiments performed. This example illustrates how simulated experiments can help to diagnose and isolate potential causes of observed phenomena.

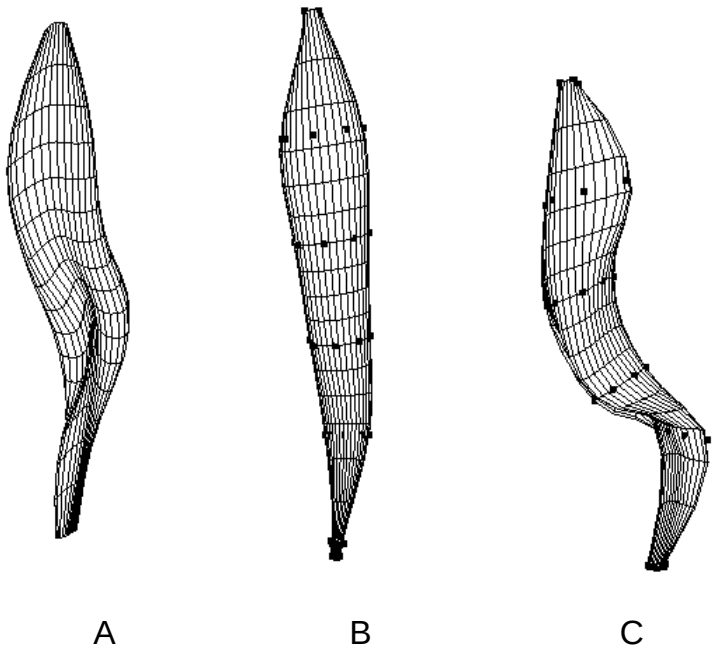


Figure 5.5: Computation of muscle rest shapes: A) B-spline solid constructed from Visible Human contour data. B) Rest shape computed by averaging link rest lengths radially and for each longitudinal section. C) Subsequent re-deformation of rest shape.

The introduction of global volume-restorative forces can add structural stability to muscle shape. In previous work with spring-mass systems[83], cross-links are often added to prevent shearing and collapsing of shape. One problem of using viscoelastic networks in isolation are that solid matter is approximated with a discrete set of mass points and weightless links. The absence of matter between the mass points provides a lack of restorative forces that should normally exist in a solid continuum, especially in an incompressible solid (Figure 5.2). The addition of volume-restorative forces can replace the role of the cross-links for structural stability. We need only provide a few links to bias the directions of shape change during contraction and let the global volume forces determine the corrective displacements of points to maintain a volume-conserving shape.

5.7 Anchored Fields

For applications where accuracy is not crucial, we may want to produce physically-based local deformations in a solid and need tools to simultaneously influence many generalized coordinates at once. We designed a simple modelling tool called the *anchored field* that creates a local, spherical deformation field by specifying the radius of influence of a field and constraining the distance the point source of the field can travel with respect to an anchor point (Figure 5.7). A spring-damper system exists between the anchor point **a** and the field source **p** which has a mass of m:

$$m\ddot{\mathbf{p}} = k(\mathbf{a} - \mathbf{p}) - d\dot{\mathbf{p}} \tag{5.18}$$

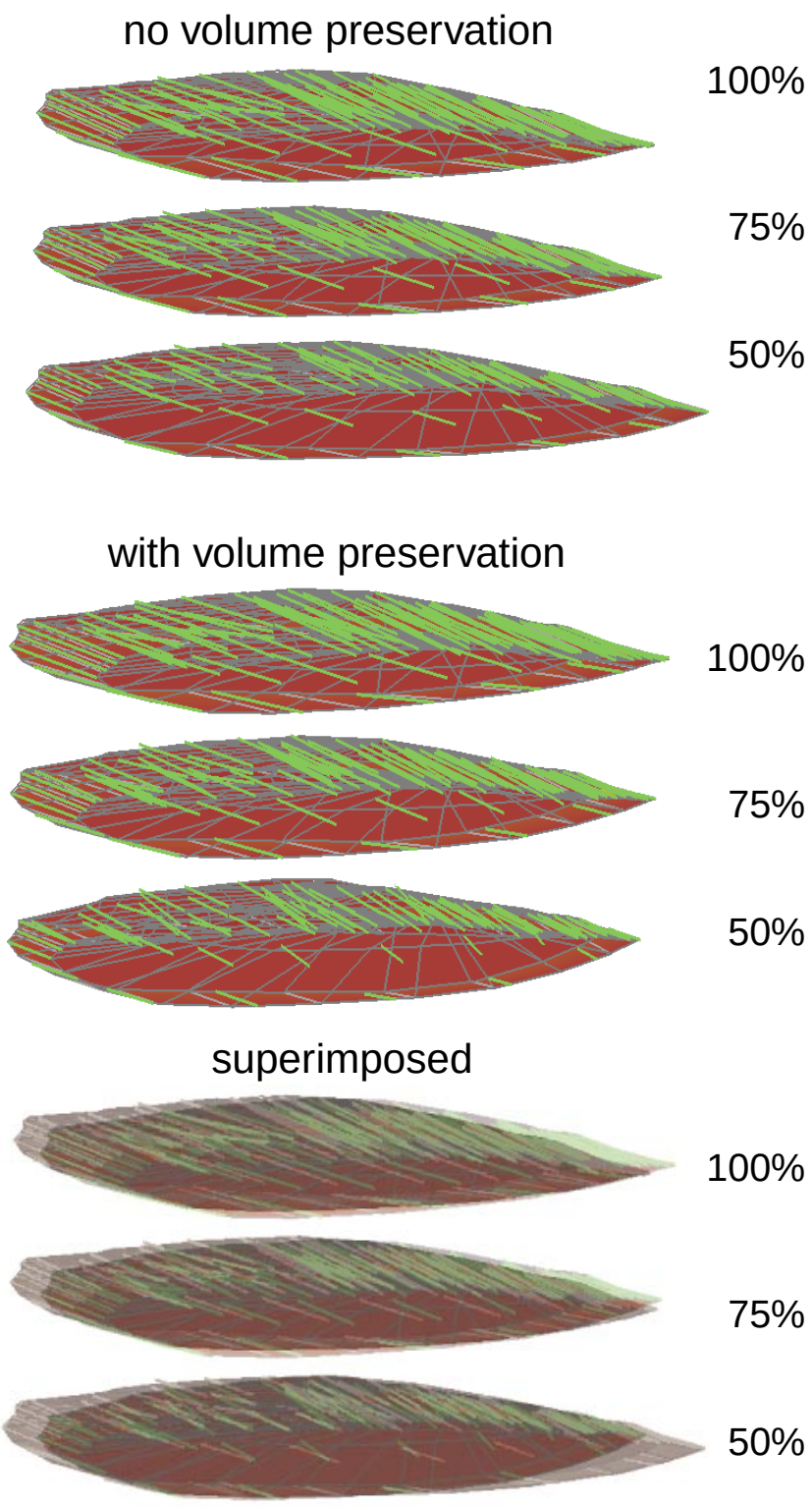


Figure 5.6: Muscle contraction without (top image) and with (middle image) volume preservation. The top image (light) is superimposed on the middle image (dark) for comparison (bottom image). Selected fibre links connecting spatial points are shown at 100%, 75%, and 50% contraction. Notice that in the case with volume preservation, the fibre orientations change due to volume-preserving forces, especially at 50% contraction.

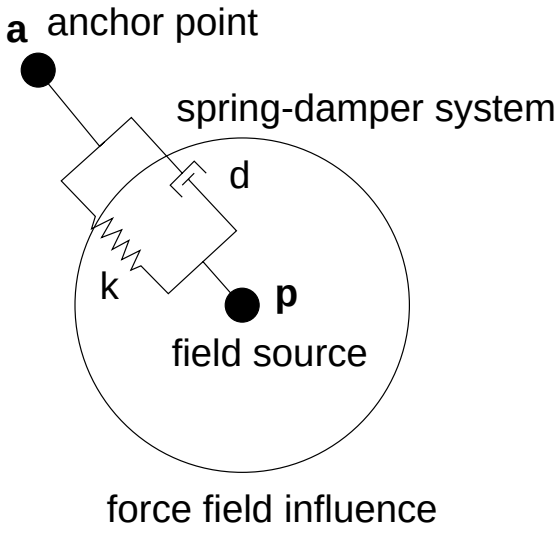


Figure 5.7: The anchored field.

The interaction of the mass, m, and the spring, k, and damping, d constants dictates whether the oscillation behaviour between the two points is underdamped, overdamped or critically damped:

$$\begin{array}{ll} d^2 - 4mk < 0 & \text{underdamped} \\ d^2 - 4mk > 0 & \text{overdamped} \\ d^2 - 4mk = 0 & \text{critically damped.} \end{array} \tag{5.19}$$

The deformation field is applied to all spatial points (or control points) in the solid that resides under the radius of influence of the anchored field. Once a point leaves the field's influence, it reverts back to its original position.

A particular point, $\mathbf{q}_i$, will be displaced by a vector, $\mathbf{d}_i$,

$$\mathbf{q}_i^{new} = \mathbf{q}_i^{old} + \mathbf{d}_i, \tag{5.20}$$

where

$$\mathbf{d}_i = \frac{\mathbf{q}_i^{old} - \mathbf{p}}{\|\mathbf{q}_i^{old} - \mathbf{p}\|}(radius - \|\mathbf{q}_i^{old} - \mathbf{p}\|). \tag{5.21}$$

The formulation of $\mathbf{d}_i$ has the effect of perturbing the point $\mathbf{q}_i$ to the boundary of the deformation field. The further the point $\mathbf{q}_i$ lies from the field source $\mathbf{p}$, the less displacement it will experience until the displacement disappears altogether at length $radius$ from $\mathbf{p}$. This ensures that the points smoothly transition back to their original positions as they gradually leave the field's influence.

During simulation, the motion of $\mathbf{p}$ is simulated using Equation 5.18 to generate damped oscillations as the field deforms the solid. The user can interactively change the radius of the field, the relative positions of the anchor point and field sources, or the spring-damper characteristics of the connection between the two points (Figure 5.8). We have created variants of the anchored field that constrain the position of the anchor point $\mathbf{a}$ to iso-curves within the solid. In particular, we can constrain the anchor point to stay on the axis of the B-spline solid to move the field source in the solid's natural coordinate system. This effectively reduces the

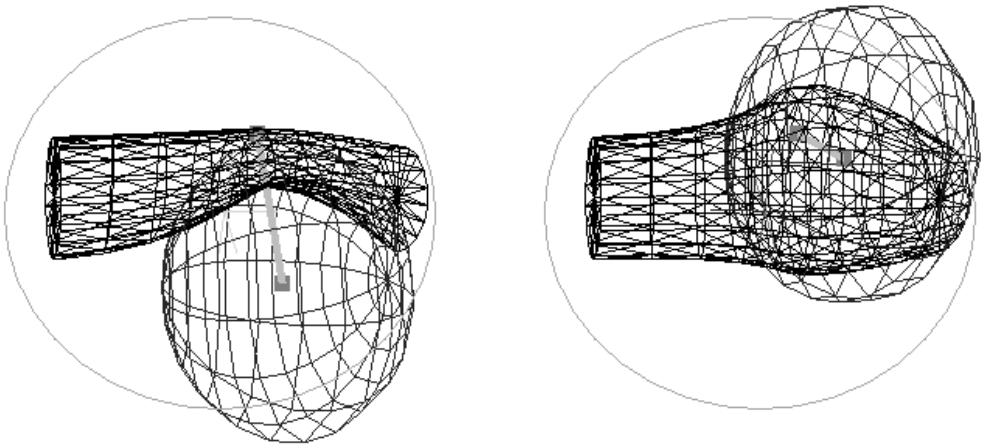


Figure 5.8: The anchored field being used to model a B-spline solid.

degrees of freedom of the tool to a single meaningful slider that is customized to the solid's shape and simplifies the three-dimensional use of the tool.

The key differences of anchored field and free form deformations (FFDs) are in the domain of influence of each tool. The user has control over the radius of influence and the strength of the field using anchored fields. For FFDs, the entire object is constrained to be embedded within the lattice at all times. To change the range of influence of an FFD control point on the deformation of its internal geometry, the lattice must be rebuilt with different basis functions or the lattice points must be readjusted and the internal geometry remapped with respect to the new lattice. As B-spline solids are tensor-product volumes, anchored fields can also be applied to FFD lattice points.

5.8 Profile-curve driven dynamics

Volume-restorative forces, viscoelastic networks and anchored fields offer methods to constrain the deformation of shape. During numerical simulations, the use of viscoelastic networks can become problematic if the muscles are attached to skeletal limbs undergoing fast, transient motions. During these quick motions, it is important that the muscle's shape remains structurally stable yet still have enough compliance to exhibit inertial effects such as damped vibrations.

Terzopoulos and Witkin[116] developed a hybrid formulation that models objects using a rigid reference component and a deformable component. The reference positions are expressed relative to a local body frame which experiences rigid body dynamics. Using the same idea, we can connect a spring between an anchor point, $\mathbf{a}$, and its assigned spatial point in $\mathbf{s}$:

$$\mathbf{f}_s = k_{anchor}(\mathbf{a}_i - \mathbf{s}_i), \tag{5.22}$$

where the subscript i denotes a particular pair of $\mathbf{a}$ and $\mathbf{s}$. The anchor position $\mathbf{a}$ will be redefined as the solid's volume changes and underlying limbs undergo rigid body motion. By adjusting the stiffness and damping, we can respectively control how quickly the solid responds to external movement and how quickly oscillations subside.

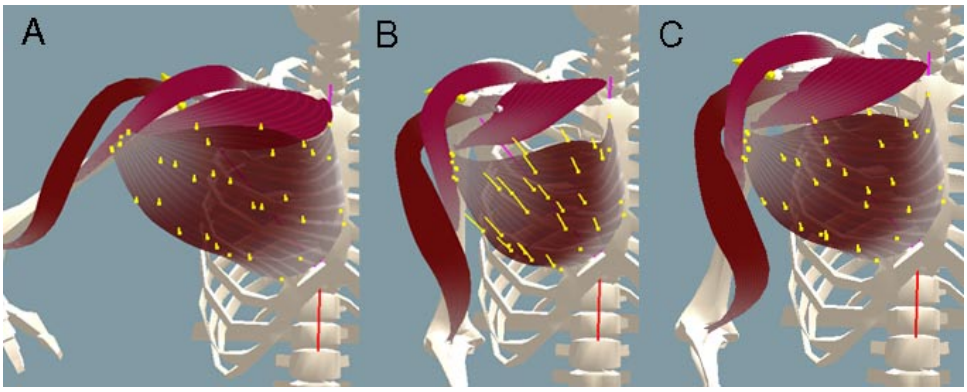


Figure 5.9: Downward arm motion causes inertial-induced "jiggles" in *pectoralis major* muscle (from A-C). Mass points and restorative forces are in yellow.

Equation 5.22 provides a restorative force for each spatial point. During fast transient motions, there will be an initial displacement of the spatial point that will create a restorative spring force to drive the point back toward the anchor point. This allows inertial effects to be visualized in musculotendons attached to a skeleton animated using dynamic simulation or kinematic manipulation of the joints. In contrast, simulating a sequence of static equilibrium problems for each frame of an animation would not exhibit inertially-induced oscillations – the "jiggles"[1].

Calculation of Anchor Positions

We would like the solid to change as the underlying skeleton moves, changing the position and orientation of the attachment areas as well as the musculotendon's length. A natural source for an initial anchor position is through the use of the profile curves' $CVSF$ (see Section 4.6):

$$\mathbf{a}_i = CVSF(\tilde{u}_i, \tilde{v}_i, \tilde{w}_i). \tag{5.23}$$

The parameters $(\tilde{u}_i, \tilde{v}_i, \tilde{w}_i)$ correspond to the normalized material coordinates of the spatial points $\mathbf{s}_i$.

Profile curves consist of significantly fewer defining points than the entire solid, providing a high-level interface for shape definition as opposed to the manual manipulation of the control and sample points. We constrain the origin and insertion curves to be rigidly attached to their bones as well as the end points of the axial curve.

For the axial curve, there are two alternatives. The axial points on the curve can be associated with the closest bone and expressed in the bone's frame of reference. This may be suitable if the axes do not undergo large displacements on changes in configuration during joint movements. However, the axial curve will only statically deform as the underlying bones change position.

Alternatively, we can treat the points of the axial curve as mass points and allow the curve to have its own dynamic behaviour. This allows independence of the axial dynamics from the

[1]This term is attributed to Caleb Howard, digital artist.

underlying skeleton. For example, the curve can vibrate in tension during a static pose in the skeletal limbs. The axial curve can also sag under gravity, creating a corresponding effect in the global shape of the B-spline solid. The dynamics of the axial curve can create overall global changes in shape which are layered with local vibrations created from Equation 5.22.

5.9 Reaction constraints

In order to control interpenetration of spatial points in the solid with external objects, we need a mechanism to apply local constraints on these points. We can constrain their behaviour using a technique developed by Platt and Barr[99] called *reaction constraints*. We briefly review the main concepts. The idea is to compute the net force, F_{output}, acting on a point as a sum of unconstrained and constrained force components,

$$F_{output} = F_{unconstrained} + F_{constrained}. \tag{5.24}$$

Given the current sum of all external and potential-derived forces in F_{input}, the force is examined and the valid forces that do not violate a desired constraint are projected out in $F_{unconstrained}$. Another force, $F_{constrained}$, is added to critically damp the point to the constraint surface.

The constrained force is computed as,

$$F_{constrained} = k\mathbf{d} + c\frac{d}{dt}\mathbf{d}, \tag{5.25}$$

where $\mathbf{d}$ is a vector (or scalar) describing the deviation of the spatial point from the constraint surface. The coefficients k and c are the strength of the constraint and damping. If $c = \sqrt{2k}$, critically-damped motion occurs[99]. The constraint force is similar to the use of Baumgarte constraint stabilization used in simulation software[57] to correct for numerical drift that can eventually violate constraints during simulation.

The final force, F_{output} is applied to the spatial point, replacing the original F_{input}. It is important that the corresponding reaction force,

$$F_{reaction} = F_{input} - F_{output}, \tag{5.26}$$

is applied to the object interacting with the B-spline solid at the spatial point to obey Newton's third law[2].

5.10 Collision forces

Musculotendons are part of an organic machine made up of an assembly of bones and other musculotendons. Consequently, we need to detect and compute the physical reactions of musculotendons at bone-muscle and muscle-muscle contact regions. Our approach for collision detection and resolution is a compromise between an exact computation for every point on the B-spline solid's surface and the other extreme of no interpenetration prevention at all. We monitor collision conditions with bone and muscle only at the boundary spatial points, $\mathbf{s}$. If collision conditions are met, we apply a local point-to-plane constraint on that spatial point.

[2]For every action, there is an equal and opposite reaction

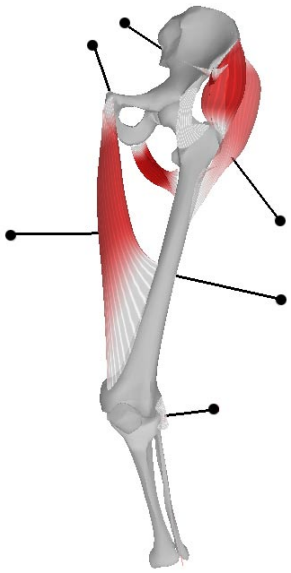


Figure 5.10: The closest bone or musculotendon to an external point can be quickly found.

5.10.1 Closest Point Detection

For a given spatial point on a B-spline solid, we need to find the closest candidate point on a nearby object where contact can occur. Johnson and Cohen[66] developed an algorithm framework for minimum distance computation using lower-upper bound trees (LUB-trees). The advantage of their approach is that objects are not limited to being convex and the framework can be used with a heterogeneous set of geometric representations. We use LUB-trees to perform proximity queries of arbitrary spatial points to a set of objects, consisting of either polygon bone meshes or B-spline solids for musculotendons (Figure 5.10). Details of how the minimal distance algorithm works with the framework can be found in [66]. We briefly summarize the approach.

Each geometric object (bone or musculotendon) is treated as a top-level node, where each top node is the root of a tree containing a hierarchy of bounding volumes. Lower and upper bounds on the distance of the point to each volume are computed. As the algorithm proceeds, we can progressively remove a candidate volume from the search list whenever its lower bound is found to be greater than another volume's upper bound distance. When no more volumes can be removed, we can further refine existing volumes and repeat the process until only one unrefinable volume remains.[3] This volume contains the closest point.

We define a procedure, $[B_1, B_2] = refine(B)$, that takes the bounding volume, B, and refines it into two smaller subvolumes, B_1 and B_2 that are each smaller than B and partition the original contents of B. If the volume can no longer be refined, $[\,] = refine(B)$, this indicates that the volume contains either a B-spline solid or a single triangle from a polygonal mesh.

We seek to find the minimal distance between an arbitrary point $\mathbf{p}$ and a collection of geometric objects. Since the algorithm deals with bounding volumes, it does not assume that

[3] In the event there is more than one remaining volume, we choose the first volume in the list.

the volumes are part of the same object. Therefore, we can pool the bounding volumes for several bones and musculotendons together to create the set of candidate objects, O, to find the closest point to $\mathbf{p}$. We introduce the function

$$[objectID, localpt] = FindClosest(\mathbf{p}, O), \tag{5.27}$$

where $\mathbf{p}$ is a given point in world coordinates and the function returns a unique object identifier, $objectID$, and a local coordinate, $localpt$, that is expressed relative to the object's coordinate system. For bone geometry, the local coordinates are relative to the frame of reference of the geometry which originates on the centre of mass and is aligned with the principle axes of the geometry's inertia tensor. Details of the computation of the body frame are in Chapter 6. For the B-spline solid, the local coordinates are the material coordinates $\mathbf{u}$. The remaining volume in the proximity algorithm will contain either a B-spline solid or a single triangle from a polygon bone mesh.

5.10.2 Lower and upper bound computation

Given a bounding volume, B, we define functions, $lower(B, \mathbf{p})$ and $upper(B, \mathbf{p})$ that return the lower and upper bound distances of the bounding volume to the point $\mathbf{p}$. If the volume contains a B-spline solid, we use the routine **closest**$(\mathbf{p})$ defined in Section 3.5.4 to find the closest point on the solid to $\mathbf{p}$ exactly without needing to refine the volume further. Therefore, the lower and upper bounds will be set to the computed distance between $\mathbf{p}$ and the closest point on the B-spline solid:

$$lower(B, \mathbf{p}) = upper(B, \mathbf{p}) = \|\mathbf{p} - \mathbf{closest}(\mathbf{p})\|. \tag{5.28}$$

When these bounding volumes enclose a polygonal mesh, we use a hierarchical tree of oriented bounding boxes (OBBs)[46] to represent the bounding volume and to provide a way to refine the volume into smaller subvolumes (Figure 5.11). An oriented bounding box has its axes aligned to the eigenvectors of the covariance matrix of the vertices of the polygonal mesh. Each OBB can be recursively subdivided into two child OBBs. The polygons within one box are partitioned into two groups and OBBs are constructed on each subgroup. OBB can be used with non-convex polygonal objects, which is an important requirement for bone geometry. For a polygonal mesh, the bounding volumes will either be an OBB or contain a single triangle (at the leaves of the OBB tree).

For an OBB, we compute the lower bound by finding the shortest distance from the point $\mathbf{p}$ to the OBB by solving a constrained least-squares problem,

$$f(\mathbf{p}) = (\mathbf{o} + s\mathbf{d_1} + t\mathbf{d_2} + u\mathbf{d_3} - \mathbf{p})^2, \tag{5.29}$$

where $\mathbf{o}$ is a reference origin point on the bounding box and $\mathbf{d_1}$, $\mathbf{d_2}$, $\mathbf{d_3}$ are the direction vectors that correspond to the orientation and length of the three orthogonal edges of the bounding box. The solution vector (s^*, t^*, u^*) is restricted to $s^* \in [0, 1], t^* \in [0, 1], u^* \in [0, 1]$, and the closest point, $\mathbf{p}^*$, is computed as:

$$\mathbf{p}^* = \mathbf{o} + s^*\mathbf{d_1} + t^*\mathbf{d_2} + u^*\mathbf{d_3}. \tag{5.30}$$

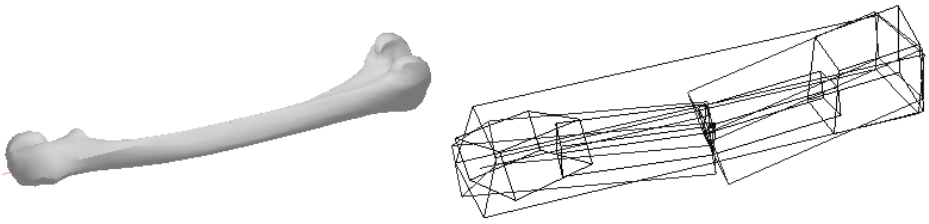


Figure 5.11: The right image displays the hierarchical assembly of oriented bounding boxes (OBBs) for the bone geometry on the left.

Here, $lower(B, \mathbf{p})$ is defined as

$$lower(B, \mathbf{p}) = \|\mathbf{p} - \mathbf{p}^*\|. \tag{5.31}$$

To compute the upper bound, $upper(B, \mathbf{p})$, we arbitrarily choose a polygon vertex residing in the OBB and take its distance to the point, $\mathbf{p}$. Ideally, we would like the bound to be tighter, but the computational requirements to search for the lowest upper bound would outweigh the time it takes to simply choose a vertex.

For finding the closest point on a triangle to a given external point $\mathbf{p}$. We use Eberly's[29] algorithm. The procedure projects the point onto the plane of the triangle. If the point lies within the triangle, the projected point is the solution. Otherwise, the closest point to the projected point is located on the boundary of the triangle. Details of Eberly's algorithm can be found in Appendix B. As in the B-spline solid case, the distance to this closest point corresponds to both the upper and lower bounds on the minimal distance.

5.10.3 Algorithm for *FindClosest*

The function,

$$[objectID, localpt] = FindClosest(\mathbf{p}, O), \tag{5.32}$$

makes use of the procedure, $AddToSortedList(ActiveList, B)$ that decides whether to add a candidate volume, B, to a sorted list, $ActiveList$ or not add the volume if it deduces that B cannot contain the closest point based on comparisons with existing lower and upper bounds in $ActiveList$. We present the algorithms for $FindClosest$ and $AddToSortedList$.

The algorithm for $FindClosest$ is as follows:

1. Input: an external point, $\mathbf{p}$, a set of objects O_i in O
2. Set $WorkList = [\,]$.
3. Compute bounding volume trees, $B_i = BoundingVol(O_i)$ for every object.
4. Add all B_i to $WorkList$.
5. For all i, compute the lower bounds on distance between B_i and $\mathbf{p}$, $B_i.lowerbound = lower(B_i, \mathbf{p})$.
6. For all i, compute the upper bound on minimum distance between B_i and $\mathbf{p}$, $B_i.upperbound = upper(B_i, \mathbf{p})$.

7. Set $ActiveList = [\,]$
8. Call $AddToSortedList(ActiveList, B_i)$ for all B_i in $WorkList$.
9. Reset $WorkList = [\,]$.
10. For all B_i in $ActiveList$, refine bounding volumes.
11. if $refine(B_i) \neq [\,]$,
12. add B_i to $WorkList$.
13. if $refine(B_i) = [C_i, D_i]$,
14. compute upper and lower bounds for C_i,D_i.
15. add C_i,D_i to $WorkList$.
16. Repeat steps 7-15 until all members of $ActiveList$ can no longer be refined.

With each pass of the algorithm, $ActiveList$ will be reduced in size as fewer volumes are added by $AddToSortedList$. The procedure $AddToSortedList(ActiveList, B)$ determines if B should be added to the current $ActiveList$. If it is added, it is placed in sorted order according to its lower bound value.

The algorithm for $AddToSortedList(ActiveList, B)$ is as follows:

1. Input: The current sorted list contents, $ActiveList$ and a candidate volume B.
2. If $ActiveList = [\,]$, add B to $ActiveList$ and return.
3. Set W to be the first item in $ActiveList$.
4. while (W contains a volume in $ActiveList$) {
5. If $B.lowerbound \geq W.upperbound$, return.
6. If $B.upperbound < W.lowerbound$
7. insert B before W and discard W and all volumes after it.
8. return
9. If $B.lowerbound < W.lowerbound$
10. insert B before W
11. return
12. Set W to next item in $ActiveList$.
13. }
14. Add B to end of $ActiveList$
15. return

$AddToSortedList$ attempts to add volumes to $ActiveList$, which is sorted in ascending order according to the lowerbound value. In the listing, line 5 prevents B from being added to $ActiveList$ if there is no possibility that B can contain the closest point because its lower

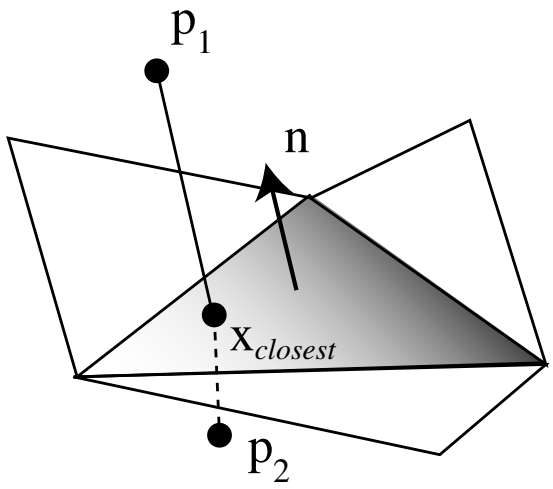


Figure 5.12: The point-to-plane constraint is applied conditionally. In the case of p_1, the point is on the outward side of the closest shaded triangle, so reaction constraint forces are not applied. For point, p_2, the point is penetrating into the triangle and must be constrained to the plane of the triangle.

bound is greater than an existing upperbound on the shortest distance. Line 6 can potentially cull out large groups of candidate volumes in $ActiveList$ when the new volume B contains an upperbound on the distance which is less than the lowerbound of the current volume, W, being examined in $ActiveList$. By preventing unnecessary volumes from being added to the list, the search for the closest point is quickly reduced to a few possible candidate volumes.

In $FindClosest$, once we have one remaining bounding volume in $ActiveList$ that cannot be further refined, we find the exact point that corresponds to the shortest distance from this volume. In our applications, the bounding volume will contain either a triangle from the polygonal mesh models of the bones or a B-spline solid coinciding with the musculotendons.

5.10.4 Point-to-plane Reaction Constraints

Once the closest point is found on an object, we can find the local plane information at that point in the form of the normal of the tangent surface on a B-spline solid or the normal to the plane of the triangle in a polygon mesh. This permits a point-to-plane reaction constraint to be established. The current force acting on a spatial point is modified by applying a reaction constraint force to project out the components of the original force that would have violated the plane constraint. Implementation details can be found in [99].

Given a plane equation $P(\mathbf{x})$ with a unit normal $\mathbf{n}$ located at the closest point, $\mathbf{x}_{closest}$, we apply this constraint conditionally when we detect interpenetration of the external point, $\mathbf{p}$ within the local surface. This is done by examining the sign of $P(\mathbf{p})$ where $P(\mathbf{x})$ is the plane equation:

$$P(\mathbf{p}) = \mathbf{n} \cdot (\mathbf{p} - \mathbf{x}_{closest}). \tag{5.33}$$

By ensuring that the surfaces of the bone and musculotendon are oriented so that the normals

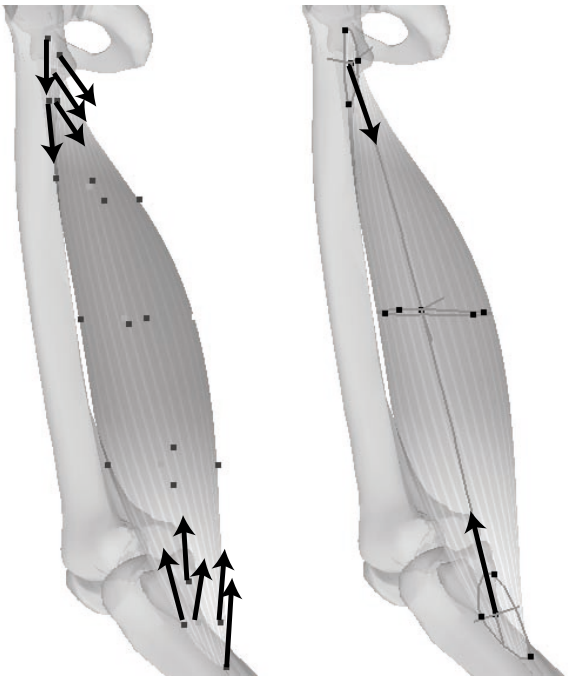


Figure 5.13: Application of musculotendon forces on bones: The left image applies forces at all spatial points attached to the bone. The right image applies forces only at the end points of the axial profile curve.

are directed outwards, we can detect when an external point has gone underneath the local plane. If $P(\mathbf{p}) \leq 0$, the point $\mathbf{p}$ is either on the plane or below it. Otherwise, the point lies on outward side of the plane and no reaction constraint is needed (Figure 5.12).

5.11 Application of Forces on Bones

Musculotendons are the primary force actuators that create locomotion and changes of body posture. For forward dynamic simulation of musculoskeletal systems, provisions to model these forces can be established by directly taking into account the B-spline solid geometry.

We provide two approaches. The first approach creates pulling forces that act on the origin and insertion attachment areas of the tendon to the bone surface. The second approach only applies forces at an axial contact point to the bone surface and is used in conjunction with profile curves. Figure 5.13 illustrates the differences. The first approach offers a greater degree of resolution of forces, while the second approach is less-computationally expensive and requires fewer parameters to be defined.

Low-level forces

We apply the forces to the bone at all the musculotendon spatial points **s** in contact with the bone. This includes the spatial points at the junction between the tendon and bone surface, as well as surface spatial points on the B-spline solid that come in contact periodically with the bone during collision.

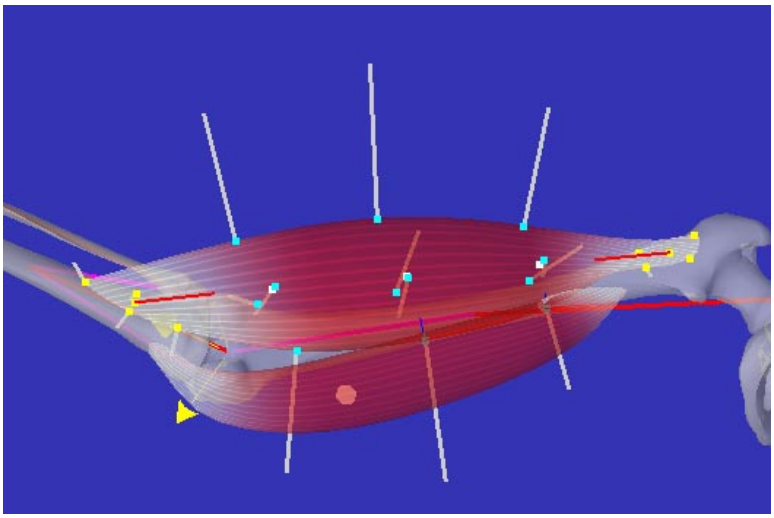


Figure 5.14: Forces generated on the musculotendons through volume-preserving potential forces, contractile forces, and contact forces. Simulation runs at 4.23 Hz where the majority of time (at least 40%) is spent updating the collision contact points.

At the attachment areas, the spatial points on the contact surface can be considered to be embedded in the bone as well as the tendon. Therefore, we transmit the resultant forces on the spatial point to the coincident point on the bone. For the surface contact points, we apply the reaction force created by the point-to-plane constraint in Section 5.10.4. The reaction forces of the muscles pressing against the bone allows the skeletal system to see the inertial forces of the musculotendons as external forces. This allows us to simulate the musculotendon B-spline solids as independent systems from the rigid body dynamical system of the skeleton. At the same time, the external forces represent the effects of the mass properties of muscle influencing motion in adjacent bones (Figure 5.14). In biomechanics, accounting for the mass properties of musculotendon in simulations of musculoskeletal systems has rarely been done.

Forces through profile curves

The above approach may require many parameters to be specified and simulated. If we are mainly concerned with the net resultant forces at the origin and insertion areas, we can model a single force generator for the entire musculotendon.

Using profile curves, the axial curve can be used as an approximation to the centroid curve of the musculotendon. We can create a force model dependent on the length and velocity of shortening of the axial curve using an elastic strain model or a more accurate Hill-based force model[138]. These forces can then be applied at the attachment ends of the axial curve. To ensure conservation of momentum, the vector sum of the forces at the contact points with the bone must be zero. This will occur as the axial curve shortens enough to generate tension as it pulls against the attached bone. Figure 5.15 illustrates how forces derived from the profile curves are used to generate motion in a lower leg skeleton assembly. Further discussion of these results can be found in Chapter 8.

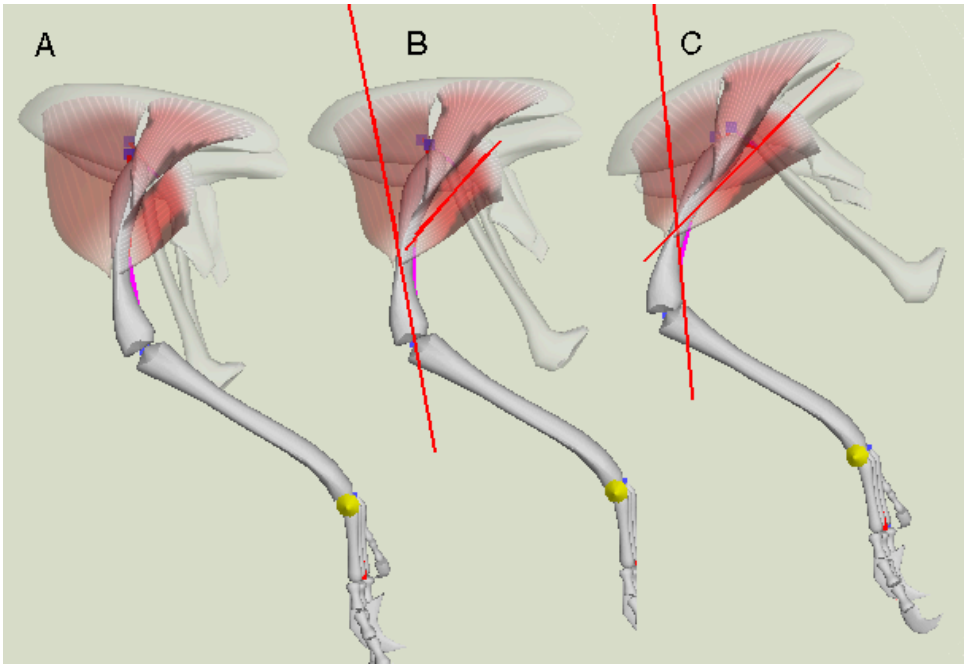


Figure 5.15: Forces based on the profile curve information of the B-spline solid musculotendons are used to create motion in a lower limb skeleton assembly.

5.12 Summary

This chapter describes the physical modelling of the B-spline solid musculotendon model. We have described how the equations of motion of a B-spline solid can be formulated with a diverse set of force models. Potential energy forces can be used to create forces of deformation, including volume-preserving forces. Viscoelastic networks can be designed to model nonuniform or directional forces within a muscle, such as contractile forces at various orientations or the different material properties of tendon and muscle fibres. Anchored fields create controllable, local deformations with physically-based oscillations. Profile-curve driven dynamics provide numerical stability of solids under fast, transient motions while displaying dynamic, inertial effects. Reaction constraints enable point-to-plane constraints at selected spatial points and are used to model collision effects between the musculotendons with bones, as well as with other muscles. Finally, the musculotendons can be treated as force actuators that create loads on the attached bones for limb motion in skeletons.

Chapter 6

Bones, Ligaments, and Skin

> No knowledge can be more satisfactory to a man than that of his own frame, its parts, their functions and actions.
>
> **Jefferson - Letter to Thomas Cooper**

Muscles and tendons actively create motion and change body shape in humans and other animals. However, they do not act in isolation. They interact with other passive biological materials, such as bones, ligaments, and skin (with the fat layer). This chapter discusses modelling considerations for these passive structures.

6.1 Bones

Bones create the skeleton framework to which musculotendon and ligaments are attached and provide the solid link segments that create an articulated figure. As the stiffness of bone material is several times larger than neighbouring musculotendon, we consider bones as rigid objects. This allows each bone's state to be described with position and orientation only.

The centre of mass of the bone is taken to be the reference point for position. Orientation can be described with a set of three local, orthonormal vectors originating from the centre of mass. This creates a local coordinate system for the bone. This frame of reference is positioned and rotated within an articulated figure skeleton. Existing models of bones are readily available as polygon meshes. Consequently, we assume our bone geometry consists of polygonal meshes where each polygon is consistently oriented so that face normals points outward (Figure 6.1).

6.1.1 Calculation of mass properties

The mass properties we typically need for use in physical simulation are the centre of mass and inertia tensor for the bones. Typically, the only information available is the bone's boundary surface. In these situations, the assumption of uniform bone density is made. In reality, bone has non-uniform density throughout its volume. For example, birds have lower interior bone densities due to development of internal air pockets (a process known as *pneumatization*), creating light, but structurally strong, wide bones[10].

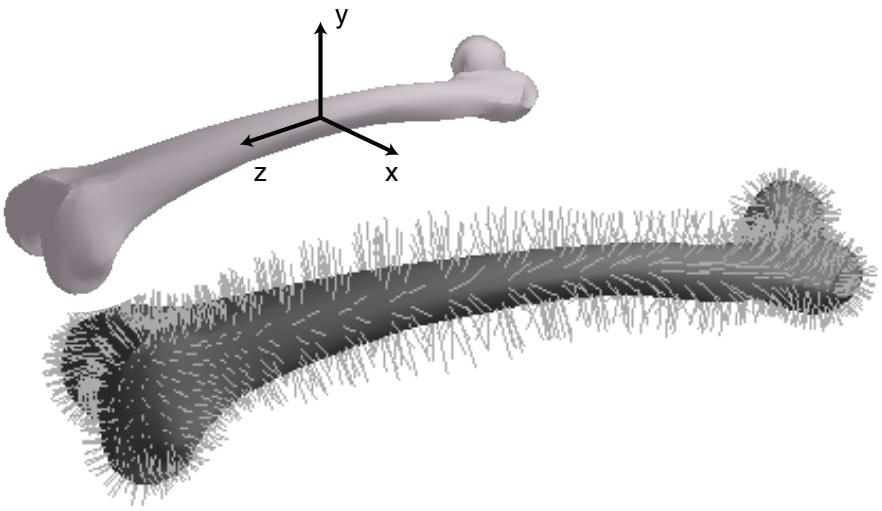


Figure 6.1: Bones are polygonal meshes with normals facing outward. Each bone will be transformed to have a local axis with origin at the centre of mass and axes corresponding to the principal directions of its inertia tensor.

We present two methods to calculate these mass properties. The *continuous bone geometry* method assumes uniform density and a closed boundary surface for the bone. The *discrete bone geometry* method approximates the bone's shape with a discrete cloud of mass points. The latter method can accommodate cases where nonuniform density distribution in bone is desired by including points of varying mass within the bone's volume.

Finding the centre of mass, $\mathbf{r}$, for the bone determines an origin for a body reference frame centred on the bone. Computing the inertia tensor matrix produces moments of inertia on the diagonal and products of inertia on the off-diagonal entries[1]:

$$J = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}. \tag{6.1}$$

Furthermore, we can use the eigenvectors (principal axes) of the matrix, J, to define a new body frame of reference for the bone where the inertia tensor is a sparse, diagonal matrix[44]. The products of inertia become zero, and the moments of inertia are the eigenvalues (principal moments) on the diagonal. The sparse inertia tensor accelerates computations during simulation.

6.1.2 Continuous bone geometry

If the bone is represented with a closed polygonal mesh, the mass and volume properties can be computed with an exact integral taken over the bone's boundary surface. To determine that

[1]We follow the convention that the products of inertia are preceded with a negative sign. Some structural analysts omit the negative sign.

a polyhedron is closed, every edge must be shared by two faces. If an edge only has one associated face, that face is adjacent to a hole or open boundary. The assumption is made that the object has uniform density because nothing can be deduced about the internal volume density distribution from just its boundary surface. We describe several of these mass property integrals which need to be evaluated.

Mass

The mass, m, of a bone is computed from its volume, V, and density, ρ,

$$m = \rho V. \tag{6.2}$$

The volume is computed by integrating over the entire domain enclosed by the object's boundary surface,

$$V = \int_B dV, \tag{6.3}$$

where $B \subseteq \Re^3$ is the domain of the bone.

Centre of mass

The centre of mass, $\mathbf{r}$, for the bone is calculated as follows:

$$\mathbf{r} = \frac{\rho}{m} \begin{pmatrix} \int_B x dV & \int_B y dV & \int_B z dV \end{pmatrix}^T. \tag{6.4}$$

The inertia tensor matrix entries are evaluated as:

$$\begin{array}{rcl} I'_{xx} & = & \rho \int_B (y^2 + z^2)\, dV \\ I'_{yy} & = & \rho \int_B (z^2 + x^2)\, dV \\ I'_{zz} & = & \rho \int_B (x^2 + y^2)\, dV \\ I'_{xy} = I'_{yx} & = & \rho \int_B xy\, dV \\ I'_{yz} = I'_{zy} & = & \rho \int_B yz\, dV \\ I'_{zy} = I'_{xz} & = & \rho \int_B zx\, dV. \end{array} \tag{6.5}$$

Notice that the matrix is symmetric. For simulation, the inertia tensor needs to be computed with respect to the body frame's origin located at the centre of mass. This can be computed directly from Equations 6.5 using *transfer-of-axis relations*[84], using the centre of mass, $\mathbf{r} = (r_x, r_y, r_z)^T$,

$$\begin{array}{rcl} I_{xx} & = & I'_{xx} - m(r_y^2 + r_z^2) \\ I_{yy} & = & I'_{yy} - m(r_z^2 + r_x^2) \\ I_{zz} & = & I'_{zz} - m(r_x^2 + r_y^2) \\ I_{xy} & = & I'_{xy} - m r_x r_y \\ I_{yz} & = & I'_{yz} - m r_y r_z \\ I_{zx} & = & I'_{zx} - m r_z r_x. \end{array} \tag{6.6}$$

The inertia tensor matrix, J, can be diagonalized to make the products of inertia disappear. Since J is symmetric and contains real values, we can use the *Jacobi method* to perform the

diagonalization[100]. The Jacobi method is a sequence of orthogonal, similarity transformations of J, called *Jacobi rotations*, which gradually reduce the off-diagonal elements of successive J to zero. The Jacobi method always works for real-symmetric matrices. Although the algorithms are computationally expensive for large matrices, the inertia tensor J is only 3×3 in size and can be quickly diagonalized with this method. Figure 6.1 illustrates a bone with its principal axes.

Mirtich[84] developed a technique for computing the integrals involved for the mass, centre of mass and inertia tensor by integrating only over the boundary domain. Using the *Divergence theorem*[111], the volume integrals are converted into a sum of surface integrals over the individual polygons of the bone geometry. Each surface integral over a polygon is rewritten as an integral over a planar projection of the surface facet. Finally, *Green's theorem*[111] allows the planar integrations to be replaced by a sum of line integrals around the edges of the polygon. Mirtich's algorithm is particularly useful for bone geometry as it has no restrictions on convexity or genus of the shape.

6.1.3 Discrete bone geometry

If the physical mesh is not a closed surface, Mirtich's approach will not work. An alternative is to treat each vertex of a mesh as a mass point. Care must be taken to ensure a uniform vertex distribution over the bone's surface (and ideally within the volume of the bone) to minimize the approximation error for the inertia tensor. If this is not possible, varying masses can be assigned to different regions of bone to compensate for any nonuniform distribution of vertices. The equation for the centre of mass, $\mathbf{r}$, and the inertia tensor, J, are discretized versions of the continuous counterparts:

$$\mathbf{r} = \frac{\left(\sum m_i x_i \quad \sum m_i y_i \quad \sum m_i z_i \right)^T}{\sum m_i}, \tag{6.7}$$

and

$$\begin{aligned}
I'_{xx} &= \sum m_i (y_i^2 + z_i^2) && (6.8)\\
I'_{yy} &= \sum m_i (z_i^2 + x_i^2) && (6.9)\\
I'_{zz} &= \sum m_i (x_i^2 + y_i^2) && (6.10)\\
I'_{xy} = I'_{yx} &= \sum m_i x_i y_i && (6.11)\\
I'_{yz} = I'_{zy} &= \sum m_i y_i z_i && (6.12)\\
I'_{zy} = I'_{xz} &= \sum m_i z_i x_i, && (6.13)
\end{aligned}$$

where m_i is the mass for vertex i with coordinates (x_i, y_i, z_i). Subsequent conversion to a diagonal inertia tensor about the centre of mass follows the procedure for continuous geometry.

A final alternative is to obtain inertia moments and centre of mass information from anthropometric studies. Unfortunately, the majority of data available only offers the gross mass properties of a limb, lacking information about individual bones.

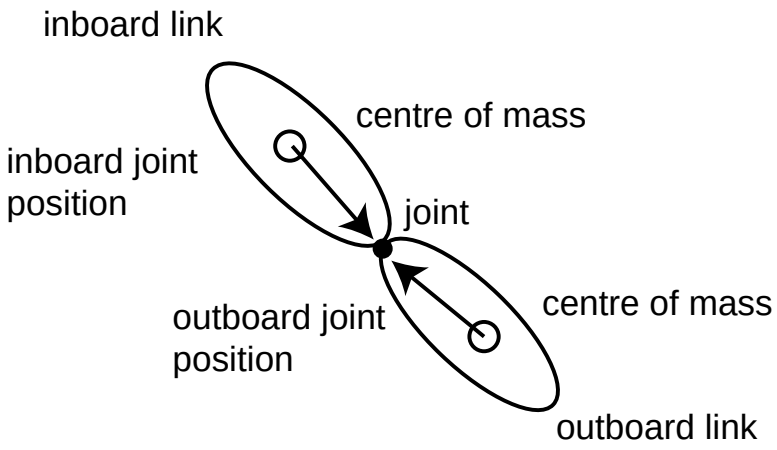


Figure 6.2: The joint position is specified in the body frame coordinates of both of its adjacent links (the inboard and outboard links).

6.2 Articulated figure

The skeleton is represented as an articulated figure comprising several links interacting with each other through joints. We restrict the articulated figure to have a tree topology, with a single root with one or more possible chains of links (Figure 2.5). The root link is considered the parent. Each link has one parent and zero or more child links. Each link contains optional geometry and a parent joint that defines relative position and orientation relationships with its parent link.

6.2.1 Joint modelling

A joint is ultimately represented by a transformation matrix. However, it is convenient to parameterize the joint transformation with one to six degrees of freedom to define and constrain this transformation. We implement joints by allowing direct manipulation of the degrees of freedom and converting the variables to a transformation matrix.

Of particular interest is the ability to interactively change the orientation or position of a joint's axis of rotation. This has applications in biomechanics where a joint articulation between two bones can experience a changing axis or centre of rotation, as is the case with the knee[88].

Joint centre modification

Commercial simulators like SD/FAST[57] position a joint relative to the body frame coordinates of the adjacent links (Figure 6.2). We adopt this joint convention to have the ability to freely modify the joint rotation centre by recomputing the new joint position in terms of the body frame coordinates for each link. The body frame coordinates of the joint are used to adjust the translation component of the transformation matrix to properly position and rotate the link's geometry relative to the joint. The body frame origin remains at the link's centre of mass. As discussed in Section 6.1.1, this permits a sparse diagonal inertia tensor. In contrast,

many kinematics-based editors for articulated figures locate the origin of the body frame at the joint position. While this arrangement may simplify matrix transformations, it limits the type of joints that can be modelled and produces dense inertia tensor matrices.

Joint axis modification

For ball-and-socket joints, the orientation is represented with a quaternion, $\mathbf{q}_{joint}$. To allow direct manipulation of the orientation, we provide three visible reference axes that correspond to the column vectors of the current orientation. Selecting an axis defines it as the new rotation axis for subsequent interactive editing. Although we can manipulate all three degrees of freedom simultaneously, we have found it easier to conduct a series of single axis rotations, affecting one degree of freedom at a time. Each incremental orientation can be converted directly to a quaternion,

$$\mathbf{q}_{new} = (\cos(\theta/2), \sin(\theta/2)a_x, \sin(\theta/2)a_y, \sin(\theta/2)a_z), \tag{6.14}$$

where (a_x, a_y, a_z) is the current rotation axis and θ the angle of rotation in radians. Using quaternion multiplication[128], $\mathbf{q}_{new}$ can be multiplied with $\mathbf{q}_{joint}$ to produce the orientation that would result from rotating $\mathbf{q}_{new}$ relative to $\mathbf{q}_{joint}$:

$$\mathbf{q}_{joint}^{new} = \mathbf{q}_{joint}\,\mathbf{q}_{new}. \tag{6.15}$$

The three reference axes can be oriented independently of the joint to change the axes of rotation.

The ability to freely change the joint position and axis of rotation is useful for modelling biomechanical joints as these characteristics do not stay constant during joint motion. It is possible to place a joint centre of rotation inside a bone instead of on the articular bony surface to simulate various joint dislocations (Figure 6.3).

Joint interpenetration detection

To prevent interpenetration of bones during joint articulation, we utilize the oriented bounding boxes previously computed for the bone geometry (see Section 5.10.2). The RAPID implementation [46] of oriented bounding boxes allows intersection tests between two oriented bounding box trees. However, in a musculoskeletal system, there can be a potentially very large number of bone pairs that can collide. Testing every combination would be prohibitively expensive. We use the V-COLLIDE library[60] to quickly find pairs of objects that may be interpenetrating. World axis-aligned bounding boxes are maintained for each bone in the scene. These boxes are used to provide a quick overlap test to identify bones potentially in collision. All potential pairs are then subjected to a more accurate oriented bounding box intersection test.

If we use physically-based collision resolution schemes between bones involved in a joint, objects would continue moving in reaction to collision forces due to inertia, making it difficult to precisely control pose when articulating a skeleton. The computational time for simulation would reduce the interactivity of the joint editing process. This justifies the use of a simpler, kinematic mechanism for joint manipulation. We can achieve very fast, but limited, collision detection and resolution by storing the last, previous set of joint angles that were collision free.

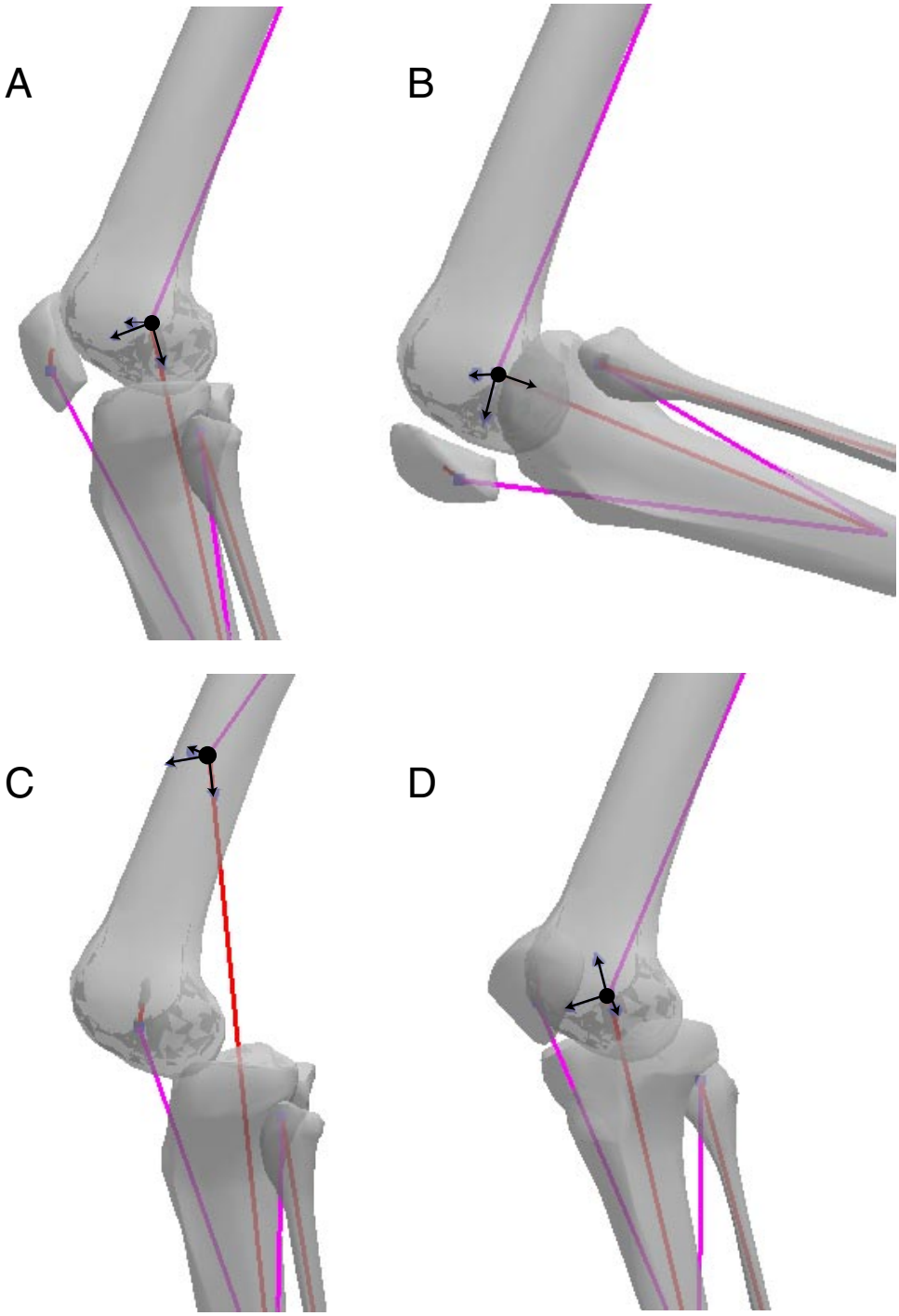


Figure 6.3: The joint centre is (A) can be interactively moved to produce different joint articulations (B and C). In (D), the joint axes were re-oriented to allow rotations about new axes.

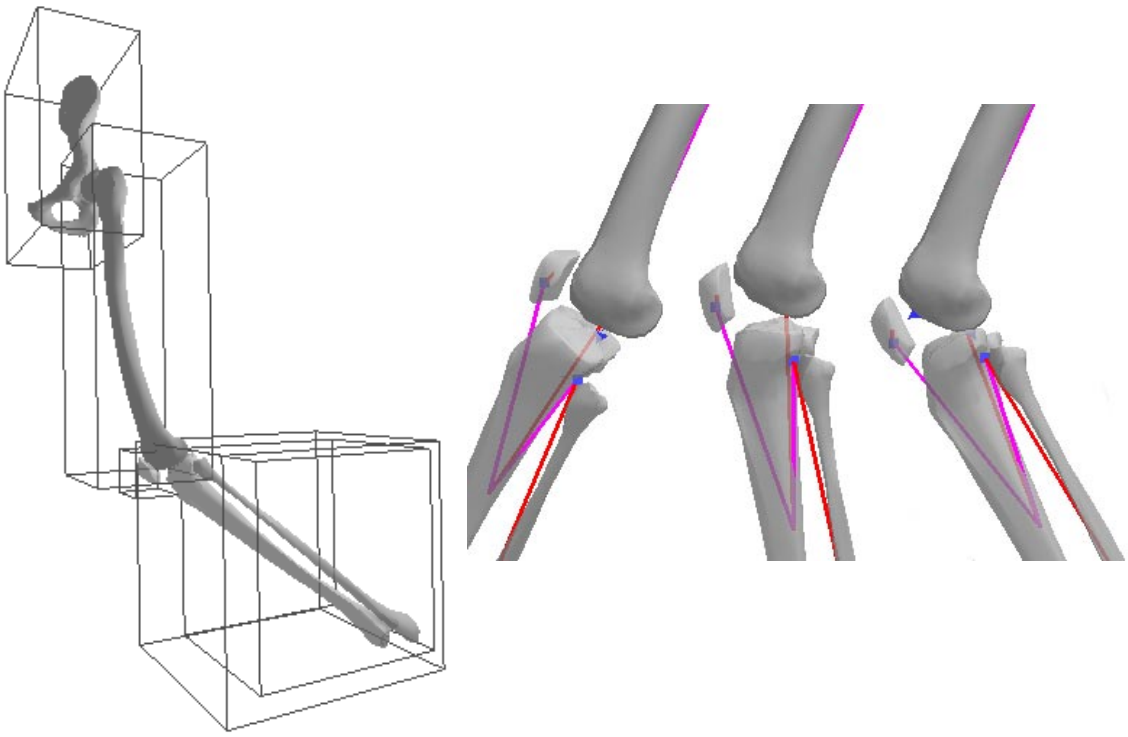


Figure 6.4: On the left are the axis-aligned bounding boxes used for detection of potential pairs of bones in collision. The sequence of joint articulations on the right demonstrate how the joint has restricted movement to prevent bone interpenetration.

If a collision is detected within the skeletal system during joint manipulation, we reset the skeleton to the stored set of joint angles (Figure 6.4). As long as there are small incremental changes from pose to pose while manipulating the bones of the skeleton, a good sense of collision detection and resolution is maintained.

6.3 Ligament modelling

Ligaments differ from musculotendons in that the former is passively elastic while the latter can actively contract. Although both tendon and ligaments contain collagen, elastic ligaments can contain twice as much elastin than collagen, giving them spring-like properties. Functionally, ligaments are used to guide joint movement and maintain the stability of joints during movement. Visually, they are cord-like in appearance and can twist along their lengths. They tend to be shorter than musculotendon because they connect closely adjacent portions of bone.

We also model ligaments geometrically using the B-spline solid primitive, using profile curves to interactively sketch them on the joint articular surfaces (see Section 4.6). Figure 6.5 illustrates ligaments modelled for a human knee. For physical modelling, we can either use an embedded viscoelastic network or profile curve physics. In contrast to musculotendon, the force models consist only of elastic and viscous elements, as there is no active contractile component as in muscle.

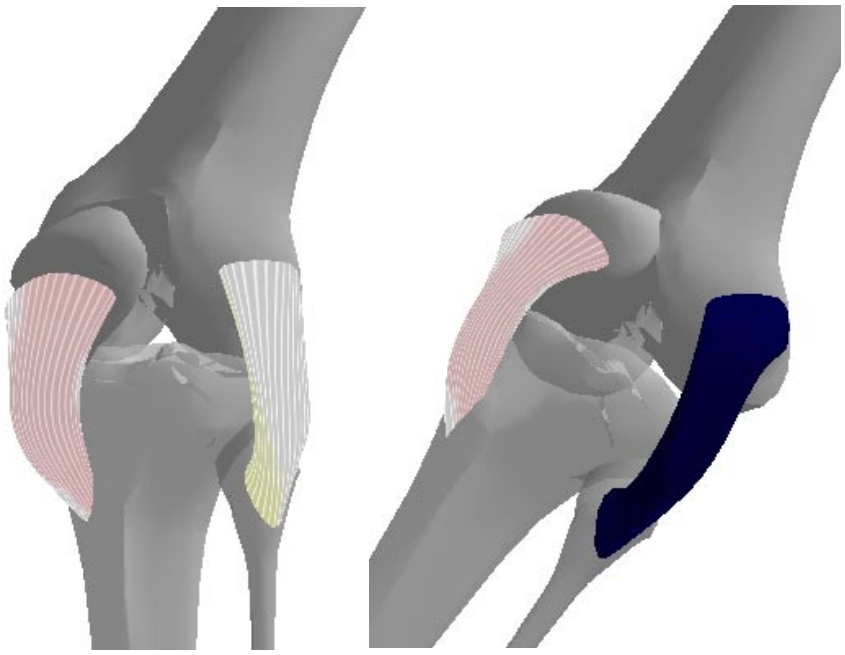


Figure 6.5: The outer collateral and patellar ligaments are modelled using B-spline solids. On the right, the strain in the outer collateral ligament exceeds the safety level, which is indicated by a colour change.

For biomechanical analysis, it is useful to visualize *strain* in the ligaments which is defined as:

$$\sigma = \frac{l^L - l_o^L}{l_o^L}, \tag{6.16}$$

where l^L is the ligament length and l_o^L is a pre-defined rest length. If strain exceeds a certain percentage, we can change the colour of ligaments to indicate overstrain conditions. To measure l^L, we take the computed arclength of the axial curve from the ligament's profile curves. The arclength is evaluated numerically using Simpson's rule[100]. Figure 6.5 visualizes an injury-causing strain condition in the knee.

6.4 Skin modelling

Skin is the superficial layer that covers the layers of skeleton, musculotendon and fat to visually manifest the body's form. The contours and valleys that create body shape are directly influenced by the musculotendons underneath. Modelling of individual layers within skin (the *epidermis* and the *dermis* layers) is beyond the scope of this thesis, but explicit modelling of these layers has been accounted for in previous research in physically-based facial animation[75] and subsurface scattering of light in skin reflectance models[49].

We restrict our representation of skin to be a boundary surface geometric model. We will use a polygonal mesh, but the methods we present to deform skin can be generalized to para-

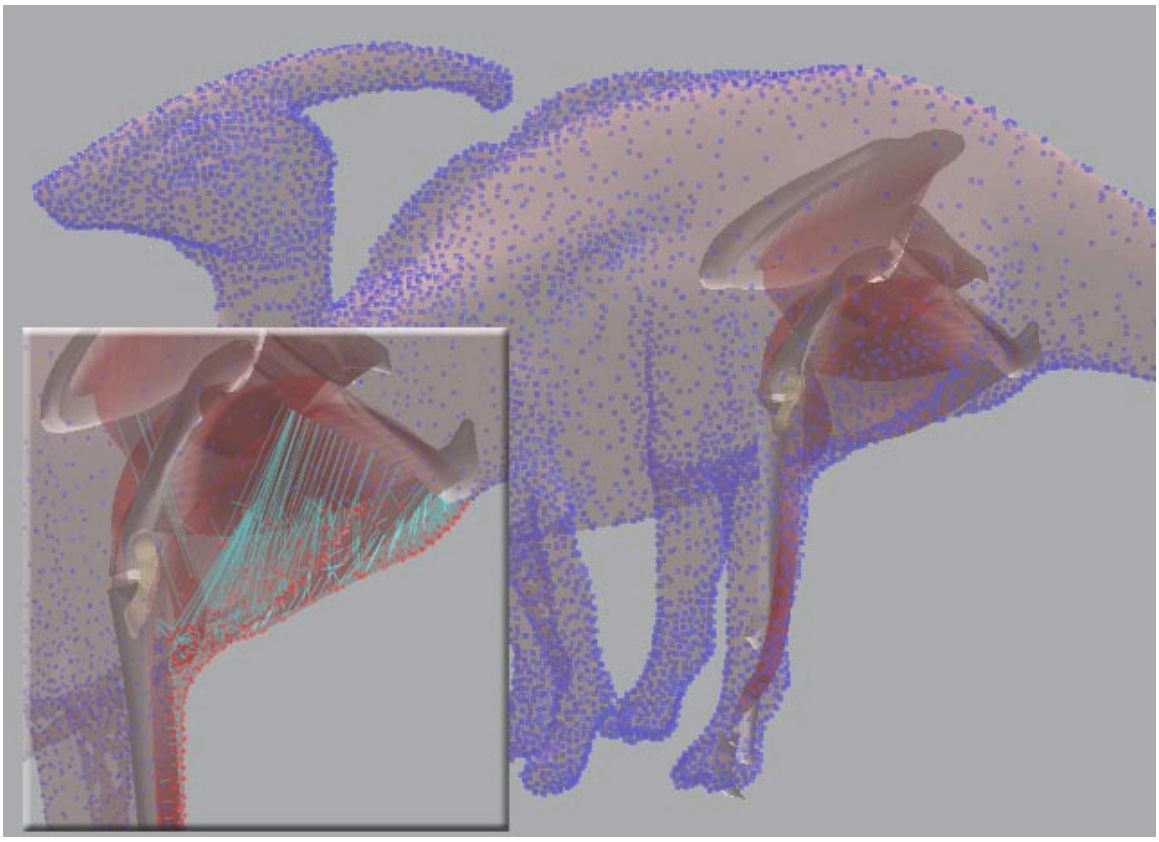


Figure 6.6: A geometric mesh representing the skin of the dinosaur can be deformed by associating skin points with the closest musculotendon or bone. Inset: Lines connect up selected skin points with their closest underlying features.

metric surfaces[69] and subdivision surfaces[27]. The choice of geometric representation is an important factor in the effectiveness of expression of features such as wrinkles, creases and folds. Wrinkles tend to develop in a perpendicular direction to muscle fibres or tendons[42]. Folds can disappear as portions of skin are stretched.

In relation to anatomical modelling, we aim to influence the skin geometry by the skeleton, musculotendon and fat layers only. Portions of the skin need to be associated with these underlying tissues. The fat layer, located between skin and muscles, can have variable thickness throughout the body.

In the following methodology of deforming skin, we require that the geometric representation can be modified to interpolate a set of points, $\mathbf{p}$. These points are adjusted in response to movements in the underlying musculoskeletal system. For a polygonal mesh or interpolating subdivision surface, we can use the mesh vertices. For a B-spline surface, we use spatial points, described in Section 3.5.2. From the assembly of skeleton, ligament and musculotendon, we use the $FindClosest$ function (Equation 5.27, defined in Section 5.10.1, to associate each skin point (Figure 6.6), $\mathbf{p}$, with the closest point on the nearest bone, musculotendon or ligament. Suppose for a particular skin point, $\mathbf{p}_i$, we locate a closest feature point, $\mathbf{f}_i$. By using different connectivity constraints between these two points, we can emulate various phenomena. For example, we can choose to constrain the skin points to coincide with the closest feature point

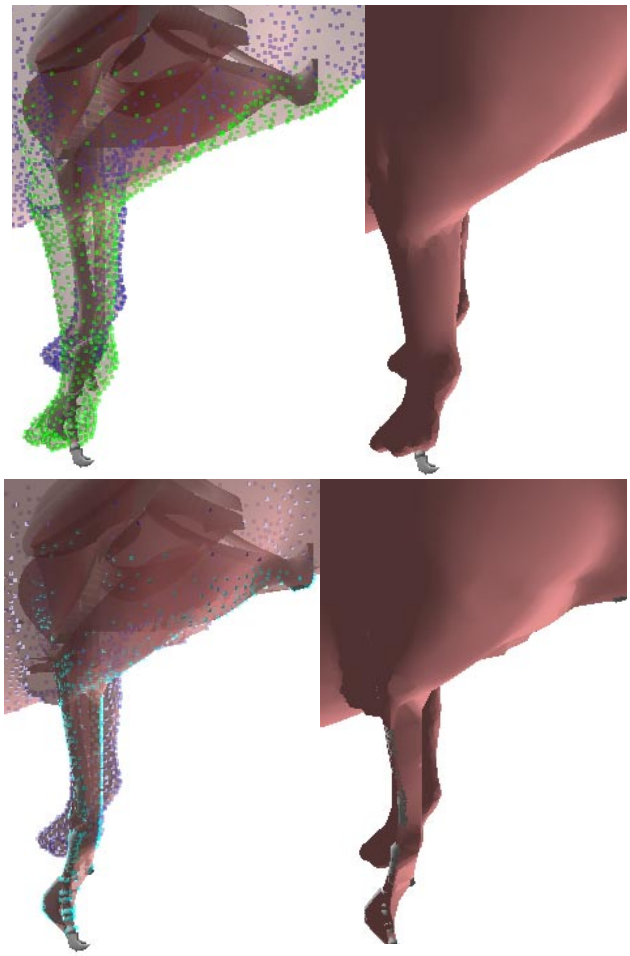


Figure 6.7: Each point of the skin geometry can be associated with its closest feature. Constraining the skin points to the closest feature point produces a "shrink-wrapping" effect. The top image shows the skin before being "shrink-wrapped" to appear as the bottom image. The same configuration with opaque and transparent skin is displayed for comparison.

on bone or muscle, "shrink-wrapping" the skin over the anatomy (Figure 6.7). This technique assumes that skin points are relatively well-distributed and densely sampled. The skin should also be relatively close to the underlying anatomy. Otherwise, distortions or visible tearing can occur.

Alternatively, the relative position of $\mathbf{p}_i$ with respect to $\mathbf{f}_i$ can be maintained by expressing $\mathbf{p}_i$ in the local coordinates of a frame of reference centred on $\mathbf{f}$. In the latter case, we can define the frame axes by taking the cross-product of the unit normal and tangent vectors at the feature location. We use the tangent plane and normal of the B-spline solid surface point for musculotendons and of the closest triangle for bone geometry. In this manner, the underlying anatomy can be used as a feature-based deformation technique for skin.

In the following methodology of deforming skin, we require that the geometric representation can be modified to interpolate a set of points, $\mathbf{p}$. These points are adjusted in response to movements in the underlying musculoskeletal system. For a polygonal mesh or interpolating

subdivision surface, we can use the mesh vertices. For a B-spline surface, we use spatial points, described in Section 3.5.2. From the assembly of skeleton, ligament and musculotendon, we use the $FindClosest$ function (Equation 5.27, defined in Section 5.10.1, to associate each skin point (Figure 6.6), $\mathbf{p}$, with the closest point on the nearest bone, musculotendon or ligament. Suppose for a particular skin point, $\mathbf{p}_i$, we locate a closest feature point, $\mathbf{f}_i$. By using different connectivity constraints between these two points, we can emulate various phenomena. For example, we can choose to constrain the skin points to coincide with the closest feature point on bone or muscle, "shrink-wrapping" the skin over the anatomy (Figure 6.7). Alternatively, the relative position of $\mathbf{p}_i$ with respect to $\mathbf{f}_i$ can be maintained by expressing $\mathbf{p}_i$ in the local coordinates of a frame of reference c Dynamical effects for fat can be created by adding a spring-damper system anchored at the skin points using the same technique we applied for musculotendons in Section 5.8. For areas of negligible fat, the dynamics already present in underlying musculotendon models can be propagated to the skin points directly through the deformation process. Consequently, it may not be necessary to have dynamical models embedded directly in the skin geometry unless fat layers will have a significant effect. The results of this model are shown in Chapter 8.

6.4.1 Limitations

The simple skin deformation scheme presented does not prevent skin self-interpenetration. Implicit constraints between skin points or force fields can be used to handle self-collisions. In particular, techniques developed for cloth animation[101] can be applied. The generation of realistic folds and wrinkles may require additional preconditioning of the skin geometry. For example, a polygon mesh used for skin can have finer tessellation in areas where folds, creases and wrinkles will be developed. Subdivision surfaces are a potential, useful representation for skin because they allow multiresolution depiction of details, arbitrary topology and local refinement of skin areas[27].

6.5 Summary

We have presented various models for the passive components of an anatomical modelling system: bones, ligaments, fat and skin. These elements are necessary to completely define body shape in addition to musculotendons. Each bone is re-transformed into a body frame of reference that produces diagonal inertia tensors to accelerate simulations. Interactive joint modelling provides a direct method for users to change the joint centres and axes of rotation of a joint. The versatility of the B-spline solid model is illustrated by reusing them to represent ligaments. Length measurements are acquired from the axial profile curve to visualize strain conditions in the ligament to illustrate injury-causing conditions. Finally, we can envelope the underlying bone and musculotendon layers with fat and skin. The ability to deform a skin model motivated by underlying, anatomical features is useful for applications in character modelling and animation of humans and other animals.

Chapter 7

System Architecture

> The best carpenters make the fewest chips.
>
> **German Proverb**

In previous chapters, we described the individual components that comprise the anatomical modelling and animation system. These individual models must co-exist in a common software environment that supports intercommunication between elements. For example, musculotendons must be aware of the bones to which they are attached. The bone's rigid body motion has a direct influence on the deformation of the muscle. Conversely, the musculotendons need to apply forces that can alter the rigid motion of the attached bones during physical simulation. This chapter outlines the system architecture, DANCE (Dynamic Animation and Control Environment)[86], on which the anatomical modelling system is built.

7.1 Design Considerations

As with any large-scale application, the standard requirements of modularity and extensibility are important. Given the exploratory nature of research, the ability to change or enhance system components is important. Communication between various components in the system should be minimized or involve minimal passing of data to localize the effects of changes in a component. To achieve these goals, an object-oriented software architecture was established, identifying a set of abstract, base classes (Figure 7.1) that are managed by a central driver class (Figure 7.2).

The DANCE system features a command-line scripting language interface based on the Tcl language[90] (Figure 7.2). A scripting language provides flexibility in formulating tasks for the application and allows useful concepts such as iterations, flow control and procedure calls. Graphical user interface (GUI) elements such as panels of buttons and scroll bars are completely external to the system, allowing customization of the GUI for different applications. As GUI elements are selected, they generate corresponding script commands that are processed by DANCE. In addition, for operations that require constant visual feedback in the anatomical modeller, a direct manipulation interface for visual operations like joint editing, musculotendon sketching, and physical simulation was provided.

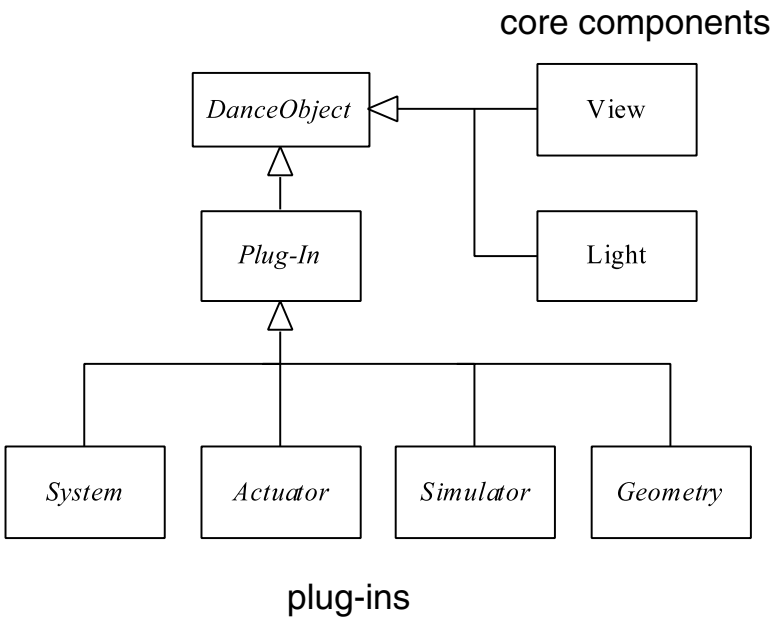


Figure 7.1: System class types: System components are subclasses and partitioned into plug-ins and core components.

We have found that a majority of system features could be reduced to one of several base classes that we defined. By keeping the number of classes small, the conceptual view is greatly simplified and communication between different combinations of class types is minimized. The base classes were chosen to contain mutually-exclusive sets of features to leave software developers with an unambiguous choice of implementation paths for any particular feature. Actual features are created by subclassing instances of the abstract base class in the form of dynamically-linked external modules, known as *plug-ins*. As plug-ins are dynamically loaded, there is no need to re-compile other plug-ins or the central driver code. Plug-ins enable developers to focus on specific implementation details of features without excessive exposure to the central driver or internals of other features of the system. From a plug-in's perspective, every other feature provides services that require no knowledge of implementation.

The central driver class, *dance*, coordinates activities of the base classes. Each different type of base class is assigned its own manager so that retrieval of a specific instance is initiated by querying the corresponding manager (Figure 7.2). All system diagrams in this chapter follow the *Universal Modelling Language (UML)* standard[34]. Appendix C provides a primer of the notation we use from UML.

7.2 DanceObject Class

Every object in the system is derived from a fundamental, *DanceObject* class. The purpose of having a universal class in the DANCE system is to ensure that every class can always retrieve basic information protocols from every feature, such as a unique identifier and type information. A common base class allows container classes to be built that can store heterogeneous items by making the containers accept only *DanceObjects*. For example, in the routines for im-

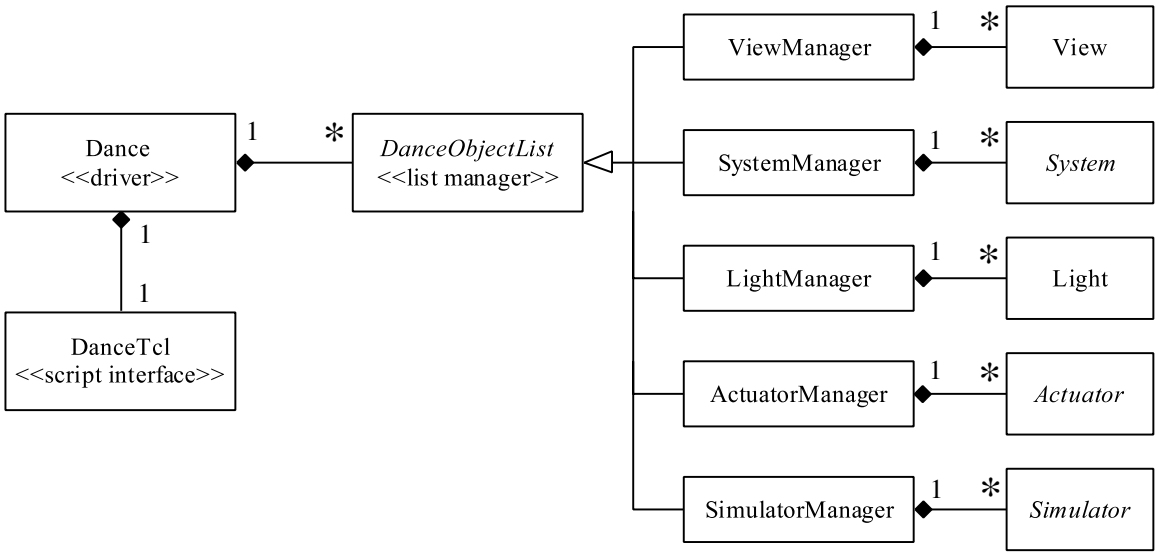


Figure 7.2: List managers for different system components are subclassed from the *DanceObjectList* class. The *DanceTcl* class contains the Tcl scripting commands for the DANCE system.

plementing *FindClosest* (Section 5.10.1), utilize an internal list of *DanceObjects* (comprising musculotendons and bones). By defining the list contents to be of type DanceObject, newer implementations or different classes can be added without needing to change the algorithm's data structures.

7.3 Base Class

From the common *DanceObject* class, several important base abstract classes are derived that define the major roles of all components in DANCE. Although the specific application we describe here is an anatomically-based modeller and animation system, the same software architecture can be used to create other physically-based applications.

The subclasses are implemented either as plug-ins or core components (Figure 7.1). All subclasses can actually be plug-ins, but we anticipate that some components will be universally needed by many applications and have built them into the main system. Core component classes reside as part of the main executable along with the DANCE driver class and contain the final implementation, not being subclassed further. The advantage of using plug-ins is that modification or enhancements to a model can be done independently of other components and the implementations can be subsequently shared with other practitioners (Figure 7.3).

The main plug-ins consist of: *System*, *Actuator*, *Simulator*, and *Geometry* classes. These are expected to present the largest variation in implementation instances and are subclassed extensively. As the classes are abstract, a simple interface is enforced from the perspective of the DANCE driver. This allows container class operations to be performed at a high-level of abstraction. For example, routines to visually display all systems only need call the `output` method for each system instance stored with the *SystemManager* list (Figure 7.2).

The *System* class represents any entity that can be described by a set of state variables and

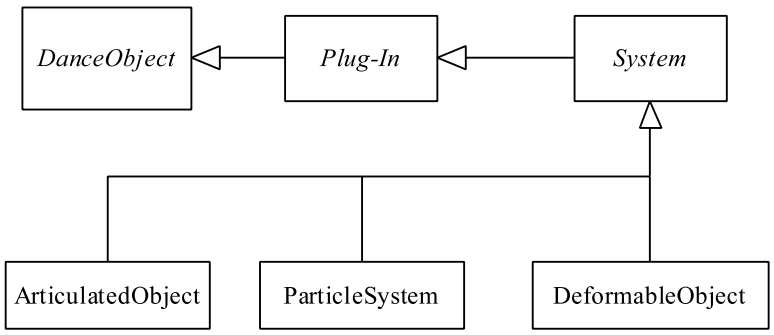


Figure 7.3: Plug-ins are a subclass of the base *DanceObject* class. This abstract class is instanced to implement different components. In this example, the *system* abstract class is a type of plug-in that in turn can be subclassed to create articulated objects, particle systems, and deformable objects.

that will undergo physical change of state. Any physical system that can be described with a set of generalized coordinates is valid. In the anatomical modeller, the articulated skeleton is a system whose configuration is specified with the degrees of freedom of each joint in the figure.

The *Geometry* class provides a visual representation for parts of the system. For example, each link in the articulated skeleton has its own bone geometry. We subclass Geometry to produce an *IndexedFaceSet* that can represent standard polygonal meshes. Within this class, we have implemented the routines for computing mass properties (Section 6.1.1) and encapsulating collision detection routines for storing oriented bounding boxes (Section 5.10.2).

Actuators exert forces or torques on systems. They are the main creators of motion in a physically-based system. A diverse range of physical phenomenon can be modelled. Actuators in our system range from simple, gravitational fields to the biomechanical force models of the musculotendons. Actuators can have their own geometric representations and state variables. For example, B-spline solids and their control points are used to depict musculotendons and ligaments. An important distinction between actuators and systems is that the former can exert forces or torques on the environment. Controllers can be constructed as special kinds of actuators that contain instructions for the generation of forces and torques based on state and time variables. Actuators can manage other actuators through hierarchical relationships.

Simulators update the state of systems over time, using either physically-based or kinematic processes. In our anatomical system, the articulated figures can be simulated by integrating the equations of motion to provide physically-based motion in the skeleton. Alternatively, a kinematic simulator can drive a system through a pre-defined set of motion curves. The ability to have several different simulators coexisting allows the state variables of all systems to be paritioned among different simulators. Systems with very different natural time frequencies in their motion can be assigned appropriate simulators using integration techniques specially designed to control the numerical stability of that subset of state variables. Both the skeleton and B-spline solids have their own simulators in our application.

There are several, minor classes uses for interaction and graphical display: *views* and *lights*. Multiple viewpoints of the scene are handles with the *view* class. Each view can be rendered differently (solid or wireframe shaded), and illuminated with several light sources. Each view has a direct manipulation interface for navigating the scene and manipulating various sys-

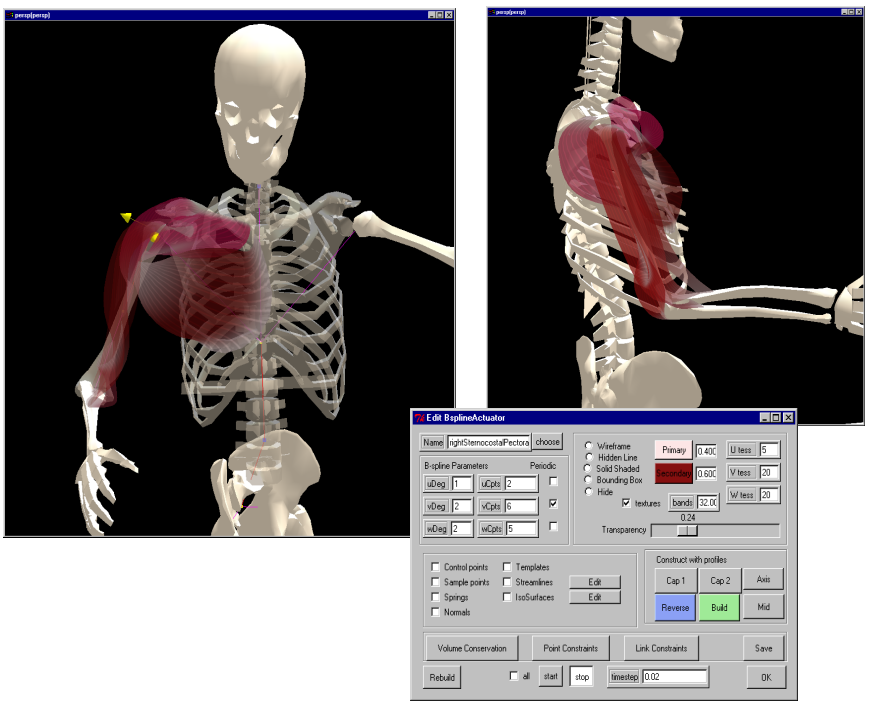


Figure 7.4: Each view can have a different perspectives of the same scene. Scene elements, such as joints, can be directly manipulated. GUI elements are completely external to the main DANCE system.

tem and actuators parameters. The camera angle can be changed both in static and animated displays, allowing unobstructed views for observation. Figure 7.4 illustrates a multiple view session with the anatomical modeller.

7.4 Application Programming Interfaces

To ensure multiplatform operation, application programming interfaces (APIs) were chosen that are available on all major operating systems. OpenGL was used for 3-D graphics, Tcl/Tk for the scripting and graphical user interface, and GLUT for window management and input event handling. Consequently, DANCE can be run on Windows and Unix-based operating systems.

7.5 Summary

In this chapter, we provided an overview of the DANCE system architecture on which the anatomically-based system was constructed. The architecture is important for enabling efficient communication and coordination of various system features. An open architecture enables different parts of the system to be continually enhanced and modified to experiment with different approaches to modelling anatomical components such as muscle models. The actuator base class allows us to model muscles as force-generating elements that can influence the

state variables of systems like the skeleton. An interesting potential use of the ability for actuators to control other actuators is the development of coordination algorithms to explore control of muscles to create useful motions.

Chapter 8

Results

> An acre of performance is worth the whole world of promise.
>
> **James Howell**

This chapter illustrates several examples demonstrating different aspects of the anatomic modelling system. We present three different applications of the system from each of the anatomical, biomechanical and computer graphics perspectives.

8.1 Contraction of soleus muscle

For muscle simulations requiring high accuracy, especially to capture internal fibre orientations, we used the fibre set CVSF (developed in Section 4.5) to reconstruct B-spline solid muscles that preserve fibre orientation in the volume as well as on the surface. In particular, we have constructed a virtual model of the posterior soleus from a human cadaver. We modelled the aponeurosis layers (tendon plates) on the top and bottom faces of the posterior soleus as a passive, elastic layer of viscoelastic links. A global damping coefficient was applied to each spatial point of the muscle model to account for internal friction and viscosity between and within muscle fibres.

For the contractile forces, we implemented a Hill-based muscle model as described in [124]. Since our muscle model offers finer structural detail than the lumped-architectural models previously used in biomechanics [93, 24, 124], we made several adjustments to the Hill model used. Although there are some series elastic effects in muscle fibre, the contribution from tendon dominates. Since we model the tendon attachments separately from the muscle fibres, we only need the contractile element (Section 2.2.8) in our Hill model to create a simpler Hill configuration. We did not include the parallel element because we are performing a contraction simulation where parallel element effects are negligible or non-existent. Being able to examine muscle at finer architectural detail permits better study of the inter-relationships of muscle fibre parameters for functional study.

Unfortunately, several muscle parameters provided in [124] could not be applied to our simulation. Parameters such as maximum isometric length and force were computed from averaged measurements or taken in directions different from the fibre directions in our model. The pennation angle parameter is inapplicable in our soleus reconstruction because we model

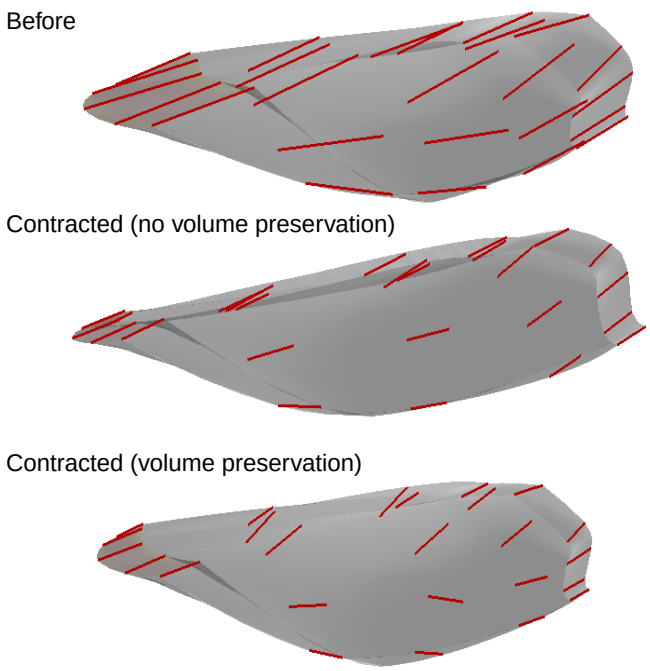


Figure 8.1: Soleus model (no aponeurosis) is contracted with and without volume conservation. Simulation runs at a rate of 14.5 Hz on a Pentium III 500 MHz CPU.

the fibre orientations explicitly in three dimensions. In contrast, the pennation angle is another averaged parameter for fibre angles. In order to avoid manually specifying parameters for each of the numerous individual fibres of our model, we used single ratios that set the isometric length parameters to be a percentage of the current fibre lengths at the start of our simulation. For the maximum isometric force, we used the same value for all fibres. Unfortunately, the accuracy of our parameter approximations is uncertain because there is currently no available experimental data to compare with. One goal of providing this model is to help direct the design of new experiments that can be used for future validation and detailed functional study of muscle.

In Figure 8.1, we display the results of two simulations of a contracting posterior soleus with and without volume preservation. We also removed the elastic aponeurosis layer. Figure 8.2 graphs the change in volume during the simulation. At all times, the deviation in volume never exceeded one percent of the target volume.

In comparison, Figure 8.3 depicts the same set of simulations with an passive elastic aponeurosis layer. Figure 8.4 depicts the volume graph. For both scenarios with and without aponeurosis, the effect of volume preservation is the pronounced change of fibre orientation during contraction. Due to additional constraints in the top and bottom layers of the aponeurosis-equipped soleus, changes in fibre orientation are slightly greater than without an aponeurosis. These changes in fibre orientation qualitatively match similar fibre orientation changes observed in ultrasound images of contracting muscle fibres (Figure 8.5). Since the Hill model features a limited range of muscle lengths where active force development can occur, initiating contraction by raising the activation level of the fibres resulted in the fibres automatically stopping their contraction once they had shortened enough. In the aponeurosis-

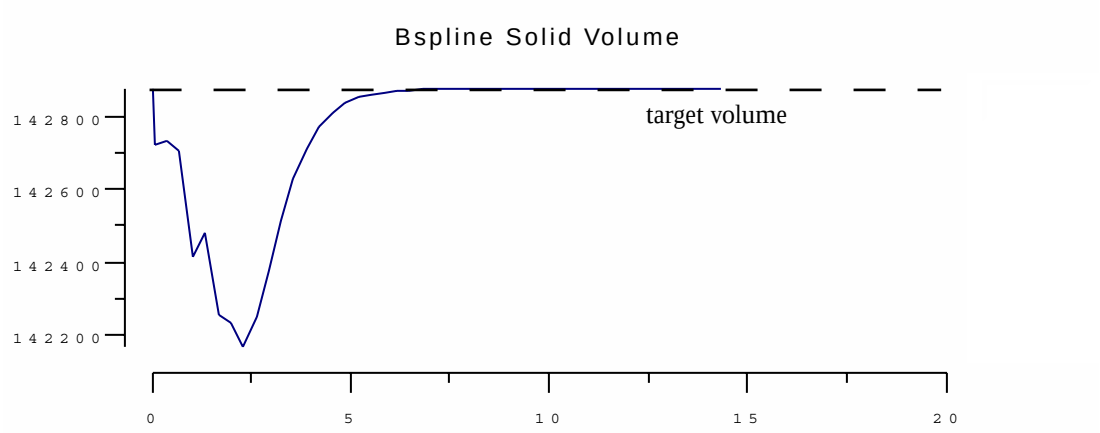


Figure 8.2: Volume graph depicting changing volume during simulation of a posterior soleus (no aponeurosis). The target volume level is denoted by the dashed line. Volume computations can be performed at rates higher than 100 Hz on a Pentium III 500 MHz CPU.

equipped soleus, active force development continued during equilibrium. In contrast, tension disappeared altogether in the non-aponeurosis soleus. A possible explanation for this phenomenon is that muscles fibres need to actively pull against the tension caused by the aponeurosis layer. In the absence of an aponeurosis, the muscle fibres are free to contract against a reduced load. We found that damping played an important role in achieving equilibrium by dissipating the energy created during contraction.

The volume graphs demonstrate that it takes a slightly longer time for the aponeurosis-equipped soleus to reach the target volume. This is due to the volume-preservation forces working against the elastic restorative forces in the aponeurosis. In both graphs, the initial deviation in volume is caused by the onset of muscle contraction. Gradually, the volume restorative forces adjust the shape to reach the target volume. The rate at which this occurs is directly related to the stiffness parameter assigned to the volume-preserving gradient force (Section 5.5).

This example demonstrates the effective use of the B-spline solid muscle for detailed architectural and functional studies of muscle. The next example shows how the model can be used in a biomechanical situation with an articulated skeleton.

8.2 Equilibrium Point Hypothesis Testing

The *Equilibrium Point Hypothesis* [56] states that limb configurations are the result of an equilibrium condition created by the forces acting on a skeleton, including forces due to active musculotendons. As a limb moves from pose to pose, the changing muscle activations supposedly create a new equilibrium condition that corresponds to the new pose. Although later studies have shown that the *Equilibrium Point Hypothesis* is not valid in every situation, it is still a convenient idea for explaining various muscle coordination phenomena.

In Figure 8.6, we have outfitted a skeleton with a pair of antagonistic musculotendons. We used a simple elastic model that generates force if the musculotendon length deviates from

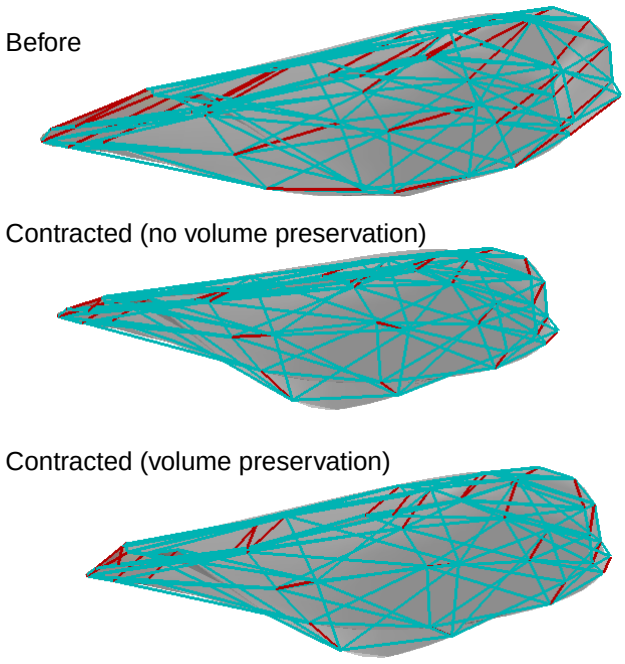


Figure 8.3: Aponeurosis-equipped soleus model is contracted with and without volume conservation. Simulation runs at a rate of 12.8 Hz on a Pentium III 500 MHz CPU. This rate is slightly lower than the soleus without aponeurosis (Figure 8.1) due to the larger number of viscoelastic links in the aponeurosis.

a target rest length. The musculotendons exhibit slackness (generate no force) whenever the musculotendon is shorter than the target length. The left image of Figure 8.6 shows the skeleton without any muscles in static equilibrium with gravity. The centre image has the limb equipped with passive muscles. Even in the absence of active force generation, the passive muscles exhibit enough tension to change joint angles. In the third case, we activate the muscles and achieve a new pose for the limb. Small damping forces were added to the joints to simulate the friction created between joint articular surfaces and sinovial fluids. The damping also aids the musculotendons by allowing the forces to be dedicated mainly to limb motion rather than stabilizing excess joint motion due to a lack of energy loss that would normally occur in nature. In Figure 8.7, we replaced the simple muscle force model with a Hill-based model used by [124] and simulated the resulting musculoskeletal system. From the baseline simulation in the top left image, we varied a single parameter of the left musculotendon in the other three images. With this technique, sensitivity analysis can be done to determine the possible functional effects of changing muscle parameters. This can be used for reconstructive surgery planning to determine any changes in range of motion due to altered muscle characteristics after surgery. Notice that the final poses created separately by reducing activation and reducing maximum isometric force result in similar poses. Due to the redundancy of muscles and the force-length-velocity dependencies, these simulations can be used to determine alternative strategies for completing motion tasks if one property of muscle becomes deficient. Simulations can also be performed to help determine muscle parameters that would be required to achieve various motions or tasks.

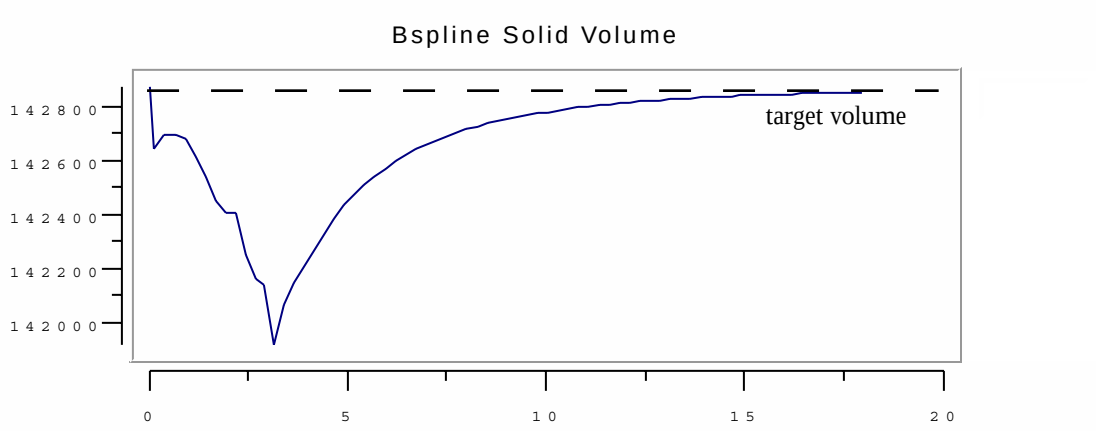


Figure 8.4: Volume graph depicting changing volume during simulation of a posterior soleus with aponeurosis layers attached. The target volume level is denoted by the dashed line.

8.3 Deforming Skin

In computer graphics, anatomically-based deformations can be achieved to deform a surrounding skin model. Figure 8.8 illustrates the various layers that can be modelled in the construction of a synthetic creature for animation. By binding the points of the skin geometry to the underlying skeleton or musculature as describe in Section 6.4, we can deform the skin as the underlying anatomy changes. Figure 8.9 depicts various instances of skin deformation due to underlying anatomy in portions of a *Parasaurolophus* dinosaur model (Figure 6.6). The top images show the effects of "shrink-wrapping" the skin close to the musculature. This has the visual effect of revealing greater muscle definition. The bottom image retains the relative distance of skin points to the underlying geometry while deforming the model. Generally, several portions of the same model can be assigned different constraints to account for varying distributions of fat along the body. Figure 8.10 shows how the effects of stretched tendon in limb motions can be propagated to a skin model.

8.4 Summary

This chapter presents a cross-section of different examples that illustrates the unique contributions of the anatomic modeller. The contracting soleus demonstrates a high level of detail that captures accurate fibre orientations, shape and functional behaviour in anatomy. Multiple muscle simulations with an articulated skeleton can be performed for motion analysis or to test different control strategies for muscle coordination in biomechanics. The muscle models not only have visual characteristics, but are capable of physically applying forces to generate motion in the attached skeleton. For more creative applications in computer graphics, the anatomic modeller can be used as a feature-based deformation technique for modelling skin deformations in animals. The examples present evidence that the musculotendon models can be scaled to different levels of detail for various applications.

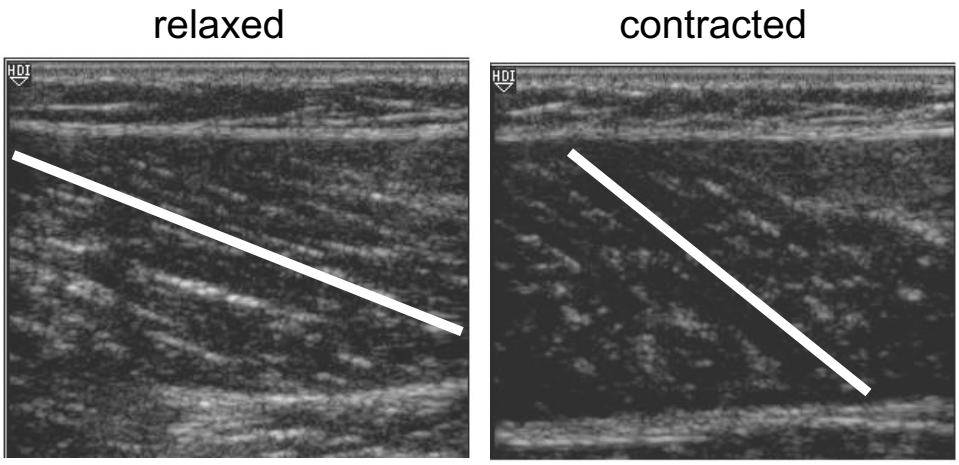


Figure 8.5: Muscle fibres are shown in relaxed (left) and contracted (right) states in ultrasound imagery. Notice the pronounced change in fibre orientation (white line) from the relaxed to contracted state.

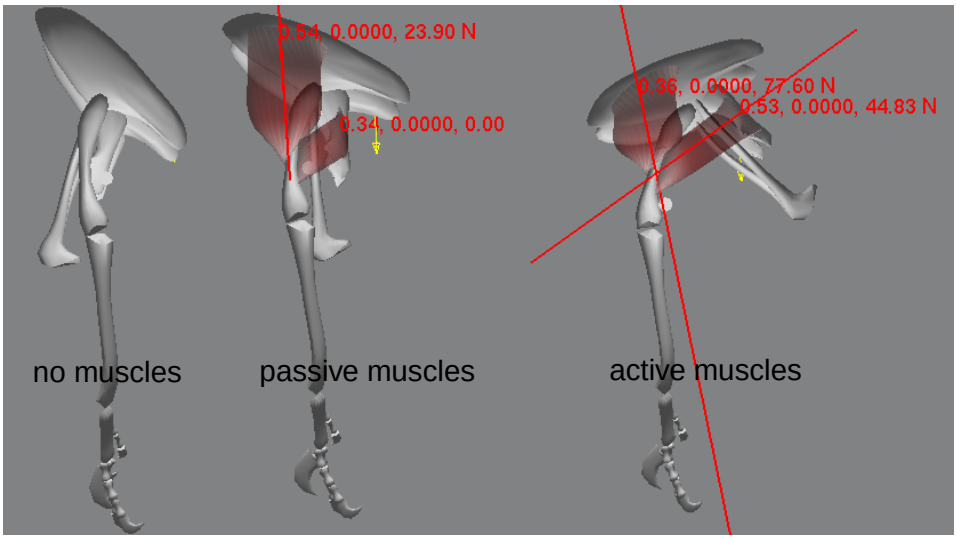
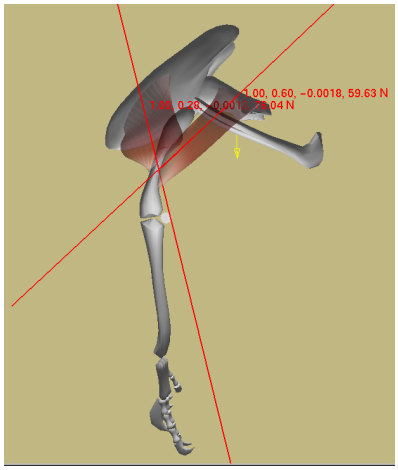
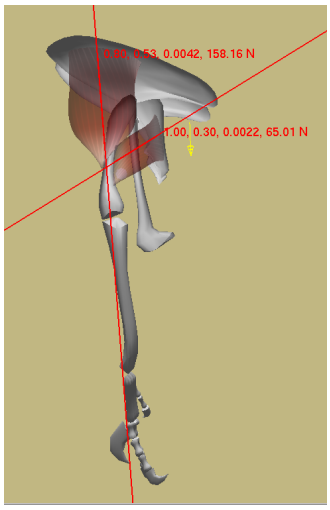


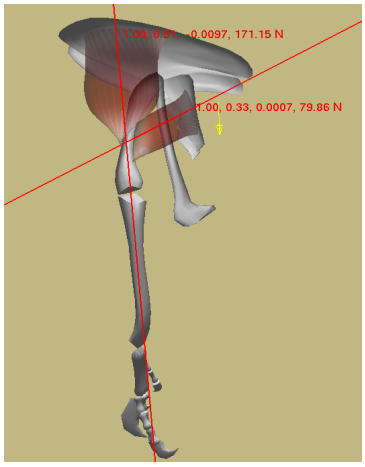
Figure 8.6: The presence of activated muscle can change the pose configuration of a skeletal limb.



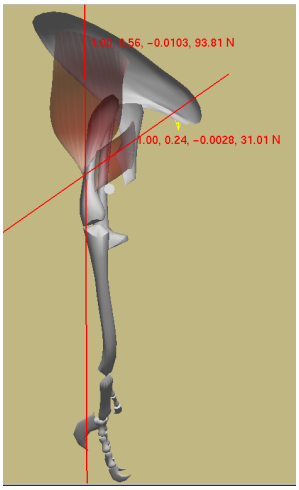
baseline simulation



80% activation



130% isometric rest length



50% maximum isometric force

Figure 8.7: One Hill model parameter is adjusted from the baseline simulation in the top left corner.

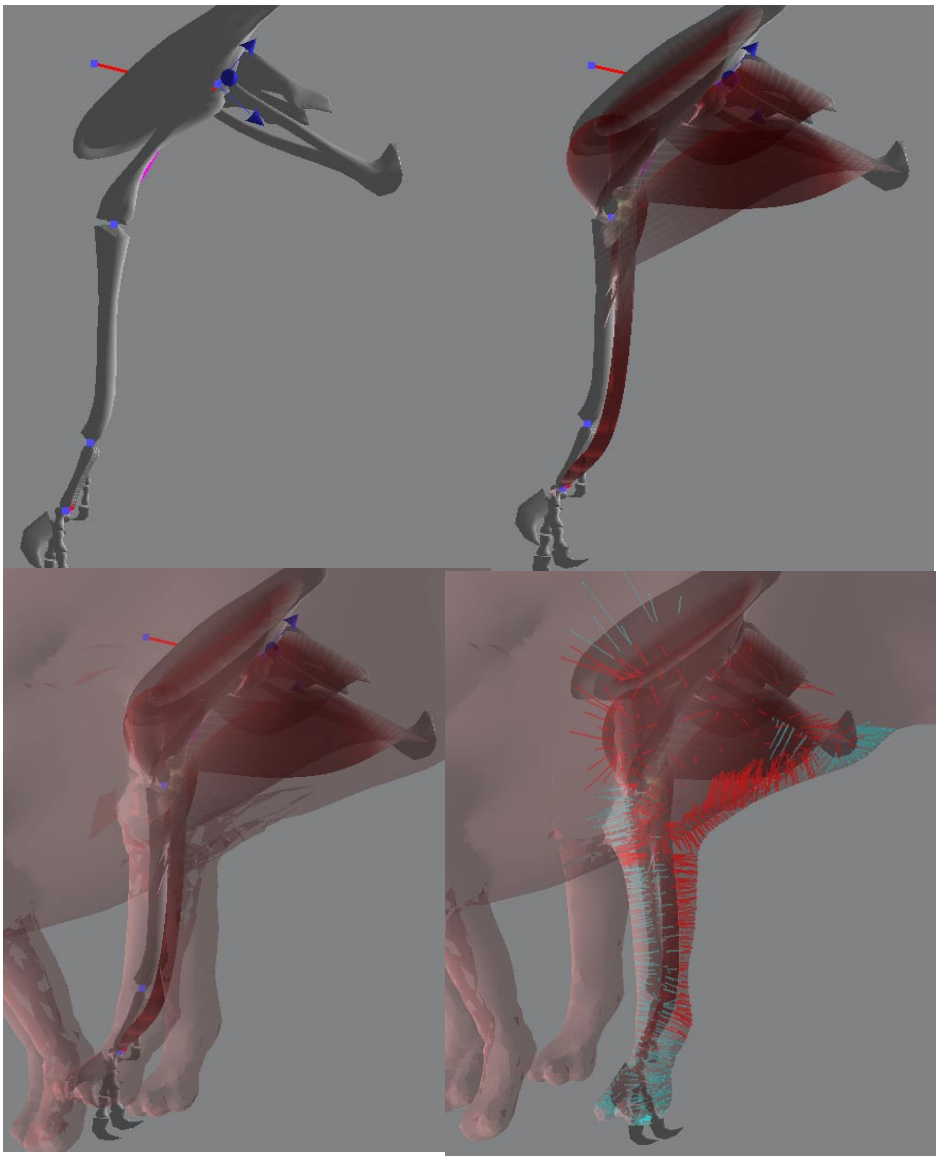


Figure 8.8: The construction of an animal can be thought of as a layering process from bone to musculature to fat and skin. In the bottom right, the red links indicate associations with musculotendon while the blue links are with bone.

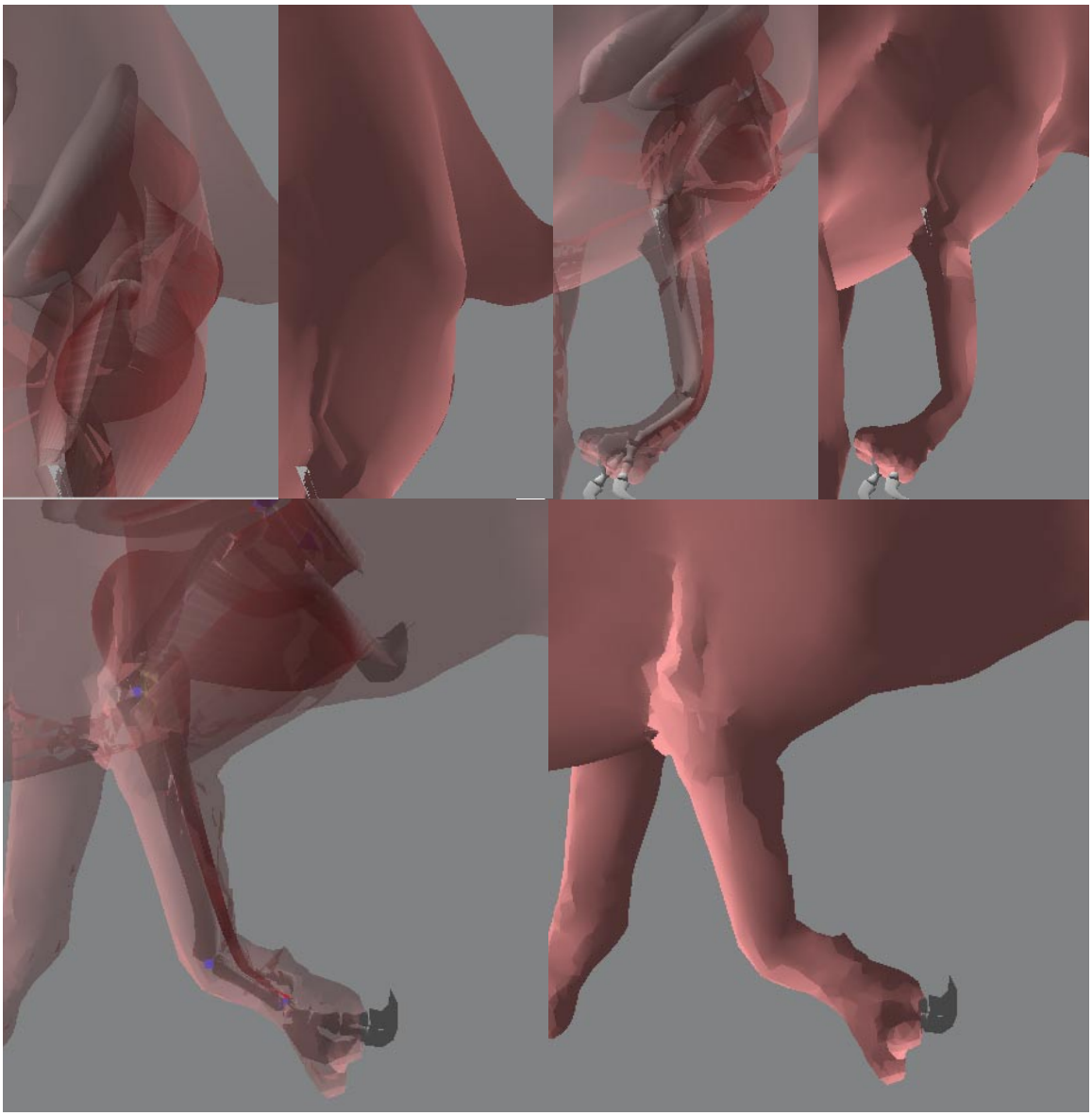


Figure 8.9: Various examples of how underlying musculature and bones can deform a surrounding skin model. Scenes are shown with adjacent views of transparent and opaque skin.

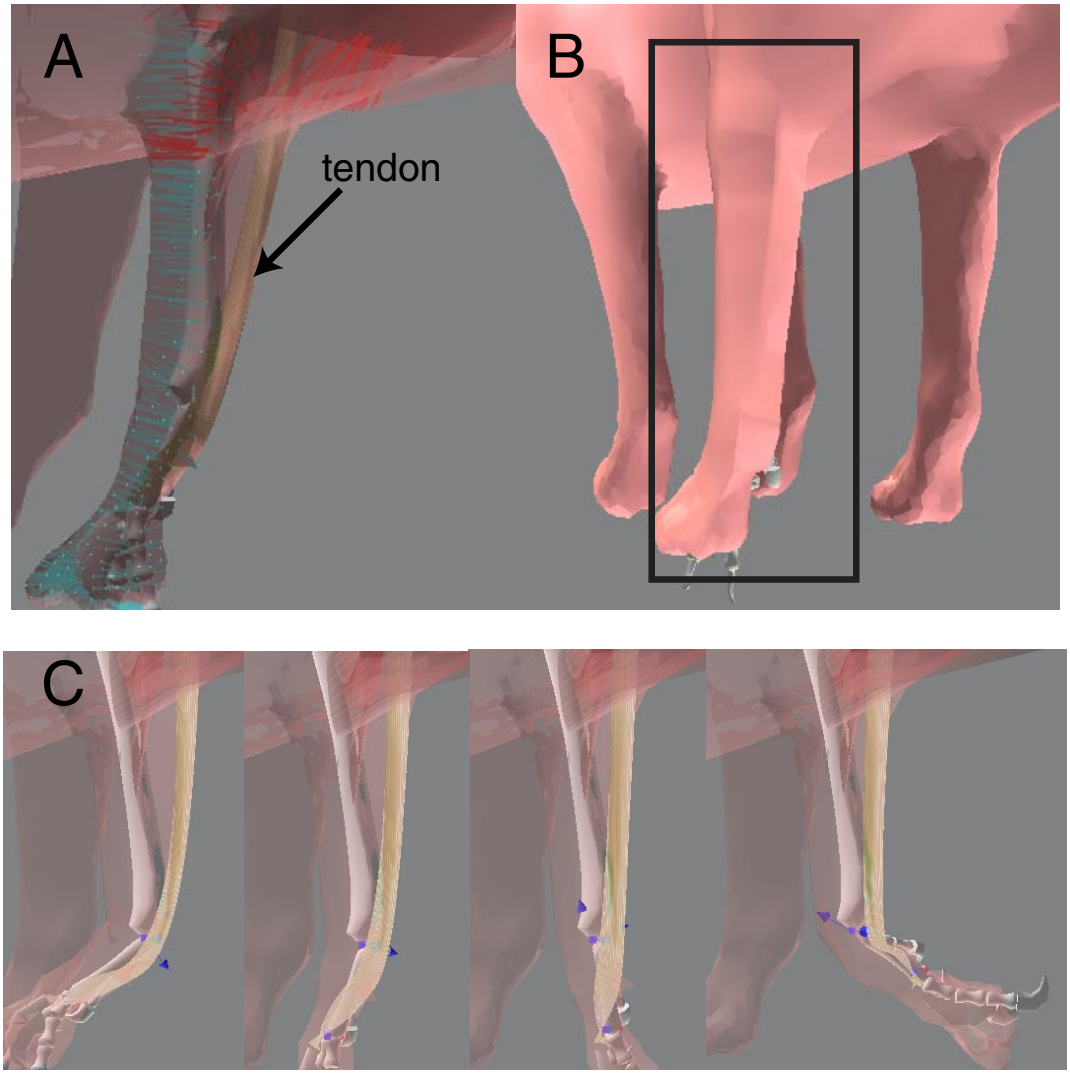


Figure 8.10: A yellow tendon straightens out as it is stretched and deforms the surrounding skin near it. A) The links between the skin and underlying anatomy. B) Superficial view of the tendon creasing the skin along the back of the left leg. Skin updates are performed at a rate of 8.3 Hz on a Pentium III 500 MHz CPU.

Chapter 9

Conclusion

> Nothing tends so much to the advancement of knowledge as the application of a new instrument.
>
> **Sir Humphrey Davy**

The process of creating an anatomic modeller revealed important problems that needed to be identified and addressed. One of the first steps is deciding the scope and relative importance of anatomic components to model in the system. The musculotendon unit is a key ingredient of the modeller, being responsible for the dual roles of superficial body shape definition and force actuator of locomotion. Recognizing this important, intertwined relationship between form and function initiated a search for a suitable mathematical primitive for musculotendon that culiminated in the choice of the B-spline solid (Chapter 3). Once the B-spline solid was selected, several subproblems arose that required solutions: the development of methods to define initial musculotendon shapes from various data sources (Chapter 4) and the formulation of physically-based models for shape deformation and force generation (Chapter 5).

9.0.1 Geometric shape definition

For geometric shape definition, we devised a data-fitting methodology that solves for the control points of the B-spline solid given a set of spatial points. The spatial points are generated from a Continuous Volume Sampling Function (CVSF) that can be defined from different data sources. This mechanism provides a clear procedure for converting discrete, sampled data to a continuous solid model. We illustrated the power of the technique using data from the Visible Human data-set, digitized fibre sets, and profile curves. The fact that B-spline solids can be fitted to these very different data sources demonstrates the versatility of the CVSF procedure as it is applied to various representations of muscle.

9.0.2 Physically-based models

Being able to represent a smooth, solid shape with a compact set of control and spatial points allows multiple musculotendon simulations to be conducted without a prohibitively large number of parameters, as may be the case with some finite element methods. The spatial points

provide direct manipulation handles for shape modulation, but also allow masses and forces to be assigned to these positions. For example, spring-mass networks or reaction constraints can be defined directly at these spatial points. For global forces based on potential energy functions, Lagrangian equations of motion can be formulated for the parameters of B-spline solids. We developed in Chapter 3 a closed-form expression for volume and used it to develop potential energy functions that enabled dynamic simulation of volume-conserving shape deformations.

9.0.3 Collision and proximity tests

For collision detection and proximity tests, we created bounding volumes using oriented bounding boxes for polygonal bone geometry and least-squares techniques for B-spline solids. Collecting all the bounding volumes into a lower-upper bound tree allowed many, successive tests to be conducted rapidly. Being able to find the closest feature point on a musculotendon or bone to an arbitrary 3-D point allows useful operations to be performed. Local point-to-plane reaction constraints can be established between objects for collision resolution and skin points can be associated with underlying anatomy to deform the skin as the anatomy moves and changes shape.

9.0.4 Inter-relationships between anatomic components

From a software engineering perspective, there was the practical problem of developing a software architecture that would enable inter-communication between the various components of an anatomic modeller. To attach musculotendons to bone, we described an intuitive direct manipulation interface for sketching musculotendons onto the surfaces of bones using profile curves to specify insertion, origin and axial curves (Section 4.6). In Chapter 7, we described an object-oriented software architecture, where functionality can be added or modified through plug-ins. In particular, musculotendons are treated as force actuators that have their own state variables and geometric representation.

9.0.5 Other anatomic system elements

Although a large portion of this thesis concentrated on musculotendons, models for other anatomic components were investigated (Chapter 6). We computed mass properties for bone geometry and redefined the local frames of reference to produce diagonal inertia tensors for efficient computation during simulation. Joint transformations were defined that account for changing rotation axes and centres of rotation during joint articulations. Ligaments can be modelled using the same B-spline solid primitives with different physical properties. Finally, we present how the underlying assembly of skeleton, musculotendon, and ligaments can be used as a feature deformation technique for propagating changes in body shape to a surrounding skin geometry muscle.

Techniques that bind skin geometry to underlying anatomically-based structures can exhibit undesirable stretching or creasing. This occurs because individual points on the skin are transformed without considering the viscoelastic forces they may experience from adjacent portions of skin. By approximating the skin as an elastic mesh, relaxation techniques

demonstrated by Wilhelms and van Gelder[130] can evenly distribute the skin mesh to eliminate excessive stretching, but may smooth out desirable details. One solution is to allow an animator to interactively map different elasticity properties over the entire skin geometry to add regional control to the amount of stretching desired.

9.0.6 Form and function

These various contributions to the anatomic modeller, consisting of both geometric and physical models, can be used to culture further investigation in the relationship between form and function of the anatomic elements of humans and other animals. Returning to the three perspectives of anatomy, biomechanics and computer graphics, the modelling system developed in this thesis takes steps towards addressing needs in each of these cases.

Detailed functional study of muscle fibre architecture throughout a muscle's volume can be done directly from digitized, cadaveric specimens. Since pennation effects can be studied on a per-fibre level, this finer degree of resolution can provide potentially more accurate estimates of the force-length properties of musculotendon because it avoids using a single, averaged pennation angle term for calculating the forces generated by an entire muscle. In surgical procedures, such as tendon lengthening surgery, simulations with a more refined model can improve the chances of obtaining desired post-operative behaviour of modified musculotendons.

The use of profile curves to specify insertion and origin regions of muscle more accurately captures the mechanical situation in the tendon-bone interface than using piecewise line segments. The dynamics within a single muscle can be simulated independently from the attached skeleton, allowing effects such as isometric contraction, muscle activation independent of joint motion, and inertial oscillations in muscle. As volume preservation and collision reaction is taken into consideration, we can begin to investigate the inter-muscle forces generated in packed muscles, in addition to contractile forces of muscle. Deforming skin directly due to underlying bone, musculotendon and fat can help animate humans and other animals realistically in an automatic fashion for animation applications in computer graphics.

9.1 Future work

There remains many areas that can be improved and investigated for creating the ideal anatomic modeller.

9.1.1 Musculotendon model

The B-spline solid model for musculotendons attempts to combining the geometric and physical aspects of muscles. Previously, models focused on one aspect to the exclusion of the other. Currently, to model muscles which branch out from a common tendon region, a separate B-spline solid would be required for each branch, with various constraints applied to attach them together. It would be advantageous to model a branching network of muscles with a single, unifying mathematical framework, where the branches are implicitly part of the model. Perhaps subdivision solids[78] can be used, but a global parameterization for the muscle's volume would need to be created to reference individual muscle fibres.

9.1.2 Skin model

Skin geometry models that are adapted to fold, wrinkle and bend with the contours of underlying anatomy need to be designed. Subdivision surfaces and other multiresolution schemes that allow more definition to be seen in regions of high curvature are promising candidates for adaptable skin models. As an alternative to modelling geometry directly, displacement or bump mapping techniques can be used to add fine wrinkles and folds to the skin geometry. In this case, the challenge lies in procedurally generating these texture maps based on geometric characteristics of the skin.

9.1.3 Reconstruction applications

The existence of an anatomic modeller that allows information to be shared between its various components encourages interesting reconstruction projects. Synthetic animals or muscular reconstruction of fossilized skeletons can produce visualizations of non-existent animals in a more vivid and animated manner than previous methods. If reconstruction of an animal follows basic musculature patterns of related species with similar skeletal structure, a high-level parameterization of anatomically-accurate muscle deformation in animals can be achieved. These evolutionary relationships in musculature between closely-related species of animals can potentially be captured and parameterized to allow automatic muscle reconstruction of a given skeleton.

9.1.4 Control and functional study

In the area of motor control research, muscle models that have physically-based properties can be used to develop mechanisms for muscle coordination and control in animals. This would require whole body simulations of multiple muscle systems where each muscle is capable of receiving an activation signal with accurate contraction dynamics. Using these muscles as force actuators, the resulting motions created in the underlying skeletons can be observed.

On a finer scale, functional studies that investigate the role of muscle fibre architecture and the corresponding forces and shape changes produced, may lead to explanations of the natural design of anatomical structures in the body. With B-spline solid models produced from the fibre set data, motor unit modelling can be applied to distinct regions of muscle fibres. This would allow study of innervation patterns within a single muscle and their influence on the gradual recruitment of fibres during muscle contraction. A completely functional model of muscle that accounts for fibre architectural changes during contraction under various conditions will help predict force production more accurately. Being able to perform simulations in a completely virtual environment would allow important investigations of muscle behaviour under unusual conditions, such as microgravity (Earth orbit conditions).

9.1.5 Need for validation

With the existence of a system that can allow these kind of musculoskeletal simulations, it is important that methods are developed for proper validation of the results with biomechanical analysis of real muscles. In many cases, the data necessary for comparative analysis is currently

not available. The parameters used in our physical models should serve as a guide for experimental design on proper data collection techniques for validation. Important parameters to determine include spring and damping coefficients used in dynamic simulation and the proper Hill parameters used in muscle force models. The latter is especially important because fibre force production is handled at a finer resolution in the B-spline solid model than traditional lumped-parameter models. As the model permits generation of nonuniform force generation within a muscle as well as inter-muscle forces due to contact with other bones and muscles, new experiments in biomechanics need to be designed to obtain accurate measurements of the corresponding situations in humans and animals, preferably *in vivo*.

To take advantage of existing data, virtual simulations of classical experiments [45] for determining the force-length-velocity properties of muscle can be performed. These experiments can include isometric muscle length and isotonic muscle load conditions. Virtual sensors can be placed on the muscle model to coincide with the physical locations of measurement apparatus in the original experiments. In this manner, data points can be plotted to reconstruct graphs displaying force-length or force-velocity relationships. If the simulated data can be shown to be quantitatively similar to classical results, a case for validation of the model can be made.

9.1.6 Digital humans and animals

The components included in this anatomic modelling system are responsible for locomotion and body shape. To provide a complete system for general medical applications, we would require many other internal organs for the successful emulation of a fully-functional digital animal. In addition to the effort required to model individual organs, the software architecture that will enable these organs to inter-relate with one another would have to be established. Different components must be aware of each other physically and will depend on each other to function properly. With these challenges in mind, the concepts presented in this thesis are offered as a starting point for further research in this exciting, multidisciplinary area.

Appendix A

Volume Computation of a B-spline Solid

Recall that we can compute the volume of a B-spline solid as follows:

$$\mathcal{V} = \mathcal{C}^T \cdot \mathcal{B}. \tag{A.1}$$

We will show that an element of $\mathcal{B}$ becomes zero whenever any combination of three u, v and w basis functions exists that lie totally within their parameter boundaries. Geometrically, this translates to all three basis functions having a domain that lies entirely within the solid, excluding its boundary. Therefore, we will show that the volume of a B-spline solid can be computed solely as a function of its boundary control points. In the following proof, we collapse multiple integral symbols to a single integral and remove the indices of integration for clarity.

A.0.7 Proof

Recall that an entry of $\mathcal{B}$ has the form:

$$\begin{aligned}
\mathcal{B}_{ijklmnopq} &= \int_I \det(u, v, w) dwdvdu \\
&= \int \underline{u}_{ijk}\underline{v}_{lmn}\underline{w}_{opq} dwdvdu + \int \underline{u}_{lmn}\underline{v}_{opq}\underline{w}_{ijk} dwdvdu + \int \underline{u}_{opq}\underline{v}_{ijk}\underline{w}_{lmn} dwdvdu \\
&- \int \underline{u}_{opq}\underline{v}_{lmn}\underline{w}_{ijk} dwdvdu - \int \underline{u}_{lmn}\underline{v}_{ijk}\underline{w}_{opq} dwdvdu - \int \underline{u}_{ijk}\underline{v}_{opq}\underline{w}_{lmn} dwdvdu.
\end{aligned} \tag{A.2}$$

In subsection 3.4.1, we showed how each term in (A.2) can be converted into the product of three separate integral operations by factoring the integrand into three separate univariate functions. For example, we do this for one of the terms of (A.2):

$$\begin{aligned}
&\int \underline{u}_{ijk}\underline{v}_{lmn}\underline{w}_{opq}dwdvdu \\
&= \int \frac{d}{du}B_i^u(u)B_j^v(v)B_k^w(w)B_l^u(u)\frac{d}{dv}B_m^v(v)B_n^w(w)B_o^u(u)B_p^v(v)\frac{d}{dw}B_q^w(w)dwdvdu \\
&= \int \frac{d}{du}B_i^u(u)B_l^u(u)B_o^u(u)du \cdot \int \frac{d}{dv}B_m^v(v)B_j^v(v)B_p^v(v)dv \cdot \int \frac{d}{dw}B_q^w(w)B_k^w(w)B_n^w(w)dw.
\end{aligned} \tag{A.3}$$

We will use the following notational shorthand for conciseness:

$$\begin{aligned}
\mu_{ijk} &= \frac{d}{du}B_i^u(u)B_j^u(u)B_k^u(u)du \\
\nu_{lmn} &= \frac{d}{dv}B_l^v(v)B_m^v(v)B_n^v(v)dv \\
\omega_{opq} &= \frac{d}{dw}B_o^w(w)B_p^w(w)B_q^w(w)dw.
\end{aligned} \tag{A.4}$$

Rewriting (A.2) using this notation, we have:

$$\begin{aligned}
&\mathcal{B}_{ijklmnopq} \\
&= \int \mu_{ilo}du \int \nu_{mjp}dv \int \omega_{qkn}dw + \int \mu_{lio}du \int \nu_{pjm}dv \int \omega_{knq}dw + \int \mu_{oil}du \int \nu_{jmp}dv \int \omega_{nkq}dw \\
&- \int \mu_{oil}du \int \nu_{mjp}dv \int \omega_{knq}dw - \int \mu_{lio}du \int \nu_{jmp}dv \int \omega_{qkn}dw - \int \mu_{ilo}du \int \nu_{pjm}dv \int \omega_{nkq}dw.
\end{aligned} \tag{A.5}$$

We can rewrite (A.5) as a new determinant:

$$\mathcal{B}_{ijklmnopq} = \begin{vmatrix} \int \mu_{ilo}du & \int \mu_{lio}du & \int \mu_{oil}du \\ \int \nu_{jmp}dv & \int \nu_{mjp}dv & \int \nu_{pjm}dv \\ \int \omega_{knq}dw & \int \omega_{nkq}dw & \int \omega_{qkn}dw \end{vmatrix}. \tag{A.6}$$

If we can show that any of the rows or columns in (A.6) are linearly dependent, then the determinant will evaluate to 0. First, let us look at the μ terms in each column of the determinant and apply the chain rule:

$$\int \mu_{ilo}du = \int \frac{d}{du}B_i^u(u)B_l^u(u)B_o^u(u)du = \int B_l^u(u)B_o(u)d(B_i^u(u)). \tag{A.7}$$

Similar applications to the other μ terms results in:

$$\begin{aligned}
\int \mu_{lio}du &= \int B_i^u(u)B_o(u)d(B_l^u(u)) \\
\int \mu_{oil}du &= \int B_i^u(u)B_l(u)d(B_o^u(u))
\end{aligned} \tag{A.8}$$

We can expand the integrals using integration by parts:

$$\begin{aligned}
&\int \mu_{ilo} du \\
&= B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} - \int B_i^u(u) d(B_l^u(u)B_o^u(u)) \\
&= B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} - \int B_i^u(u)B_l^u(u) d(B_o^u(u)) - \int B_i^u(u)B_o^u(u) d(B_l^u(u)). \\
&= B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} - \int \mu_{oil} du - \int \mu_{lio} du
\end{aligned} \tag{A.9}$$

Similarly,

$$\begin{aligned}
\int \mu_{lio} du &= B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} - \int \mu_{oil} du - \int \mu_{ilo} du \\
\int \mu_{oil} du &= B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} - \int \mu_{lio} du - \int \mu_{ilo} du
\end{aligned} \tag{A.10}$$

Similar expressions can be derived for the ν and ω entries. We now have expressions that relate the entries of the three columns of the determinant matrix. For the determinant to be zero, an *entire* column must be linearly dependent with the other columns.

Showing linear dependency of the columns

The final step is to show that the columns of the determinant matrix in (A.6) will always be linearly dependent whenever there exists a combination of three u, v, and w basis functions that are defined entirely within the solid's boundaries.

From the previous section (see equation (A.9)), the relationship between the columns of (A.6) can be expressed as:

$$\begin{aligned}
\int \mu_{ilo} du &= B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} - \int \mu_{lio} du - \int \mu_{oil} du \\
\int \nu_{jmp} dv &= B_j^v(v)B_m^v(v)B_p^v(v)|_{v_0}^{v_{r+1}} - \int \nu_{mjp} dv - \int \nu_{pjm} dv \\
\int \omega_{knq} dw &= B_k^w(w)B_n^w(w)B_q^w(w)|_{w_0}^{w_{r+1}} - \int \omega_{nkq} dw - \int \omega_{qkn} dw.
\end{aligned} \tag{A.11}$$

If we can show that

$$B_i^u(u)B_l^u(u)B_o^u(u)|_{u_0}^{u_{r+1}} = B_j^v(v)B_m^v(v)B_p^v(v)|_{v_0}^{v_{r+1}} = B_k^w(w)B_n^w(w)B_q^w(w)|_{w_0}^{w_{r+1}} = 0, \tag{A.12}$$

we will have linearly-dependent columns and the determinant in Equation (A.6) will be zero under those same conditions.

In the special case of a periodic knot vector, the corresponding triple product expression in (A.12) for the periodic parameter will always evaluate to zero since the parameter boundaries will coincide. We will examine the remaining cases for non-periodic B-splines:

1. The triple product expression will have a non-zero value only when all the basis functions of that expression in condition (A.12) have domains that include one of the solid's surface boundaries. In the case of a cylindrical solid, these surface boundaries are the

iso-surfaces: $u = u_0$, $u = u_{r+1}$, $w = w_0$, and $w = w_{r+1}$. If a non-zero value occurs for any triple product expression, condition (A.12) cannot be satisfied and the determinant will also be non-zero.

2. If there exists a combination of three u, v and w basis functions whose domains lie entirely within their respective parameter boundaries. This implies that these basis functions will be zero at the boundaries, causing the triple products to be zero in condition (A.12). Therefore, condition (A.12) will be satisfied.

We have shown that condition A.12 holds whenever a combination of three u,v, and w basis functions in (A.12) exist that have domains that lie entirely within their respective parameter boundaries, producing a linear dependency in the columns of (A.6), so $\mathcal{B}_{ijklmnopq} = 0$. Therefore, we can compute the volume of a B-spline solid using only the elements of $\mathcal{B}$ whose basis functions all have domains that are non-zero somewhere along the boundary surface. Since the volume calculation in (A.1) contains a dot product, $\mathcal{B}$ and $\mathcal{C}$ can be shortened to include only the non-zero elements of $\mathcal{B}$ and the corresponding indexed elements of $\mathcal{C}$, accelerating volume computations.

Appendix B

Calculation of Shortest Distance from Point to Triangle in 3D

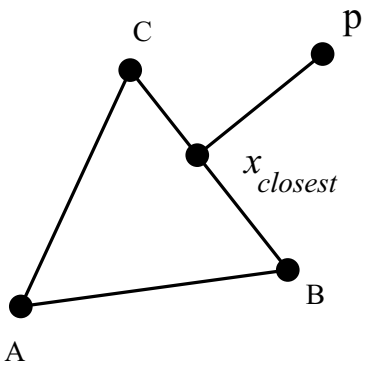


Figure B.1: Triangle ABC

The following summary of the algorithm for finding the closest point $\mathbf{x}_{closest}$ on a triangle to an arbitrary point $\mathbf{p}$ in world coordinates is based on the algorithm provided be Eberly[29]. Given a triangle ABC (Figure B.1), a parametric equation describing the plane of the triangle is:

$$\mathbf{tri}(s,t) = A + s\vec{AB} + t\vec{AC}. \tag{B.1}$$

The triangle's parametric domain in st space is bounded as follows (Figure B.2):

$$(s,t) \in D \equiv \{(s,t)|s \in [0,1], t \in [0,1], s+t \leq 1\}. \tag{B.2}$$

The squared distance function for the point $\mathbf{p}$ to the point $tri(s,t)$ is a quadratic equation:

$$\begin{aligned} d(s,t) &= \|\mathbf{tri}(s,t) - \mathbf{p}\|^2 && \text{(B.3)} \\ &= as^2 + 2bst + ct^2 + 2ds + 2et + f, && \text{(B.4)} \end{aligned}$$

where

$$a = \vec{AB} \cdot \vec{AB} \tag{B.5}$$

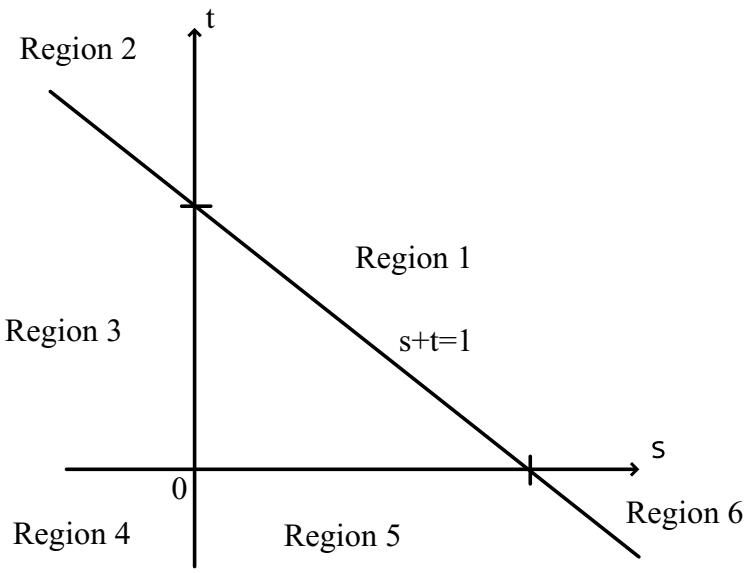


Figure B.2: st domain for triangle

$$b = \vec{AB} \cdot \vec{AC} \quad \text{(B.6)}$$
$$c = \vec{AC} \cdot \vec{AC} \quad \text{(B.7)}$$
$$d = \vec{AB} \cdot (\mathbf{A} - \mathbf{p}) \quad \text{(B.8)}$$
$$e = -\vec{AC} \cdot (\mathbf{A} - \mathbf{p}) \quad \text{(B.9)}$$
$$f = (\mathbf{A} - \mathbf{p}) \cdot (\mathbf{A} - \mathbf{p}). \quad \text{(B.10)}$$

Since $d(s,t)$ is a quadratic continuous, differentiable function, we can minimze the functions by setting the gradient, ∇d, to **0**:

$$\nabla d = 2(as + bt + d, bs + ct + e) = (0,0). \quad \text{(B.11)}$$

The solution,

$$s^* = \frac{be - cd}{ac - b^2} \quad \text{(B.12)}$$
$$t^* = \frac{bd - ae}{ac - b^2}, \quad \text{(B.13)}$$

produces the closest point, $\mathbf{tri}(s^*, t^*)$, on the plane to point $\mathbf{p}$. If (s^*, t^*) lies within the triangle's domain D, then $\mathbf{tri}(s^*, t^*)$ is the closest point and the final solution. If (s^*, t^*) lies outside the domain D, the closest point must lie on the triangle's boundary. In this case, the point $\mathbf{tri}(s^*, t^*)$ will lie in one of five possible planar regions outside the triangle (Figure B.2). The region can be deduced by examination of the values of s and t.

The function $d(s,t)$ has a minimum at $\mathbf{tri}(s^*, t^*)$ and forms iso-level curves ($d(s,t) = L$, L is constant) which are ellipses in the st plane. For regions 1,3 and 5 in Figure B.2, the closest point on the triangle will occur at the s,t coordinates where the lowest value level-set curve tangentially contacts the corresponding edge bordering the respective region: edge $s = 0$ for

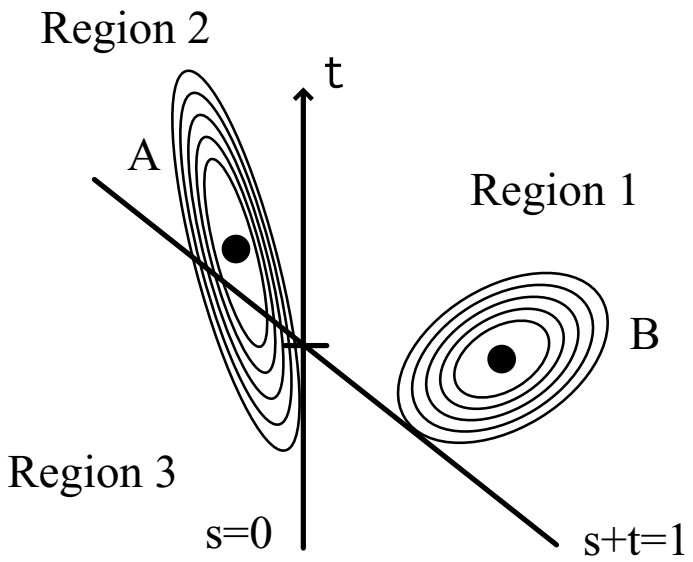


Figure B.3: Level curves of distance gradient

region 3, edge $t = 0$ for region 5, and edge $s + t = 1$ for region 1 (as in the set of curves marked B in Figure B.3).

For regions 2, 4 and 6, the level curves will contact one of the two edges closest to that region or the vertex common to both edges. For example, if the global minimum of d lies in region 2, the elliptical level curves will first contact either edge $s + t = 1$, edge $s = 0$ or vertex $(0, 1)$. In Figure B.3, the set of elliptical curves labelled A first contact the edge $s = 0$. Mathematically, this can be translated to the condition that only one of $(0, -1) \cdot \nabla d(0, 1)$ or $(1, -1) \cdot \nabla d(0, 1)$ is negative. The vectors, $(0, -1)$ and $(1, -1)$ are the direction vectors for the edges defined by $s = 0$ and $s + t = 1$ respectively, originating from their common vertex. If a dot product with the edge vector is negative, the closest point will reside on the corresponding edge. Once we have located the closest edge, we can find the closest point to the line segment representing the edge. If both dot products are positive, the closest point is the common vertex shared by the edges. Similar arguments hold for regions 4 and 6.

Appendix C

UML Primer

To document object-oriented designs, we use the *Unified Modelling Language (UML)*[34]. Classes are represented as rectangles and various relationships between classes are recognized. Figure C.1 describes a subset of the class notation available in UML and used in this thesis.

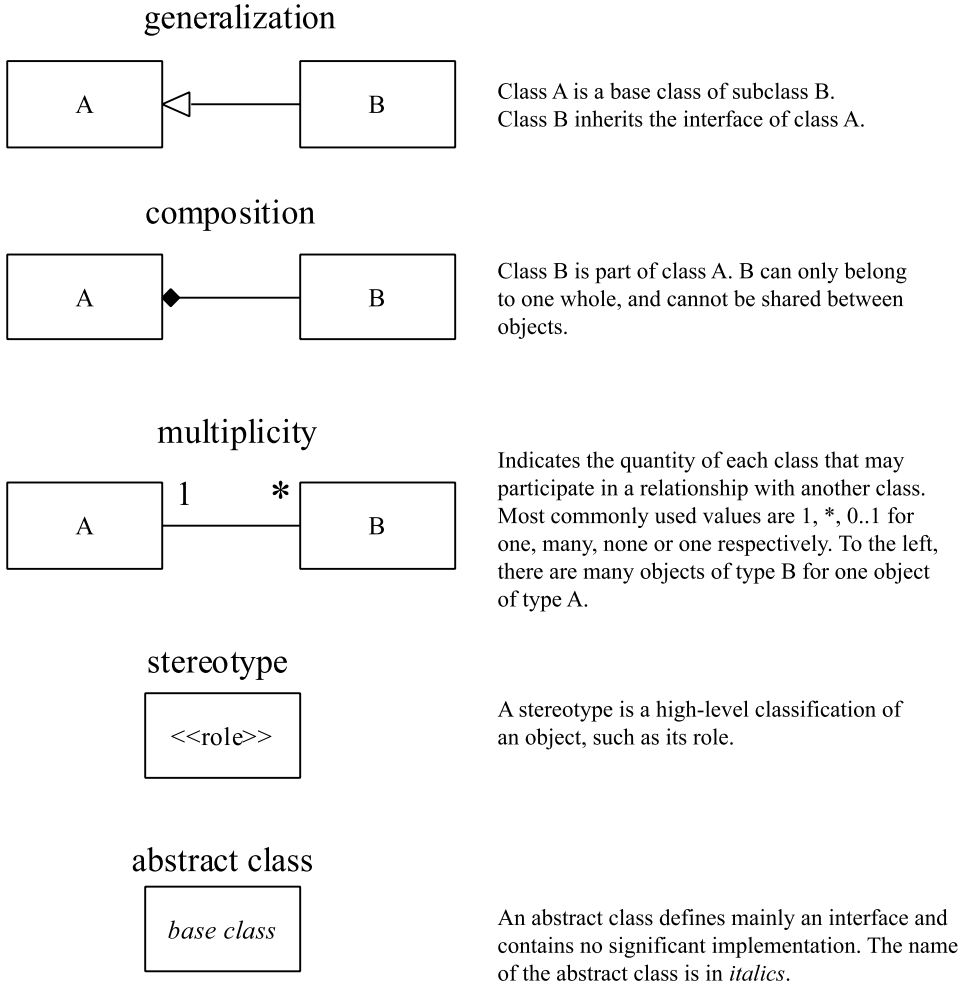


Figure C.1: UML class relationships

Bibliography

[1] Y.I. Abdel-Aziz and H.M. Karara. Direct linear transformation into object space coordinates in close-range photogrammetry. In *Symposium on Close-Range Photogrammetry*, pages 1–19, 1971.

[2] R. McN. Alexander. *Animal Mechanics*. Sidgwick and Jackson, 1968.

[3] R. McN. Alexander. Three uses for springs in legged locomotion. *The International Journal of Robotics Research*, 9(2):53–61, 1990.

[4] W. W. Armstrong and M. W. Green. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer*, 1(4):231–240, 1985.

[5] N. I. Badler, C. B. Phillips, and B. L. Webber. *Virtual humans and simulated agents*. Oxford University Press, 1992.

[6] A. S. Bahler. Modeling of mammalian skeletal muscle. *IEEE Transactions on Bio-Medical Engineering*, BME-15(4):249–257, 1968.

[7] A. S. Bahler, J. T. Fales, and K. L. Zierler. The dynamic properties of mammalian skeletal muscle. *Journal of General Physiology*, 51:369–384, 1968.

[8] K.A. Ball and M.R. Pierrynowski. Estimation of six degree of freedom rigid body segment motion from two dimensional data. *Human Movement Science*, 14:139–154, 1995.

[9] T. Beier and S. Neely. Feature based image metamorphosis. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 35–42, 1992.

[10] A. D'A. Bellairs and C. R. Jenkin. The skeleton of birds. In A. J. Marshall, editor, *Biology and Comparative Physiology of Birds*, pages 241–300. Academic Press, 1960.

[11] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[12] J. Bloomenthal. Calculation of reference frames along a space curve. In Andrew S. Glassner, editor, *Graphics Gems*, pages 567–571. Academic Press, Inc., 1990.

[13] J. Bloomenthal and K. Shoemake. Convolution surfaces. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 251–256, 1991.

[14] D. Bourguignon and M.-P. Cani. Controlling anisotropy in mass-spring systems. In *Proceedings of the 11th Eurographics Workshop on Animation and Simulation*, pages 113–123. Springer-Verlag, August 2000.

[15] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, September 1978.

[16] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In J. Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 243–252, July 1989.

[17] D. S. Chandrasekharaiah and L. Debnath. *Continuum Mechanics*. Academic Press, 1994.

[18] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, Benton L. Leong, Michael B. Monagan, and Stephen M. Watt. *First Leaves: A Tutorial Introduction to Maple V*. Springer-Verlag, 1992.

[19] D. T. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 89–98, July 1992.

[20] M. Cianchi. *Leonardo The Anatomy*. Giunti Gruppo Editoriale, 1998.

[21] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 187–196, August 1990.

[22] R. D. Crowninshield. Use of optimization techniques to predict muscle forces. *Transactions of the ASME*, 100:88–92, may 1978.

[23] J. L. Davis. *Introduction to Dynamics of Continuous Media*. Macmillan Publishing Company, 1987.

[24] S. L. Delp and J. P. Loan. A graphics-based software system to develop and analyze models of musculoskeletal structures. *Comput. Biol. Med.*, 25(1):21–34, 1995.

[25] S. L. Delp, J. P. Loan, M. G. Hoy, F. E. Zajac, E. L. Topp, and J. M. Rosen. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *IEEE Transactions on Biomedical Engineering*, 37(8):757–767, 1990.

[26] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., 1983.

[27] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 85–94, 1998.

[28] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects. In *Graphics Interface '99*, pages 1–8, 1999.

[29] D. Eberly. Distance between point and triangle in 3d. `www.magic-software.com/gr_dist.htm`, 1999.

[30] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on visualization and computer graphics*, 3(3):201–214, 1997.

[31] G. Farin. W. boehm: Differential geometry i. In *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, chapter 11, pages 173–183. Academic Press, Inc., second edition, 1990.

[32] W. O. Fenn and B. S. Marsh. Muscular force at different speeds of shortening. *Journal of Physiology*, 85:277–297, 1935.

[33] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 205–212, August 1988.

[34] M. Fowler and K. Scott. *UML Distilled: Applying the Standard Object Modeling Language*. Addison-Wesley, 1997.

[35] C.B. Frank and N.G. Shrive. Ligament. In Benno M. Nigg and Walter Herzog, editors, *Biomechanics of the Musculo-skeletal System*, chapter 2, pages 107–126. John Wiley and Sons, Ltd., second edition, 1999.

[36] Y. C. Fung. *Biomechanics: Mechanical properties of living tissues*. Springer-Verlag, 1981.

[37] M.-P. Gascuel. An implicit formulation for precise contact modeling between flexible solids. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 313–320, 1993.

[38] M.-P. Gascuel, A. Verroust, and C. Puech. A modelling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2:82–91, 1991.

[39] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[40] M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 263–270, July 1985.

[41] A. S. Glassner. *Principles of Digital Image Synthesis*, volume 1. Morgan Kaufmann Publishers, 1995.

[42] E. Goldfinger. *Human Anatomy for Artists*. Oxford University Press, 1991.

[43] R. N. Goldman. Identities for the b-spline basis functions. In A. W. Paeth, editor, *Graphics Gems V*, pages 163–167. Academic Press, Inc., 1995.

[44] H. Goldstein. *Classical Mechanics*. Addison-Wesley Publishing Company, second edition, 1980.

[45] A. M. Gordon, A. F. Huxley, and F. J. Julian. The variation in isometric tension with sarcomere length in vertebrate muscle fibres. *Journal of Physiology*, 184:170–192, 1966.

[46] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 171–180, 1996.

[47] J.-P. Gourret, N. M. Thalmann, and D. Thalmann. Simulation of object and human skin deformations in a grasping task. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, pages 21–30, 1989.

[48] J. Griessmair and W. Purgathofer. Deformation of solids with trivariate b-splines. In *Eurographics '89*, pages 137–148, 1989.

[49] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 165–174, 1993.

[50] H. Hatze. A myocybernetic control model of skeletal muscle. *Biological Cybernetics*, 25:103–119, 1977.

[51] H. Hatze. *Myocybernetic control models of skeletal muscle: Characteristics and applications*. University of South Africa, 1981.

[52] W. Herzog. Muscle. In Benno M. Nigg and Walter Herzog, editors, *Biomechanics of the Musculo-skeletal System*, chapter 2, pages 127–147. John Wiley and Sons, Ltd., second edition, 1999.

[53] W. Herzog and J. Gál. Tendon. In Benno M. Nigg and Walter Herzog, editors, *Biomechanics of the Musculo-skeletal System*, chapter 2, pages 127–147. John Wiley and Sons, Ltd., second edition, 1999.

[54] A. V. Hill. The force-velocity relation in shortening muscle. In *First and Last Experiments in Muscle Mechanics*, chapter 3, pages 23–41. Cambridge at the University Press, 1970.

[55] J. K. Hodgins and M. H. Raibert. Biped gymnastics. *The International Journal of Robotics Research*, 9(2):115–132, 1990.

[56] N. Hogan. Mechanical impedance of single- and multi-articular systems. In *Multiple Muscle Systems: Biomechanics and Movement Organization*, chapter 9, pages 149–164. Springer-Verlag, 1990.

[57] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. Sd/fast. URL: www.symdyn.com, Symbolic Dynamics, Inc., 1991.

[58] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, 1989.

[59] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 177–184, 1992.

[60] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-collide: Accelerated collision detection for vrml. In *Proceedings of ACM Symposium on Virtual Reality Modeling Language*, pages 117–123, 1997.

[61] A.F. Huxley. Muscle structure and theories of contraction. *Progress in Biophysics and Chemistry*, 7:255–318, 1957.

[62] Waterloo Maple Inc. Maple v release 5. Computer software, 1998.

[63] P. M. Isaacs and M. F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 215–224, July 1987.

[64] H. Iwamoto, R. Sugaya, and H. Sugi. Force-velocity relation of frog skeletal muscle fibres shortening under continuously changing load. *Journal of Physiology*, 422:185–202, 1990.

[65] R. K. Jensen and D. T. Davy. An investigation of muscle lines of action about the hip: a centroid line approach vs the straight line approach. *Journal of Biomechanics*, 8:103–110, 1975.

[66] D. E. Johnson and E. Cohen. A framework for efficient minimum distance computations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3678–3684, 1998.

[67] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[68] K.R. Kaufman, K. An, and E.Y.S. Chao. Incorporation of muscle architecture into the muscle length-tension relationship. *Journal of Biomechanics*, 22(8/9):943–948, 1989.

[69] K. Komatsu. Human skin model capable of natural shape variation. *The Visual Computer*, 3:265–271, 1988.

[70] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 313–324, 1996.

[71] H. J. Lamousin and Warren N. Waggenspack Jr. Nurbs-based free-form deformations. *IEEE Computer Graphics and Applications*, pages 59–65, November 1994.

[72] C. Lanczos. *The variational principles of mechanics*. Dover Publications, Inc., fourth edition, 1970.

[73] J. Lander. Skin them bones: game programming for the web generation. *Game Developer*, 5(5):11–16, May 1998.

[74] P. Lee, S. Wei, J. Zhao, and N. I. Badler. Strength guided motion. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 253–262, August 1990.

[75] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 55–61, 1995.

[76] Y. T. Lee. A simplified b-spline computation routine. *Computing*, 29:365–371, 1982.

[77] R. N. Leekam, A. M. Agur, and N. H. McKee. Using sonography to diagnose injury of plantaris muscles and tendons. *American Journal of Roentgenology*, 172:185–189, 1999.

[78] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 181–188, 1996.

[79] W. Maurel, D. Thalmann, P. Hoffmeyer, P. Beylot, P. Gingins, P. Kalra, and N. M. Thalmann. A biomechanical musculoskeletal model of human upper limb for dynamic simulation. In *Computer animation and simulation*, pages 121–136, 1996.

[80] M. McKenna and D. Zeltzer. Dynamic simulation of a complex human figure model with low level behavior control. *Presence*, 5(4):431–456, 1996.

[81] T. A. McMahon. Fundamental muscle mechanics. In *Muscles, Reflexes, and Locomotion*, chapter 1, pages 3–26. Princeton University Press, 1984.

[82] K. Meijer, H. J. Grootenboer, H. F. J. M. Koopman, B. J. J. J. van der Linden, and P. A. Huijing. A hill type model of rat medial gastrocnemius muscle that accounts for shortening history effects. *Journal of Biomechanics*, 31:555–563, 1998.

[83] G. Miller. The motion dynamics of snakes and worms. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, pages 169–178, 1988.

[84] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.

[85] J. J. More, B. S. Garbow, and K. E. Hillstrom. Minpack. Numerical software library, 1980. Argonne National Laboratory.

[86] V. Ng-Thow-Hing and P. Faloutsos. Dynamic animation and control environment (dance). In *SIGGRAPH 2000 Technical Sketch: Conference Abstracts and Applications*, page 198, 2000.

[87] B. M. Nigg and S. K. Grimston. Bone. In Benno M. Nigg and Walter Herzog, editors, *Biomechanics of the Musculo-skeletal System*, chapter 2, pages 64–85. John Wiley and Sons, Ltd., second edition, 1999.

[88] M. Nordin and V. H. Frankel. Biomechanics of the knee. In *Basic Biomechanics of the Musculoskeletal System*, chapter 6, pages 115–134. Lea and Febiger, second edition, 1989.

[89] U.S. National Library of Medicine. The visible human project. MRI, CT, and axial anatomical images of human body, October 1996.

[90] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, first edition, 1994.

[91] M. G. Pandy and F. C. Anderson. Three-dimensional computer simulation of jumping and walking using the same model. In *VIIth International Symposium on Computer Simulation in Biomechanics*, pages 92–95, 1999.

[92] M. G. Pandy and F. E. Zajac. Optimal muscular coordination strategies for jumping. *Journal of Biomechanics*, 24(1):1–10, 1991.

[93] M. G. Pandy, F. E. Zajac, E. Sim, and William S. Levine. An optimal control model for maximum-height human jumping. *Journal of Biomechanics*, 23(12):1185–1198, 1990.

[94] M. van de Panne and E. Fiume. Sensor-actuator networks. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 335–342, August 1993.

[95] M. van de Panne, E. Fiume, and Z. Vranesic. Reusable motion synthesis using state-space controllers. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 225–234, August 1990.

[96] H. K. Pedersen. Decorating implicit surfaces. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, volume 29, pages 291–300, 1995.

[97] C. S. Peskin and D. M. McQueen. A general method for the computer simulation of biological systems interacting with fluids. In *Proceedings of SEB Symposium on Biological Fluid Dynamics*, Leads, England, 1994.

[98] J. S. Petrofsky and C. A. Phillips. The force-velocity relationship of skeletal muscle. In Chandler A. Phillips and Jerrold S. Petrofsky, editors, *Mechanics of skeletal and cardiac muscle*, chapter 2, pages 19–58. Thomas, 1982.

[99] J. C. Platt and A. H. Barr. Constraint methods for flexible models. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, pages 279–288, 1988.

[100] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[101] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97*, pages 177–189, 1997.

[102] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 349–358, 1991.

[103] A. Rappoport, A. Sheffer, and M. Bercovier. Volume-preserving free-form solids. In *Solid Modelling '95*, pages 361–372, 1995.

[104] D. F. Rogers and L. A. Adlum. Dynamic rational b-spline surfaces. *Computer-aided Design*, pages 609–616, August 1990.

[105] F. Scheepers, R. E. Parent, W. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 163–172, August 1997.

[106] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, August 1986.

[107] Uri Shani. Splines as embeddings for generalized cylinders. *Computer Vision, Graphics and Image Processing*, 27:129–156, 1984.

[108] J. Shen and D. Thalmann. Interactive shape design using metaballs and splines. In *Implicit Surfaces '95*, pages 187–193, 1995.

[109] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 405–414, 1998.

[110] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, volume 32, pages 395–404, 1998.

[111] J. Stewart. *Calculus*. Brooks/Cole Publishing Company, 1987.

[112] J. D. Talbot. Accurate characterization of skin deformations using range data. Master's thesis, University of Toronto, 1998.

[113] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[114] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 205–214, 1987.

[115] D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[116] D. Terzopoulos and A. Witkin. Physically-based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51, 1988.

[117] F. Thomas and O. Johnston. *The Illusion of Life: Disney Animation*. Hyperion, New York, 1981.

[118] C. Truesdell and W. Noll. *The Non-Linear Field Theories of Mechanics*. Springer-Verlag, 2nd edition, 1992.

[119] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, pages 43–50, July 1994.

[120] R. Turner and E. Gobbetti. Interactive construction and animation of layered elastically deformable characters. *Computer Graphics Forum*, 17(2):135–152, 1998.

[121] A. van den Bogert. *Computer Simulation of Locomotion in the Horse*. ADDIX, 1989.

[122] A. J. van den Bogert. Simulation. In Benno M. Nigg and Walter Herzog, editors, *Biomechanics of the Musculo-skeletal System*, chapter 4, pages 594–617. John Wiley and Sons, Ltd., second edition, 1999.

[123] F.C.T. van der Helm and R. Veenbaas. Modelling the mechanical effect of muscles with large attachment sites: application to the shoulder mechanism. *Journal of Biomechanics*, 24(12):1151–1163, 1991.

[124] A. J. van Soest and M. F. Bobbert. The contribution of muscle properties in the control of explosive movements. *Biological Cybernetics*, 69:195–204, 1993.

[125] A. J. van Soest, A. L. Schwab, M. F. Bobbert, and G. Jan van Ingen Schenau. The influence of the biarticularity of the gastrocnemius muscle on vertical-jumping achievement. *Journal of Biomechanics*, 26(1):1–8, 1993.

[126] T. I. Vassilev. Fair interpolation and approximation of b-splines by energy minimization and points insertion. *Computer-Aided Design*, 28(9):753–760, 1996.

[127] T. Wakabayashi, N. Max, and N. Hayashi. Computer simulation and animation of muscle cross-bridge motion. *The Journal of Visualization and Computer Animation*, 1(1):9–14, August 1990.

[128] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.

[129] J. Wilhelms. Animals with anatomy. *IEEE Computer Graphics and Applications*, 17(3):22–30, 1997.

[130] J. Wilhelms and A. Van Gelder. Anatomically based modeling. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 173–180, August 1997.

[131] J. M. Winters. Hill-based muscle models: a systems engineering perspective. In J. M. Winters and S. L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*. Springer-Verlag, 1990.

[132] J. M. Winters and L. Stark. Muscle models: What is gained and what is lost by varying model complexity. *Biological Cybernetics*, 55:403–420, 1987.

[133] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. In *ACM Computer Graphics (Proc. 1990 Symposium on Interactive 3D Graphics)*, volume 24, pages 11–21, 1990.

[134] A. Witkin and M. Kass. Spacetime constraints. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 159–168, August 1988.

[135] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 243–252, August 1990.

[136] R. D. Wottiez, P. A. Huijing, H. B. K. Boom, and R. H. Rozendal. A three-dimensional muscle model: a quantified relation between form and function of skeletal muscles. *Journal of Morphology*, 182:95–113, 1984.

[137] G. T. Yamaguchi, A. G. U. Sawa, D. W. Moran, M. J. Fessler, and J. M. Winters. A survey of human musculotendon actuator parameters. In J. M. Winters and S. L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, pages 717–739. Springer-Verlag, 1990.

[138] F. E. Zajac. Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control. *Critical Reviews in Biomedical Engineering*, 17(4):359–411, 1989.

[139] Q. Zhu, Y. Chen, and A. Kaufman. Real-time biomechanically-based muscle volume deformation using fem. *Computer Graphics Forum*, 17(3):275–284, 1998.

[140] R. F. Ziegler. Character animation using transformation based linear dynamics. Master's thesis, University of Toronto, 1997.

[141] C. J. Zuurbier and P. A. Huijing. Influence of muscle geometry on shortening speed of fibre, aponeurosis and muscle. *Journal of Biomechanics*, 25(9):1017–1026, 1992.