ANATOMICALLY-BASED HUMAN

BODY DEFORMATIONS

THESE No 2573 (2002)

PRESENTEE AU DEPARTEMENT D'INFORMATIQUE

ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ES SCIENCES

PAR

Amaury AUBEL

Ingénieur informaticien IIE
originaire de France

acceptée sur proposition du jury :

Prof. D. Thalmann, directeur de thèse
Prof. N. Ayache, rapporteur
Prof. D. Bechmann, rapporteur
Dr. K. Arminian, rapporteur
Prof. D. Hersch, rapporteur
Prof. P. Fua, rapporteur

Lausanne, EPFL
2002

Table of contents

Abstract

The modeling of the human being is one of the most arduous tasks in Computer Graphics. The human body is so complex a machinery that no current biomechanical or computer graphics model comes even close to its true nature. Additionally, we are subconsciously attuned to the details of humans because of our extreme familiarity to human shapes. Hence, we know immediately if one detail is missing when we see a virtual human on screen. We believe that an approach to human body modeling that considers all the major anatomical layers so as to produce skin deformations can get around the “missing detail” problem in a natural manner.

This thesis proposes a new multi-layered body representation for producing automatic, fast, appropriate deformations of a geometric envelope (*skin*) from a moving hierarchical, articulated structure (*skeleton*). Our model centers around the three general anatomical structures which create surface form: the skeleton, the musculature and the fatty tissues. We model and deform each of these layers using fast, ad-hoc methods. The final model is completed by an overlying skin that automatically adjusts itself to changes in the underlying layers.

We show that four joint models allow to approximate the motion of the skeleton satisfactorily. We particularly expose the benefits of well-chosen parameterizations. Our joint models moreover allow the easy specification, visualization and enforcement of anatomically accurate limits. The application of coupled limits for the swing motion and the axial rotation of the limbs is demonstrated.

We introduce simple yet efficient techniques for deforming automatically and realistically the muscle layer once the movement of the human figure is specified. We abstract a muscle by two layers: a skeleton, which is defined by a set of centroid curves called *action lines*, and a surface mesh, which represents the muscle shape. We show that the deformation of the muscle shape can be completely driven by the set of action lines. Our approach unifies biomechanics and computer graphics since the action lines can be used for computing both the muscle force and the deformation. In the end, our model can capture all muscle shapes and produces visually convincing deformations at (nearly) interactive rates.

The results of a recent work in soft body dynamics are applied to the dynamic deformation of fatty tissues in areas where these are abundant. The skin is deformed by a two-step geometric algorithm according to the current state of the skeleton and muscle layers. In the first stage, the skin is deformed using a variant of the well-known *skinning* algorithm. In the second stage, the skin is sculpted by the underlying musculo-skeletal system using a normal-directed ray-casting procedure. The algorithm works with any kind of mesh and permits to output a fixed topology.

Version Abrégée

La modélisation de l'être humain est l'une des tâches les plus difficiles en Infographie. Le corps humain est une mécanique tellement complexe qu'il n'existe aucun modèle en biomécanique ou en infographie qui s'en approche. En outre, de par notre familiarité avec la silhouette humaine, nous sommes attentifs à ses moindres détails. Aussi décelons-nous immédiatement l'absence de tel ou tel détail chez un humain virtuel. Nous pensons que ce problème de "détail manquant" peut être résolu par une approche prenant en compte les principales couches anatomiques du corps humain lors de la modélisation.

Dans cette thèse, nous proposons un nouveau modèle multi-couches permettant de générer automatiquement et rapidement des déformations réalistes d'une enveloppe géométrique appelée *peau* à partir d'une structure articulée en mouvement appelée *squelette*. Notre modèle s'articule autour des trois structures anatomiques qui sculptent l'enveloppe extérieure: le squelette, les muscles et les tissus adipeux. Chacune de ces couches est explicitement représentée et déformée par une méthode rapide et appropriée. Une peau s'adaptant automatiquement aux changements des couches internes recouvre le tout.

Nous introduisons quatre modèles d'articulation pour simuler correctement le mouvement du squelette. Nous nous attacherons à exposer les avantages de certaines paramétrisations. Par ailleurs, nos modèles d'articulation facilitent la spécification, la visualisation et l'application de limites articulaires correctes d'un point de vue anatomique. Nous démontrons l'application de limites couplées pour le mouvement général d'un membre et son mouvement de torsion.

Nous introduisons des techniques simples mais efficaces pour déformer de façon automatique et réaliste la couche musculaire une fois que le mouvement du personnage est spécifié. Notre modèle de muscle se compose de deux couches: un "squelette", défini par un ensemble de lignes centroïdes appelées *lignes d'action*, et un maillage de surface représentant la forme du muscle. Nous montrerons que la déformation globale du muscle peut être entièrement gouvernée par ses lignes d'action. Notre approche est aussi bien adaptée aux exigences de la biomécanique que de l'infographie puisque les lignes d'actions peuvent servir et à deformer le muscle et à calculer la force musculaire produite. Au final, notre technique permet de déformer n'importe quel muscle de facon convaincante et interactive.

Les résultats d'un travail récent dans le domaine des déformations élastiques sont appliqués aux tissus graisseux. La peau est déformée, en deux étapes, par un algorithme géométrique. La première étape de l'algorithme se résume à l'application d'une variante du célèbre algorithme de *skinning*. En second lieu, la peau est sculptée par les muscles et le squelette à l'aide d'un algorithme de lancer de rayons le long des normales à la surface. L'algorithme peut être appliqué à n'importe quel type de maillage et garantit une topologie fixe.

Remerciements

Je tiens à remercier en premier lieu le Professeur Daniel Thalmann, directeur du laboratoire, qui m'a permis de travailler sur des projets de recherche variés et passionnants pendant cinq ans. Je souhaite le remercier en outre pour avoir dirigé mon travail de thèse.

J'adresse également mes remerciements aux membres du jury pour le soin qu'ils ont apporté à l'examen de cette thèse.

Je tiens à remercier chaleureusement Marcelo Kallmann et Ronan Boulic pour avoir relu ce manuscrit et avoir suggéré des modifications pertinentes. Par ailleurs, je souhaite exprimer ma vive reconnaissance à l'égard de Nicolas, Thierry et Mireille, les talentueux "designers" avec qui j'ai eu le plaisir de collaborer. Leurs encouragements constants n'ont pas été inutiles. De manière plus générale, je souhaite remercier bon nombre de mes collègues pour les innombrables discussions techniques et scientifiques au cours de ces cinq années:

- Marcelo, Paolo, Ronan et Ralf. Leurs vastes connaissances ont été d'un précieux secours en maintes occasions.
- Thierry et Walter qui ont su m'intéresser à la biomécanique et à l'anatomie.
- Lorna, Raquel, Ralf, Paolo, Benoît, Richard et Captain Fu pour nos discussions en quatre dimensions sur les orientations :-)

Je tiens aussi à saluer certaines personnes que j'ai rencontrées en Suisse:

- Paolo, Philippe, Rémy, Christophe, Luca, Benoît et Etienne qui ont partagé avec moi la passion pour la fondue au fromage (qu'elle soit moitié-moitié, champenoise ou a la bière) !
- Fabrice, Christian M., Christian B., Ali, Anderson, Raquel et Ralf pour les bons moments passés sur les courts de tennis, les terrains de basket, les pentes enneigées, la glace et l'eau.
- Angela, Selim, Maja, Christian M., Christian B. et Cédric, d'anciens collègues du LIG.
- Jean-Sé pous ses superbes photos et non moins superbes pièces de théâtre.
- Joaquim, Rémy et Richard pour nos affinités pianistiques.
- Lorna pour m'avoir patiemment aidé dans mon apprentissage de la langue de Goethe.
- Raquel pour ses lecons d'Espagnol meme si je n'ai pas toujours été un élève assidu !
- Chris, Gaël, Tom, Sumedha, Sunil et Laurent de Miralab.
- Won-Sook et sa famille pour m'avoir aidé lors de mon hospitalisation à Séoul.

Enfin, je remercie ma famille proche (et moins proche) ainsi que mes amis en France pour m'avoir soutenu et encouragé pendant ces nombreuses années passées à l'EFPL.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Importance and relevance

Virtual humans are, perhaps paradoxically, becoming more and more real and common in our lives. Indeed, the number of fields and disciplines in which 3D synthetic characters have spread is truly stunning: video games, films, television, commercials, virtual reality, teleconferencing, ergonomics, medicine, biomechanics. The list goes on and on. All these applications roughly fit in one of the three following main classes: film production, real-time applications and simulations involving humans.

Computer-generated images have gained wide acceptance in the film industry. Box-office hits such as *Jurassic Park* and *Titanic* paved the way. More recently, *Final Fantasy*, an entirely computer-generated film, featured photo-realistic virtual humans of unprecedented realism. The amazing hair animation and the nearly flawless skin textures alone undeniably made the movie an engineering and technical feat. Yet, owing to the plummeting cost of computer-generated imagery, synthetic characters are no longer used only in expensive films for entertainment but also increasingly in commercials, kinematic scenes for video games, and films with a didactic purpose. For example, safety instructions on most long-distance flights are now demonstrated by digital actors on small personal TV sets.

In the second category, we find all the applications that require the embodiment of one or several participants (such an embodiment is called *avatar*) on the one hand and real-time interaction on the other hand. Video games and collaborative virtual environments are classical examples but many more applications fall under the same category. As a rule, the vast majority of virtual reality applications need some sort of representation of the user and the ability to interact with the environment in real-time. Virtual surgery is a typical example where the patient's body or at least part of it has to be represented and the response to the surgeon's movements needs to be as rapid as possible. Military simulations provide us with another example of immersion of trainees in a real-time environment involving virtual humans. More generally, we see the emergence of a class of virtual reality applications to train people for specific tasks or situations. Digital surrogates are also becoming more and more common on the

Internet: Synthetic characters dispense advice to consumers navigating through websites, turn into virtual reporters reading the news [Ananova 00], or act as coaches in roleplay [Extempo 99]. Teleconferencing, too, is perhaps on the verge of being revolutionized by the introduction of digital doubles. These help to restore human interaction and make the meeting more productive, while possibly cutting down transmission bandwidth requirements [Capin 95].

Finally, virtual humans are needed in many kinds of simulations. For example, computer-aided ergonomics utilizes virtual humans for work place assessment while the automotive industry replaces expensive cumbersome dummies with virtual mannequins in car crash simulations. Medicine also benefits from the technology in orthopedic rehabilitation, anatomy, clinical analysis of abnormal movement patterns, etc. In addition to medical purposes, motion analysis, assisted by a 3D representation of the human body, is also used for improving the performance of athletes in competitive sports.

1.1.2 The challenge of human body modeling

Researchers have tried to model (and animate) realistic human beings almost since the introduction of graphics displays. Back in the seventies, Badler [Badler 79] and Herbison-Evans [Herbison 78] already investigated ways to represent and deform 3D human models with volumetric primitives. Many other researchers have proposed new models since then. Today, research in this field is basically as active as ever, both in the academic world and in the industry.

The reason why human body modeling still remains one of the greatest challenges in Computer Graphics is probably two-fold. First and foremost, the human body is of such biological complexity that no current model, whether a computer graphics model or a biomechanical one, comes even close to its true nature. The second reason is that we, as human beings, are all experts when it comes to human shapes. The sensitivity of our eyes to human figures is such that we can usually identify unrealistic body shapes (or motions) in a split second. Hence, the problem of producing a visually appealing representation of the human body with convincing deformations, albeit nearly as old as computer graphics, is still unresolved.

Nevertheless, recent technological breakthroughs have solved some issues. Laser scanning systems coupled with 3D reconstruction algorithms (see [Krishnamurthy 96] and [Carr 01] for instance) allow today's graphics artists to display realistic virtual actors in static poses. Magnetic and optical motion capture systems help to produce smooth, natural-looking animations. However, this is only half of the problem, as deforming the skin of virtual characters in a realistic fashion mostly remains a manual, tedious, labor-intensive process.

Nowadays, the most widespread technique for skin deformation in the gaming industry consists in assigning each vertex of the polyhedral mesh, which represents the skin, a number of bones with appropriate weights [Lander 98]. During animation, a vertex mesh is thus moved based on transformations of the underlying bone system. This simple technique is often referred to as *skinning.* Likewise, commercial modeling packages commonly used for the kine-

matic industry, e.g. Maya [Alias 01], also rely on the *skinning* technique even though graphics artists complement it with other tools such as deformers, blend shapes and other shape interpolation techniques [Lewis 00]. It is remarkable, perhaps truly unique, that the same technique is employed throughout the spectrum of computer graphics applications, ranging from real-time 3D games to film post-production. And yet, skinning is tedious and time-consuming because weights are usually found through a trial-and-error process. The graphics artist often spends hours to manually adjust the skinning weights, vertex by vertex. Moreover, in areas with considerable mobility like the shoulder, some combination of weights may produce good results in some skeleton postures and yield unacceptable ones in others, leading to considerable frustration by designers.

1.2 Overview of our Approach

Anatomy is the biological science concerned with the form, position, function, and relationship of structures in the human body in an erect and motionless stance. *Artistic anatomy* is a specialized discipline concerned only with those structures that create and influence surface form when the body moves into different stances. Thus, for centuries, painters and sculptors have studied artistic anatomy to improve their work. Inspired by their approach, we have based our research on anatomy.

As human beings, we are subconsciously attuned to the details of humans. Hence, we know immediately if one detail happens to be missing when we see a virtual human on screen. Remarkably enough, we are not always able to explain what is actually missing because the recognition process is subconscious. An approach to human body modeling that considers all the major anatomical layers so as to produce skin deformations gets around the “missing detail” problem in a natural manner.

This thesis proposes a new multi-layered body representation for producing automatic, fast, appropriate deformations of a geometric envelope (*skin*) from a moving hierarchical, articulated structure (*skeleton*). Our model centers around the three general anatomical structures which create surface form: the skeleton, the musculature and the fatty tissues. We explicitly model and deform each of these layers (somewhat independently as reflected by the structure of this document) using ad-hoc methods. The final model is completed by an overlying skin that automatically adjusts itself to changes in the underlying layers.

1.3 Objectives of this Research

Human body modeling and deformation is customarily split into three separate problems: face, body and hands. The human face is distinguishable in that it conveys emotions through expression wrinkles, which are not found in any other body part. This is why facial animation is generally taken care of by a specific deformation system. Analogously, hands animation is usually handled by another specific deformation module because of their extraordinary mobility by comparison with other body parts. In this work, we have focused on the body excluding

its extremities.

The main goals of this thesis are outlined as follows:

Study of the human joints and motion range leading to the choice of appropriate computer models.

Bones are explicitly modeled in our anatomically-based approach. This requires the choice of an articulated structure that accurately reflects the anatomy of the human skeleton. We show how to build up on the basis provided by the H-Anim standard [HAnim 98] in order to derive an anatomically accurate articulated structure on top of which, the skeleton, muscle and skin layers can be placed.

Another point of interest concerns the possible motion range of human joints. Limits in existing computer joint models are notoriously inaccurate. They are usually specified as minimal and maximal angles around three orthogonal axes. These limits do not faithfully describe the range of motion a joint allows [Maurel 00]. Nor is the visualization of the reachable space easy or even possible. We aim to remedy these shortcomings by proposing joint models with natural parameterizations that allow the easy specification, visualization and enforcement of anatomically accurate limits.

Design of efficient methods that deform human muscles automatically and realistically based on the skeleton posture while allowing the easy computation of the produced forces.

Human muscles vary extremely in their form and function. Our goal is to develop methods that are generic enough to handle the great variety of human muscles and produce a visually convincing deformation at (nearly) interactive rates on modern computers. Additionally, we are looking for an elegant way to unify the approaches to muscle force modeling in biomechanics and approaches to muscle deformation in computer graphics.

Development of methods that automatically deform the skin of the character once the deformation of the inner layers has been computed.

For any animation of the human figure, the character's skin should deform naturally and realistically according to the current state of the underlying skeleton and muscle layers. No restriction or assumption concerning the topology of the skin should be made so as to permit the use of skin meshes produced by different modalities (modeling software, laser scanning device, etc.). The deformation should moreover reflect the dynamic behavior of the underlying fatty tissues in areas where these are significant.

Meeting the requirements of subsequent related tasks.

It is highly desirable to output a skin mesh with a fixed topology as it eases many subsequent operations. What we mean by fixed topology is a mesh whose number of vertices remains constant during animation and whose connectivity between vertices is not altered.

Firstly, 2D image textures can be mapped onto the skin surface to give a more natural look to an otherwise plastic or metal looking skin. This process is accomplished by assigning each skin vertex 2D coordinates referring to a pixel in the image texture. A constant mesh topology implies that the set of texture coordinates needs not be recomputed for each frame. Secondly, bump mapping [Watt 92], which fakes small bumps at the surface of an object by perturbing the surface normals using a 2D grayscale image (*bump map*), provides a supplementary way of enhancing the life-likeness of skin. Once again, the correspondence between the skin mesh and the bump map is eased by a fixed topology. Lastly, the simulation of a virtual clothing layer on top of the skin also benefits from a fixed topology of the body mesh.

1.4 Organization of this Document

The plan of this document is as follows.

- In Chapter 2, we review the main previous works in computer graphics for the problem of skin deformation.
- In Chapter 3, we focus on the skeleton layer.
- In Chapter 4, the emphasis is laid on modeling the human musculature. We introduce novel techniques for automatically deforming the muscle layer based on the posture assumed by the skeleton.
- In Chapter 5, we describe different techniques for modeling the fat and skin layers.
- In Chapter 6, we show the main results of this work. We first present our joint editor, then our interactive muscle deformation modeler, named *MuscleBuilder,* and finally show some examples of fat and skin deformation.
- In Chapter 7, we summarize the contributions of this thesis, and suggest future research directions.

1.5 Mathematical Notation and Conventions

Throughout this document, we use the *column vector* convention and *right-handed coordinate frames*. Scalars are denoted by small letters such as s. Vectors are denoted by small boldface letters such as $\boldsymbol{v}$. The components of a 3D vector are usually noted v_x, v_y and v_z. The three basis vectors of a coordinate frame are denoted by $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. Matrices and frames are denoted by capital letters such as M, while points are denoted by capital boldface letters such as $\boldsymbol{P}$. When referring to the coordinates of a point, we shall use the point notation rather than the vector notation. Hence, $\boldsymbol{P}$ denotes both a point and its coordinates. Similarly, homogeneous coordinates are denoted by overlined capital boldface letters such as $\overline{\boldsymbol{X}}$.

The inner product of two n-dimensional vectors is defined as $\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{n} v_i w_i$. The norm of a vector is chosen to be the Euclidean norm: $\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}}$.

Chapter 2

State of the Art in 3D Character Modeling and Deformation

The problem of human body modeling, and by extension of 3D character modeling, is commonly decomposed into two sub-problems: modeling the skin of the character in a given pose on the one hand, and deforming this skin for any other pose on the other hand. Yet, these two sub-problems are basically not decoupled since choices made for one are likely to impact the resolution of the other. Hence, we do not consider separately the modeling stage and the deformation phase in the following.

2.1 Surface Models

Many character models rely on just two layers:

- An articulated structure or *skeleton*, which forms the backbone of the character animation system. This structure is frequently arranged as a hierarchy and is covered by:
- An external geometric envelope or *skin*, whose deformations are driven by the underlying articulated structure.

2.1.1 Rigid deformations

The simplest model of skin consists in a collection of polygonal meshes placed on top of the skeleton, with generally one mesh by body part. By rigidly anchoring each mesh to a specific joint, one obtains a skin that roughly follows the motion of the underlying articulated structure for a very low (the lowest actually) computational cost. However, body parts tend to interpenetrate in the vicinity of joints or appear unconnected during movement. Furthermore, muscle bulging is completely ignored in this approach. For all that, such models are still common in Web applications [HAnim 98].

2.1.2 Local surface operators

Most problems that come along with rigid deformations can be solved by using a continuous deformation function with respect to the joint values. Komatsu applied such a function to

deform the control points of biquartic Bezier and Gregory patches [Komatsu 88]. A distinction between bending and twisting motions is made and local deformations rules are applied to the control points accordingly. The outcome is a smooth shape (thanks to the Bezier patches) that bends and rounds near joints upon flexion, with some crude approximation of muscle swelling.

At roughly the same time, the Thalmanns introduced the concept of *Joint-dependent Local Deformation* (JLD) [Magnenat-Thalmann 87] [Magnenat-Thalmann 88]. Like with Komatsu's approach, JLDs deform the skin algorithmically. In a first pass, the skin vertices are mapped to skeleton segments, thus restricting the influence of a joint to the two segments it connects. Afterwards, the vertices are deformed (using their parameterization in terms of segments) by a function of the joint angles. The numerous function parameters allow to adjust the amount and the extent of rounding near the joints as well as muscle inflation. The approach is demonstrated effectively on the body [Magnenat-Thalmann 87] and on the hand [Magnenat-Thalmann 88] with an exponential function applied on inner finger vertices to mimic muscle inflation.

Komatsu and the Thalmmans showed that fairly realistic skin deformations (see Fig. 2.1) could be derived from the application of specialized algorithms. However, this approach suffers from three problems. Firstly, each type of joint (or even each joint) needs to be handled by a specific function. Secondly, the mathematical functions are perhaps too limited to depict the complexity and individual variability of real anatomy. Last but not least, the graphics designer cannot easily control the deformations since they are determined algorithmically.

Figure 2.1 Marilyn's skin is deformed by JLDs [Magnenat-Thalmann 87].

This last limitation is somewhat alleviated in the work of Forsey where the skin is approximated by hierarchical B-spline patches, whose control points move as a function of joint angles [Forsey 91].

2.1.3 Skinning

Skinning is another type of surface deformer that works locally. It differs from earlier local surface operators in that the algorithm is generic enough to be applied to all kinds of joints and in that full control of the deformations is handed over to the graphics artist.

Skinning is basically an interpolation algorithm. A multitude of different names have been coined for it (even though they all refer to the same technique): skinning [Lander 98], skeleton-subspace deformation [Lewis 00], smooth binding [Alias 01], transform blending [Sloan 01], matrix blending, etc. The basic idea can be summarized as follows. Every skin vertex $\boldsymbol{P}$ is expressed as a linear combination of offset points $\boldsymbol{P_i}$, each of which is rigidly transformed by an associated skeletal coordinate frame:

$$\boldsymbol{P} = \sum_{i=1}^{n} w_i M_i \boldsymbol{P_i} \text{ with } \sum_{i=1}^{n} w_i = 1 \tag{2.1}$$

More precisely, the 3D world coordinates $\boldsymbol{P}$ of the vertex are transformed into the coordinate systems M_i of n relevant joints (usually, $n = 2$) in an initial skeletal posture. The weights w_i, which add up to one, are assigned to the various joints that influence the vertex. When the skeleton is moved, the new position of the skin vertex is found by the same Eq. (2.1).

As demonstrated by video games, skinning is a very simple technique that works rather well. Its initially low computational cost is further reduced by hardware support of matrix blending on recent graphics cards. Skinning is also fairly versatile. Thus, it has been employed to approximate clothes deformation with some success. More, dynamics can be faked by moving the joints dynamically, which in turn leads to a dynamic motion of the bound vertices. To do so, new imaginary joints are usually introduced in the articulated structure. More generally, new joints can be inserted into the articulated structure in order to simulate secondary deformation such as breathing or muscle action [Alias 01].

Recent work focuses on increasing the speed of the skinning procedure. Sun *et al.* [Sun 99] restrict the number of computations by mapping a high-resolution mesh onto a lower-resolution control mesh using the concept of normal-volume. Singh and Kokkevis introduce surface-based FFDs to deform skins [Singh 00]. These surface-oriented control structures bear a strong resemblance to the geometry they deform and can be constructed from the deformable geometry automatically. Similarly to [Sun 99], the advantage of the approach is that one can skin the low-resolution surface control structure and use it to deform the high-resolution object. Houle and Poulin produce skinned meshes with continuous level of detail [Houle 01]. When simplifying the mesh, the global displacement $\Delta\boldsymbol{P}$ of a skin vertex after an edge collapse is simply injected back into Eq. (2.1):

$$\boldsymbol{P} + \Delta\boldsymbol{P} = \sum_{i=1}^{n} w_i M_i (\boldsymbol{P_i} + \Delta\boldsymbol{P_i}) \text{ where } \forall i \in [1 \ldots n],\ M_i \Delta\boldsymbol{P_i} = \Delta\boldsymbol{P}$$

The skinning technique has, however, severe limitations. First of all, assigning weights is at best semi-automatic. A good deal of manual work is ordinarily required to attain acceptable deformations. Indeed, the graphics artist often spends hours to manually adjust the weights, vertex by vertex. Moreover, in areas with considerable mobility like the shoulder, *some* combination of weights may produce good results in certain skeleton postures and yield unacceptable ones in others, leading to considerable frustration by designers. As identified by Lewis *et al.* [Lewis 00], the problem basically is that the desired deformation may not necessarily lie in the subspace defined by the skeletal frames. Consequently, one may very well tweak the skinning weights endlessly, since the “right” combination does not always exist. Two visual defects, depicted in Fig. 2.2, can be recognized as typical products of deformations actually lying outside the skeletal subspace: The elbow collapses when the forearm is bent or twisted [Komatsu 88] [Lewis 00] [Sloan 01]. Nevertheless, due to the simplicity of the method, skinning remains

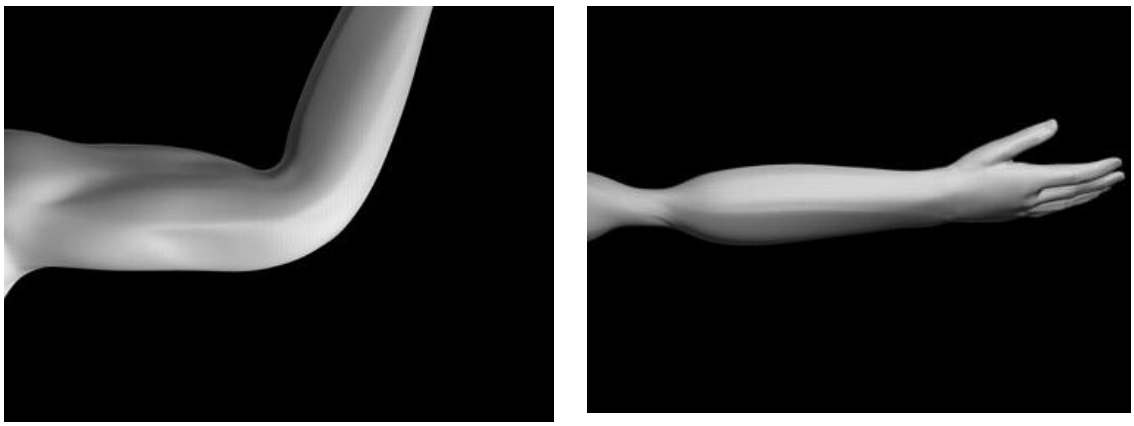


Figure 2.2 Characteristic defects of the skinning algorithm (reproduced from [Lewis 00]). Left: The elbow collapses upon flexion. Right: The elbow collapses when the forearm is twisted.

one of the most widespread techniques for skin deformation.

2.1.4 Contour deformation

The human trunk and limbs exhibit a roughly cylindrical shape. This property can readily be exploited by considering the different body parts as generalized cylinders and manipulating the cross-sections to approximate skin deformations. For example, Thalmann and Shen group the skin vertices in contours [Thalmann 96]. By setting the orientation and position of each contour, they obtain a smooth deformation of human limbs and trunk. As deformations are not computed on an individual vertex basis but for grouping contours, real-time results are easily achieved [Kalra 98].

The algorithm for setting the position and orientation of a cross-section is surprisingly simple. If we consider the deformation of the arm, as illustrated in Fig. 2.3, we have three joints and two segments whose directions are $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$. Let $\boldsymbol{n_u}$, $\boldsymbol{n_o}$ and $\boldsymbol{n_l}$ be the normal vectors of the cross-sections at the ends of the segments. Typically, the end normals are set to the directions of the segments (i.e. $\boldsymbol{n_u} = -\boldsymbol{l_1} / \|\boldsymbol{l_1}\|$ and $\boldsymbol{n_l} = \boldsymbol{l_2} / \|\boldsymbol{l_2}\|$) while the middle normal is assigned the normal of the bisecting plane:

$$\boldsymbol{n_o} = \frac{\boldsymbol{n_u} + \boldsymbol{n_l}}{2}$$

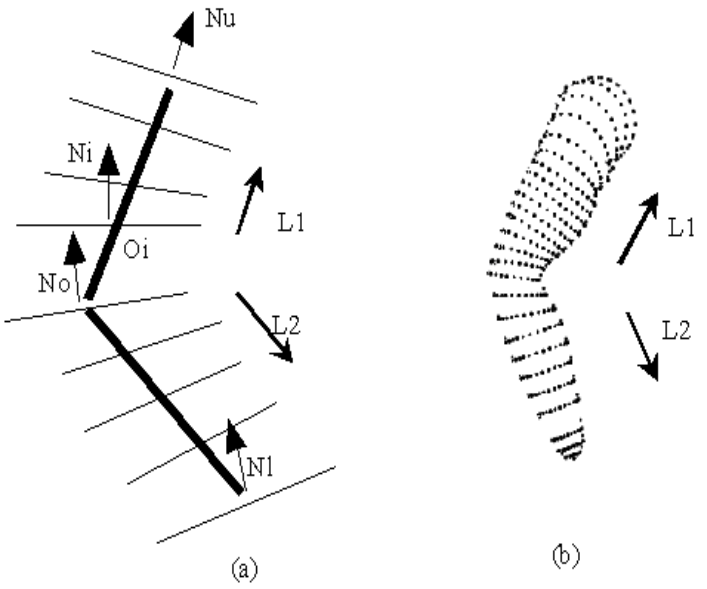


Figure 2.3 Arm deformation with cross-sections.

Then, simple linear interpolation of the two closest end normals gives us the normal $\boldsymbol{n_i}$ for the i-th intermediate cross-section. A similar operation is performed for tangent vectors so that a local frame is eventually constructed for each cross-section. The origin of the frame can be conveniently placed on the skeleton segment. Equipped with these local frames, it becomes straightforward to compute local coordinates for every vertex $\boldsymbol{V_{i,j}}$ of the i-th contour and use these for subsequent deformations.

This technique produces very smooth deformations at a lower computational cost than the more mainstream skinning algorithm because deformations are not computed one skin vertex at a time but for an entire contour. Besides, as vertices are transformed into local coordinate systems, one of the characteristic defects of the skinning algorithm disappears: when the arm is twisted, the cross-sections rotate accordingly, so no shrinking of the arm is produced. On the other hand, the elbow still collapses upon flexion (see Fig. 2.3 b), yet the effect can easily be reduced by adequately scaling the local coordinates within a cross-section. This scheme can also be used for coarsely simulating muscle inflation. On the downside, a contour is solely influenced by the two closest joints, which may lead to poor results in some regions like the shoulder. Furthermore, a specific organization of the mesh is required, which may create additional stitching problems between the limbs and the trunk.

2.1.5 Deformations by example

Recently, there has been a growing interest for producing skin deformations by blending predefined examples or *keyshapes* [Lewis 00] [Sloan 01]. These keyshapes are simply triangle meshes in various skeletal poses. They are acquired by digitizing devices such as laser scanners [Talbot 98], or sculpted by hand in traditional modeling softwares. The basic idea for

deformation is that the keyshapes form an abstract space, from which new shapes can be created by interpolation or extrapolation. Unlike 3D morphing algorithms which essentially try to solve a correspondence problem, the deformation-by-examples approach is confined to the problem of smooth interpolation and possibly of coherent extrapolation. A unique topology (i.e., the same number of vertices and the same connectivity) shared by all keyshapes is, therefore, prerequisite.

As keyshapes are arbitrarily and, thus, irregularly placed in the skeletal parameter space (i.e., the joint angles), shape deformation can be thought of as a scattered data interpolation problem. Radial basis functions are typically used for scattered data interpolation, since these are controllable, smooth, and have a large literature of techniques and extensions. Lewis *et al.* settle on a gaussian radial basis function [Lewis 00] while Sloan *et al.* select a radial basis with a cross-section of a cubic B-spline [Sloan 01].

These shape blending techniques are quite powerful and elegantly solve the typical problems associated with skinning algorithms (see Fig. 2.4). They also perform very fast by comparison with multi-layered deformation models and even achieve real-time results on modest PCs by taking advantage of matrix blending hardware acceleration [Sloan 01]. One of the great strengths of the method is that the designer can add as many keyshapes and as many details as desired so as to accurately control the deformation process. So, unlike skinning, muscle inflation is faithfully reproduced from the predefined examples and can even be exaggerated by extrapolation in the abstract space. But this strength is also the weakness of the method since the number of reference shapes needed grows exponentially with the number of parameters (e.g. degrees of freedom in the articulated structure), thus imposing a high workload on the artist.

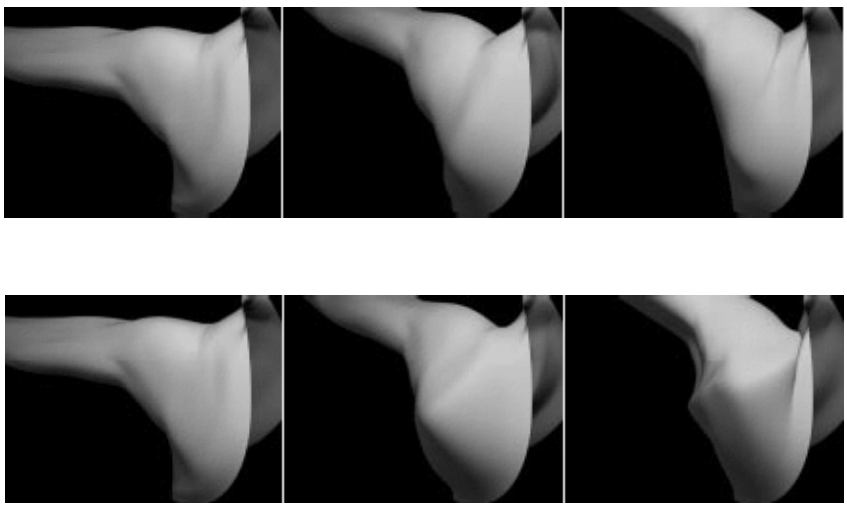


Figure 2.4 Comparison on shoulder deformation between the skinning algorithm (bottom row) and the PSD algorithm (top row) [Lewis 00] using two extreme keyshapes (reproduced from [Lewis 00]).

2.2 Volumetric Models

The first volumetric models [Badler 79] [Herbison 78] relied on elementary geometric primitives such as ellipsoids and spheres to approximate the shape of the body. They were developed in the early age of computer graphics when systems still had very limited capabilities. Implicit surfaces present an interesting generalization of these early models. Last, volumetric models often allow to gracefully handle collisions.

2.2.1 Implicit surfaces

An *implicit surface* (also known as *iso-surface*) is defined by a function that assigns a scalar value to each 3D point in space. An iso-surface is extracted from the level set of points that are mapped to a same scalar value.

The scalar field is generally created from a number of discrete sources (known as *skeletons*) such as points or lines. Each skeleton emits a potential field with a certain distribution described by a *field function.* The metaball formulation is a widely used example of field function: it is a high-order (four or more) polynomial of the distance to the skeleton. In the end, the scalar value at any given point in space is found by summing the contributions of all skeletons. For more details about implicit surfaces and their use in computer graphics, we direct the reader to [Bloomenthal 97].

Implicit surfaces are frequently used to represent organic forms because of their natural smoothness. Early on, Blinn created a “blobby man” [Blinn 82] from an implicit surface generated by point skeletons with an exponentially decreasing field function. A decade later, Yoshomito showed that a complete realistic-looking virtual human could be created with metaballs at a reduced storage cost [Yoshomito 92]. He demonstrates the approach with a ballerina made up of 500 ellipsoidal metaballs and a few ellipsoids on a low-end PC. The author reckons that about 100,000 polygons would have to be used to obtain a comparable visual quality. More complicated implicit formulations have been used as well. Bloomenthal simulates a human arm [Bloomenthal 91] and hand [Bloomenthal 93] with convolution surfaces applied to skeletal primitives that approximate the bones, tendons, muscles, and veins (see Fig. 2.5).

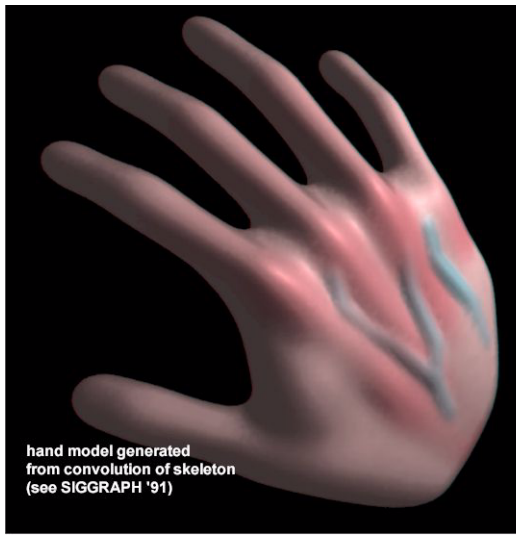


Figure 2.5 Hand modeled with convolution surfaces [Bloomenthal 93].

Implicit surfaces possess many properties that make them suitable for body modeling. Their main distinctive quality is that they smoothly blend into one another because the field functions have C1 or higher continuity, thus yielding very aesthetic shapes. An exception is found

in [Thalmann 96] where there is only geometric continuity so as to increase speed. Another advantage is that implicit surfaces defined by point or polygon skeletons are simple to edit. Last, they offer a very compact formulation, which requires little memory and storage capacity. On the other hand, many issues arise when the model is animated. First of all, unwanted blending typically occurs as the character's skeleton moves. In Fig. 2.5 for instance, the fingers are likely to blend. Shen *et al.* [Thalmann 96] prevent this by dividing the body into coherent parts (right upper leg, left upper leg, etc.) and labeling skeleton points with specific body parts, thus strictly restricting the influence of a metaball to a localized region. More generally, unwanted blending is avoided through the use of *blending graphs*, which specify how the contributions from different skeletal primitives are to be summed [Cani-Gascuel 97] [Wyvill 98]. Polygonizing the implicit surface is another issue that is further complicated by the animation. Since the polygonization of the implicit surface is required for the application of texture maps among other things, a fixed topology of the polygonization must be maintained during the animation. Hybrid techniques mixing surface deformation models and implicit surfaces are generally brought in to resolve this problem. In [Thalmann 96], B-spline patches are fitted to the iso-surface using contour deformation and a ray-casting procedure, while in [Leclerq 01] a skinning algorithm is applied on a polygonal mesh followed by a normal-directed projection of the skin vertices onto the implicit surface.

2.2.2 Collision models

Some volumetric models allow to gracefully handle collisions between different models (or different parts of a same model) and accordingly generate deformed surfaces in contact. Cani-Gascuel integrates elastic properties directly into the formulation of distance-based implicit surfaces [Gascuel 93] and thus establishes a correspondence between the radial deformation and the reaction force. In this way, a stable, precise, C1 contact surface is elegantly defined between colliding models and the resulting reaction forces can be integrated in a physically-based simulation at the next animation step. Various extensions, which preserve the C1 continuity of the surface, allow to propagate the deformation in the region surrounding the contact surface [Gascuel 93], to locally control the volume variation [Cani-Gascuel 97], and to introduce anisotropy in the deformation [Cani-Gascuel 98].

Recently, Hirota *et al.* proposed a non-linear finite element model of a human leg [Hirota 01] derived from the Visible Human Database. The demonstration of their system highlights the contact area between the calf and the posterior side of the thigh when the knee is flexed and achieves a high level of realism in the deformation (see Fig. 2.6). In order to estimate the amount of penetration at every skin vertex, the authors introduce the notion of *material depth* (see Fig. 2.7), which is a continuous approximation of the distance fields in a deformed configu-

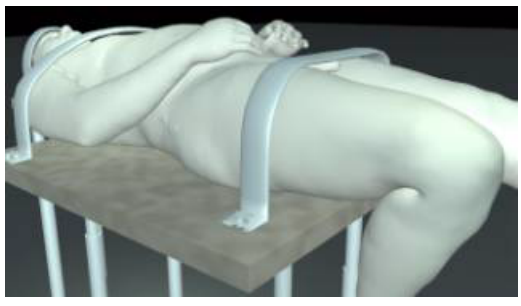


Figure 2.6 Leg deformation including contact surface using FEM [Hirota 01].

ration of the model based on the distance fields in the undeformed configuration. Penalty forces, which are accordingly computed from the penetration function, are then integrated continuously over the model surface.

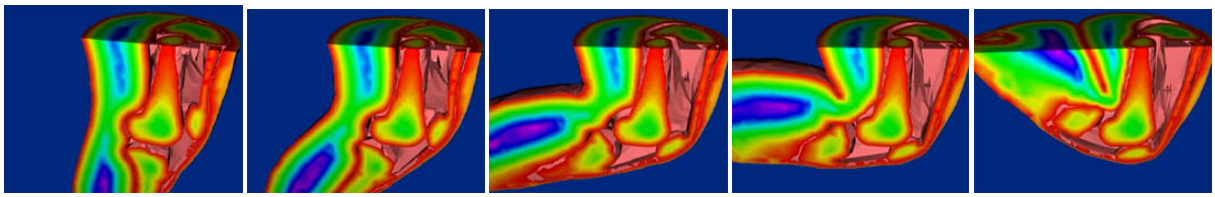


Figure 2.7 The material depth (encoded in pseudo color) is used for computing the contact surface when the knee is bent [Hirota 01].

Frisken *et al.* propose to use adaptively sampled distance fields (ADF) representations [Frisken 00] for handling soft body collisions [Frisken 01]. Analogously to implicit surfaces, inside/outside and proximity tests can be performed rapidly using ADFs because the notion of interior and exterior is inherent to the formulation. Furthermore, potential contact regions can be quickly localized by exploiting the spatial hierarchy of the data, while the region of overlap can be accurately represented by simply substracting the ADFs of different objects [Frisken 01]. Finally, as for [Hirota 01], a continuous penalty force can be defined over the whole volume overlap.

2.3 Multi-Layered Models

Chadwick *et al.* first coated their character with an additional muscle layer [Chadwick 89]. Since then, most researchers have used a layered approach to character modeling, with more or less attention to the anatomical accuracy of their model. Some models rely on a combination of ordinary computer graphics techniques like skinning and implicit surfaces, and tend to collapse several anatomical layers into one. Others are more inspired by the actual biology of the human body and attempt to represent and deform every major anatomical layer, and to model their dynamic interplay.

2.3.1 Skeleton layer

The skeleton is defined by an articulated structure, which consists in a hierarchy of segments (e.g. [Thalmann 96]) or, in a few rarer instances, of parallelepipeds [Gascuel 91]. The articulated structure is sometimes covered by material bones approximated by simple geometric primitives or triangle meshes [Hirota 01] [Porcher 98] [Scheepers 96] [Wilhelms 97a] [Wilhelms 97b].

Work on the skeleton layer is mainly concerned with the immaterial articulated structure, in particular with the accurate characterization of the range of motion of specific joints. In the field of biomechanics, Engin and coworkers [Engin 89] have studied the directional limits of

the upper arm while Wang *et al.* [Wang 98] have measured its twisting limits. In computer graphics, different models of joints limits have been suggested: Korein [Korein 85] uses spherical polygons as boundaries for the directional component of spherical joints like the shoulder while Maurel and Thalmann have recourse to joint sinus cones for the shoulder and scapula joints [Maurel 00]. The coupling between joints has also been investigated. Monheit and Badler [Monheit 91] [Badler 93] propose to distribute the total bending, rolling and twisting of the spine on the individual vertebrae according to user-defined weights. Maurel *et al.* constrain the scapula to slide on the thorax using inverse kinematics [Maurel 00]. Coupling in the shoulder complex is also handled by Scheepers *et al.* who separate the scapula from the arm skeleton and define its motion functionally [Scheepers 96].

2.3.2 Muscle layer

Geometric models

In the Critter system [Chadwick 89], the foundations for the muscles and fat deformations are based on Free Form Deformations (FFD) [Sederberg 86]. In practice, a muscle is encased in a pair of adjoining FFDs oriented along the skeleton link. In total, seven planes of control points slice the muscle. The two control planes at either end function to ensure C1 continuity with other connected muscles. The deformation of the central cubical volume via the three remaining control planes produces muscular deformation. Kinematic deformations are controlled by establishing a relationship between the control points of the mid planes and the joint angles in the skeleton while dynamic deformations result from the elastic deformation of an embedded mass-spring network built from the FFD control points. Similarly, Moccozet models the behavior of the hand muscles [Moccozet 96] using Dirichlet Free Form Deformation (DFFD) (DFFD is an interesting generalization of FFD that removes the severe limitation imposed on the shape of the control box and provides a more local control of the deformation). The resulting geometric deformations look convincing despite the complicated branching structure of the hand.

Figure 2.8 Muscle layer abstracted by FFDs. Left: Bragger bones [Chadwick 89]; Right: Hand deformation based on Dirichlet FFDs [Moccozet 96].

Implicit surfaces have been recognized as highly suitable for modeling organic forms. Hence, they have been extensively used to model the muscle layer too. In [Turner 93], muscles are approximated by implicit primitives like spheres and super quadrics. In [Thalmann 96], grouped ellipsoidal metaballs with a simplified quadratic field function are used to mimic the gross behavior of bones, muscles, and fat. The contraction and release behavior of muscles is simulated by binding the nine degrees of freedom of an ellipsoid (i.e., rotation, translation, and scaling) to the degrees of freedom of the skeleton. For example, the bulging and the flattening of the leg muscles can be engendered by ellipsoidal metaballs whose scaling parameters are tied to the knee flexion (see Fig. 2.9). The technique reaches its limits in regions of high mobility (the shoulder, typically), in which each ellipsoidal primitive is simultaneously influenced by several joints. In that case, linear interpolation of the contributions of the various joints as in [Thalmann 96] may lead to unsatisfactory deformations. More recently, Leclerq *et al.* also modeled muscles with ellipsoidal metaballs [Leclerq 01]. They do not say, however, how or even whether muscle inflation is faked.

Figure 2.9 Leg muscles mimicked by deforming metaballs [Thalmann 96].

When the deformation is purely geometric, muscle models tend to use the ellipsoid as the basic building block. It is a natural choice because an ellipsoid approximates fairly well the appearance of a fusiform muscle[1]. Besides, the ellipsoid has an analytic formulation that lends itself well to inside/outside tests. When the primitive is scaled along one of its axes, the volume of the primitive can moreover be preserved by adequately adjusting the lengths of the two remaining axes (see Fig. 2.11). The ratio of the height to width can be kept constant in the same manner. This is why two research teams use a volume-preserving ellipsoid for representing a fusiform muscle [Scheepers 97] [Wilhelms 97a]. In contrast to earlier approaches, the work of these two teams is peculiar in that the emphasis is laid on building anatomically accurate replicas of the musculature of humans or animals. Scheepers and his colleagues, for instance, detail every superficial muscle of the upper body and explicitly introduce tendons that connect the muscle bellies to the bones (see Fig. 2.10). For multi-belly muscles such as the pectorals in the chest, they position a set of ellipsoids along two spline curves [Scheepers 97].

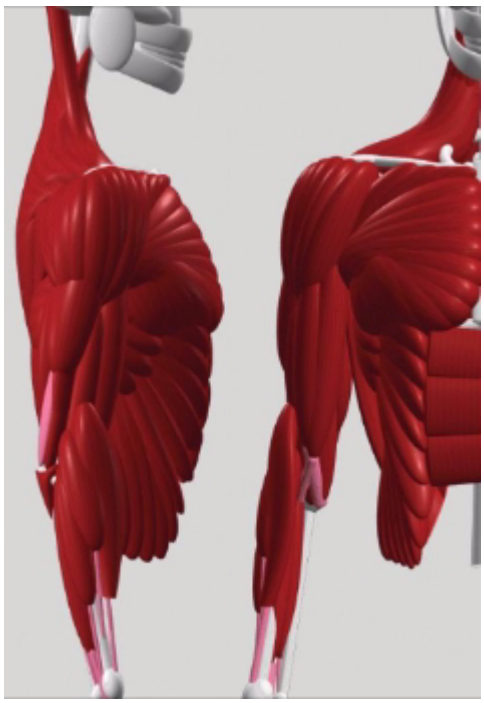


Figure 2.10 Anatomically-based modeling of the human musculature using ellipsoids [Scheepers 97].

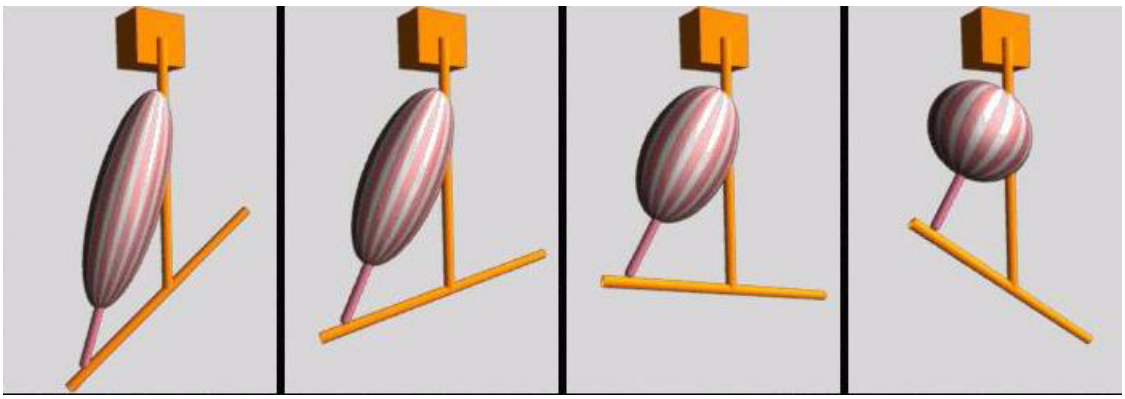


Figure 2.11 A volume-preserving ellipsoid is a good approximation of a fusiform muscle (reproduced from [Scheepers 97]).

1. The shape of a fusiform muscle tapers at either end. Fusiform muscles are common in the limbs.

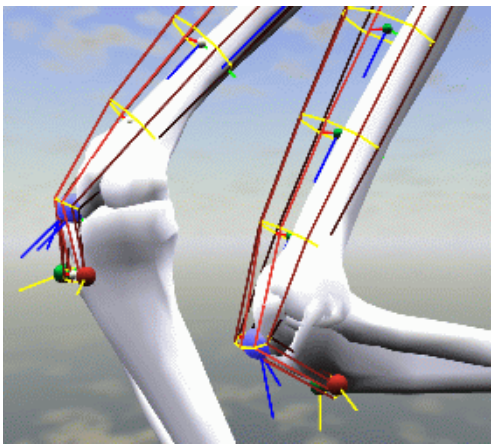


Figure 2.12 A deformed-cylinder muscle model. Pivot points (in blue) ensure a smooth bending of the muscle around joints [Wilhelms 97b].

Both research teams nevertheless admit that the ellipsoid, albeit a good approximation in simple cases, fails to capture the shape of more complicated muscles. Hence, Scheepers *et al.* also provide a general model that consists in tubularly-shaped bicubic patches capped with elliptic hemispheres at either end [Scheepers 97] and use it for modeling muscles that bend such as the *brachialis* in the arm. Exact volume preservation remains possible because the general muscle model still has an analytic description. Likewise, Wilhelms resorts to generalized cylinder versions of the muscles in her latest work on animal modeling [Wilhelms 97b]. A muscle is made up of seven elliptic cross-sections that consist in turn of a certain number of points that can be edited to adjust the overall shape. Initially, the axis of the cylinder runs in a straight line from the origin on the proximal bone to the insertion on the distal bone. Two pivot points can, however, be inserted to bend the axis (see Fig. 2.12) and diminish the penetration of other anatomical structures. During animation, whenever a joint lying between muscle origins and insertions moves, a new deformed cylinder shape is computed. Based on the change in length of the cylinder's axis, the width and thickness of every cross-section is scaled to maintain approximately constant volume.

The generalized cylinder muscle model had already been proposed previously by Dow and Semwal [Dow 93]. In their work, cross-sections are represented by B-spline curves controlled by two parameters, the BaseSize and the GrowthFactor. The former represents the fixed minimal distance between the spline control point and the cylinder axis while the latter governs the motion of the control point further away from the axis according to the tension of the muscle. Collisions between muscles are detected based on polygonal versions derived from the cross-sectional splines, and control points are accordingly pushed towards the cylinder axis.

A distinctive, and perhaps unique, feature of the system by Scheepers and colleagues [Scheepers 97] is that, besides isotonic contractions, isometric contractions of the muscles can also occur. A tension parameter, which controls the ratio of a muscle's height to its width, is bound to articulation variables and thus allows to alter the form of a muscle that does not undergo any change in length. This effect is noticeable in the biceps as the hand is clenched into a fist (see Fig. 2.13).

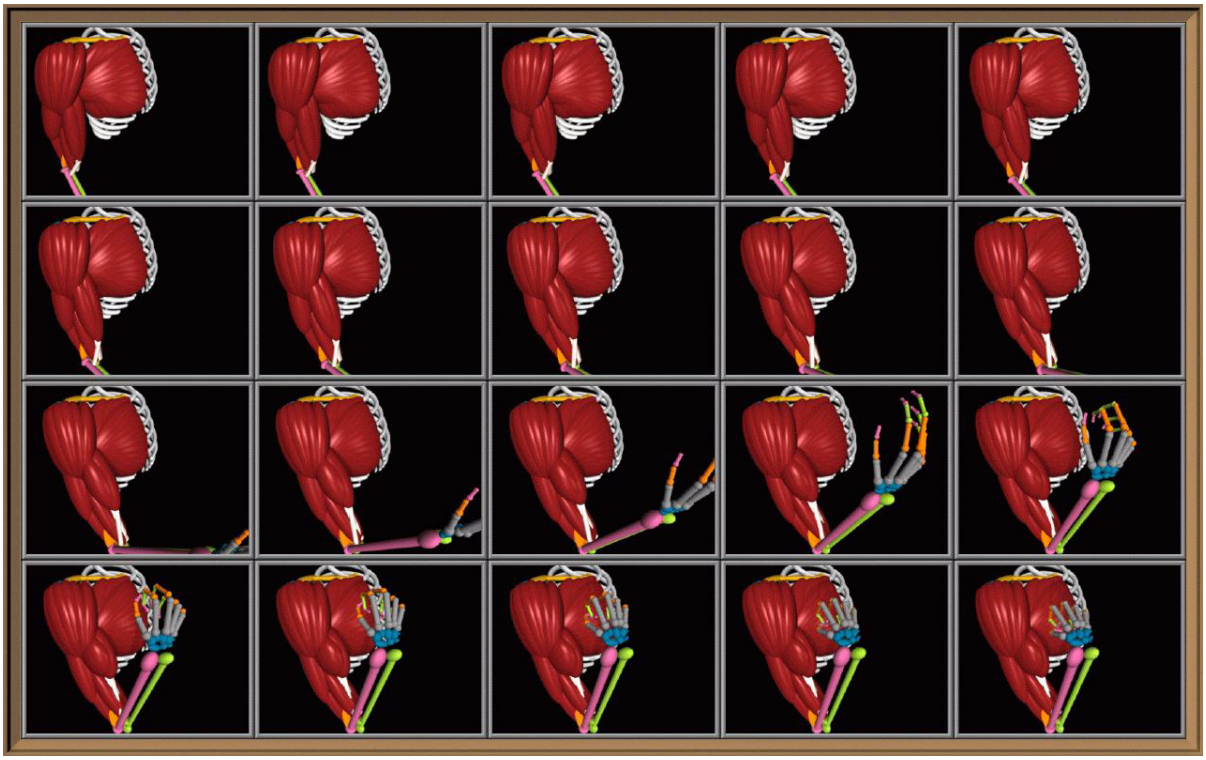


Figure 2.13 Isotonic contraction of the arm muscles followed by an isometric contraction when the hand is clenched into a fist [Scheepers 97].

In the field of biomechanics, the SIMM software [Delp 00] stands out as a possible platform for a computer graphics simulation of muscles. In SIMM, the geometry of a muscle-tendon unit is characterized by a series of points (at least two), which are connected by line segments. These segments define the admissible path for the muscle. They are deflected by wrapping surfaces (ellipsoids and cylinders) to prevent the penetration of surrounding structures (see Fig. 2.14) and further constrained by via points, which are anchored to specific joints and activated in a user-defined range of motion. The segments are not covered by more meaningful representations, however.

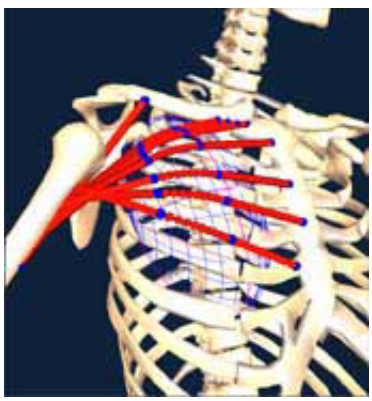


Figure 2.14 Representation of the pectoral in SIMM [Delp 00]

Physically-based models

Gourret and the Thalmanns [Gourret 89] first generated tissue deformation by applying the engineering Finite Element Modeling (FEM) technique. A human hand grasping an elastic ball is shown as an example. A linear constitutive law for the flesh tissue is used in the context of small strains and quasi-statics analysis produces for each frame a large linear system, which

relates the displacement of the nodes to the forces via the stiffness matrix. In matrix form:

$$K\boldsymbol{u} = \boldsymbol{f} \tag{2.2}$$

Where K is the stiffness matrix and $\boldsymbol{u}$ the displacement vector. Boundary conditions can be imposed by assigning fixed values to some components of $\boldsymbol{u}$. Theoretically, Eq. (2.2) is valid provided that the displacement field and the displacement gradient be small. Obviously, neither condition is met when soft tissues deform. This two-fold hypothesis is, however, often assumed in computer graphics so as to avoid the non-linearities produced by finite (i.e., large) deformations. Similarly, biological material is in reality nonlinear but a linear stress-strain relationship is frequently chosen for simplicity's sake.

Zhu *et al.* [Zhu 98] deform the *anconeus* muscle using FEM and volume graphics. They use eight-node 3D brick elements to represent a muscle as a collection of voxel elements. Chen and Zeltzer also rely on the FEM technique to obtain muscle deformation [Chen 92]. They use twenty-node 3D bricks and a muscle model that is based on the work of Zajac in biomechanics [Zajac 89]. In both cases, muscles work in isolation and are not covered by a skin.

More recent works [Hirota 01] [Lemos 01] deal with the geometric non-linearity induced by large deformations and/or model more realistic non-linear soft tissues. The non-linear FEM solver in [Lemos 01] uses eight-node brick-like elements while tetrahedral elements are used in [Hirota 01]. In both cases, for each integration step, the equation of motion is linearized by taking Newton-Raphson steps [Press 92]. Lemos and coworkers apply their non-linear finite element solver to deform the *soleus* and *gastrocnemius* leg muscles of a cat and underscore the different deformations resulting from the different fibers arrangements in the two muscles. Hirota *et al.* not only deform the leg muscle layer but also the other anatomical structures with a finite element mesh made up of 40,000 tetrahedra and 10,000 nodes that comprise the major bones of the leg, some muscles, tendons, and ligaments, and a monolithic skin-fat layer (see Fig. 2.15).

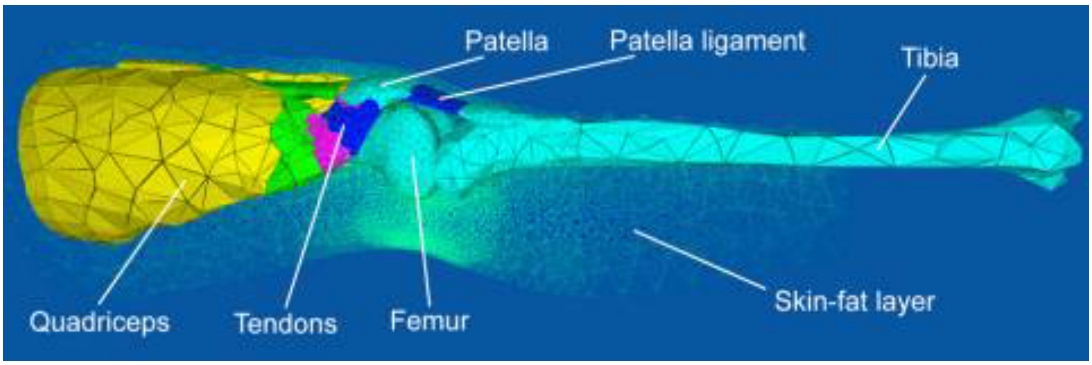


Figure 2.15 Finite element model of the leg in [Hirota 01].

Gascuel, Verroust, and Puech [Gascuel 91] rely on the simpler mass-spring model to deform the muscles and flesh of the character. They associate deformable components to each skeleton

link. Cones of flesh are arranged in a star-shaped way around the center of mass of the link and radially deformed by damped sprigs (see Fig. 2.16). The deformations are propagated from one flesh cone to the others by a second module so as to meet dynamic or geometric criteria (e.g. constant volume). The deformable components are well-suited for detecting collisions and responding to them.

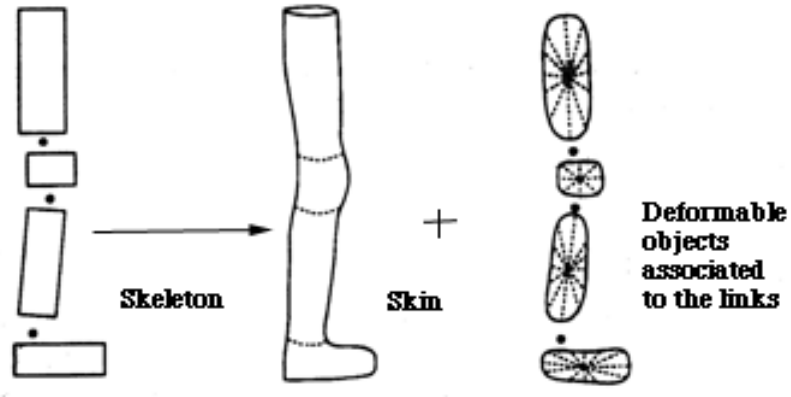


Figure 2.16 Deformable components [Gascuel 91]

Nedel and Thalmann [Nedel 98] introduced the idea of abstracting muscles by an action line (a polyline in practice) representing the force produced by the muscle on the bones, and a surface mesh deformed by an equivalent mass-spring network. An elastic relaxation of the surface mesh is performed for each animation frame, thus yielding a collection of static postures. The authors acknowledge that their model can only work on fusiform muscles. Furthermore, it must be noted that the authors do not explicit how they constrain the surface mesh to follow the action line when it consists of more than one segment. An interesting feature of their mass-spring system is the introduction of *angular springs* which help to control the volume variation of the muscle and to smooth out the geometric discontinuities of the surface. An angular spring works by connecting a point $\boldsymbol{P}$ on the surface to a virtual anchor, whose position is computed as the average of the two neighbors of $\boldsymbol{P}$ in the horizontal or vertical direction. In this way, it exerts a force on the surface vertex that tends to restore the initial curvature. It is important to note that angular springs do not necessarily work well for large deformations: if the local curvature changes from a valley to a summit (or the reverse), the spring may come to a rest although the curvature has become inverted.

Ng-Thow-Hing [Ng-Thow-Hing 00] uses B-spline solids for modeling muscles. He points out that, unlike many other representations, the B-spline solid (see Fig. 2.17) can capture the multiple muscle shapes (fusiform, triangular, bipennate, etc.) and can also render various sizes of attachments. Among other things, the aponeurosis, which is a wide sheet of tendon, is easily modelled. Muscle reconstruction involves fitting the control points of the B-spline solid to raw data acquired by different modalities (the data are typically represented by a volumic cloud of points). A 3D mass-spring-damper network implanted in the B-spline solid forms the basis for muscular deformation. The network does not correspond to the B-spline control points because these may be immaterial but to spatial points of maximum influence. Unfortunately, an inevitable consequence of the duality between the B-spline control points and the spatial material points is an increase of the computational complexity. Varying force magnitude in the network results in non-uniform physical effects. While most physics-based models simulate a sequence of static equilibrium problems (i.e., quasi-statics), the approach of Ng-Thow-Hing allows to observe viscosity or inertia effects such as creep or oscillations. The author also incorporates muscle/{muscle,bone} collision forces as reaction constraints [Platt88]. However, no explicit solution is given as to how multiple collisions between muscles can be resolved within the same integration step.

Figure 2.17 3D B-spline models of leg muscles [Ng-Thow-Hing 00]

2.3.3 Fat layer

Few models explicitly represent fatty tissues and model their behavior. In fact the fat layer is frequently blended into the muscle layer as in [Chadwick 89] and [Thalmann 96] for example.

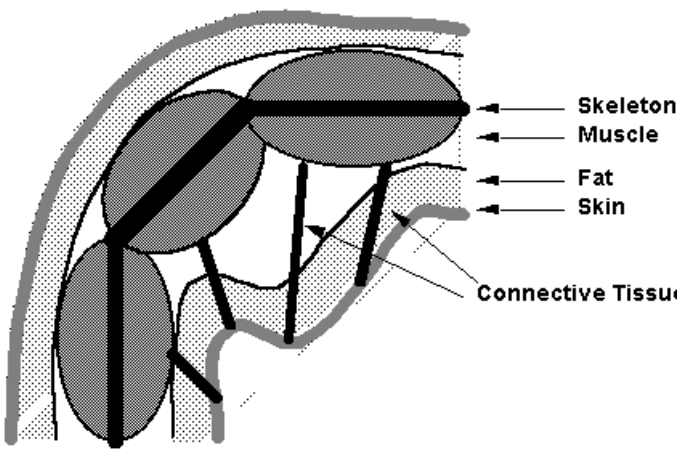


Figure 2.18 Layers in LEMAN [Turner 93]

In the LEMAN system, the fat layer is modeled as a thickness below the skin layer [Turner 93]. The thickness of the fat layer is adjusted globally, or locally, skin vertex by skin vertex. When the model is animated, the behavior of the fatty tissues is approximated by hookian springs connecting skin vertices to the nearest perpendicular points of the muscle layer (see Fig. 2.18). In practice, the fat springs are not actually created but equivalently replaced by reaction constraints

[Platt 88] applied to the skin vertices that penetrate the muscle layer surface displaced by the fat layer thickness.

Similarly, Wilhelms and Van Gelder [Wilhelms 97b], Lee *et al.* [Lee 95], as well as Ng-Thow-Hing [Ng-Thow-Hing 00] model the elastic behavior of the fatty tissues by springs anchoring the skin vertices to the muscle layer beneath. For [Wilhelms 97b], the anchor points on the muscle layer are computed using a parametric trilinear transformation over two adjacent slices of deformed-cylinder muscles. In the work of Lee and co-workers [Lee 95], a tri-layer mass-spring lattice approximates the muscle, fat and skin layers of the face. Different elasticity parameters are associated with each layer, thus reflecting the heterogeneity of the tissues. A particular feature of the springs in the fat layer is that they are readily extensible at low strains, and yield an increasing restoring stress after a certain threshold. This biphasic behavior is closer than linear springs to the true stress-strain relationship of the human skin.

When implicit surface techniques are used for extracting the skin from the muscle and skeletal layers, an offset can easily be applied to account for the thickness of the adipose tissues. Scheepers and his colleagues [Scheepers 97] adjust the radius of influence of the density functions derived from the implicit versions of the muscle primitives. Wilhelms and Van Gelder [Wilhelms 97b] voxelize the region around the character in a rest pose and extract an iso-surface at some distance from the bones and muscles by choosing an appropriate threshold (see Fig. 2.3.4).

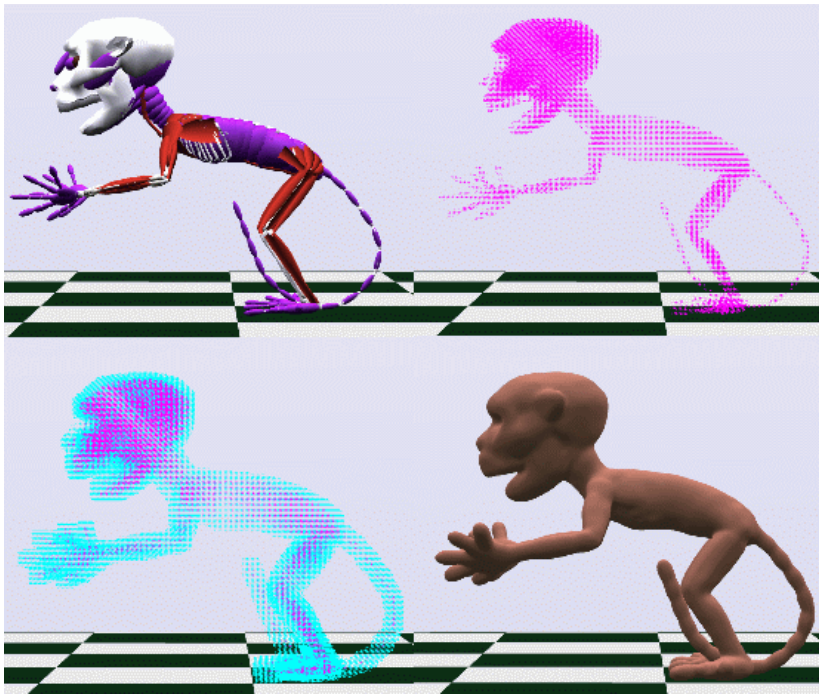


Figure 2.19 Skin extraction by voxelizing the inner layers in a rest pose [Wilhelms 97b]. The initial density function (upper right) is blurred and moved away from the underlying components (lower left) by a filter and an appropriate threshold.

2.3.4 Skin layer

Skin has been modeled by every type of surface:

- Polygonal [Chadwick 89] [Hirota 01] [Wilhelms 97b],
- Parametric [Gascuel 91] [Henne 90] [Scheepers 97] [Thalmann 96],
- Subdivision [DeRose 98] [Leclerq 01],
- Implicit [Bloomenthal 93] [Cani-Gascuel 98].

The main advantage of polygonal surfaces is that they can be directly processed by the graphics unit. So, when speed or interactivity are called for, polygon meshes are most eligible. However, various schemes may be needed to smooth out the surface discontinuities. Wilhelms *et al.* [Wilhelms 97b] filter the voxelization of the muscle and skeleton layers by a Gaussian kernel (see Fig. 2.3.4). The side-effect of the filter is that it also removes -somewhat indiscriminately- the possible fine details in the muscle layer.

Parametric surfaces, such as bicubic patches, are appealing candidates for modeling the skin because they naturally yield smooth shapes. Note that in the end, however, B-spline or Bezier patches must be polygonized for rendering purpose. Shen and Thalmann [Thalmann 96] relax the continuity constraints usually placed on soft object field functions and derive a lower degree polynomial field function for the inner layers because they compensate the loss of C1 continuity of the inner layers by the use of cubic B-spline blending for the skin (see Fig. 2.20).

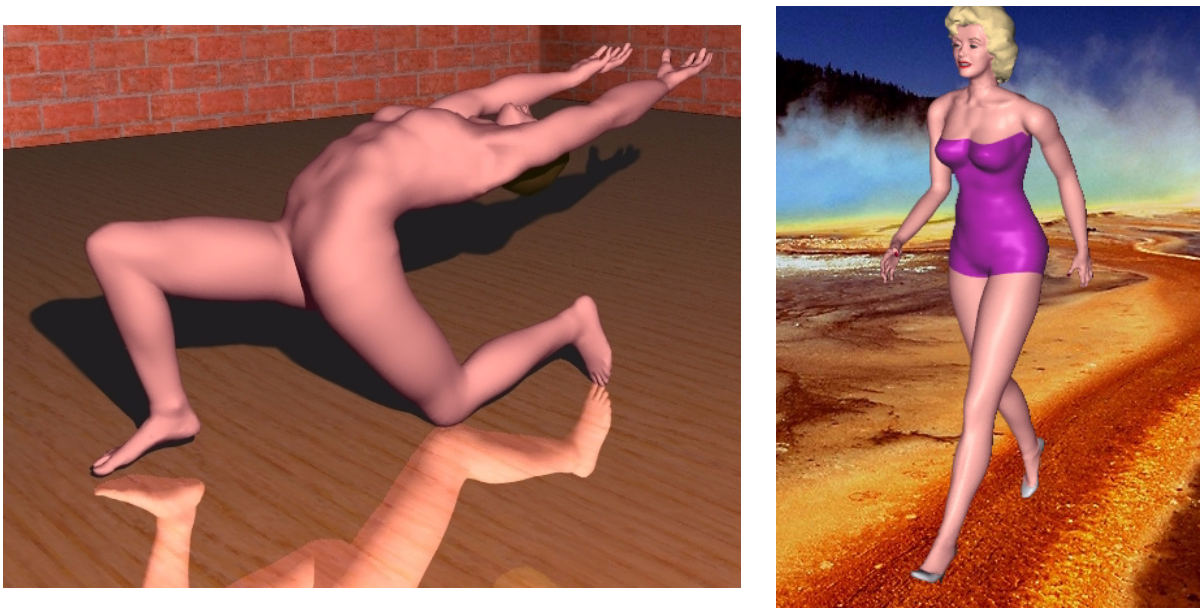


Figure 2.20 Smooth geometric body deformations produced by B-spline surfaces fitted to an isosurface [Thalmann 96].

As noted before, implicit surfaces are quite appropriate for representing organic forms. They

nevertheless suffer from a number of problems, which become conspicuous when the model is animated. The chief obstacle to their use is that the application of texture maps is difficult or even impossible (see Section 2.2.1). This is why they are rarely used for directly extracting a skin and used more often for deeper invisible anatomical layers.

Subdivision surfaces are perhaps the most befitting representation for the skin layer (see Fig. 2.21). They combine several advantages: smoothness can be guaranteed by recursively subdividing the surface, a polygonal version suitable for rendering is automatically derived without further computations, and interpolating schemes can be used [Zorin 96] (unlike B-spline surfaces whose control points do not lie on the surface).

Figure 2.21 The skin of Geri is created from Catmull subdivision surfaces [DeRose 98].

There basically exist three ways to deform the skin in multi-layered models:

- Surface deformation models are first applied to the skin. In a second stage, the skin is projected back onto the inner anatomical layers. Alternatively, both stages may happen concurrently, as in [Thalmann 96].
- The skin is deformed by a mechanical model of elastic membrane and constrained to stay a certain distance away from the material beneath.
- The skin is defined as the surface of a volume finite element/mass-spring model of the body.

The former deformation model is used by Leclerq *et al.* [Leclerq 01] who subdivide a coarse skin mesh using an N-adic decomposition of the triangles and project the created vertices along their normals onto an implicit surface generated from ellipsoidal metaballs. In [Thalmann 96], regular skin cross-sections are sampled by casting rays from the skeleton segments in a star-shaped manner. The orientation of the cross-sections is determined by contour deformation (see Section 2.1.4). The outermost points where rays intersect the implicit surface, which is also defined by ellipsoidal primitives, are subsequently used as skin B-spline control points (see Fig. 2.22).

The second case is mainly illustrated by [Henne 90], [Turner 93], and [Wilhelms 97b]. Wilhelms and Van Gelder simulate the motion of the skin by elastic relaxations of an equivalent mass-spring system [Wilhelms 97b]. Each skin vertex is anchored to the closest underlying component and each edge of the skin mesh becomes a spring whose stiffness is set to the area of the two adjacent triangles divided by the edge's length squared [Van Gelder 98]. This formula provides a more accurate model of uniformly elastic skin that would uniform stiffness for

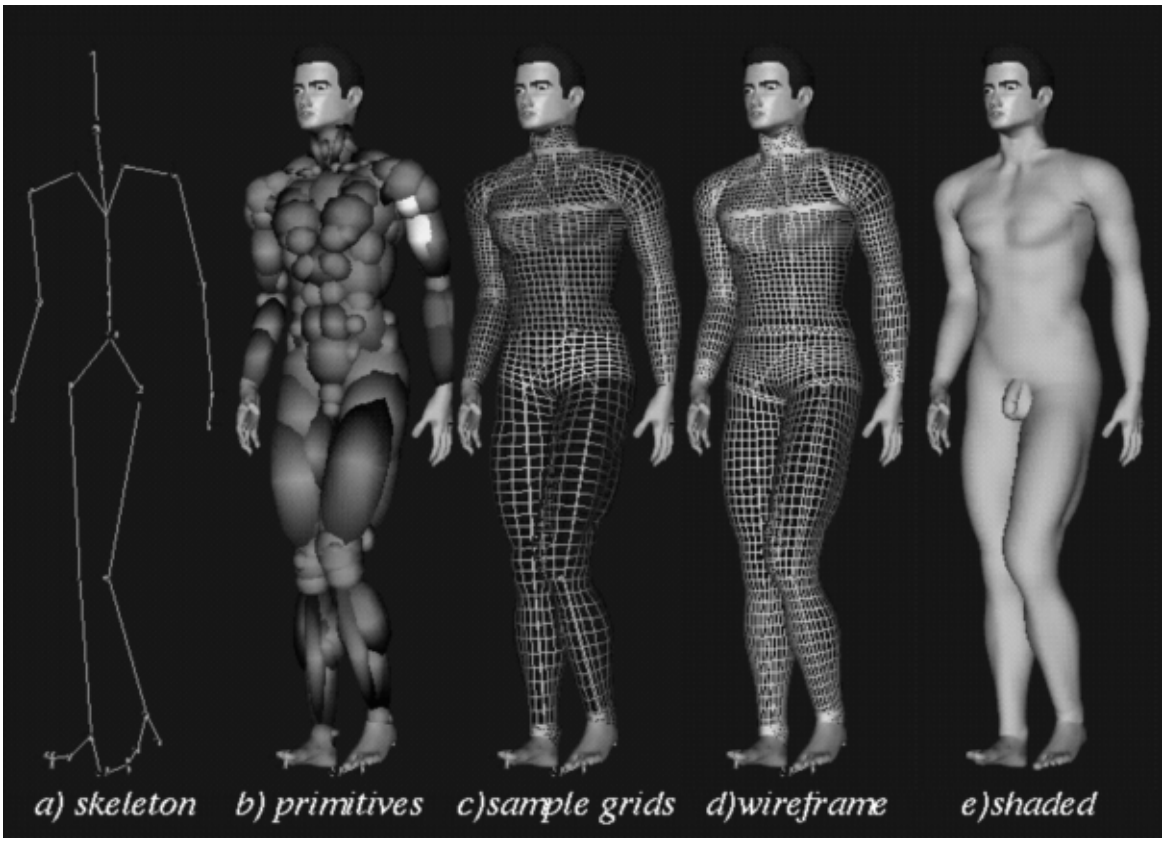


Figure 2.22 The layered model by Shen and Thalmann [Thalmann 96]. Ellipsoidal primitives (b) form an implicit surface, which is sampled by a ray-casting process. The sample points (c) are used as B-spline control points. The B-spline surface is ultimately polygonized at the desired resolution (d and e).

all springs (see Fig. 2.23). In the LEMAN system designed by Turner and Thalmann [Turner 93], the skin is deformed by a mechanical model based on [Terzopoulos 87] that leaves it free to slide over the underlying surfaces (see Fig. 2.24). The skin resistance to bending is ignored in the mechanical simulation since real skin bends much more easily than it stretches. Unlike [Wilhelms 97b], the emphasis in LEMAN is laid on producing a dynamic skin motion with typical squash and stretch effects [Lasseter 87]. The main limitation of the system is that the skin mesh is topologically restricted to a rectangular grid.

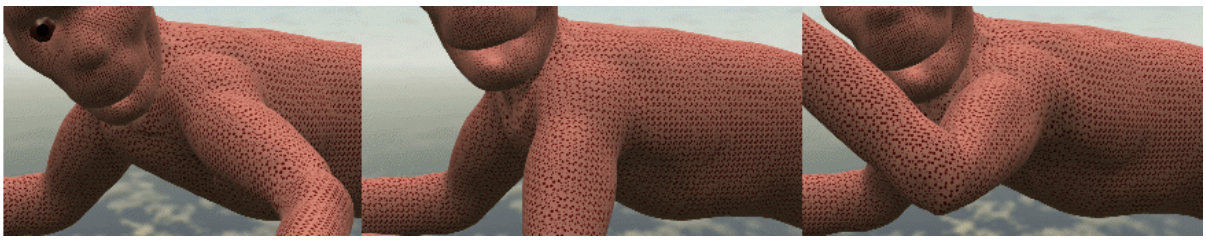


Figure 2.23 Three frames of the monkey shoulder animation. The skin mesh is deformed by a mass-spring network with varying stiffness and elastically anchored to the underlying muscles [Wilhelms 97b].

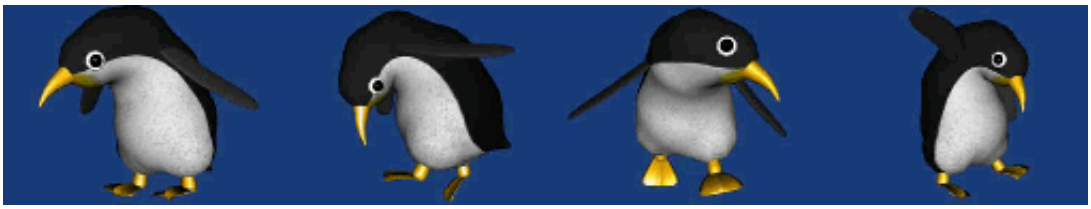


Figure 2.24 Elastic skin that slides over the underlying surfaces [Turner 93].

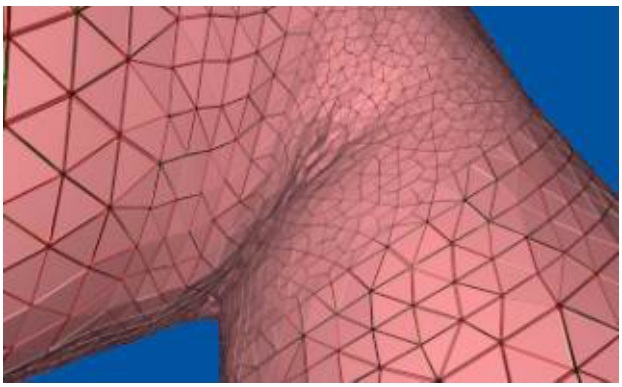


Figure 2.25 The skin buckles and creases in the work of Hirota *et al.* [Hirota 01].

Finally, the third and last deformation model is exemplified by the excellent work of Hirota *et al.* [Hirota 01]. In their system, the skin triangle mesh is chosen to be a subset of the sides of the tetrahedral finite elements that mesh the interior of the body. Unlike other works, they manage to obtain fold and crease patterns in the skin. In counterpart, their simulation is far from real-time, reaching a whooping computation time of 376 minutes on a 300MHz R12000 processor for a fifty frame animation of leg flexion. Another example is the work of Lee *et al.* [Lee 95], where the elastic triangular prism elements that approximate the skin and fatty tissues match the triangles in the facial mesh.

2.4 Conclusion

We have presented the main works in computer graphics on 3D character modeling and deformation. To conclude, we draw a comparison between the various approaches that have been proposed and outline the desirable features of a good multi-layered model.

2.4.1 Depth of simulation

As mentioned by Lewis *et al.* [Lewis 00], the depth of simulation is a prevalent issue in computer graphics. Kinematic approaches to animation, e.g. inverse kinematics, received much attention in the early days of computer animation while physics-based models that simulate the dynamics of motion only appeared as computational power became more readily available. Inversely, early rendering methods like ray-tracing and radiosity were based on physics and sub-disciplines like optics while more recent techniques aim at capturing the complexity of 3D models with shallow image-based representations.

Similarly, in character deformation both deep and shallow approaches have their place. Recent shallow approaches like the Pose Space Deformation [Lewis 00] paved the way for fast synthesis of realistic-looking deformation by turning the deformation problem into a modeling problem (this consequently means that the accuracy of the deformation entirely depends upon the graphics artist). At the other extreme, researchers have proposed anatomically correct musculo-skeletal models that produce great-looking results. These deep models promise universally accurate simulation. However, producing anatomically plausible models is a daunting task as shown by the huge body of literature in biomechanics.

Like in other areas such as animation, the future of body deformations probably lies in a combination of shallow and deep approaches. One can very well conceive an anatomically-based system (like the one we propose in this thesis) whose output is used for generating the various keyshapes required by recent surface deformation techniques, thus combining the precision of deep simulation models with the speed of shallow deformation models.

2.4.2 Comparative analysis

Table 2.1 summarizes the existing surface deformation models while Table 2.2 shows a comparison between the different multi-layered models that have been proposed. We do not compare surface models and multi-layered models since their complementarity has already been discussed.

Surface Deformation Model	Computation	Complexity	Physical Realism	Visual Realism	User Control
Rigid body parts	Very Low	Lowest	Low	Very Low	- (None)
Specialized surface operators (e.g. JLD)	Low	Low to Medium	Low	Medium	- (None)
Skinning	(Very) Low	Low	Low	Medium	Medium

Table 2.1 Comparative analysis of surface deformation models

Surface Deformation Model	Computation	Complexity	Physical Realism	Visual Realism	User Control
Contour manipulation	(Very) Low	Low	Low	Medium	Medium
Keyshapes interpolation (e.g. PSD)	Low to Medium	Low to Medium	Low	- (depends on artist)	High
Elastic surfaces (e.g. LEMAN)	Medium	Medium	Medium	Medium	Low to Medium

Table 2.1 Comparative analysis of surface deformation models

Predictably, surface deformation models have a rather low computation time. Most of them execute in real-time for moderately detailed skin meshes on low-end hardware. The major flaw of specialized surface operators is that the deformation is algorithmically determined. As a result, graphics designers cannot interactively improve the deformation. Additionally, mapping a deformation algorithm designed for a specific model to another model is bound to be problematic. Skinning and contour deformation techniques approximately produce the same visual quality (with a slight edge for contour manipulation) at roughly the same computational cost. They both allow a fine control through the manipulation of weights. In counterpart, both require a good deal of work as the designer may have to get down to the vertex level for bettering the deformation. Though physically more realistic, 2D mass-spring systems and other elastic surface models have seldom produced visually-appealing results. Keyshape interpolation in an abstract space stands out as the best compromise among surface models. Nevertheless, as noted before, the visual quality entirely rests with the graphics artist. Moreover, the number of keyshapes grows exponentially with the number of parameters. These parameters include the joint axes around which rotations may occur and possibly less obvious ones such as the degree of muscle activation. Eventually, this may lead to an explosion in the number of required keyshapes. Also, the impact of the interpolation function on the deformation is not well-known.

Deformation Model	Computation	Complexity	Physical Realism	Visual Realism	User Control
FFD, DFFD (e.g. Critter)	Low to Medium	Medium	Low	Medium to High	Medium
Metaballs (e.g. [Thalmann 96])	Medium	Low to Medium	Low	Medium to High	Medium to High

Table 2.2 Comparative analysis of multi-layered deformation models

Deformation Model	Computation	Complexity	Physical Realism	Visual Realism	User Control
Kinematic a) Ellipsoid b) Generalized Cylinder	(Very) Low	Low	Low	Medium to High	Medium
FEM	Very High	High	High	High	Low
Mass-spring systems (muscles)	Medium (2D) to High (3D)	Medium	Medium	Low to medium	Low (3D) to Medium (2D)

Table 2.2 Comparative analysis of multi-layered deformation models

Among multi-layered models, those based on FEM generally produce good-looking deformations. They are, however, quite slow and unwieldy. Because of the high computational cost associated to the FEM technique, most models simulate the deformation of only a few muscles, usually working in isolation. Most of them would probably become intractable if one tried to simulate the numerous muscles of the human body at the same time. In fact, only recently has a relatively complete model been demonstrated [Hirota 01]. It is important to note, however, that there exists a huge body of literature on FEM and that various acceleration techniques have been proposed, especially in the context of virtual surgery. For example, Bro-Nielsen and Cotin [Bro-Nielsen 96] apply condensation techniques to a volume finite element model and thus restrict the computations to the surface nodes while preserving the same behavior as for the original solid volumetric model. Another example is the work of Cotin *et al.* [Cotin 99a] in which a linear quasi-static deformation is computed in real-time as a linear combination of pre-computed deformations along three independent axes. Another considerable drawback of FEM models is their poor reusability. Transforming a skinny character into a muscular character for instance requires remeshing the model and re-assigning elastic and mass properties to the finite elements. Muscle models based on mass-spring networks run faster than FEM models but have not produced really convincing deformations thus far. On the contrary, geometric deformation models have shown promising results ([Scheepers 97] for example).

2.4.3 Desirable properties of a multi-layered model

Before anything else, we listed a certain number of constraints and desirable properties for a good multi-layered model:

- Highly detailed skin as it is often the only visible layer.
- Simulation time proportional to the importance of the anatomical layer. Roughly speaking, the simulation time should decrease as we get farther away from the skin.
- Important visual clues of the skin such as grain, color variation, veins, hair, beauty spots and age wrinkles should not be overlooked. They are, however, beyond the scope of this

research work. For simplicity, we will render them using adequate texture maps. Note that age wrinkles have been simulated in [Boissieux 00] and [Wu 97].

- The topology of the skin mesh must remain unchanged during deformation because this is desirable for operations like skin texturing and virtual clothes simulation.
- Scalability and reuse of internal components. A few parameters should suffice to change from a skinny character to a muscular character, or from a tall character to a short character.
- Surface self-intersection prevention. Contact surfaces between different body parts may form in the vicinity of joints.
- Simulation of fatty tissues. Most of the work to date on inner anatomical layers has been concerned with the musculature. Obviously, the importance of fatty tissues on the surface form has been underestimated.
- Bi-directional propagation of the deformation between layers. If the skin is displaced because of the application of external forces (e.g. virtual clothes pressing on the body), the deformation should be propagated to the fatty tissues and muscles.

Chapter 3

Modeling of the Skeleton

We begin this chapter by introducing the necessary anatomical notions concerning the human skeleton. Next, we describe the articulated structure that forms the backbone for animating our character. In the core of the chapter, we introduce four joint models that provide a reasonable approximation of the actual human joints in terms of motion and limits.

3.1 Osteology

The *skeleton* determines the general shape of the body. It is made up of *bones* connected by *joints* which allow relative motion between different parts of the skeleton. In this section, we take a closer look at these components.

3.1.1 Bones

The general framework of the body consists largely of a series of bones supplemented in certain regions by *cartilage*, a soft flexible tissue that lines the inner surface of joints and cushions them. There are 206 bones in an adult body (see Fig. 3.1). They not only support the body weight and enable movement, but also protect the internal organs and store vital nutrients. One can roughly divide the skeleton into two main parts. The central bones of skull, ribs, vertebral column and sternum form the axial skeleton while the bones of the arms and legs, along with the scapula, clavicle, and pelvis make up the appendicula skeleton.

Anatomists usually classify bones into four categories according to their shapes: long, flat, short, or irregular [Gray 00]. *Long bones* are found in the limbs and act very much as levers moved by the associated muscles. Regarding their shape, long bones have a shaft and expanded extremities called *epiphyses*. The shaft is usually not straight but slightly curved, the curve generally taking place in two planes. The epiphyses assist the purposes of articulation and afford broad surfaces for muscular attachment. The bones belonging to this class are: the clavicle, humerus, radius, ulna, femur, tibia, fibula, metacarpals, metatarsals, and phalanges. *Short bones* have no shaft and are found in parts where the skeleton is intended for strength and compactness with limited motion as in the carpus and tarsus. *Flat bones* are broad curved plates which provide large surfaces for muscular attachment and extensive protection of the inner organs. Examples of flat bones include the scapula, sternum and ribs. The remaining

bones may be classed as *irregular bones*: such are the bones of the face, those that form the vertebral column, and small bones of the hand and foot.

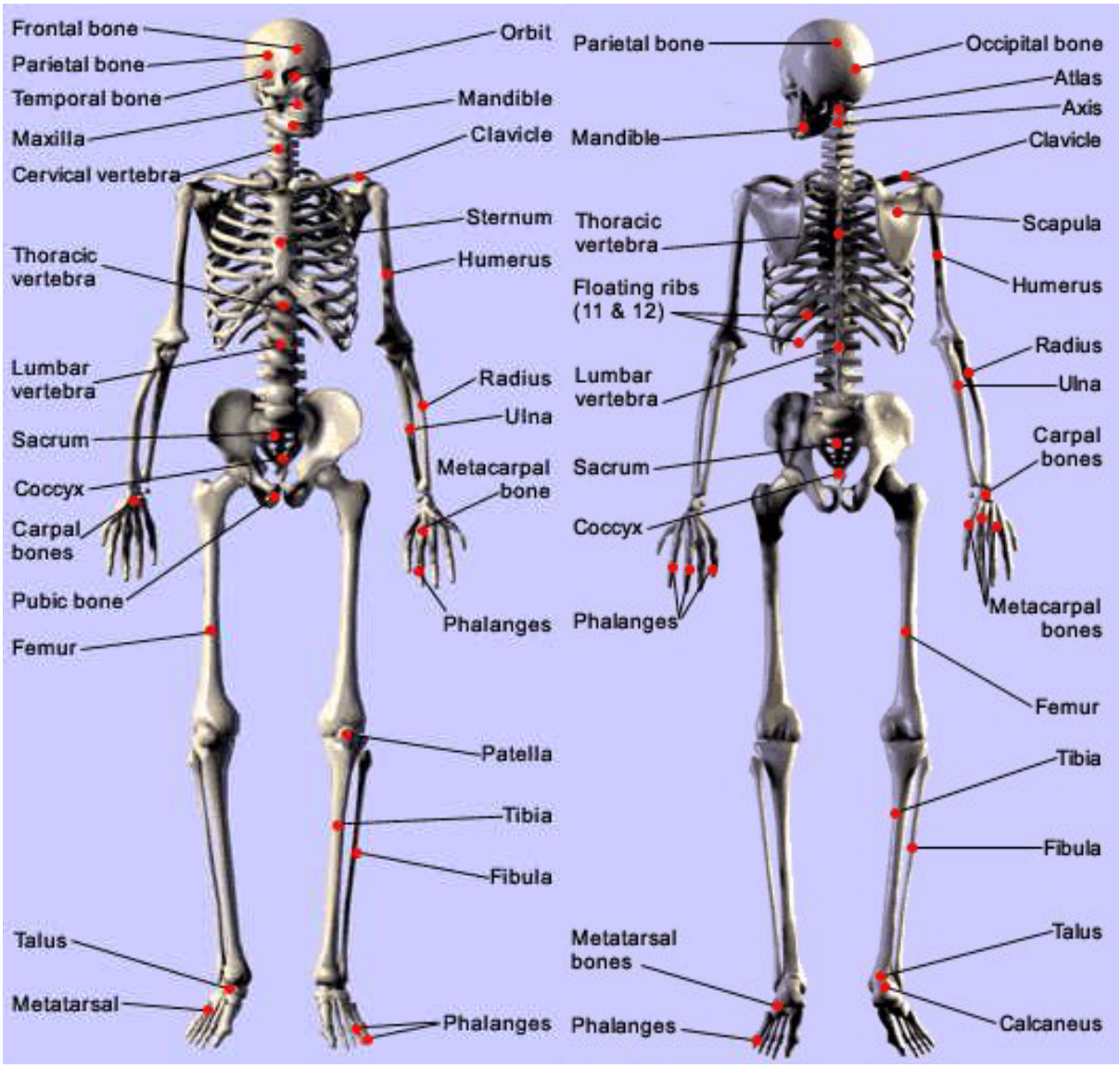


Figure 3.1 Anterior and posterior view of the human skeleton.

3.1.2 Joints

The location where two bony parts meet is called joint or *articulation*. Human joints are usually split into three categories by anatomists according to their range of motion: inmovable, slightly movable, or freely movable. Alternatively, joints are classified according to their structure. At any rate, both classifications are very similar due to the close relationship between the structure of a joint and the range of motion it allows. *Inmovable joints* (bony or fibrous joints) are typically found in the skull where adjacent margins of the bones are almost in contact, being separated merely by a thin layer of fibrous membrane. These joints allow no appreciable

motion. *Slightly movable joints* (cartilaginous joints) are situated in places where slight movement combined with great strength is required, typically in the vertebral column. Note that the movement between the individual bones may be small, but the amount of possible movement in the column as a whole is considerable because of the high number of joints in close proximity. In the *freely movable joints* (synovial joints) the surfaces of bones are completely separated and bound by strong fibrous bands called *ligaments*. The expansions of the bones that form the articulation are usually covered by a layer of cartilage. The range of motion of synovial joints is wide in comparison with other joint types. However, ligaments limit or check the movement of these articulations in many instances. For example, the very powerful iliofemoral ligament reinforced by the less powerful pubofemoral ligament limit the extension of the hip. Other soft tissues, especially muscles, may also restrict the joint mobility. Thus, the range of motion of the shoulder joint is restricted by the ligaments and muscles that span this articulation rather than by the shape of the gleno-humoral socket.

The above classification of joints is, however, not necessarily the most suitable one for computer graphics. For instance, inmovable joints are of little importance from our standpoint as we are only concerned with the forms produced by their union. In fact, it is more appropriate to classify joints according to the type of motion they allow. Six classes of joints can be distinguished [Scheepers 96] [Gray 00] (see Fig. 3.2):

Hinge Joints: A hinge joint is a monoaxial joint. The rotation takes place about an axis perpendicular to the long axis of the bones involved. This transverse axis allows flexion/extension motion, usually with a considerable extent. The direction which the distal bone takes in thismotion is seldom in the same plane as that of the axis of the proximal bone; there is usually a certain amount of deviation from the straight line during flexion. The interphalangeal joints and the elbow joint between the humerus and the ulna are examples of hinge joints. The knee and ankle joints are less typical, as they allow a slight degree of rotation or of side-to-side movement in certain positions of the limb.

Pivot Joints: A pivot joint is also a monoaxial joint but, in this form, the rotation takes place about a longitudinal axis. It supports angular movement around the long axis of a bone, allowing one bone to rotate in the ring of another bone. Both articulations of the radius with the ulna (proximal and distal) are pivot joints.

Ball-and-socket Joints: A ball-and-socket joint allows rotational movement in all planes. A bone with a rounded end (ball) fits in a cuplike cavity (socket) of another bone, hence the name ball-and-socket. The range of motion depends to a large extent on the depth of the socket; a shallower socket increases the range of possible motions, but the stability of the joint suffers. The shoulder is a shallow ball-and-socket joint. The hip is another ball-and-socket joint.

Ellipsoidal Joints: An ellipsoidal joint is a modified ball-and-socket joint, where an oval-shaped knob on one bone is received into an elliptical cavity on another bone. This design allows all movements allowed by ball-and-socket joints, except axial rotation. The wrist joint is an example of this form of articulation.

Saddle Joints: A saddle joint allows exactly the same types of motion as ellipsoidal joints but the structure of the articulating ends are different. Flexion, extension, adduction, abduction, and circumduction are therefore allowed; but no axial rotation. Each articulating end in a saddle joint has double curvature, and when fitted together, convex curvature in one meets concave curvature in the other. The carpometacarpal joint of the thumb is a saddle joint; it permits opposing the thumb against each of the other fingers.

Gliding Joints: A gliding joint admits of only moderate gliding movement; it is formed by the apposition of plane surfaces, or one slightly concave, the other slightly convex, the amount of motion between them being limited by the ligaments or osseous processes surrounding the articulation. It is the form present in the joints between the articular processes of the vertebræ, in most carpal joints, and in most tarsal joints.

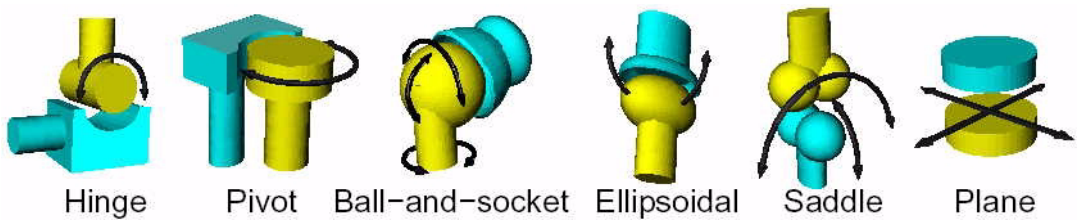


Figure 3.2 Classification of movable joints (extracted from [Scheepers 96]).

3.2 An Articulated Structure Based on the H-Anim Standard

As in virtually all character animation systems, the motion of our character is entirely controlled by a hierarchical articulated structure. We rely on the H-Anim standard as the basis for the articulated structure.

The H-Anim specification [HAnim 98] is the result of a joint effort in an attempt to standardize the representation and animation of virtual humans. The specification as of the 1.1. version is based on the Virtual Reality Modeling Language (VRML) and is therefore oriented towards Web applications. However, the standard can be used by any piece of software provided that the VRML file containing the human body definition can be parsed and correctly interpreted. One of the great benefits of using an H-Anim compliant skeleton lies in the ability to reuse any animation produced for this standard.

An H-Anim file contains a list of joints arranged in a hierarchical fashion, a list of segments also arranged hierarchically and gathering all the information concerning the representation of body parts, and finally a list of specific landmarks on the body that serve primarily for animation purposes e.g. as end-effectors for Inverse Kinematics. The specification also suggests four different levels of animation in order to address the needs of a wide range of applications. We settle on the highest level of animation, in which most movable joints of the human anatomy are included. Finally, the H-Anim standard allows the appendage of joints to the otherwise

fixed hierarchy. In other words, new joints can be added to the hierarchical structure, yet only as leaves.

We add radioulnar joints to the H-Anim hierarchy in order to solve the common problem of anatomically accurate pronation-supination of the forearm. By procedurally linking the rotation of the radioulnar joint with the twisting motion of the elbow, anatomically correct pronation and supination poses are achieved (see Fig. 3.3). This ad-hoc scheme is similar to the one employed by Scheepers *et al.* [Scheepers 96]. It only differs in that a radioulnar joint in our approach is a sibling joint of the wrist instead of being its parents due to the restriction imposed by the H-Anim standard.

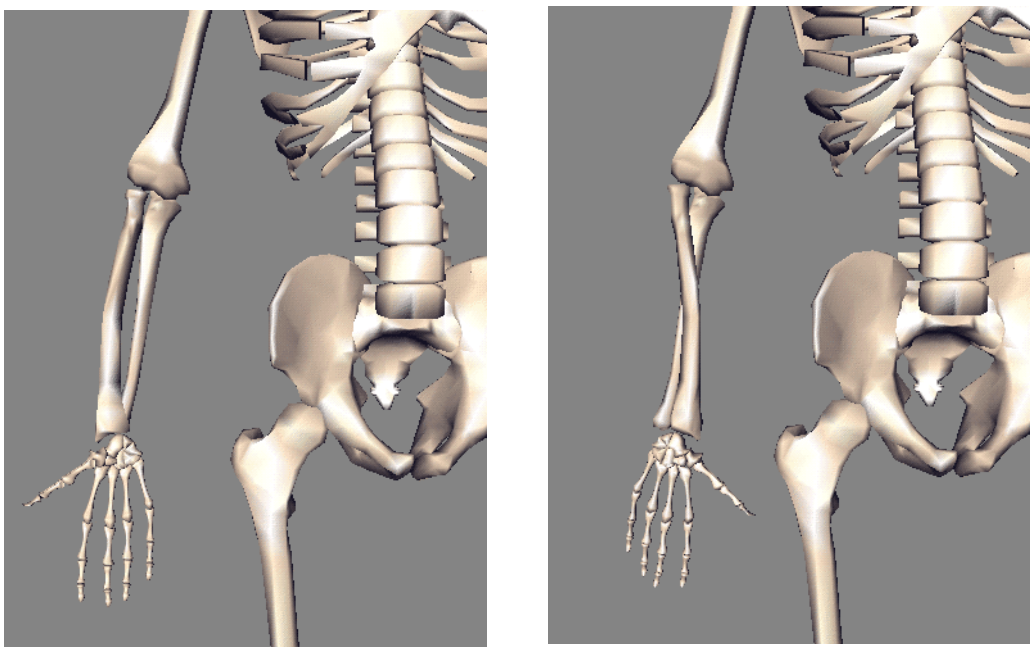


Figure 3.3 Anatomically accurate supination and pronation poses.

3.3 Joint Models

Joint modeling is a complex task that involves several issues. First, one has to estimate the precise location of the joint center as well as the axes of rotation (or translation). This is further complicated by the fact that these axes may be moving. For example, the geometric location of the axis of the knee joint moves slightly while the thigh and the lower leg rotate about it [Kroemer 90]. Second, a minimal yet pertinent set of parameters has to be chosen to describe as faithfully as possible the kind of motion a joint allows. This amounts to deciding on the number of degrees of freedom, and then finding the best suited parametrization. Last but not least, reasonable joint limits have to be determined, specified, and eventually enforced. This issue is further complicated by the coupling of limits within a joint (intra-joint coupling) and by the coupling of joint limits between different joints (inter-joint coupling).

3.3.1 Degrees of freedom

A *degree of freedom* (DOF) corresponds to one of the three non-collinear axes around which a rotation may take place, or along which a translation may occur. A joint in a 3D articulated structure has up to six degrees of freedom, half of which are intended for rotation, the other half for translation. The total number of DOFs of the articulated structure is the sum of the DOFs of each joint. Note that the H-Anim specification leaves it up to the application to decide on the number of DOFs for every joint.

3.3.2 Euler angles

Although various joint parameterizations have been put forward by the graphics community over the last two decades, surprisingly few models have found their way into commercial or academic human animation packages. These packages [Alias 01] [Boulic 91] continue to rely almost exclusively on Euler angles for parameterizing rotational joints. However, the wide adoption of Euler angles by the graphics community is historically rather than rationally determined [Shoemake 85]. In fact, we feel Euler angles are the worst possible parameterization. We briefly recall hereafter its main flaws. More details can be found in [Dam 98], [Grassia 98] and [Shoemake 85].

Euler angles represent a general rotation as successive rotations about three basis axes. The order of rotation axes is important, i.e. different orders produce different rotations. For this reason, a fixed convention is usually adopted, e.g. a ***zyx*** order. Unfortunately, different communities have chosen different angle schemes, so a certain confusion reigns.

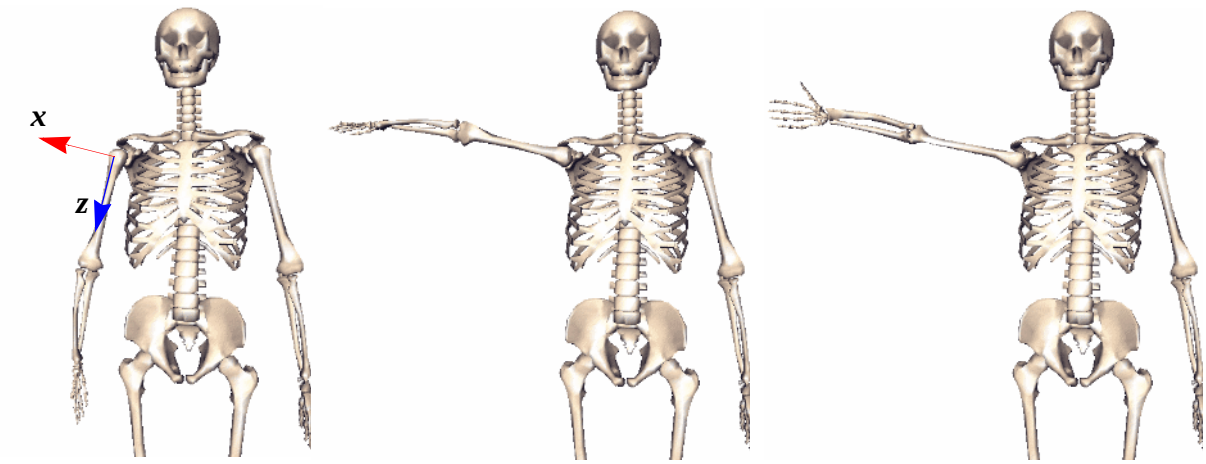


Figure 3.4 The right shoulder joint undergoes the gimbal lock. Euler angles are set as follows. Left = $[0\ 0\ 0]$; Middle = $[0\ \pi/2\ 0]$; Right = $[\pi/3\ \pi/2\ 0]$ **and** $[0\ \pi/2\ \pi/3]$.

Another well-known problem is the occurrence of “gimbal lock” [Watt 92]. Gimbal lock is encountered when the second axis of rotation becomes aligned with one of the two remaining axes, which leads to the loss of one degree of freedom. The shoulder joint parameterized by

three Euler angles provides a pertinent example. Let the start posture (i.e., when the three Euler angles equal zero) be such that the arm is lying by the side of the body (left picture of Fig. 3.4) and let us assume we use a ***xyz*** angle scheme [Shoemake 94] where a rotation around ***x*** flexes or extends the arm, a rotation around ***y*** yields an abduction/adduction motion, and a rotation around ***z*** twists the arm. In such a case, the articulation locks up when the shoulder is abducted or adducted by $\pi/2$ (middle picture of Fig. 3.4). Flexion is then "lost" and becomes equivalent to an axial rotation (right picture of Fig. 3.4).

We now introduce our four joint models.

3.3.3 Revolute joint (one DOF)

The revolute joint allows a rotation about an arbitrarily fixed axis. The natural parameterization of the joint is the angle of rotation θ with respect to a reference configuration. Limits are enforced by confining the angle of rotation to a restricted range. Hinge and pivot joints, which have one rotational DOF, are both modeled by revolute joints in our system.

In the H-Anim standard [HAnim 98], joint axes are necessarily aligned with the axes of the global (or world) coordinate system in the default position of the humanoid. We introduce the notion of *anatomical frame*, common to the four joint models, in order to be able to specify a more meaningful joint orientation, in which the rotation axes and limits are more easily expressed. The corresponding anatomical matrix M_{ana} describes a rotation that transforms the joint frame into the world frame in the reference pose. We arbitrarily choose the rotation axis of a revolute joint to be the ***z*** axis of the local frame. Hence, the local transformation of a revolute joint is: $R_{\mathbf{z}}(\theta)$. This gives the following chain of transforms:

$$R_{\mathbf{z}}(\theta) \equiv {M_{ana}}^{-1} \times R \tag{3.1}$$

$$\theta \rightarrow \theta_{clamped} \in \left[\theta_{min} \ \cdots \ \theta_{max}\right] \tag{3.2}$$

$$R_z(\theta_{clamped}) \times M_{ana} \tag{3.3}$$

Eq. (3.1) transforms any rotation R into the local coordinate system of the revolute joint. The joint parameter θ is then extracted from the resulting transformation. This step can be seen as the application of "hard" limits: the incoming rotation about any possible axis is restricted to a rotation about a specific axis, in our case, the ***z*** axis of the local frame. Eq. (3.2) clamps the parameter θ to an authorized range. The rotation is ultimately transformed back into the original coordinate system by Eq. (3.3).

3.3.4 Knee joint (two DOFs)

From an anatomical point of view, two types of motion can be distinguished for the knee: flexion/extension on the one hand and axial rotation of the outgoing segment on the other hand. There exists, however, a coupling between these motions, which becomes conspicuous as the leg approaches its limit in extension. For the last twenty degrees of extension or so (known as the terminal motion), an involuntary axial rotation occurs [Maciel 02].

Our knee joint model performs two successive rotations about two orthogonal axes. However, the knee joint is not equivalent to two independent revolute joints located at the same point because the twist limits are functionally coupled with the extension. In practice, the range of axial motion becomes more and more restricted as the leg approaches full extension.

3.3.5 Swing joint (two DOFs)

The swing joint allows all rotational motions except axial rotation. Both saddle and ellipsoidal joints are logically modeled by a swing joint in our system. In terms of parameterization and limits, a swing joint is strictly equivalent to a ball-and-socket joint whose axial motion is restricted altogether.

3.3.6 Ball-and-socket joint (three DOFs)

The ball-and-socket joint has three rotational degrees of freedom. It allows an axial motion (or *twist*) about the limb axis (one DOF), as well as a *spherical* motion (or *swing*) that determines its direction (two DOFs). We arbitrarily choose the direction of the segment (i.e., limb) to be aligned with the ***z*** axis of the local frame.

There exist many different ways of parameterizing an orientation in 3D space: rotation matrices, Euler angles, unit quaternions [Shoemake 85], axis-angle or exponential map [Grassia 98], swing-twist [Korein 85]. In his early work on the investigation of reachable space [Korein 85], Korein already advocated the swing-and-twist parameterization for three-DOF joints like the human shoulder and hip because it eases the specification and enforcement of limits. Grassia [Grassia 98] points moreover out that the unique singularity of the swing-twist parameterization is (or more exactly *can be*) located far beyond the natural limits of the shoulder. Baerlocher also discusses the occurrence of singularities for different parameterizations including swing-twist [Baerlocher 01] and addresses the problem of clamping illegal orientations to joint boundaries.

Based on the work of Baerlocher [Baerlocher 01], we too rely on a swing-and-twist parameterization for ball-and-socket joints, for it strikes a good balance between simplicity and accuracy. The rotation R of a ball-and-socket joint is decomposed into a swing and a twist as follows:

$$R = R^{twist} R^{swing}$$

$$\text{where } R^{twist} = R_{\mathbf{z}}(\theta) \text{ and } R^{swing} = \begin{bmatrix} s_x & s_y & 0 \end{bmatrix}^T \tag{3.4}$$

The swing motion is performed by a rotation parameterized by the axis-angle of Eq. (3.4) Note that the rotation axis for the swing always lies in the ***x*-*y*** plane perpendicular to the major axis of the limb. A formula for converting a quaternion into swing and twist components can be found in [Baerlocher 01]. The axial rotation, which succeeds the swing motion, occurs around the (arbitrarily chosen) ***z*** axis of the local frame.

The one singularity of the axis-angle swing parameterization is located on direction:

$$\boldsymbol{d} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T, \text{ where } s_x^2 + s_y^2 = \pi^2 \tag{3.5}$$

Eq. (3.5) tells us that if we are to avoid the singularity, all we need to do is pick a canonical (i.e., $s_x = s_y = 0$) position for the joint so that the limb may never point in the same direction as the $-\boldsymbol{z}$ vector under normal physiological conditions. For the shoulder joint, we conveniently choose an abduction of $\pi/2$ with respect to the H-Anim default pose [HAnim 98]:

$$M_{offset}^{shoulder} = R_{\boldsymbol{z}}\left(-\frac{\pi}{2}\right)$$

The resulting canonical posture resembles the middle picture of Fig. 3.4. The final chain of transformations for a ball-and-socket joint is:

$$R^{twist} R^{swing} = M_{ana}^{-1} M_{offset}^{-1} R \tag{3.6}$$

$$R^{twist} = R_{\boldsymbol{z}}(\theta) \rightarrow R_{\boldsymbol{z}}(\theta_{clamped}) \text{ and } R^{swing} = \begin{bmatrix} s_x & s_y & 0 \end{bmatrix}^T \rightarrow \begin{bmatrix} s_x & s_y & 0 \end{bmatrix}_{clamped}^T \tag{3.7}$$

$$R_{\boldsymbol{z}}(\theta_{clamped}) R_{clamped}^{swing} M_{offset} M_{ana} \tag{3.8}$$

Eq. (3.6) transforms an incoming rotation R into the joint coordinate system. This is followed by a decomposition into twist and swing parameters, which are clamped to admissible values in Eq. (3.7). The joint's local transformation is then transformed back into the original coordinate system by Eq. (3.8).

3.3.7 Comparison of swing-twist with other parameterizations

Grassia discusses in [Grassia 98] the suitability of various rotation parameterizations with respect to different applications and concludes that there is no ideal parameterization. He also emphasizes that no single parameterization is free of singularity in R^3. By contrast, unit quaternions and rotation matrices avoid singularities (i.e., they are safe from gimbal lock) since they work in a different space.

Rotation matrices form a group under matrix multiplication known as $SO(3)$. Mathematically, $SO(3)$ is characterized by matrices whose columns (or rows) are mutually orthogonal unit magnitude vectors, and whose determinant equals one. This makes $SO(3)$ an ill choice as six non-linear constraints must be enforced to keep the matrix orthonormal. Unit quaternions lie on the unit hypersphere S^3 embedded in the four-dimensional Euclidean space R^4. Obviously, placing joint limits directly in this unit quaternion space is difficult. Lee showed how to specify conic, axial and spherical limits [Lee 00]. Yet, imposing more meaningful limits is challenging.

On the contrary, accurate limits such as spherical polygons are easily expressed with the swing-twist parameterization and can quickly be enforced [Baerlocher 01]. Equally important, the swing-twist decomposition gracefully avoids the problem of gimbal lock because its singu-

larity is located well beyond the natural limits of the shoulder (or hip) joint for a well-chosen canonical position. It is important to note that the amount of twist given by the parameterization is not absolute. It is actually defined with respect to a neutral or zero twist posture given by the swing motion. It therefore is a good idea to select the canonical posture (i.e., $s_x = s_y = 0$) as the approximate center of the shoulder joint sinus cone in order to avoid large amounts of induced twist.

Besides limits, another typical application consists in retrieving the motion of a limb over time, and more specifically, the axial rotation and swing motion. Using Euler angles for this task is common albeit inappropriate. To support this assertion, let us examine the axial rotation of the upper arm during a captured tennis serve motion. Fig. 3.5 shows the evolution of the shoulder twist during the movement when three Euler angles parameterize the shoulder joint. As can be seen, a sudden, steep drop occurs after about six seconds. This is further confirmed by the plot of the shoulder flexion, in which an abrupt rise is located at the same point in time. Checked visually, the posture of the shoulder joint turns out to be close to the singularity during these nearly-discontinuous angle variations. The problem basically is that, when the second Euler angle comes close to the singularity, the other two Euler angles start to vary wildly. In the neighborhood of the singularity, a small change of orientation results in steep changes of two Euler angles, hence the sudden drop in Fig. 3.5. Contrariwise, the rate of change of swing-twist parameters remains small within the whole human range of motion because the joint posture always remains far from the singularity.

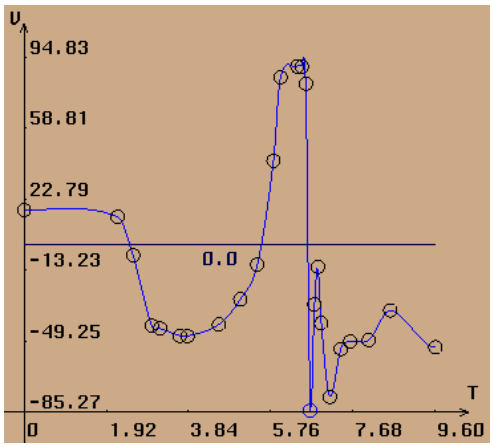


Figure 3.5 Plot of the shoulder twist angle in degrees with respect to time during a tennis serve motion.

The last point of importance about the swing-twist parameterization is that the decomposition of an orientation into swing and twist components is unique[1]. This is ensured by arbitrarily constraining the unit quaternion $\boldsymbol{q} = ((q_x, q_y, q_z), q_w)$ from which the swing and twist components are extracted to the positive hemisphere, i.e. $\boldsymbol{q}$ is replaced by $-\boldsymbol{q}$ when $q_w < 0$. This is allowed because two antipodal points in S^3 encode the same rotation. Since the twist is given by $\tau = 2\,\mathrm{atan2}(q_z, q_w)$ [Baerlocher 01] where $\mathrm{atan2}(y, x)$ returns the polar angle of point $\boldsymbol{P}(x, y)$ in the range $[-\pi, \pi]$ the possible values of τ are thus restricted to $[-\pi, \pi]$ instead of $[-2\pi, 2\pi]$. Therefore, the unicity of the decomposition is guaranteed except at the singularity where $q_z = q_w = 0$.

In summary, the swing-and-twist decomposition is a natural parameterization which decouples the axial rotation of the limb and its movement in space. Additionally, limits are intuitively expressed and rapidly enforced with this parameterization. Further, we shall see in the

1. This evidently does not hold at the singularity.

next chapters that it is also appropriate for other applications including skin deformation. However, it must be stressed that the swing-and-twist parameterization is not a panacea. Depending on the intended application, other parameterizations can be more suitable.

3.3.8 Determination of joint limits

Imposing meaningful limits on the articulations turns out to be of paramount importance if one wants to avoid unrealistic muscle and skin deformations resulting from unreachable skeleton stances. The determination of joint limits has been extensively investigated in several disciplines ranging from biomechanics, to ergonomics, to anthropometry, and to computer graphics.

3.3.9 Limits of shoulder ball-and-socket joint

The shoulder joint has received considerable attention from researchers. Engin *et al.* [Engin 89] measured the directional limits of the arm and established a first shoulder kinematic data base. Based on the data from Engin and coworkers, Maurel and Thalmann [Maurel 00] explicitly model directional limits with joint sinus cones while Baerlocher [Baerlocher 01] restricts the reachable space of the upper arm using spherical polygons. An illegal direction of the upper arm can be corrected in both methods by projection in 2D [Maurel 00] or in 3D [Baerlocher 01].

Wang and Maurin [Wang 98] complemented Engin's study by finding the axial motion range for a number of definite positions of the upper arm. They mechanically measured the maximal inward and outward twists for arm directions spaced 20 rotational degrees apart in the horizontal and vertical plane. The collected data are assembled into two surfaces (inward and outward) using a regression fitting method with an orthogonal homogeneous polynomial basis. The results show that the axial motion range depends strongly on the position of the upper arm in the shoulder sinus cone and varies on average from 94 to 157 degrees. They also stress that the individual variability is low (seven people in the experiment). Their method is, however, fairly inaccurate due to the low number of samples, the imprecision of the mechanical measurement device and the use of a simple regression method.

Herda *et al.* [Herda 01] astutely merge directional and axial limits into a single unified representation. They measure the motion range of the shoulder complex using an accurate optical motion capture. The recorded orientations are converted into quaternion space and a closed, continuous implicit surface based on metaballs is fitted to the resulting cloud of points. Thus, the implicit surface in quaternion space represents the complete space of valid orientations. The method is powerful because an orientation can be checked against limits very quickly using a simple inside/outside test. Furthermore, an invalid orientation can be clamped to a close valid one (the closest in quaternion space) by a simple gradient-directed projection onto the implicit surface. However, the method captures the limits of the entire shoulder complex and is therefore suitable only if one approximates the region by a single joint.

In our approach, directional limits are enforced through the use of spherical polygons [Baerlocher 01] [Korein 85]. The spherical polygon of the shoulder (see Fig. 3.6) was designed after

the data collected by Engin and colleagues [Engin 89].

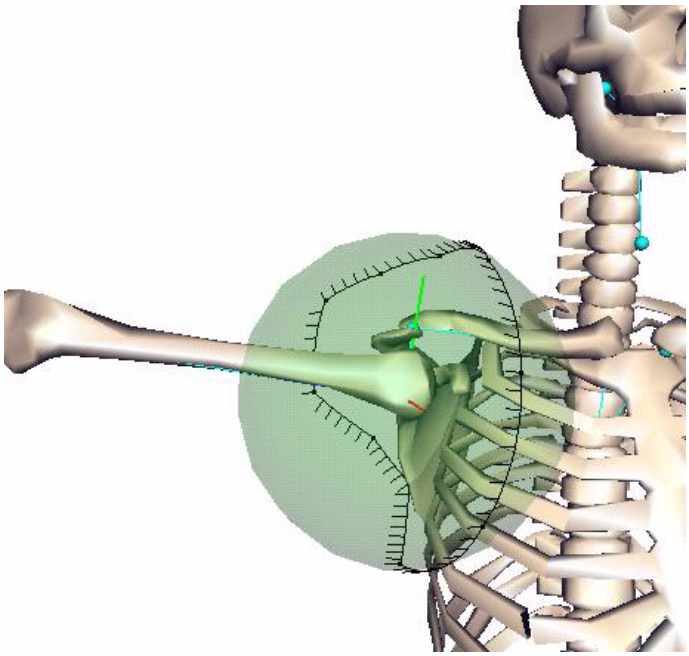


Figure 3.6 Directional limits of the shoulder as a spherical polygon.

The axial rotation of the upper arm is limited by two height fields in the ***x*-*y*** swing plane defined by Eq. (3.4). One height field is positive and delimits the possible outward twist while the other one is negative and describes the maximal inward twist. Each height field is modeled by specifying key values in the swing axis-angle plane using the visual feedback provided by the bones. During animation, a precise twist limit is readily computed for any given swing by interpolation of the *k*-closest keys. For a swing vector $\boldsymbol{s}$, we compute the weight of the *i*-th key as follows:

$$w_i(\boldsymbol{s}) = \frac{1}{\|\boldsymbol{s} - \boldsymbol{s_i}\|} - \frac{1}{\|\boldsymbol{s} - \boldsymbol{s_k}\|}, \; 0 < i < k$$

where *k* is the *k*-closest key in the swing axis-angle plane and $\boldsymbol{s_k}$ is the corresponding swing vector. The weights are then normalized so that they add up to one:

$$\tilde{w}_i(\boldsymbol{s}) = w_i(\boldsymbol{s}) \Big/ \left(\sum_{j=1}^{k} w_j(\boldsymbol{s}) \right)$$

This yields weights that vary smoothly between key values while interpolating the user-specified keys as shown in Fig. 3.7.

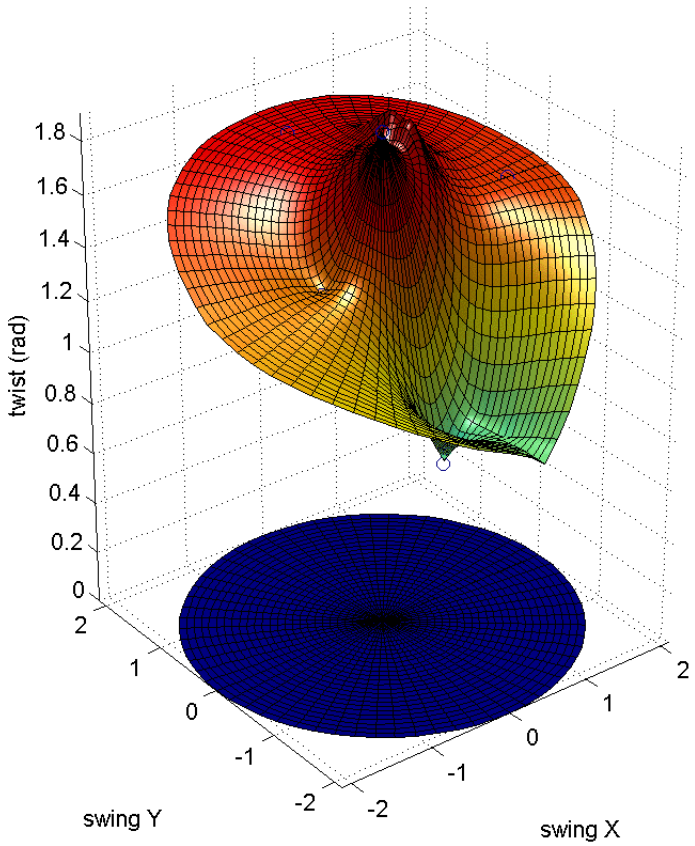


Figure 3.7 The twist height field is computed by *k*-closest interpolation of *n* key values (blue circles). Here, we model the positive outward twist of the upper arm using $k = 4$ and $n = 5$. The swing values are restricted to an "average" reachable space, i.e $\|s\| < 0.65\pi$.

3.3.10 Chosen joint types

The joints of the fingers (except for the base of the thumb) and toes, as well as the elbow and radioulnar joints are modeled by revolute joints in our approach. The shoulder and hip joints are modeled by ball-and-socket joints. The joints of the spinal cord are either swing or ball-and-socket joints. Most of them are assigned quite short a range of motion. Finally, the wrist joint and the carpometacarpal joint of the thumb are approximated by swing joints while the ankle joint is set to the ball-and-socket type.

3.4 Bones

Representing bones is important for three reasons when choosing an atomically-based approach to human body modeling. First and foremost, bones influence the skin shape in many areas of the body. Besides, while some bony edges directly sculpt the shape of the overlying skin at all times, others may do so only in certain postures. For example, the clavicle is clearly defined when the arms are lying by the side but contributes less to the surface form when the shoulder is abducted. Second, bones give important visual clues for the subsequent attachment

of tendons and muscles to the skeletal system. As a consequence, inaccurate modeling of the skeleton inevitably leads to flaws in the muscle layer. It is thus necessary to pay special attention to the representation of bones, even for those that do not directly sculpt the skin. Last, bones provide visual feedback when checking against joint limits. In particular, while it is difficult to estimate the axial rotation of limbs using only a stick figure of the skeleton, the process becomes much easier when bones are explicitly represented.

3.4.1 Representation

Up to now, research teams have approximated bones by polygonal meshes [Porcher 98] [Scheepers 96] and by a combination of simple geometric primitives like ellipsoids and cylinders [Scheepers 96] [Thalmann 96] [Turner 93] [Wilhelms 97a]. Both representations possess specific advantages. Volumetric representations such as ellipsoids possess an analytic formulation that lends itself well to the fast and exact computation of intersections with other simple geometric objects. Also, they naturally integrate the notion of inside/outside. Polygonal meshes possess the distinctive advantage of being more readily available and supported by virtually all modeling softwares. We represent the skeleton with 94 polygonal meshes. The bones of the hands and head are moderately detailed whereas the bones of the foot are roughly approximated by a single mesh.

3.4.2 Motion

Every bone is simply anchored to a specific joint of the underlying articulated structure. This is a reasonable approach because bones can be considered rigid objects for the most part.

3.5 Conclusion

In this chapter, we have shown how to model the different components of the human skeleton. We have proposed four joint models that reasonably approximate the real human joints. For complex joints with three rotational degrees of freedom, we have seen that the swing-and-twist decomposition provides a suitable, intuitive parameterization that has a one-to-one mapping to rotation matrices and easily avoids singularities when restricted to the human range of motion. We have also exposed a simple way to specify coupled limits of ball-and-socket joints. Additionally, our model allows to play back any keyframe animation based on the H-Anim standard.

Chapter 4

Modeling of the Musculature

We open this chapter by first exposing general anatomical considerations regarding the human musculature. In the second section, we focus more specifically on the muscles of the upper body and detail all the major muscles in terms of shapes and function. We then briefly introduce our geometric model that can capture most muscle shapes while the core of the deformation method is exposed in the Section 4.4. The two following sections address practical issues. We close the chapter on the derivation of a real-time version of the deformation model.

4.1 Myology

While bones form the general framework of the body, muscles refine the general shape of the surface form. As a matter of fact, muscles account for nearly half of the total mass of the male adult body and fill in almost completely the gap between the skeleton and the skin [Richer 81].

Our bodily needs demand that muscles accomplish different chores, so we are equipped with three types of muscles: cardiac, smooth and skeletal muscles. All kinds of muscles, albeit functionally different, exhibit the same fundamental constitutive and mechanical properties [Maurel 98]. *Cardiac muscles*, found only in the heart, power the action that pumps blood throughout the body. *Smooth muscles* surround or are part of the internal organs. They are found in the stomach, bladder, and blood vessels. Both cardiac and smooth muscles are called involuntary muscles, because they cannot be consciously controlled. *Skeletal muscles* on the other hand carry out voluntary movements. Skeletal muscles are attached to bones by connective tissues. In this way, they make us capable of a variety of actions by simply contracting and becoming shorter, pulling the bones they are attached to towards each other.

Of all kinds of muscles, artists need only be concerned with skeletal muscles because the other types do not create or influence surface form. Hence, when speaking of muscles in the rest of this document, we shall implicitly refer to skeletal muscles unless otherwise mentioned.

4.1.1 Skeletal muscles

The principal function of skeletal muscles is to move the limbs, trunk, head, respiratory appa-

ratus, and eyes. They are the body's most abundant tissue, comprising about 23% of a woman's body weight and about 40% of a man's body weight. Each skeletal muscle is served by nerves which link the muscle to the brain and spinal cord. Commands initiated in the brain are thus transmitted along nerves and muscles generate forces as a response. When the nervous system properly coordinates the activation of many muscles, the result is smooth, purposeful movement.

Structurally, skeletal muscles consist of a red, striated, fleshy, contractile, central part named *belly* and of white, glistening, stiff, fibrous bands at the extremities called *tendons.* The tendons connect the belly to the bones. Being nearly devoid of elasticity, they act very much as force transmitters. The belly is the contractile part of the muscle and produces the force necessary to move the skeleton. The attachment of the muscle to the more stationary bone is called the *origin* while the other end is called the *insertion*. Generally, the origin (resp. insertion) is situated on the proximal (resp. distal) bone. When the area of attachment is very large, a tendon can be replaced by an *aponeuresis*, which is a sheet-like fibrous membrane of a pearly white color, resembling a flattened tendon.

There are approximately 600 skeletal muscles in the human body (see Fig. 4.4). They differ greatly in size. Some muscles are very large, such as the *gastrocnemius*, the major muscle forming the calf in the lower leg. Others are very small, such as the muscles of the eyelid. Skeletal muscles also vary extremely in their form. Some muscles are triangular (e.g. *deltoid*) while others are rectangular (e.g. *rectus abdominis*) or trapezoidal (e.g. *trapezius*). In general, long fusiform muscles (e.g. *biceps brachii*) are found mainly in the limbs while short muscles appear around joints (e.g. *brachialis*) and large, flattened muscles cover the back (e.g. *latissimus dorsi*). Similarly, tendons have different shapes and sizes in different muscles. They are sometimes round, sometimes flattened. As a result of evolution, tendons tend to be longer in the lower part of the limbs, thus shifting the weight away from the hand and foot. This reduction of weight in our limbs ends and therefore of inertia effects allows us to better control their motion.

The biological diversity is also great concerning the number of tendinous extremities and heads in the muscle. Muscles can be bicipital (e.g. *biceps brachii*), tricipital (e.g. *triceps brachii*), etc. In the leg, four heads make up the belly of the *quadriceps* muscle. As for tendons, they can be completely missing and the belly can attach directly to the bone.

4.1.2 Types of contraction

Anatomists distinguish between two types of contraction: Isometric (same length) and isotonic (same tonicity) contraction. Upon *isotonic contraction*, the belly changes shape, often bulging, while the total length of the muscle diminishes so that the bones to which the muscle is attached are pulled towards each other. The shortening of the muscle during an isotonic contraction seldom exceeds a third of the initial rest length under normal physiological conditions [Richer 81]. Upon *isometric contraction*, the shape of the belly also alters because of the tension in the muscle but the length of the muscle does not change, so no skeletal motion is pro-

duced. The performance of most exercises typically involves a combination of isotonic and isometric contractions.

4.1.3 Muscle structure

Tendons are made up of loosely packed bundles of collagen, organized along side each other. The collagen fibers are arranged so that they run parallel to the “pull” of the muscle when it contracts. The belly is composed of bundles of elastic fibers named *fascicles*. Fascicles are prismatic in shape, of varying size, and are for the most part placed parallel to one another, though they have a tendency to converge toward their tendinous attachments. Each fascicle is enclosed and connected to other fascicles by a delicate web of fibrous tissue named *perimysium* The perimysium, fascicles, and blood vessels are in turn covered by a sheath of dense connective tissue, known as the *epimysium*, which invests the entire muscle.

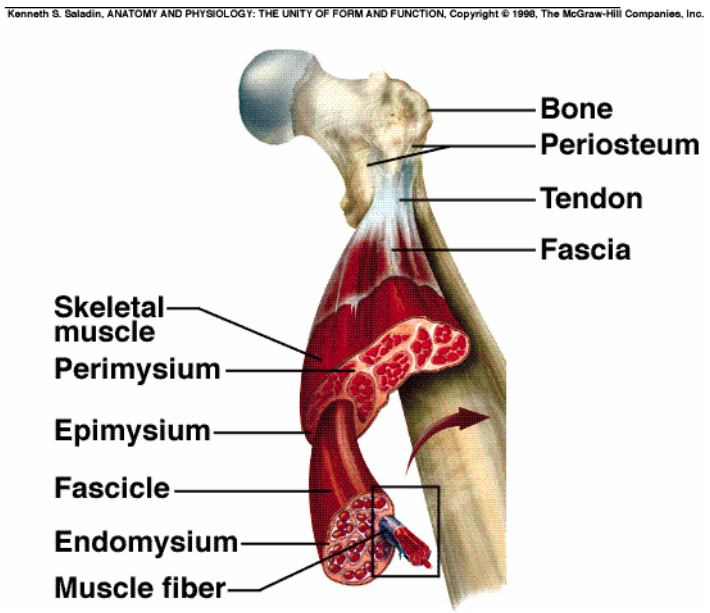


Figure 4.1 General architecture of a skeletal muscle [Saladin 98].

At a finer level (see Fig. 4.2), the muscle fibers that make up a fascicle run parallel with one another and are held together by a delicate connective tissue, the *endomysium*., similar to the perimysium but more fluid and gelatinous. The strength with which a muscle contracts is partly determined by the number of muscle fibers.

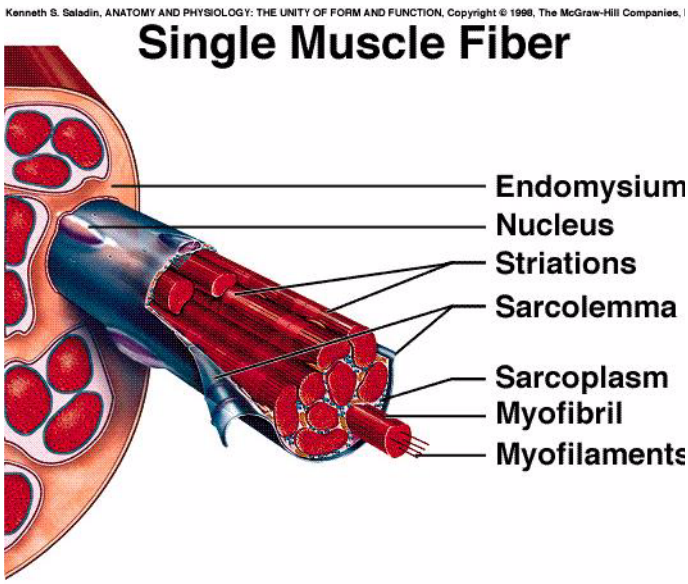


Figure 4.2 Microstructure of a muscle fiber [Saladin 98].

At the microscopic level, the contractile apparatus of each muscle fiber is subdivided into *myofibrils*, longitudinally oriented bundles of thick and thin filaments. These millions of tiny protein filaments work together to produce the motion in the body by contracting. As the filaments can only become shorter and not longer, muscles can pull but cannot push.

The connective tissues, that is the endomysium, perimysium, and epimysium, form a sheath of fibrous tissue known as the *deep fascia* that holds the muscle together. However, it must be stressed that the deep fascia also binds down collectively the muscles in each region. The deep fascia being naturally tensed, it applies pressure onto the surface of the covered muscles, thus increasing the power of the muscles.

4.1.4 Mechanical properties

All muscle tissues exhibit two fundamental properties:

- They can contract and shorten in length (because the myofilaments can slide across each other).
- After contraction, they relax and return to their former length.

More specifically, muscle tissue is characterized as nonlinear, anisotropic, and viscoelastic, much like other soft tissues. A viscoelastic material is, quite simply, both *elastic* and *viscous*. The elasticity characterizes the internal forces (*stress*) resulting from a given geometrical deformation (*strain*) while the viscosity includes the internal forces resulting from a given deformation speed. Muscle tissue is also *anisotropic* because it has properties, such as elasticity, that depend on the direction considered. Finally, it is *nonlinear* because the correlation between the force applied onto the material and the resulting deformation cannot be approximated by a straight line. In biomechanics parlance, the stress-strain relationship is said to be nonlinear.

As for tendons, due to the arrangement of collagen fibers in nearly parallel bundles, they are highly flexible but have tremendous resistance to tension. As a consequence, they are often considered to be devoid of elasticity. From a more quantitative point of view, muscle tissues are several orders of magnitude more elastic than tendons [Fung 81].

4.1.5 Pennation

There is considerable variation in the arrangement of the muscle fibers with reference to the tendons to which they are attached, even though the fibers always run parallel with one another between the origin and insertion of the muscle. This layout is named *pennation* [Zajac 89] and has considerable influence over the shape of the belly upon contraction. For example, if the muscle fibers run parallel to the long axis of the muscle, contraction causes a bulging. Yet, if the fibers form a distinct angle to the long axis of the muscle, contraction does not result in notable bulging but rather in parallel sliding of the top aponeurosis with respect to the bottom one [Lemos 01].

Four main classes of fiber arrangements can be distinguished in muscles: parallel, convergent, pennate and circular (see Fig. 4.3). Most muscles, especially those in the limbs, belong to the first category. This includes the fusiform muscle that slightly tapers near the tendons, so the fibers do not run parallel *sensu stricto*, but tend to converge towards a thinner extremity. The *biceps brachii* in the upper arm is a good example of fusiform muscle. Convergent muscles

Figure 4.3 Different arrangements of muscle fibers: parallel, (bi)pennate and convergent.

cover a broader area, but come together at one attachment point. The *pectoralis major*, the large muscle in the chest, is a typical convergent muscle; It originates from the sternal and clavicular region thus covering a wide area, and thins into a narrow twisted bundle at its insertion on the humerus. In a pennate muscle, the fibers appear oriented at a common angle to the tendons known as the *pennation angle*. When there are two pennation patterns, the muscle is said to be bipennate, as in the middle picture of Fig. 4.3. The *extensor digitorum* in the forearm and the *gastrocnemius* in the calf are (uni)pennate muscles, the *rectus femoris* in the thigh is a bipennate muscle, and the *deltoidus* in the shoulder region is a multipennate muscle. Lastly, circular muscles surround and are able to contract or close an opening of the body. The obicular muscles of the eye and mouth are examples of this variety of fiber arrangement.

4.1.6 Muscle groups

Any given movement requires the simultaneous action of many muscles. The muscle that causes a movement is called the *agonist* while the opposing muscle that stretches is the *antagonist*. Antagonistic pairs are found in virtually every skeletal muscle system in biology. In fact, any movement around a joint relies on antagonistic pairs. The biceps and triceps muscles are one example of such an antagonistic pair. Upon arm flexion, the biceps is the agonist while the triceps is the antagonist. The roles are reversed when the arm is uncurled.

Other muscles that play a role in motion are the *synergists*, which assist the prime mover, and *fixators*, which help to steady the movement. The *serratus anterior* muscle is an example of fixator in horizontal punching movements. *Flexor* and *extensor* are other denominations commonly found in anatomy books. The former describes a muscle, which assists in flexing a limb, while the latter refers to a muscle, which assists in extending a limb.

4.1.7 Blood irrigation

When the muscle contracts repeatedly as during a physical exercise, neighboring veins swell in order to accelerate the return of the blood towards the heart. In some areas of the body, they

sometimes stand out prominently beneath the skin as a result of multiple, powerful contractions of the muscles they irrigate.

4.1.8 Lines of action

A *line of action* (or *action line*), as meant by the biomechanics community, denotes the imaginary line along which the force exerted onto the bone is produced. The definition of this line is nevertheless not as strict as may seem. Many specialists assume that the muscle force acts along the straight line which connects attachment points of the tendons. This approximation aims to ease the analytical description of the model but is not always correct from anatomical and mechanical points of view. At the other end of the spectrum, Jensen and Davy [Jensen 75] define the action line as the *muscle centroid curve*. The centroid line is estimated from measurements on cadavers in rigid poses, which has two main flaws: The centroid line cannot be described analytically and is undefined in skeleton configurations other than the one in which the measurements are carried out. A variety of intermediate approaches has also flourished. The most common one defines the action line as a series of line segments (or *polyline* in computer graphics terminology) [Delp 00]. The choice and the number of segments depend on the anatomy but, as a rule, the line of action is representative of the muscle force at a cross-section.

4.2 Anatomy of the Upper Body Musculature

Here we take a closer look at the muscles of the human body (see Fig. 4.4), region by region. We limit our study exclusively to the superficial muscles and those that indirectly exert an influence on the superficial contours. We do not describe muscles in the lower part of the body because demonstrations and examples in this thesis are chiefly restricted to the upper body musculature. For more information about muscles in the lower limb, we direct the interested reader to any good textbook on artistic anatomy ([Thomson 29] or [Richer 81] for instance).

4.2.1 Muscles of the shoulder girdle

The *deltoid* is a large, entirely superficial muscle that forms the shoulder cap (see Fig. 4.4). It is triangular in shape, the base of the triangle arising from the lateral third of the clavicle, the acromion, and the scapula. The muscle is divided into three portions with different pennations, which converge to a tendon that is inserted into the humerus. The anterior and posterior portions consist of parallel fibers while the central portion is bipennate (see Fig. 4.3) thus producing a remarkably coarse texture. The action of the deltoid depends on which portion is brought into play. The central and anterior parts serve as the principal abductor of the humerus, their activity increasing progressively with the elevation of the limb and reaching a peak between 90 and 180 degrees of abduction [Netter 87]. The anterior and posterior fibers also assist in bending the arm forwards and backwards [Thomson 29].

The *trapezius* is a large thin superficial muscle with a complicated trapeze-like shape, which covers the back and the back of the neck. It arises as an extremely wide tendinous portion starting in the neck and reaching as low down as the level of the last thoracic vertebra on the spine.

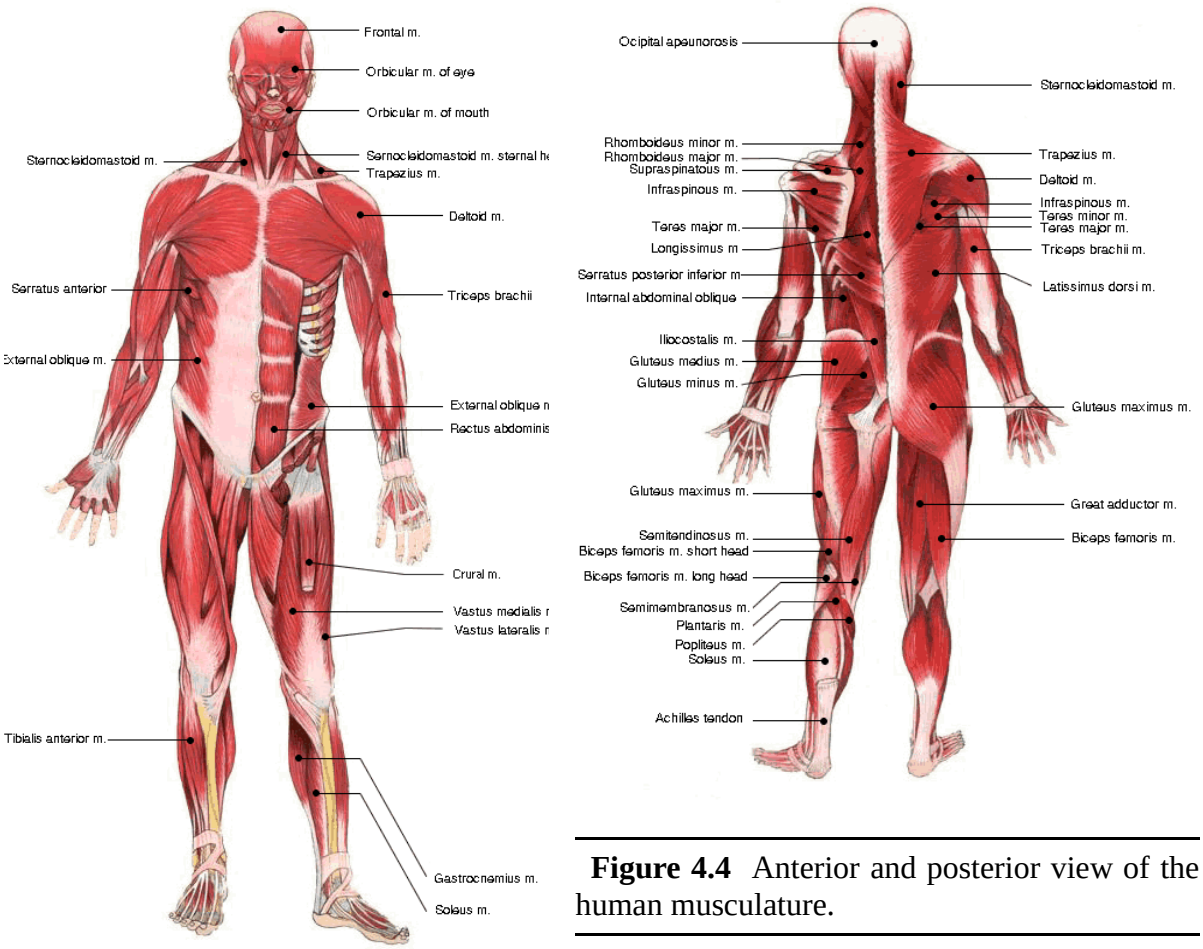


Figure 4.4 Anterior and posterior view of the human musculature.

The muscle fibers converge towards the bones of the shoulder. The *trapezius* covers other muscles which exert considerable influence over its form in return. The upper thoracic part of the muscle draws the shoulder strongly backwards and contributes to pulling or extension movements of the arm. The middle and lower segments assist in pulling and squaring the shoulders, and in the abduction of the humerus (mainly due to the middle part). Finally, the muscle acts in conjunction with other muscles to rotate the scapula.

The *latissimus dorsi* is the broad, triangular muscle of the lower part of the back. It arises as an aponeurosis with attachments to the spinous, lumbar, sacral processes and the lower six thoracic vertebrae. The fibers succeed the aponeurosis and ultimately converge to a band-like tendon, which inserts into the humerus. The muscle draws the arm downward and backward and also rotates it inward. Its full action is exemplified by the crawl stroke in swimming [Netter 87].

The *pectoralis major* forms the fullness of the upper portion of the chest. It also imparts a roundness to the lower part of the anterior hollow of the armpit. The pectoralis major has clavicular, sternocostal, and abdominal portions that converge to join on the humerus. Its primary action is to flex and adduct the humerus through its clavicular portion, but it is also capable of drawing the shoulder downward through its sternocostal portion, and of medial rotation of the

arm when this action is resisted.

4.2.2 Muscles of the abdominal wall

The two main superficial muscles of the abdominal wall are enclosed in tendinous sheets. These aponeuroses originate in the region of the lumbar vertebrae, go all the way round the waist, and reunite in a middle line in front, known as *linea alba*, which runs from the base of the pectorals down to the pubis, thus forming a separation between the left and right side of the body that is clearly visible in thin muscular men.

On either side of the linea alba are the longitudinal recti muscles (or *rectus abdominis* muscle), ensheathed by the aponeuroses of the muscles of the flank. The *rectus abdominis* muscle is peculiar in that it is interrupted by transverse tendinous intersections at definite positions. It is the muscle, which, when well defined, is routinely referred to as “washboard abdominal”.

The flank is covered by several oblique and transverse muscles, the most superficial of which being the *external oblique*. The *external oblique* is broad, flat, thin and irregularly quadrilateral, arising by eight fleshy digitations arranged in an oblique line which runs downwards and backwards. From these attachments, the fleshy fibers proceed in various directions.

The abdominal muscles support the contents of the abdominal cavity and assist in breathing too. When muscles of both sides are brought into play, flexion of the vertebral column is made possible. If muscles are contracted on only one side, a lateral movement of the trunk towards the side of muscular contraction can follow.

Flexion and extension of the vertebral column produces considerable modification of the abdominal region in terms of shape. Upon flexion, transverse folds appear, the deepest of which traverses the belly at the level of the navel. On the contrary, upon extension of the trunk, a distinct flattening and stretching occurs.

4.2.3 Muscles of the upper arm

The *biceps brachii*, or biceps of the arm, is a long fusiform muscle made up of two heads. It arises from the shoulder blade by two tendons that lie under cover of the shoulder muscles. Each superior tendon expands into a fleshy belly. Yet, the two bellies are so close to each other that they often seem to form only one piece. The bellies end in a flattened tendon, which is inserted onto the radius. While the tendons are hidden, the fleshy part of the biceps is superficial and gives its round form to the front of the upper arm. The biceps is the main flexor of the forearm but it also assists in its supination and in raising the arm at the shoulder joint.

The *triceps brachii* is a three-head muscle that extends the entire length of the back of the upper arm. Two heads arise from the humerus while the last one springs from the shoulder blade. The bellies come together in a flat tendon, which inserts into the oleocranon behind the elbow joint. This creates a flat surface over the tendon, and above this flattened area, contraction causes a clear a bulging. The triceps works in conjunction with the biceps and is the main

extensor of the forearm, but it also plays a role in the adduction of the arm.

The *brachialis* is a deeper muscle, which covers the front of the elbow joint and the lower half of the humerus. Its inferior tendon is inserted into the tuberosity of the ulna. The muscle is partially overlain by the biceps onto which it pushes when contracting, thus indirectly shaping the surface form. The main function of the *brachialis* is to flex the forearm.

The *coraco-brachialis* is a small muscle arising from the shoulder blade and inserting midway into the humerus on the inner side. The muscle becomes clearly defined when the arm is raised. As a matter of fact, it assists in raising the arm at the shoulder joint.

The *anconeus* is a very small triangular muscle between the upper am and the forearm. It covers the back of the head of the ulna.

4.2.4 Muscles of the forearm

There are 19 muscles in the forearm, which all exert an influence on the contours of the limb, either directly or indirectly. All muscles assist in moving the hand and the fingers, except the *brachoradialis*, which is an elbow flexor. The muscles can be split into six categories according to the movement they effect [Netter 87]:

- Rotate the radius on the ulna: Pronator teres, Pronator quadratus, Supinator. The latter supinates the forearm, hence its name, while the two others perform the prenation.
- Flex the hand at the wrist: Flexor carpi radialis, Flexor carpi ulnaris, Palmaris longus.
- Flex the digits: Flexor digitorum superficialis, Flexor digitorum profundus, Flexor pollicis longus.
- Extend the hand at the wrist: Extensor digitorum, Extensor indicis, Extensor digiti minimi.
- Extend the digits, except the thumb: Extensor digitorum, Extensor indicis, Extensor digiti minimi.
- Extend the thumb: Abductor pollicis brevis, Extensor pollicis brevis, Extensor pollicis longus.

Interestingly, all the pronator and superficial flexor muscles arise from the inner condyle of the humerus by a common tendon of attachment and clothe the anterior compartment of the forearm, while all the supinator and superficial extensor muscles spring from the external condyle of the humerus by a common tendon of origin and pass down the posterior side of the forearm [Thomson 29].

In the lower part of the forearm, both in the anterior and posterior regions, the fleshy bellies of the muscles are replaced by long tendons that take less space, thus imparting a fusiform aspect to the outer shape. Most tendons of insertion, especially those of the flexors, reveal themselves above the wrist when the hand is clenched into a fist. The *palmaris longus* tendon, which is

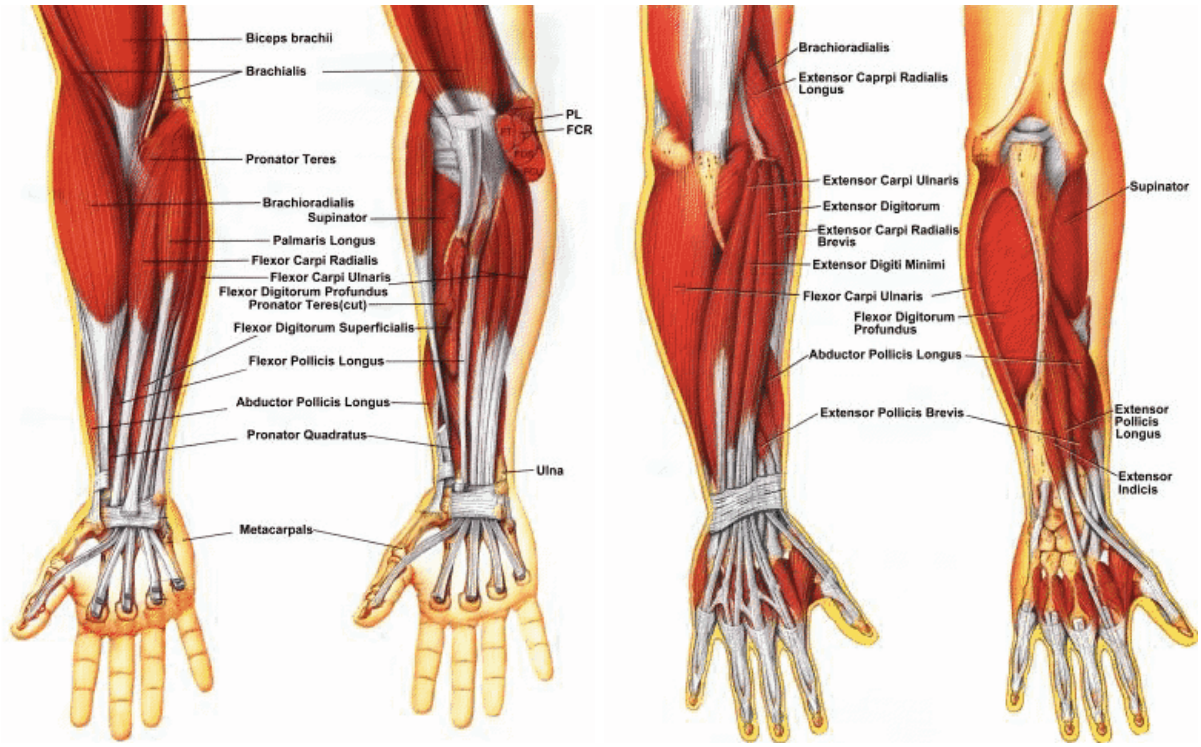


Figure 4.5 Muscles of the forearm. From left to right: superficial anterior muscles, deep anterior muscles, superficial posterior muscles, deep posterior muscles.

commonly visible at all times, is an exception. For a few extensor and flexor muscles, namely the *extensor digitorum*, the *flexor digitorum superficialis*, and the *flexor digitorum profundus*, the belly branches into several tendons in its lower part, each tendon serving a specific finger. However, the arrangement of tendons in the hand is quite complex, forming an intricate ramification with connections between tendons in different fingers. This is why it is so difficult to independently move the fingers, even impossible sometimes, as the case with the middle finger. The thumb, which is moved by clearly independent tendons, is the exception to the above rule.

4.3 Geometric Model

In light of the extreme diversity of muscle shapes, existing computer graphics models are for the most part overly simple. The ellipsoid, the most widespread model, cannot capture the shape of any single muscle of the human body. It does not even approximate fusiform muscles properly, since these have attachments of varying size and possibly several heads (e.g. *biceps brachii*). In fact, only general triangle meshes and B-spline models avoid any loss of generality.

Ng-Thow-Hing [Ng-Thow-Hing 00] pointed out that solid B-splines can represent all muscle shapes as well as more or less broad attachments. However, B-spline surfaces and solids are

completely determined by a set of control points that do not necessarily match a real location in the material, thus making properties such as mass or elasticity hard to specify. Ng-Thow-Hing solves this issue by introducing points of maximum influence that correspond to actual locations in the material. The drawback is that transformations from the B-spline's parameter space (control points) to material space (points of maximum influence) as well as the inverse transforms must constantly be performed. This duality inevitably incurs a performance cost.

In this work, for speed and simplicity, we chose to use triangle meshes for representing musculotendon units. Yet, our approach could work equally well with other surface representations such as B-spline patches or subdivision surfaces. Since our model does not require any specific organization of the mesh structure, nor a certain regularity of the discretization, we can use any mesh without having to resample, decimate, or alter the connectivity in any way. Muscles presented in this thesis were created in Maya, a commercial 3D modeler [Alias 01], and directly imported into our in-house software but we could as well have taken advantage of prior work and used muscle meshes reconstructed from medical data [Beylot 96] [Ng-Thow-Hing 00].

4.4 Two-layer Deformation Model

Research so far has mainly focused on modeling muscles in static postures, sometimes ignoring the animation problem. And yet, it is very complex to automatically derive the appropriate position and deformation of a muscle in any possible posture. In our approach, as often in computer graphics, the motion of the skeleton induces the deformations of the muscles contrary to what occurs in reality.

We abstract the muscle by two layers: a skeleton, which is defined by the action lines of the muscle on the one hand, and a surface mesh, which represents the muscle shape, on the other hand (see Fig. 4.6). The key idea of our approach is that the deformations of the surface mesh are entirely driven by the underlying action lines. Thus, the three-dimensional nature of the elasticity problem is reduced to one dimension for fusiform muscles (one action line) and two dimensions for flat muscles (several action lines). The idea of using action lines for modeling forces is not new in biomechanics. In computer graphics, a single work suggested to use these lines [Porcher 98], but then again for modeling forces. The novelty of our approach is that the action lines can serve both to deform the muscle shape and to model the produced force.

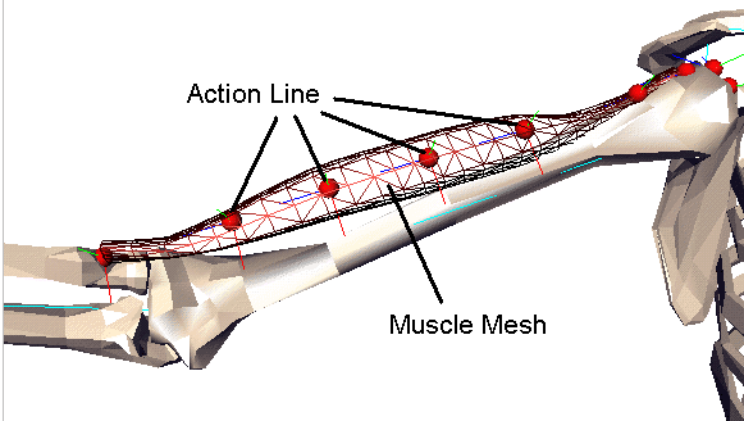


Figure 4.6 Two-layer abstraction of a muscle.

Multi-belly muscles usually have clear bundles e.g. the pectoral muscle has four main bands of fibers. However, as these bundles are wrapped into connective tissues (i.e. fascia), the muscle displays a single, continuous, smooth surface. It therefore makes sense to dissociate the main directions of contraction of the muscle from its surface, as we do.

In order to avoid possible confusion in the rest of this chapter, we shall use the term *node* when referring to a vertex of an action line, and the term *vertex* when speaking of a vertex of the surface mesh.

4.4.1 Action lines

Each action line is modeled by a polyline from which an equivalent 1D mass-spring-damper system is constructed. The nodes that correspond to the insertion and origin of the muscle are anchored to the skeleton joints so that their motion is driven only by the skeleton. We refer to these nodes as *attachment nodes*. The positions of all the remaining nodes are obtained, for each animation frame, through an elastic relaxation of the mass-spring system. These nodes are henceforth referred to as *dynamic nodes*. Usually, the first and last node of the polyline are attachment nodes while the nodes between the two end nodes are dynamic nodes. Note that dynamic nodes are always enclosed by attachment nodes.

All nodes of the action line are assigned an equal mass, which is not a problem because the mass-spring system has no physical reality but is rather used as a deformation tool. The stiffness of each spring is determined by the material type it represents, either tendon or belly. We also add attractive and repulsive force fields in order to constrain the action line. These force fields are built up from ellipsoids. Repulsive force fields prevent gross penetration of other anatomical structures while attractive fields help to refine the trajectories of the action line (see Fig. 4.7).

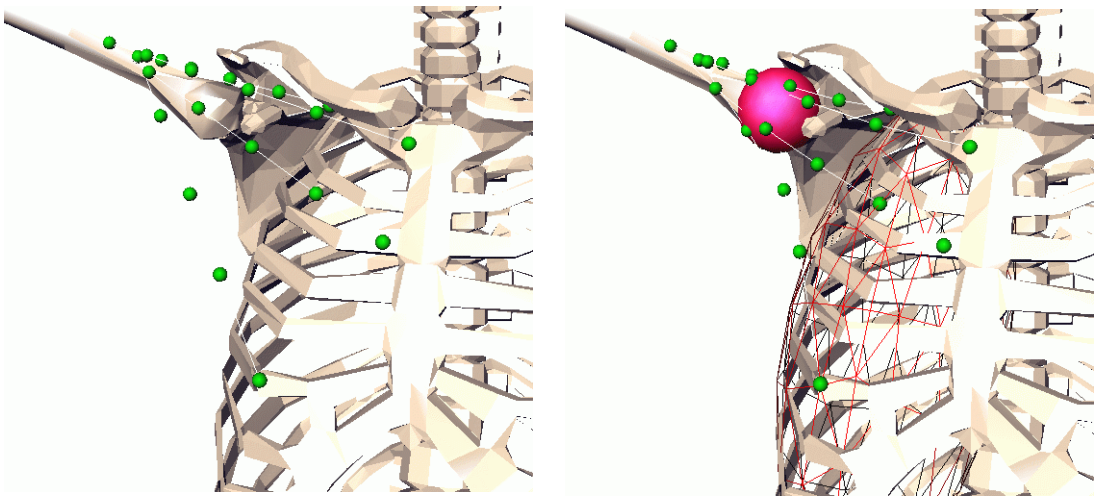


Figure 4.7 Action lines of the pectoral muscle during shoulder abduction. Left: no force fields; Right: use of two force fields (solid and wireframe ellipsoids).

Due to the nature of the discretization, sharp angles may appear in the polyline. This may occur for example when insufficiently sampling a tendon that wraps around a joint. We remedy this situation by replacing the polyline in such cases by a C1 piecewise cubic curve that interpolates the nodes of the action line with specified tangential directions [Farin 90]. More precisely, a piece of curve between two successive nodes is a cubic Bezier curve whose tangent is computed as in Section 4.4.2. This has the effect of smoothing out the action line.

4.4.2 Local frames

The positions of the nodes provide information as to how the surface mesh will expand or shrink over time. So as to infer the orientation of the mesh, we need to augment each action line node with a local frame. The construction of these local frames is a rather complex operation which we shall now describe.

We start by computing the Z-axis at each node. We proceed as depicted in Fig. 4.8. The Z-axis is set to the normal of the bisecting plane for every in-between node ($\boldsymbol{V_1}$ and $\boldsymbol{V_2}$) and to the tangent for the end nodes ($\boldsymbol{V_0}$, $\boldsymbol{V_3}$). We then proceed to compute the X-axes. To do so, we distinguish between the attachments nodes, which are rigidly anchored to the skeleton, and the dynamic nodes, which move freely under the influence of springs and force fields. The Y-axes are ultimately found by completing the right-handed coordinate systems.

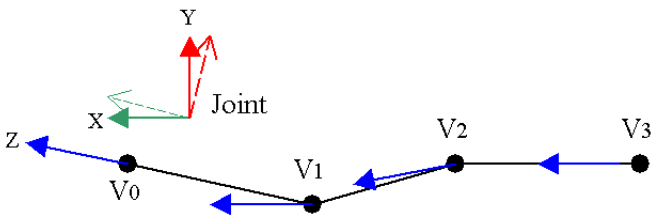


Figure 4.8 Z-axis is set to the normal of the bisecting plane. The joint frame is initially rotated so as to align one of its axis (here the X-axis) with the Z-axis of end node $\boldsymbol{V_0}$.

The X-axes are first computed for attachment nodes. We take, in the rest posture, the local frame of the joint (X and Y solid arrows in Fig. 4.8) to which the node is bound and rotate it so as to bring one of its axes in alignment with the node's Z-axis. The selected axis is the one that leads to the minimal rotation (X in Fig. 4.8 for instance). The resulting rotated frame (dashed arrows in Fig. 4.8) is expressed and saved in the joint's coordinate system. During any subsequent animation, this frame is transformed by the joint's current coordinate system, then rotated again so as to be aligned with the current direction of the node's Z-axis. This rotation is usually quite small because the smallest rotation in the rest posture was initially chosen. Thus, the local frame of every attachment node is smoothly updated as the action line moves and deforms itself.

There remains to compute the X-axes for the dynamic nodes. To this end, we make use of the orientations already computed for the two attachment nodes that enclose the dynamic nodes. One can naively try to interpolate the two end orientations using a spherical linear interpolation (slerp) [Dam 98] [Shoemake 85]. This approach fails, however, when the difference between the two end orientations nears π. As spherical linear interpolation picks the shortest path on the

quaternion unit sphere, the orientation at a node may suddenly flip from one animation frame to the next.

The problem here is akin to the computation of reference frames along a curve, whose tangent vectors are known. Bloomenthal solves this problem iteratively by minimizing rotations between two successive frames [Bloomenthal 90]. Given a start frame and tangent vectors for all frames along the curve, the orientation is propagated along the curve using small local rotations, typically the minimal rotation that aligns the tangent vectors of two successive frames (given tangent vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, the minimal rotation is defined by the axis $\boldsymbol{u}_1 \times \boldsymbol{u}_2$ and the angle $\operatorname{acos}(\boldsymbol{u}_1^T \boldsymbol{u}_2)$). Yet, we actually want the start frame to smoothly evolve along the curve until reaching a fixed orientation determined by the end frame. This cannot be done with the method proposed by Bloomenthal since it allows to fix the orientation at only one end of the chain.

Hanson introduced a mathematical framework for representing the space of possible frames along curves and surfaces [Hanson 98]. We employ the same framework for minimizing the twist when interpolating frames along a defined path. The family of frames with a preferred direction $\boldsymbol{v}$ can be expressed as the multiplication of two quaternions:

$$\boldsymbol{f}(\theta,\boldsymbol{v}) = \boldsymbol{q}(\theta,\boldsymbol{v}) \times \boldsymbol{b}(\boldsymbol{v}) \quad \text{where} \tag{4.1}$$

$$\boldsymbol{q}(\theta,\boldsymbol{v}) = (\cos\left(\frac{\theta}{2}\right), \boldsymbol{v} \cdot \sin\left(\frac{\theta}{2}\right)) \text{ and} \tag{4.2}$$

$$\boldsymbol{b}(\boldsymbol{v}) = \boldsymbol{q}(\operatorname{acos}(\boldsymbol{z} \cdot \boldsymbol{v}), \frac{\boldsymbol{z} \times \boldsymbol{v}}{\|\boldsymbol{z} \times \boldsymbol{v}\|}) \tag{4.3}$$

Eq. (4.2) describes the family of rotations that leaves the direction $\boldsymbol{v}$ invariant while quaternion $\boldsymbol{b}$ in Eq. (4.3) maps the $\boldsymbol{z}$ vector onto the chosen vector $\boldsymbol{v}$. The variable θ in Eq. (4.1) serves to parametrize a ring in quaternion space, each point of which corresponds to a particular 3D frame. Equipped with this mathematical framework, one can transform the problem of interpolation into a minimization problem with boundary conditions. We choose to minimize the following cost function:

$$\sum_{i=0}^{n-1} \operatorname{acos}(\boldsymbol{f}(\theta_i,\boldsymbol{z_i}) \cdot \boldsymbol{f}(\theta_{i+1},\boldsymbol{z_{i+1}})) \quad \text{with } \theta_0 \text{ and } \theta_n \text{ fixed} \tag{4.4}$$

which represents the sum of all turning angles undergone by the frames. Like in most optimization problems, several minima may exist for Eq. (4.4). Worse, even if we find the global minima at frames *k*-1 and *k*, these minima are not necessarily those we want because they are independently computed. The problem basically is that new interpolated frames must exhibit both a spatial and temporal coherence. In order to avoid switching from one (possibly local) minimum to a very different (possibly local) minimum in the next animation frame, one can initialize the vector $\theta^{\boldsymbol{k}} = (\theta_0^k, \theta_1^k \ldots \theta_n^k)$ at frame *k* with the solution vector $\theta^{\boldsymbol{k-1}} = (\theta_0^{k-1}, \theta_1^{k-1} \ldots \theta_n^{k-1})$ found at the previous frame. This is not sufficient, however. Visually, frames sometimes undergo sudden orientation flips, especially for quick motions of

the skeleton. To ensure time continuity between frames *k*-1 and *k*, we restate the minimization problem as follows:

$$\text{Minimize} \sum_{i=0}^{n-1} \left|\theta_i^k - \theta_i^{k-1}\right| + \sum_{i=0}^{n-1} \operatorname{acos}(\boldsymbol{f}(\theta_i^k, \boldsymbol{z_i}) \cdot \boldsymbol{f}(\theta_{i+1}^k, \boldsymbol{z_{i+1}})) \qquad (4.5)$$

$$\text{with } \theta_0^k \text{ and } \theta_n^k \text{ fixed.}$$

In solving Eq. (4.5), care must be taken to clamp $\left|\theta_i^k - \theta_i^{k-1}\right|$ to $[0 \ldots \pi]$. This is done by replacing $\left|\theta_i^k - \theta_i^{k-1}\right|$ in the evaluation of the cost function by:

$$2\pi - \left|\theta_i^k - \theta_i^{k-1}\right| \text{ when } \left|\theta_i^k - \theta_i^{k-1}\right| > \pi$$

We also investigated more geometric answers to the interpolation problem by combining the principle of iterative minimization put forward by Bloomenthal and the mathematical framework of Hanson. We propagate the X-axis direction of each end frame to the in-between nodes as follows. As we know the Z-axes of all frames, we already have for each node $\boldsymbol{V_i}$ a plane P_i normal to $\boldsymbol{z}_i$ in which the remaining $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ axis must lie. Starting from axis $\boldsymbol{x_0}$ at end node $\boldsymbol{V_0}$, we estimate the axis $\boldsymbol{x_1}$ in the plane P_1 by sampling the trigonometric circle using Eq. (4.1) and finding θ_1 so that it minimizes the dot product $\boldsymbol{x_0} \cdot \boldsymbol{x_1}$. The process is iterated for each $\boldsymbol{x}_{i+1}$ by finding θ_{i+1} that minimizes the deviation from $\boldsymbol{x}_i$. We thus propagate the orientation at node $\boldsymbol{V_0}$ to the other end node $\boldsymbol{V_n}$. We then perform the same operation, but in reverse order, and conversely propagate the orientation from $\boldsymbol{V_n}$ to $\boldsymbol{V_0}$. At this stage, we rely on the principle of temporal coherence to handle 2π modulos. Finally, a linear interpolation of the two θ values computed at each node is performed using a user-specified ratio or one that is related to the distance from the in-between node to the two end nodes along the polyline. This interpolation does not fail because it takes place in the parameter space of Eq. (4.1), namely in angular space, and not in quaternion space as for the spherical linear interpolation.

In practice, this geometric approach has proven quicker than the (mathematically more accurate) resolution of Eq. (4.5). The approach with the cost function is plagued by the nonlinearity and the discontinuity of the derivative in Eq. (4.5) which incur a rather severe resolution cost. Contrariwise, wild oscillations of the frames are completely subdued using the geometric approach with a much lower computational cost.

4.4.3 Muscle mesh

Every muscle mesh is modeled by hand and interactively placed on top of the bones. As stated before, the action lines serve as a skeleton for producing surface deformations. Let us first describe how vertices of the surface mesh are reparameterized.

As explained in Section 4.4.2, the polylines (or curves), which approximate the action lines, are augmented with frames. Hence, an action line can naturally be considered as a skeleton: nodes amount to joints while segments (or pieces of curve) correspond to bones. This allows us to reuse algorithms developed for mapping skin vertices to skeleton segments [Magnenat-

Thalmann 88] [Sun 99].

Mapping a vertex to an action line is done as follows. We project the vertex onto the XY planes of the action line nodes and select the two closest enclosing planes, as in [Sun 99]. Let d_1 and d_2 be the distances to these two planes and let $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ denote the respective positions of the nodes through which the enclosing planes pass (see Fig. 4.9). The vertex is mapped to the segment *s* joining $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ and the ratio *t* along this segment is set to $t = d_1/(d_1 + d_2)$. The vertex is then expressed in a local coordinate system whose origin is situated at:

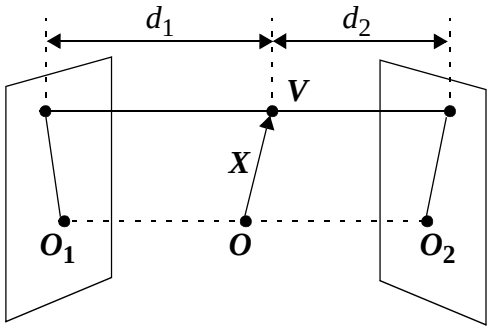


Figure 4.9 Mapping a vertex to an action line segment.

$$\boldsymbol{O} = t \cdot \boldsymbol{O_1} + (1 - t) \cdot \boldsymbol{O_2} \tag{4.6}$$

on the segment (similarly, we position the origin at ratio *t* along the piece of curve between $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ when the action line is described by a piecewise cubic polynomial) and orientation is determined by spherical linear interpolation (slerp) of the orientations at $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ with ratio *t*. Here, it is safe to rely on spherical linear interpolation because the difference between frames at $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ does not approach π for any reasonable discretization of the action line. Eventually, a vertex is reparameterized by the triplet (*s*, *t*, $\boldsymbol{X}$) where *s* indexes the segment (or piece of curve) of the action line, *t* expresses a ratio along this segment, and $\boldsymbol{X}$ denotes the local 3D coordinates of the vertex.

So far, we have explained how to map a vertex to one action line. Nevertheless, several action lines may run through the mesh depending on the form of the muscle. If so, a vertex is mapped to either one or two action lines based on the following two-pass algorithm. In the first stage, we map every vertex to every action line. As a result, we get for every vertex a set of triplets $(s, t, \boldsymbol{X})_i$ with $i = 1 \ldots n$ associated with the *n* action lines. We then order the action lines by their euclidean distance to the vertex, this distance being $\|\boldsymbol{X}\|$. At this point, it may look like a reasonable choice to simply select the two closest action lines for a given vertex. However tempting this may be, it is wrong because the distances should not be computed in euclidean space but rather along a path on the surface mesh. As an illustration of this problem of metric, let us consider the *deltoid*, a U-shaped muscle, which is usually split into three parts, namely the anterior, middle, and posterior regions. Let us also assume that we conveniently design three action lines to control the deformations of each region. Because of the particular U-shape of the muscle, some vertices belonging to the anterior region are likely to be mapped to the posterior action line (see Fig. 4.10., left picture) This peculiar U-shape is also found in the *trapezius*

muscle.

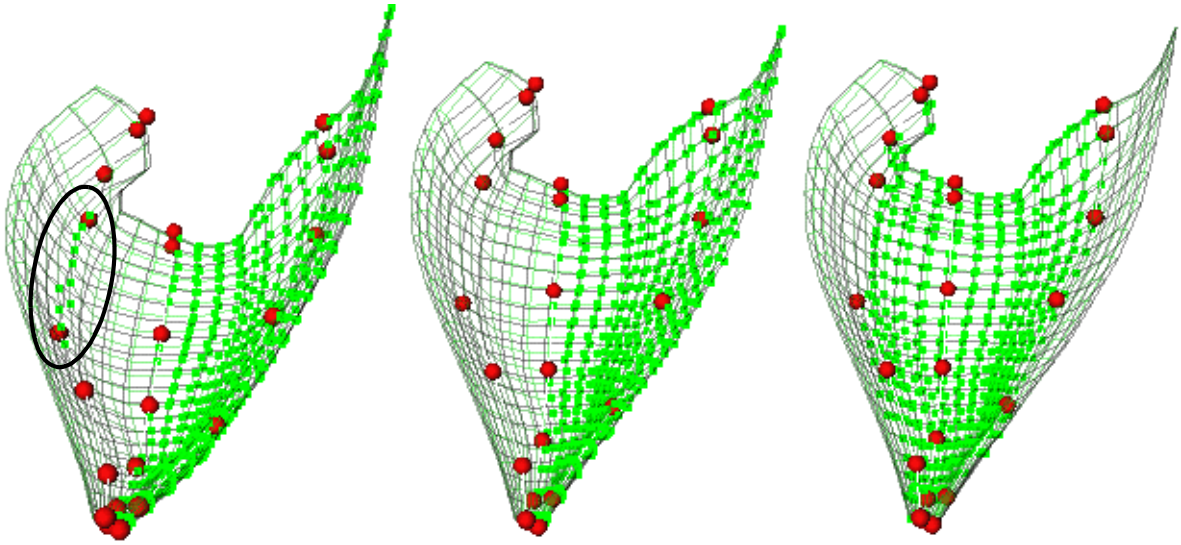


Figure 4.10 Mappings for the deltoid muscle. Left: incorrect mapping of vertices to the posterior (right) action line; Middle: corrected mapping using regions of influence; Right: region of influence of the middle action line.

We overcome this difficulty by constructing, in a second pass, the region of influence of each action line using an incremental algorithm. We initialize the region of influence of an action line with the set of vertices that are not closer to any other action line. We mark this initial region with a 0-index. Then, we scan the list of vertices and select those for which the action line is the second (third, fourth, etc. during the following iterations of the algorithm) best possible mapping. When a vertex, whose first-ring neighbors do not belong to any existing region, is found, a new region consisting of this single vertex is created with a higher index. Conversely, when a vertex connected to an existing region through one of its neighbors is found, we collapse the linked regions into the region with the lowest index. Thus, after some time, the regions with the lowest indices grow and others gradually disappear. The refinement process is iterated, i.e. vertices are scanned several times, until the regions no longer change. In the end, the region of influence takes the form of a single subset of connected vertices (see Fig. 4.10, middle picture). By stating that a vertex may be mapped to an action line if and only if it belongs to its specific zone of influence, we thus ensure that a vertex is always parameterized by the two closest action lines when following a path on the surface of the mesh.

The above algorithm works fine except that vertices located “on the border” of the mesh are mapped to two action lines instead of only one. We identify these border vertices by relying on the projections onto the various action lines (see Fig. 4.11). We simply compute the angle θ between the two directions given by the projections onto the two best action line candidates. The vertex is considered to be on the border if $\theta < \theta_{threshold}$. The threshold should be inversely proportional to the “thickness” of the mesh. We set the default value to $\theta_{threshold} = \pi/2$, which works well for thin, plate-like muscles. This is adequate as most muscles parameterized by multiple action lines have a flattened shape. The default value was

used in the right picture of Fig. 4.10.

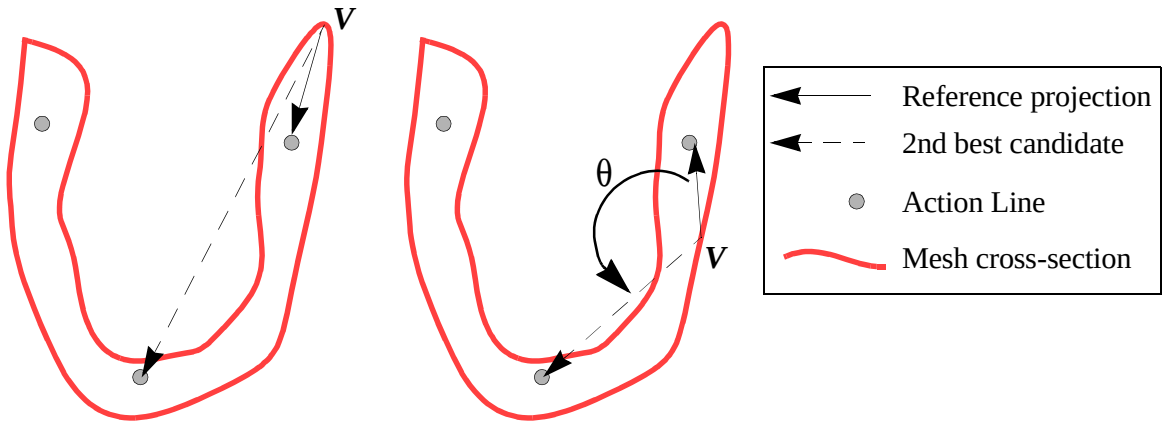


Figure 4.11 The angle θ determines the number of action lines for parameterizing a vertex ***V***. Left: vertex is mapped to one action line. Right: vertex is mapped to two action lines.

A vertex is finally parameterized by coordinates (u_1, s_1, t_1, $\boldsymbol{X}$) if it lies on the border, and by (u_1, s_1, t_1, u_2, s_2, t_2, $\boldsymbol{X}$) otherwise. In both cases, u indexes the action line, s is the index of the segment, t expresses a ratio along s, and $\boldsymbol{X}$ stores the local coordinates. When a vertex $\boldsymbol{V}$ is mapped to two action lines, the local coordinates $\boldsymbol{X}$ are computed using bilinear interpolation. Let $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ denote the local coordinates of $\boldsymbol{V}$ expressed independently with respect to action lines u_1 and u_2 and let $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ be the origins of the local frames on the action lines. Distances to action lines u_1 and u_2 are computed and normalized as follows:

$$w_1 = \frac{\|\boldsymbol{X_2}\|}{\|\boldsymbol{X_1}\| + \|\boldsymbol{X_2}\|} \text{ and } w_2 = 1 - w_1 \tag{4.7}$$

Vertex $\boldsymbol{V}$ is transformed into the local coordinate system whose origin sits at:

$$\boldsymbol{O} = w_1 \cdot \boldsymbol{C_1} + w_2 \cdot \boldsymbol{C_2} \tag{4.8}$$

and whose orientation is determined by spherical linear interpolation of the orientations at $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ with the weight w_1.

Eventually, during animation, a vertex $\boldsymbol{V}$ is moved as follows in matrix notation:

$$\boldsymbol{V} = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} & O_x \\ M_{2,1} & M_{2,2} & M_{2,3} & O_y \\ M_{3,1} & M_{3,2} & M_{3,3} & O_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\boldsymbol{X}}$$

where $\bar{\boldsymbol{X}}$ are the local homogeneous coordinates (i.e., the local coordinates $\boldsymbol{X}$ augmented with a 1), $\boldsymbol{s}$ is a scaling vector, point $\boldsymbol{O}$ is found by Eq. (4.6) or Eq. (4.8), and rotation matrix M by spherical linear interpolation as previously explained.

4.4.4 Isotonic contraction

Isotonic contraction (see Section 4.1.2) is simulated during animation by simply scaling each vertex along the axes of its local coordinate system based on the change in length of the muscle. The choice of a suitable scaling function is dictated by two requirements:

- C1 continuity to confer a visually pleasant smoothness to the deformation, and
- minimization of the volume change since biological tissues tend to conserve their volume upon deformation [Maurel 98].

We begin with surface meshes parameterized by a single action line. We can think of such a mesh as a generalized cylinder, with the action line as the axis of the cylinder. For such geometric primitives, Wilhelms *et al.* suggest to scale the width and height of each cross-section by a ratio related to the change in length of the axis of the cylinder [Wilhelms 97b]:

$$\text{scaling} = \sqrt{\frac{\text{rest length}}{\text{present length}}} \tag{4.9}$$

We know from simple integral calculus that the volume of a cylinder of radius r and length h is $(\pi r^2)h$ and that the volume of a torus of radius r with an axis of rotation of length d equals $(\pi r^2)(2\pi d)$. If we now sweep an arbitrary uniform cross-section along a C1 (or higher continuity) curve, we can extrapolate from the cylinder and torus examples that the swept volume is:

$$v = (\text{cross-section area}) \times (\text{curve arc length})$$

So, the volume of a generalized cylinder with a uniform cross-section should remain unchanged during deformation if we scale the cross-sections using Eq. (4.9). In practice, however, we have meshes that correspond to generalized cylinders with varying cross-section. Still, intuition tells us that Eq. (4.9) remains a reasonable choice for any kind of generalized cylinder.

The change of length of the action line is in reality not distributed uniformly due to the different elasticity coefficients of the springs. We take this non-uniformity into account by constructing a scaling function, similar to Eq. (4.9), but one that reflects the different changes in length of the segments. To this end, we compute the elongation, defined as the initial length divided by the current length, for every segment (or piece of curve) of the action line. Note that this length is computed anyway when evaluating the elastic force induced by the spring. Then, we establish a correspondence between the measured elongations and the action line. Using the two-coordinate parameterization of the action line (*segment* = *s*, *position within segment* = t), we map each elongation value e_i, $i = 0 \ldots n-1$ to the middle of the corresponding segment: $e_i \rightarrow (i, 0.5)$ and duplicate the elongations at either end of the action line: $e_0 \rightarrow (0, 0)$ and $e_{n-1} \rightarrow (n, 0)$. Ultimately, we interpolate the discrete elongation measurements with a piecewise cubic polynomial with C1 continuity [Farin 90]. Thus, we obtain a smooth, individual elongation value for each muscle vertex which we use as the scaling factor squared root:

$$s_x = s_y = \sqrt{e(s,t)} \tag{4.10}$$

Empirical evidence shows that our scaling function is a reasonable choice. Depending on the form of the muscle, we measured volume variations from 1-2% to about 9% when the muscles shorten by 30%, which corresponds to the maximal physiological compression rate [Richer 81].

Finally, for vertices parameterized by several action lines, we determine the scaling factor by bilinear interpolation, in the same way we computed the local coordinates: We calculate the scaling vectors $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$ for the two action lines independently by Eq. (4.10). Then, we interpolate along the other principal direction using weights as defined by Eq. (4.7).

4.5 Interactive Construction of Deformation Models

The two-layer muscle model presented in Section 4.4 provides the general framework for designing muscle deformations. However, many hidden practical issues arise when actually trying to build a deformation model. These issues lead to certain choices, as we shall see in the following sections, in which we describe the successive stages for interactively creating a complete deformation model for a given muscle.

4.5.1 Semi-automatic creation of an action line

For designing an action line, the user starts by specifying the muscle's origin and insertion and the numbers of belly and tendinous segments. A straight action line with the right number of nodes is then automatically created. At this stage, the nodes are equally spaced out along the action line. Stiffnesses of segments are automatically assigned, the stiffness of a tendon segment being set to an order of magnitude higher than a belly segment. Tendons are in reality about two orders of magnitude stiffer than muscle tissues [Fung 81]. We nevertheless found out that extremely stiff tendons increased the instability too much. In practice, increasing the stiffness tenfold suffices from a graphical point of view.

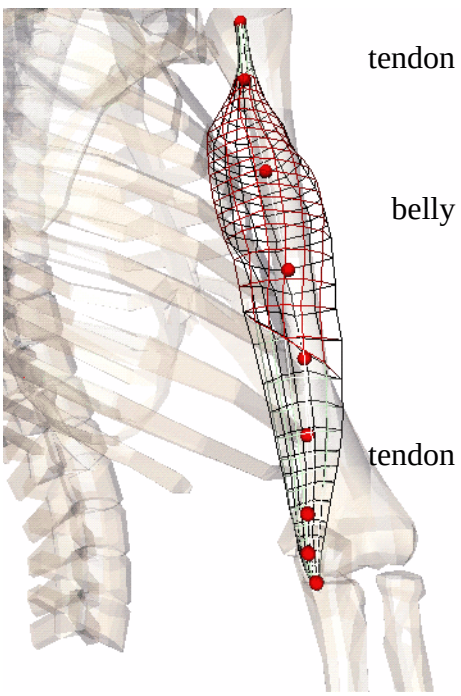


Figure 4.12 Lateral head of *triceps brachii* with five tendon segments and three belly segments.

The position of each in-between node along the straight line joining the two end nodes can then be modified interactively (see Fig. 4.12). An inverse dynamics procedure computes rest lengths for each spring so that current lengths are much higher than rest lengths and that the equilibrium is maintained for the chosen positions of the nodes. Having small rest

lengths ensures that springs remain always elongated. It is crucial because a series of springs experiencing compression creates uncontrollable, unwanted jags along the action line. Besides, human muscles *in vivo* are never in a fully relaxed state [Richer 81]. Elongated springs therefore represent the natural tension of the muscle.

4.5.2 Bending the action line

The designer adds ellipsoidal force fields to bend the (so far) straight action line into the required shape. For a fusiform muscle with one action line, this is more or less the centroid curve. By combining several ellipsoids, more complex force field shapes can be defined. In order to avoid force fields generating forces that would destroy the nice balance produced by the carefully chosen springs' stiffnesses, we introduce two modes for force fields.

In *radial mode*, the node is attracted towards (or repulsed from) the center of the closest ellipsoid. This is useful for ellipsoids that are close to a spherical shape. In *orthogonal mode*, the node is attracted towards its (orthogonal) projection on the closest ellipsoid [Hart 94]. In this mode, the node may slide freely on the surface of the force field until reaching a minimal energy configuration. This mode helps to prevent the force field from counteracting the forces produced by the springs.

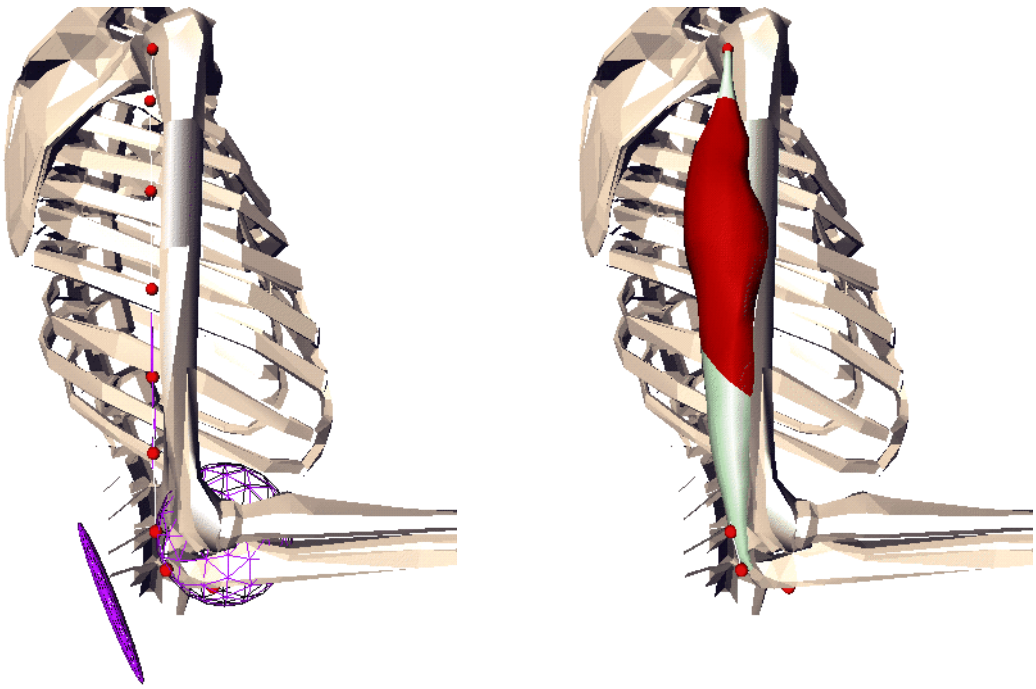


Figure 4.13 Use of ellipsoidal force fields and resulting deformation.

Fig. 4.13 shows the concrete use of force fields to bend the action line of the triceps into the desired shape. The pulley at the elbow is modeled by a repulsive force field (spherical shape in the picture plane) in conjunction with an elongated attractive force field (lower left). The elongated force field exerts a weakly attractive force that constrains the nodes to stay on the lower

part of the elbow. The resulting deformation of the mesh is displayed in the right picture. The tendon bends at the elbow in a natural fashion thus pulling the muscle belly downwards. The mesh is automatically scaled to account for the change in length of the action line.

4.5.3 Avoiding collisions

Attempting to handle all muscle-muscle and muscle-bone collisions is unreasonable. In our framework, preventing muscle-bone collisions is easy and fast using repulsive force fields (see Fig. 4.13). However, it is not highly accurate because the force fields act on the action line instead of acting on the surface mesh. The designer can try to compensate this by slightly increasing the size of the force fields but even so, some degree of penetration at the mesh level cannot always be avoided.

As for muscle-muscle interaction, we allow any force field to be anchored to and deformed by an action line. Thus repulsive ellipsoidal force fields can be positioned along the action line(s) of a deep muscle and their action immediately exerted upon more superficial muscles. This requires that the relaxation of the action lines be performed in a certain order. The resulting limitation is that the chain of influence between muscles must remain acyclic e.g. if muscle A influences muscle B, muscle B cannot in turn influence muscle A. Yet, muscle interpenetration is naturally reduced by carefully designing the deformation of each individual muscle.

4.5.4 Anchoring the mesh

The three previous stages are iterated for each head of the muscle. The process can be accelerated by reusing certain components (insertion and origin nodes, force fields, etc.) between several action lines. Afterwards, the muscle is anchored to the action line(s) in the *rest posture*. This process is executed as described in Section 4.4.3 except for muscles where the belly branches into several tendons For such muscles, the mapping algorithm is slightly modified to prevent vertices that represent the branching tendons from being mapped to two action lines (see Fig. 4.14).

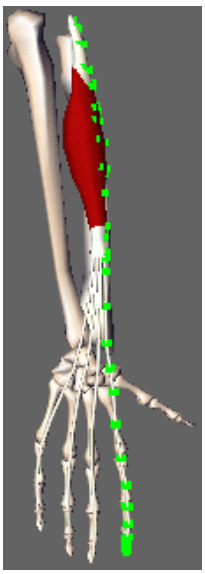


Figure 4.14 Mapping vertices to action lines with branching tendons. Exclusive mapping is done by preventing vertices, which represent the branching tendons, from being mapped to a second action line.

4.5.5 Controlling the mesh deformations

At this point, the designer selects the action line approximation that outputs the more aesthetic (or realistic) deformations of the surface mesh. In our experience, the C1 piecewise cubic curve is preferred to the polyline in the vast majority of cases.

Another adjustment that can be made at this stage is to change the joint reference frame for

computing the local frame of an attachment node (see Section 4.4.2). By default, the local frame of an attachment node is computed using the joint to which the node is bound. The biceps muscle, which attaches onto the ulna, shows that this is not always appropriate. Although the insertion of the biceps moves with the elbow and radioulnar joints, the lower tendon barely changes direction when the elbow flexes, nor does it twist much when the ulna rotates around the radius. As a consequence, better-looking mesh deformations are obtained by using the shoulder joint as the reference frame for constructing the local frame of the insertion node.

A third issue is that, in the rest posture, the action line of a fusiform muscle may not perfectly match the central axis of the mesh. This poses a problem because we uniformly scale the cross-sections of the mesh along the X- and Y-axes of the local frames, as a response to isotonic contractions (see Section 4.4.4). If the action line lies too far from the medial axis, we unintentionally introduce anisotropy in the scaling.

We alleviate the induced anisotropy by estimating the deviation from the centroid axis at regular samples along the action line. For every sample slice of the mesh, we compute the intersection of the XY plane of the local frame with the mesh triangles in order to estimate the shape of the cross-section. The centroid is determined by regularly sampling the outer boundary of the resulting cross-section. We ultimately calculate the deviation vector ***d*** from the centroid of the cross-section to the sample point on the action line (see Fig. 4.15).

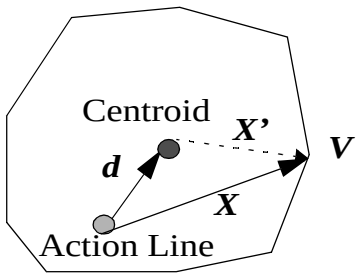


Figure 4.15 Deviation from cross-section centroid.

For any vertex ***V*** we estimate the centroid of the cross-section in which it lies by linear interpolation of the closest deviation vectors. We thus get new local coordinates ***X'*** for ***V*** expressed in the frame whose origin is the centroid of the cross-section and whose axes match those of the local frame on the action line. Eq. (4.10) is replaced by:

$$s_x = \left(\frac{X'_x}{X_x}\right)\sqrt{elongation} \text{ and } s_y = \left(\frac{X'_y}{X_y}\right)\sqrt{elongation} \qquad (4.11)$$

Eq. (4.11) corrects the artificial distortion in the scaling induced by the deviation from the centroid curve and the magnitude of the scaling. For every vertex, the coefficients of correction (X'_x/X_x) and (X'_y/X_y) need to be computed only once in the rest posture, that is when the surface mesh is anchored to the action line.

4.5.6 Isometric contraction

In actuality, the scaling of each vertex results from the combination of an isotonic contraction and of an isometric contraction. As said before, the isotonic contraction is automatically computed from the length variation of the action line. The isometric contraction on the other hand is given by an activation curve for each muscle. The (semi-)automatic determination of the activation curve based on the motion performed is described in Section 4.6.

The designer sculpts, in the rest posture, a new shape for the mesh upon a full isometric contraction. This is done in our system via a graphical interface that lets the graphics artist specify scaling values along the $\boldsymbol{x}$ and $\boldsymbol{y}$ directions of the local frame for every action line node. Typically, the height of the muscle belly is increased while its width is reduced.

4.5.7 Secondary deformation

The deformation of each individual muscle is obtained through a relaxation of its action lines. As a result, we get a collection of static postures for any animation. In other words, we only have control over the kinematics of the muscle. And yet, secondary motions such as inertia-induced oscillations and other such dynamics features are essential for the plausibility of an animation and even need to be exaggerated in computer graphics [Lasseter 87]. Nevertheless, biological tissues rapidly converge to equilibrium under external forces, so it is hard to observe viscosity or inertia effects in some regions of the body like the face or the fingers [Hirota 01]. Slow-motion videos of athletes in action disclose that oscillations tend to occur mostly in large fleshy muscles like the pectorals in the chest or the quadriceps in the thigh.

We treat the deformation of an action line as the combination of a kinematic motion (main motion) and dynamic oscillations (secondary motion) generated by a mass-spring system about the reference positions of the main motion (see Fig. 4.15). More specifically, the oscillations are generated by anchoring each node of the action line to its kinematic trajectory by a damped spring whose stiffness is related to the material type in the node's neighborhood.

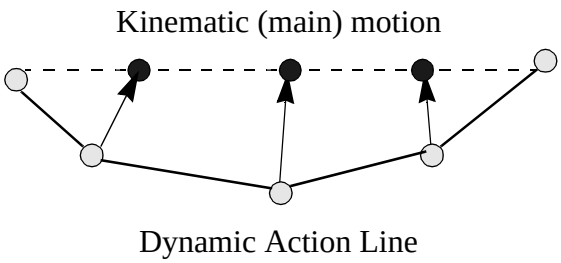


Figure 4.16 Dynamic motion of the action line with restorative forces indicated by arrows.

A somewhat similar concept, named *profile-curve driven dynamics*, was put forward by Ng-Thow-Hing but used for a different goal [Ng-Thow-Hing 00]. In his system, a muscle deforms under the action of an embedded 3D mass-spring-damper network. So, quick transient motions may result in an exaggerated creep behavior because forces propagate along the springs from the attachment nodes, thus requiring several steps before masses in the fleshy portion are set in motion. To combat this excessive delay, each node is anchored to its projection on the axial curve of the muscle. However, it is unclear how the axial curve is controlled.

Adjusting the oscillations of the dynamic action line requires some manual work by the graphics artist. For all that, we do not always get a lifelike motion, even after tweaking the elasticity parameters. In truth, the outcome frequently resembles rubberlike material more than muscle tissues in motion. We attempt to alleviate this problem by monitoring and restricting the drift from the main motion.

The problem of controlling the drift bears a resemblance to the problem of collision response

in mechanical systems for which researchers have either used direct correction of position [Baraff 98], of velocity [Volino 95], or constrained accelerations to legal directions only [Witkin 90]. There is no clear agreement as to which method is best. In our specific case though, drift may accumulate over the frames when correcting accelerations via elastic forces. It is therefore preferable to constrain the positions of the nodes directly. The final position of each node is given by linear interpolation between the dynamic position and the kinematic position.

4.6 Muscle Activation

Simulating isotonic contractions is important and yet, isotonic contractions alone hardly render the look of real muscles, as we picture them. Visually, the muscles of the character do bulge when they shorten, but the character does not seem to "tense" its muscles before the motion has actually happened. Put another way, the contractions of the muscles appear passive and not active.

4.6.1 Classification of muscular activity

In reality, most physical exercises require concurrent isotonic and isometric contractions of the muscles, especially when an action is resisted. This happens for instance when flexing the arm while holding a (heavy) object in the hand. The flexion itself is mostly the outcome of the isotonic contraction of the biceps, whereas the load induces an isometric contraction to counteract the effect of gravity.

The classification of muscular activity into isotonic and isometric contractions is an approximation. Strictly speaking, the term *isometric* refers to a condition in which the length of the muscle remains constant, whereas the term *isotonic* applies to a condition in which muscle tension (and therefore force) is kept constant [Kroemer 90]. Hence, our use of the word *isotonic* is not rigorous since the force needed for producing some movement must in fact adjust over time due to the variable velocity of the motion and the changes that occur in the mechanical conditions (pull angles and lever arms) under which the muscle functions.

4.6.2 Muscle tension from the motion kinematics

Our idea behind the simulation of isometric contractions is to impart a more active look to the deformation of the muscles. This differs greatly from the work in biomechanics where models try to predict the force produced by a muscle. Our approach is basically the reverse: Instead of applying the laws of mechanics on the musculo-skeletal system so as to produce a movement, we crudely deduce the level of activation (and so, the force in a way) of the muscles from the kinematics.

The action of each individual muscle has been carefully studied by anatomists [Netter 87] [Thomson 29]. As a consequence, we know remarkably well which muscles must be brought into play in order to accomplish some movement. We reuse this knowledge to semi-automatically construct the activation curves for the muscles according to the motion that is performed.

We associate the muscles to the joints of the skeleton. For example, the biceps of the arm is associated to the elbow, radioulnar and shoulder joints because it flexes the forearm, assists in its supination, and raises the humerus. As a joint may furthermore have several degrees of freedom, the binding is in actuality performed at this level. For example, a muscle that abducts or adducts the upper arm is mapped to the X-axis of the swing component of the shoulder joint, since only this DOF controls the elevation of the limb. The swing-and-twist parameterization of the shoulder joint proves useful in this task since each of its three components maps to a unique direction. We now describe the analysis of a key framed animation.

Let $K = [K_{ij}]$ be a keyframe sequence where K_{ij} denotes the scalar value taken by *i*-th degree of freedom of the skeleton at frame *j*. We substract columns two by two: $\tilde{K} = [K_{ij+1} - K_{ij}]$ in order to distinguish between positive and negative motions e.g. between abduction and adduction. We extract the most significant variation for the *i*-th DOF by tracking sequences of consecutive positive (resp. negative) velocities in the *i*-th line of $\tilde{K}$. Let $\boldsymbol{s}_{jp} = \left[\tilde{K}_{ij} \ \ldots \ \tilde{K}_{ij+p}\right]^T$ denote such a sequence of *p* elements.

The sequence is considered meaningful over time if $p > n$ where *n* is a user-specified threshold. Similarly, we filter out sequences that are kinematically not significant enough: $max(\boldsymbol{s}_{jp}) < threshold$.

As the tension of a muscle is not constant during the course of contraction but is continually decreasing [Gray 00], we associate to a meaningful sequence $\boldsymbol{s}_{jk}$ a linearly decreasing function *f* such that:

$$f(j) = \frac{max(\boldsymbol{s}_{jp})}{max} \text{ and } f(j+p) = 0$$

where *max* is a user-specified value that depends on the keyframe and the muscle considered. For a given muscle, we then assemble the various functions into a global function that represents the level of activation over the whole keyframe sequence.

4.6.3 Discussion

The above method is obviously very crude. A biomechanics module with force prediction would probably achieve much better and accurate results. However, our kinematics approach has the advantage of simplicity and allows the user to easily edit activation curves by manipulating a few high-level parameters (i.e., threshold values). Manual refinement of activation curves is unavoidable in case the character interacts with objects (e.g. lifting a heavy book).

4.7 Numerical Issues

4.7.1 Equation of motion

The mass-spring system that governs the evolution of an action line deforms itself according to the Lagrange equation of motion. The position $\boldsymbol{X_i}$ of the i-th (dynamic) node of the action

line is given by the following second-order differential equation:

$$\frac{d^2\boldsymbol{X_i}}{dt^2} + \gamma\frac{d\boldsymbol{X_i}}{dt} + \boldsymbol{f}_{i-1}^{\boldsymbol{elastic}} + \boldsymbol{f}_{i+1}^{\boldsymbol{elastic}} + \sum_k \boldsymbol{f}_k^{\boldsymbol{forcefield}} = 0 \tag{4.12}$$

The first term accounts for the node acceleration. The mass is omitted as all nodes are assigned a unit mass (see Section 4.4.1). The second term applies a viscous damping to the node. For reasonably small values of γ, the viscous damping increases the system stability and speeds up the convergence towards equilibrium. The last term in Eq. (4.12) includes all the forces that force fields exert on the particle while the third and fourth terms account for the elastic forces produced respectively by the previous and next action line nodes via damped springs. In our case:

$$\boldsymbol{f}_{i+1\to i}^{\boldsymbol{elastic}} = \left[k_e\left(\left\| \boldsymbol{L}_{i,\,i+1} \right\| - \frac{\left\| \boldsymbol{L}_{i,\,i+1}^0 \right\|}{\alpha} \right) - k_d\left(\frac{d\boldsymbol{L}_{i,\,i+1}}{dt} \cdot \frac{\boldsymbol{L}_{i,\,i+1}}{\left\| \boldsymbol{L}_{i,\,i+1} \right\|} \right) \right] \frac{\boldsymbol{L}_{i,\,i+1}}{\left\| \boldsymbol{L}_{i,\,i+1} \right\|}$$

$$\text{with } \boldsymbol{L}_{i,\,i+1} = \boldsymbol{X}_{i+1} - \boldsymbol{X}_i$$

and where k_e and k_d are elasticity and damping constants. Note that the rest length of the spring is divided by a coefficient α such that $\alpha \gg 1$ to avoid compression. We use $\alpha = 10$.

4.7.2 Numerical integration

Eq. (4.12) can effortlessly be transformed into two coupled first-order differential equations by introducing the velocity $\boldsymbol{V}_i$ as an intermediate variable:

$$\begin{bmatrix} \dfrac{d\boldsymbol{X_i}}{dt} \\ \dfrac{d^2\boldsymbol{X_i}}{dt^2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_i \\ -\gamma\boldsymbol{V}_i - \boldsymbol{f}_{i-1}^{\boldsymbol{elastic}} - \boldsymbol{f}_{i+1}^{\boldsymbol{elastic}} - \sum_k \boldsymbol{f}_k^{\boldsymbol{forcefield}} \end{bmatrix} \tag{4.13}$$

Eq. (4.13) being valid for any dynamic node of the action line, we drop the index *i* in the following. The numerical integration of Eq. (4.13) is performed by discretizing the time line into small steps. The differential equation is then integrated over each time step using the derivatives as evolution information. Taking for instance an Euler step of size dt from a given configuration after *n* steps brings us to:

$$\begin{bmatrix} \boldsymbol{X}^{n+1} \\ \boldsymbol{V}^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^n \\ \boldsymbol{V}^n \end{bmatrix} + dt \begin{bmatrix} \boldsymbol{V}^n \\ M^{-1}\boldsymbol{f}^n \end{bmatrix} \tag{4.14}$$

where $\boldsymbol{f}^n$ is the net sum of forces acting on a node at the *n*-th step and *M* is the mass (it is included for generality but is set to the identity matrix in our case). It is important to note that a simulation error accumulates from step to step. Hence, an exaggerated inaccuracy may lead to a numerical divergence of the simulation. The positions of the nodes then take near-infinite

values, which results in a visual “explosion”.

Obviously, the elastic relaxation of an action line should proceed as quickly as possible. On the other hand, the tool becomes useless if the simulation sometimes “explodes” on account of a numerical divergence. Two contradictory goals must therefore be satisfied: speed and stability. In addition, as the system is interactive, the user often changes parameters like the intensity or the position of a force field while an action line is being relaxed. This implies that the mechanical system must also be immune to sudden changes in its parameters.

4.7.3 Techniques for dealing with instability

The propensity to instability of a mechanical system is characterized by its stiffness, which can be defined as the ratio between the highest and the smallest eigenvalues. A system is said to be stiff if this ratio is much greater than one. In mass-spring systems, the stiffness increases with the size of the time step, the rigidity of springs, and rather counteruntuitively, when the spatial discretization gets smaller.

Several methods can be used to prevent a mechanical simulation from exploding. Setting the time step to a very small value is one such example, but, as it may dramatically impact the speed of the simulation, researchers have looked for more efficient solutions.

Runge-Kutta methods increase the number of force evaluations for each integration step, but increase the stability of the system even more. The two most widespread Runge-Kutta methods in computer graphics are the midpoint method (or second-order Runge-Kutta) and the fourth-order Runge-Kutta method. The idea behind these methods is to take several trial steps and use the supplemental information to extrapolate a higher order Taylor expansion to the solution. Rewriting Eq. (4.14) as:

$$\boldsymbol{Y}(t + dt) = \boldsymbol{Y}(t) + dt\boldsymbol{Y}'(t, \boldsymbol{Y}(t)) \tag{4.15}$$

The second-order Runge-Kutta method is then derived as follows:

$$\boldsymbol{K_1} = dt\boldsymbol{Y}'(t, \boldsymbol{Y}(t))$$

$$\boldsymbol{K_2} = dt\boldsymbol{Y}'\left(t + \frac{dt}{2}, \boldsymbol{Y}(t) + \frac{\boldsymbol{K_1}}{2}\right)$$

$$\boldsymbol{Y}(t + dt) = \boldsymbol{Y}(t) + \boldsymbol{K_2} + O(dt^2)$$

Analogously, the fourth-order Runge-Kutta formulation is derived by taking four trial steps [Press 92].

The Euler and Runge-Kutta integration methods are called explicit because they make use of derivatives only at the beginning of the step. *Implicit methods* take a different approach by trying to extrapolate the solution by estimating derivatives at the end of the step. This scheme theoretically grants unconditional stability to the system as it tries to find the future configuration by taking a step “backward” to return to the initial state. Thus, the backward Euler step is writ-

ten as:

$$\begin{bmatrix} \boldsymbol{X}^{n+1} \\ \boldsymbol{V}^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^{n} \\ \boldsymbol{V}^{n} \end{bmatrix} + dt \begin{bmatrix} \boldsymbol{V}^{n+1} \\ M^{-1}\boldsymbol{f}^{n+1} \end{bmatrix} \tag{4.16}$$

Contrary to Eq. (4.14), the force and velocity are now computed at the terminus of the time step. These quantities are, however, unknown and have to be predicted. Using a first-order approximation:

$$\boldsymbol{f}^{n+1} = \boldsymbol{f}^{n} + \frac{\partial \boldsymbol{f}^{n}}{\partial \boldsymbol{X}}(\boldsymbol{X}^{n+1} - \boldsymbol{X}^{\boldsymbol{n}}) + \frac{\partial \boldsymbol{f}^{n}}{\partial \boldsymbol{V}}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^{\boldsymbol{n}}) \tag{4.17}$$

Introducing Eq. (4.17) in the bottom row of Eq. (4.16):

$$\boldsymbol{V}^{n+1} = \boldsymbol{V}^{n} + dtM^{-1}\left(\boldsymbol{f}^{n} + \frac{\partial \boldsymbol{f}^{n}}{\partial \boldsymbol{X}}\boldsymbol{V}^{n+1}dt + \frac{\partial \boldsymbol{f}^{n}}{\partial \boldsymbol{V}}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^{\boldsymbol{n}})\right)$$

Rearranging terms finally yields the following linear system:

$$\left(M - dt\frac{\partial \boldsymbol{f}^{n}}{\partial \boldsymbol{V}} - dt^{2}\frac{\partial \boldsymbol{f}^{n}}{\partial \boldsymbol{X}}\right)\boldsymbol{V}^{n+1} = M\boldsymbol{V}^{n} + dt\boldsymbol{f}^{\boldsymbol{n}} - dt\boldsymbol{V}^{\boldsymbol{n}} \tag{4.18}$$

In computer graphics, the backward Euler integration scheme has been popularized by Baraff and Witkin in the context of clothes simulation [Baraff 98]. Since then, several strategies have been developed to solve (for each integration step) the linear system in Eq. (4.18) that stems from the implicit integration. Baraff and Witkin use a preconditioned conjugate gradient thus exploiting the sparsity of the matrix [Baraff 98]. Similarly, Volino *et al.* resort to a conjugate gradient solver but optimize its computation [Volino 00]. Desbrun *et al.* decompose the linear system produced by surface mass-spring networks into linear and non-linear terms and thus manage to avoid performing explicit matrix inversions [Desbrun 99]. Their method is, however, fairly inaccurate, and is chiefly intended for real-time applications.

More recently, Hauth and Etzmuss [Hauth 01] introduced another class of integration methods called *backwards differential formula* (bdf) to the computer graphics community. Unlike previous methods which advance the simulation from step to step, bdf methods extrapolate a new configuration from several previous states. The number of previous states used gives the order of accuracy of the method. In their paper, Hauth and Etzmuss compare a second-order bdf and a second-order implicit integration method and favor the former.

Finally, modal analysis provides another way to circumvent the instability in mechanical systems. The principle is to break down dynamics into the sum of independent vibration modes. The process is somewhat similar to Fourier analysis: The equations of motion in Cartesian space are transposed into the frequency domain. Pentland *et al.* [Pentland 89] thus replace the second-order partial differential equation for an n-node system with a set of $3n$ independent linear equations. The instability is decreased by ignoring high-frequency modes. Since the frequency is inversely proportional to the amplitude of the deformation, the global deformation of

the object remains the same. Similarly, Witkin and Welsh [Witkin 90] deliberately restrict the range of deformations using Lagrange dynamics so that high frequencies motions (i.e., instability) are eliminated.

4.7.4 Our approach

The implicit integration of our system yields a sparse diagonal banded matrix with at most 3*3 = 9 non-zero elements in a row because a node is connected through springs to the next and previous nodes only [Kass 93]. As the inversion of a diagonal banded matrix can be performed rapidly [Press 92], implicit integration looks like an ideal candidate in our situation. However, an implicit integration requires the evaluation of internal forces at the end of each time step. For this purpose, forces are replaced by an approximation as a first-order Taylor series, which requires the computation of force derivatives with respect to the nodes positions and velocities as shown by Eq. (4.18). Unfortunately, the projection of a point onto a an ellipsoid does not have a closed-form in 3D [Hart 94]. Hence, we cannot easily compute the derivatives of the forces induced by force fields in orthogonal mode (see Section 4.5.2). Still, we could mix explicit and implicit integration schemes [Eberhardt 00] to cope with this problem.

A simple comparison between explicit and implicit integration schemes showed that in the case of a 1D elastic line, the implicit integration of spring forces is not very advantageous. The limit between stability and divergence is displayed in Fig. 4.17 for three explicit and one implicit integration schemes. The results of the study clearly indicate that for time steps in the range of the graphics display refresh rate (24-100Hz), the fourth-order Runge-Kutta method (RK(4)) remains stable for stiffer springs than the implicit Euler scheme. As force fields moreover contribute significantly to the overall stiffness, using an implicit integration scheme is not useful. Another interesting result shown in Fig. 4.17 is that RK(4) is likely to run faster than the midpoint method. A single RK(4) step requires four force evaluations while a midpoint step requires half that number. Since the RK(4) scheme is more than twice as stable as the midpoint scheme, we can infer that RK(4) should perform faster.

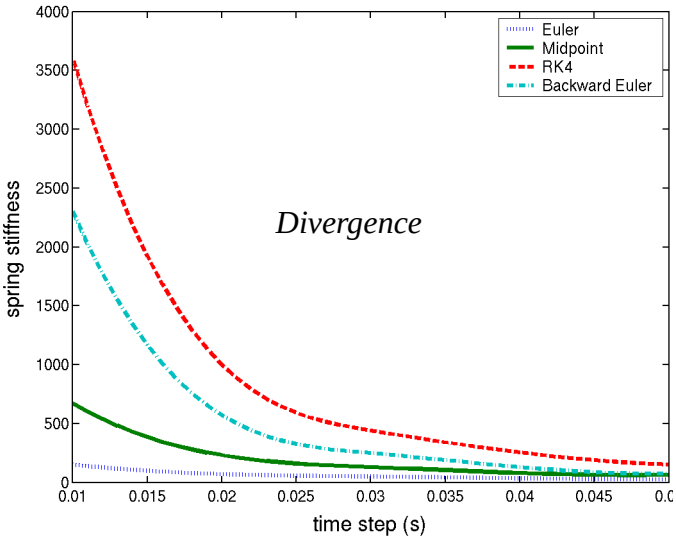


Figure 4.17 Comparative analysis of the stability yielded by four different integration schemes for an elastic polyline with six nodes.

In the end, we rely on an adaptive fourth-order Runge-Kutta scheme to integrate the equations of motion. We also tested an explicit midpoint integration but, in accordance with the results

displayed in Fig. 4.17, the RK(4) method always fared better. Analogously, we tested different methods for adapting the stepsize (note that the time step is independently adapted for each action line). The residual of the integration can be monitored and used for adjusting the time step accordingly. We implemented the adaptive fifth-order Runge-Kutta method described in [Press 92]. It uses an embedded fourth-order formula to estimate the error, the magnitude of which is then used as an indicator for varying the step size. As a simpler alternative, one can locally monitor the kinetic energy of the system [Volino 95]. In our case, the kinetic energy variation of an action line is taken to be the maximal kinetic energy variation of all dynamic nodes. Then, the adaptive stepsize control based on the kinetic energy is similar to the approach in [Baraff 98]. For an action line, we compute at the terminus of an integration step the kinetic energy variation and consider the system unstable if the kinetic energy increases more than twofold over the time step (an exception is made when the action line is completely motionless at the start of the step). In case of instability, we divide the time step by two and integrate the equation of motion again. This process is iterated until the integration is completed without sudden changes of the kinetic energy. Now, if we take n successive steps without having to reduce the time step, we try to double it. If the integration repeatedly fails in doing so, we reduce the time step again and increment the number of successful steps n that must be taken before the time step can be increased again.

Method **Keyframe**	Adaptive RK 4 (kinetic energy)	Adaptive RK 5 (residual)
Backhand (190 frames)	1440 / 7.6	2107 / 11.1
Right-hand (230 frames)	1612 / 7.0	2430 / 10.5
Serve (780 frames)	3027 / 3.8	not measured

Table 4.1 Total / per frame timings in seconds for computing the deformations of 20 muscles (32 action lines) on a 250MHz R10000 CPU.

Table 4.1 shows that the simpler method performs faster. The adaptive fifth-order Runge-Kutta method requires two extra force evaluations per step (and a few additional vector operations). As the fourth-order method requires four derivations, we can infer that a single step is at least 50% more expensive for the fifth-order scheme. This estimate is close to the timings given in Table 4.1. In other words, the gain in accuracy and stability does not offset the extra computations.

4.8 Real-Time Deformation Model

Because we approximate muscle deformation by elastic relaxations of 1D action lines, the computational cost of our model is significantly lower than the cost of previously proposed simulation models. For all that, the simultaneous deformation of many muscles cannot be realized in real-time on relatively modest hardware (see Table 4.1). Upon closer inspection, the elastic relaxation of the action lines turns out to be the bottleneck in the system.

4.8.1 Admissible centroid curve

The deformation process can be greatly accelerated by removing the elasticity in the action line. The action line is then used only to define an admissible path along which the muscle components are spread. The specification of the action line is carried out as follows. The attachment (start and end) nodes are first positioned. A certain number of intermediate nodes are then created and placed along the centroid axis joining the two attachment nodes. As each action line node is attached to a specific joint, nodes function as pivot points, thus defining a path that bends around skeletal structures (see Fig. 4.18).

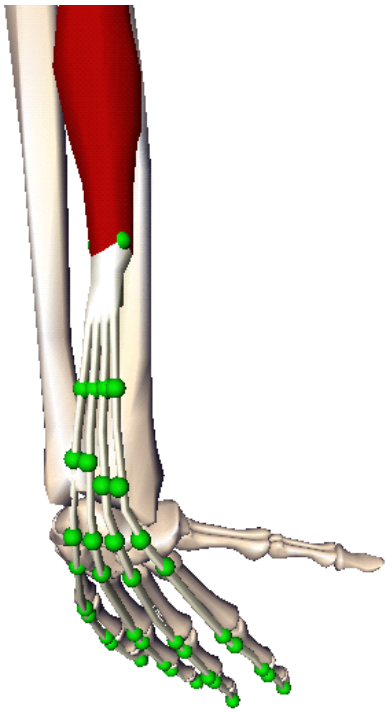


Figure 4.18 Action line nodes (indicated by green spheres) act as pivot points.

Similarly to [Delp 00], we deflect the path by ellipsoidal primitives in some places. For this purpose, we introduce *collision nodes*, which are parameterized by a ratio between 0 and 1 along the segment joining two action line nodes. Hence, their motion solely depends on the two enclosing action line nodes, not on the skeleton joints. Collision nodes are then attracted or repulsed by ellipsoids (hence their name), either to prevent penetration of other anatomical structures or to yield a more accurate path. Collision nodes are particularly useful for avoiding certain artifacts that sometimes occur when nodes get closer to each other during movement. Because collision nodes are initially constrained to remain on a straight line joining two action line nodes, they greatly help to avoid unnatural bends of tendons (see Fig. 4.19).

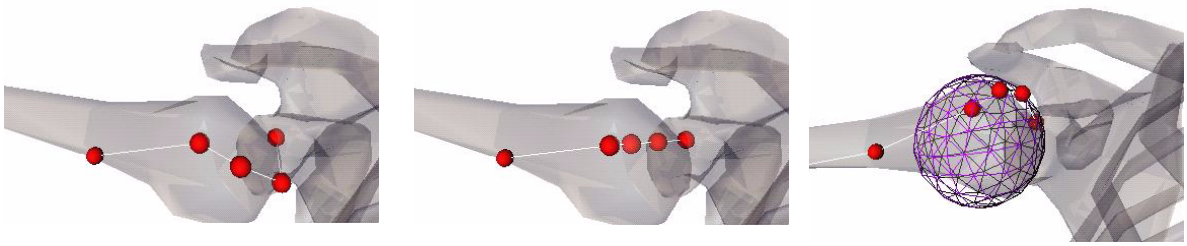


Figure 4.19 Left: compression due to shoulder abduction that results in an unnatural V-shape. Middle: The third and fourth nodes (numbering from the left) become collision nodes and therefore lie on the segment that connects the second and fifth node. Right: An ellipsoidal deflective surface alters the positions of the collision nodes.

We also allow action line nodes to be influenced by joints to a certain degree only. This per-

mits to simulate the variable adherence of muscles to the underlying bones. Fig. 4.20 shows a concrete example of this feature on the one-DOF radioulnar joint. More generally, the swing-and-twist parameterization of three-DOF joints proves useful as a node anchored to a three-DOF joint may follow completely the swing movement but twist less.

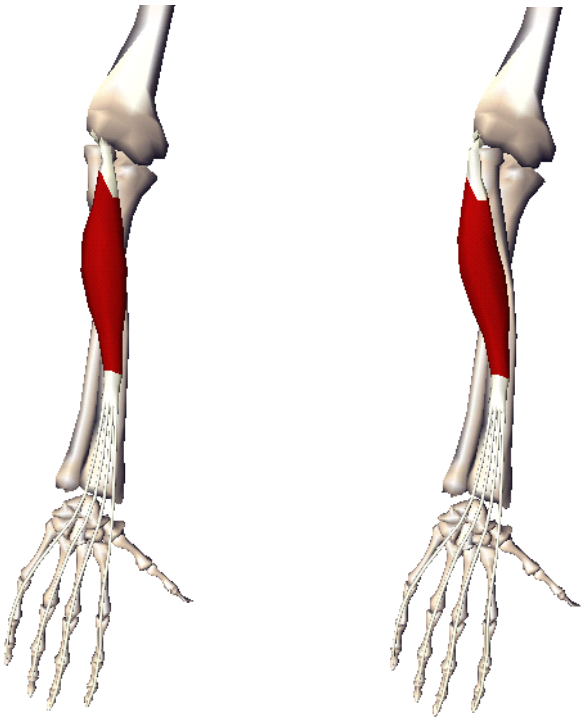


Figure 4.20 Variable adherence of the common extensor muscle to the ulna. Left: no delay. Right: action line nodes rotate less than the ulna, which better reflects the actual anatomy.

Finally, we enable displacements for certain nodes (which we refer to as *mobile nodes*) as a function of the underlying skeleton state. Each mobile node is associated with one DOF of a reference joint. The position $\boldsymbol{X}$ of the mobile node is dynamically determined as follows:

$$\text{If } \theta \in [\theta_0,\theta_{max}], \text{ then } t = \frac{\theta - \theta_0}{\theta_{max} - \theta_0}, \quad \boldsymbol{X} = \boldsymbol{X}_0 + f(t)(\boldsymbol{X}_{max} - \boldsymbol{X}_0)$$

$$\text{If } \theta \in [\theta_{min},\theta_0], \text{ then } t = \frac{\theta - \theta_{min}}{\theta_0 - \theta_{min}}, \quad \boldsymbol{X} = \boldsymbol{X}_0 + f(t)(\boldsymbol{X}_{min} - \boldsymbol{X}_0)$$

where θ denotes the DOF of interest of the reference joint (i.e., an angle or a component of the swing vector as defined by Eq. (3.4)) and where $f(t) \in [0,1]$ is a continuous function. Typical examples are linear and parabolic functions. In summary, the position $\boldsymbol{X}$ of a mobile node is interpolated between the default and extreme state of its reference joint. We distinguish two phases of θ because the muscle deformation may have different behaviors in different directions. We do not allow a mobile node to be simultaneously controlled by several DOFs to avoid the problem of blending between more than two positions.

4.8.2 Mesh deformation

Afterwards, the mesh is anchored to the action lines(s). The length of the tendinous portions and the length of the belly portion are computed in the rest posture. Note that this requires the explicit insertion of action line nodes whose location match the interface between the tendons and the belly.

When the skeleton is animated, we distribute the total elongation of the action line, whether positive or negative, on the belly portions only, which amounts to considering the tendons as totally devoid of elasticity. The effect of this dynamic reparameterization can be seen in Fig. 4.21. When the action line is approximated by a piecewise cubic curve, we numerically compute the curve length by recursive binary subdivision. The subdivision stops when the local curvature is close enough to a straight line. Note that this scheme could easily be extended to approximate nonlinear elasticity.

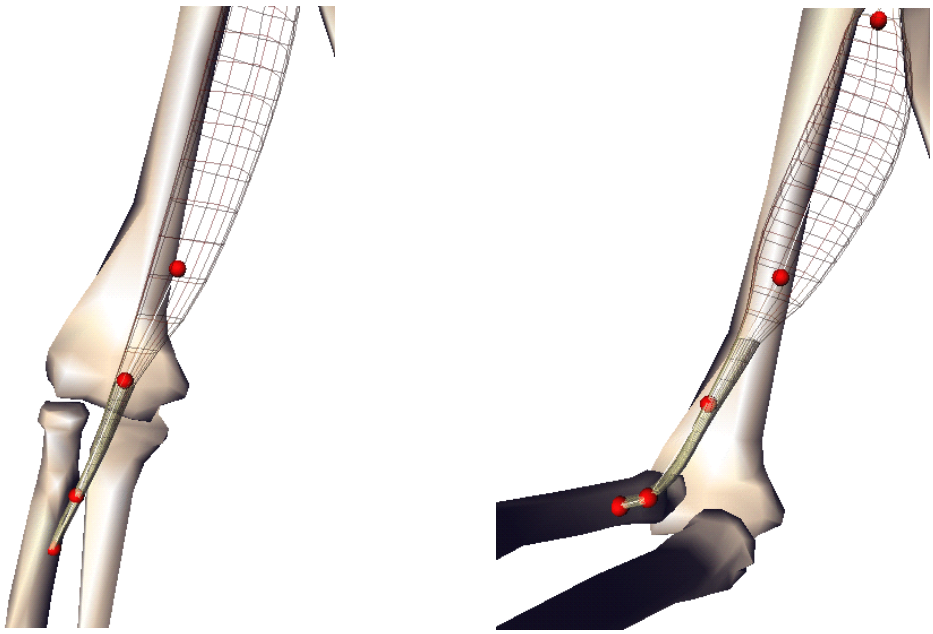


Figure 4.21 Action line nodes are dynamically remapped to preserve initial tendon lengths. Left: the third node is located at the separation between the lower tendon and the belly. Right: the mesh slides over the action line because of the remapping.

4.8.3 Comparison with the elastic action line model

In many instances, the real-time deformation model suffices to produce good-looking muscle deformations. By comparison with the elastic polyline model, the deformations of muscles that only span hinge joints (e.g. the lateral head of the *triceps brachii*) look identical. On the other hand, the deformation of more complex muscles (e.g. the *pectoralis major*) may look better with the more complicated deformation model.

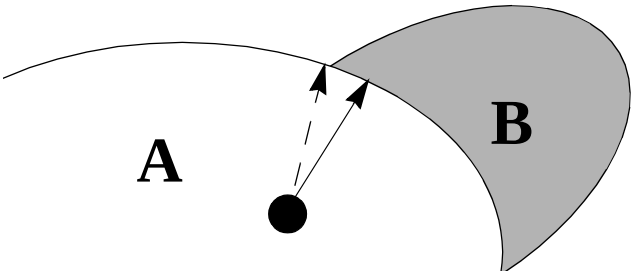


Figure 4.22 Difference between repulsive force fields (dashed arrow) and wrapping surfaces (solid arrow).

One of the differences between the elastic polyline model and the non-elastic model is illustrated in Fig. 4.22. One controls the deflection of an action line node more accurately using an elastic polyline because the node is forced outside of all repulsive force fields after a few iterations. Contrariwise, the node is projected onto the surface of ellipsoid A in the simplified model although the projection violates the repulsive constraint imposed by the other wrapping surface. To remedy this, one might be tempted to carry out several successive projections but it is important to note that the convergence of this scheme is not guaranteed (a possible work-around would be to define the union of the wrapping surfaces as a single implicit surface). More generally, the control of the action line is finer with the elastic model for the following reasons:

- ellipsoids can be combined more easily to form complex force field shapes,
- force fields can also be attractive,
- every force field has a user-specified intensity.

However, modeling time is generally shorter with the real-time deformation model. In addition, as dynamic systems are generally difficult to control because of the many parameters and the indirect control they give, graphics designers are sometimes reluctant to use them and prefer to rely on the simpler deformation model.

4.9 Conclusion

In this chapter, we have shown that muscles represented by general polygonal meshes could easily be parameterized and driven by a set of underlying action lines. We have presented two models for the action lines. The first one is controlled by 1D mass-spring systems, which are relaxed for each animation frame. For the purposes of speed and stability, we have compared different integration schemes and showed that explicit high-order adaptive schemes perform the fastest while ensuring stability. The second action line model is derived from the first one and achieves real-time results by bypassing the elastic relaxations.

Chapter 5

Modeling of the Fat and Skin

We open this chapter with an anatomical analysis of the outer tissues of the human body. We introduce, after, two distinct models for deforming the outer layers. The former reproduces the effects of dynamics on fatty tissues while smoothly deforming the overlying skin. The latter relies on a purely geometric approach and considers the fat and skin as a monolithic layer.

5.1 Physiology

5.1.1 Analysis of fatty tissues

The *fatty tissues* represent the last anatomical structure that creates surface form. They are found either between the skin and the fascia or between deep organs. Fatty tissues in the former location form the *panniculus adiposus* and play an important role on the surface form (certainly underestimated in computer graphics). The skin lies directly on the adipose tissues, which are in turn connected to the densely fibrous fascia.

Fatty tissues can be found in all individuals, whatever their age and constitution, but in various quantities. Thus, fat is significantly more abundant in women and babies, which accounts for the chubbiness of their figures. In all cases, the fat layer plays a prominent role on the form of the buttocks and breast, both for men and women [Richer 81]. From a mechanical point of view, fatty tissues are nonlinearly viscoelastic, do not resist much to tension, and are considered incompressible [Maurel 98].

5.1.2 Skin physiology

The skin is a continuous external sheet that covers the body. It consists of two layers, the *dermis* and the *epidermis*, which are prolonged by the subcutaneous fatty tissues (also called *hypodermis*). In constitutive description, human skin is a non-homogeneous, anisotropic, nonlinear viscoelastic, nearly incompressible material. Its mechanical properties vary with factors such as age, gender, obesity, hydration, disease, etc. Unlike the fat layer, skin resists strongly to stress thus protecting the inner organs from injuries. Skin anisotropy is characterized by prestress lines called Langer's lines (see Fig. 5.1). Langer's cleavage lines are clearly related to the visible crease and wrinkle lines of the skin because the extensibility of the skin is lower

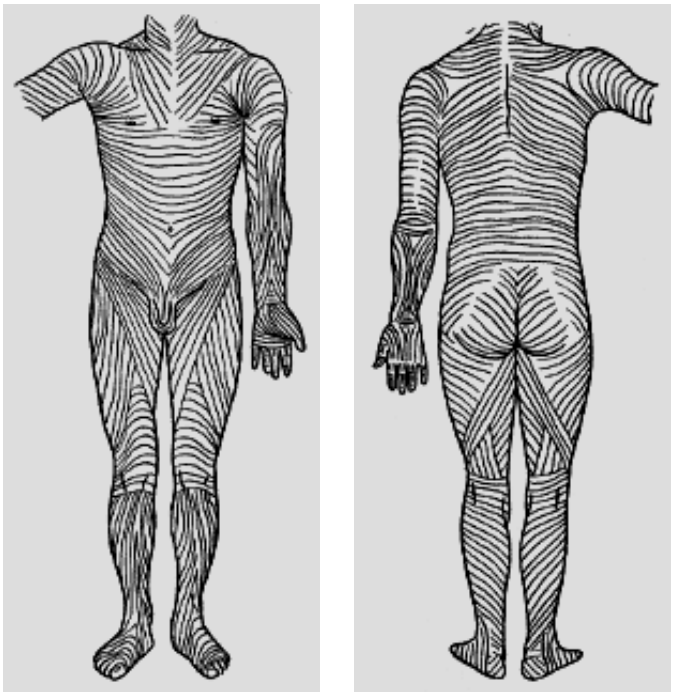


Figure 5.1 Langer's lines characterize the stress lines of the human skin.

along those directions and its stiffness is higher [Maurel 98].

The skin accounts for about 16% of the body weight [Lanir 87]. It has surface area that ranges from 1.5 to 2.0 m^2 in adults. Its thickness varies a great deal depending on the location. The skin of the palms and soles of the feet is thick (up to 6.0 mm) while the skin in the eye lids is very thin (0.2 mm).

5.1.3 Motion and deformation

When the skeleton moves, subcutaneous fatty tissues slide relatively freely over the fascia while the skin clings relatively tightly to the adipose tissues. Put differently, the skin and fat layers appear to move as a whole over the internal tissues.

The mobility of the skin and fat layer depends greatly on the location. In some places, these two layers strongly stick to the internal tissues thus creating permanent furrows and grooves. For example, the strong binding to underlying structures creates wrinkles at the wrist, on the palm, on the inner side of the fingers, under the buttocks, in the armpit and the groin. Other permanent skin wrinkles exist even though their presence is not necessarily due to a strong adherence of the outer layers. These ever-present lines can be seen on the knuckles of the fingers, at the bend of the arm, on the abdomen, at the base of the neck, etc. Note that all these wrinkles are located near joints and become more conspicuous as the articulation is flexed or extended.

Last, the macro-structure of the skin highly depends on the age. Aging causes a reduction of the muscular mass in all individuals. As the skin elasticity diminishes as one gets older, the skin cannot flex enough to recover the gaps induced by the loss of muscle mass. This phenomenon creates age wrinkles, which are particularly noticeable in the face.

5.2 Skin Modeling

There basically exist three ways to model a skin layer in computer graphics. One can design it from scratch or modify an existing mesh in a 3D modeler. Another popular way to create a character mesh is to laser scan a real person/model. This produces a dense mesh that, once cleaned, faithfully represents a human figure. For example, Krishnamurthy *et al.* capture the most meaningful features of a 3D character by fitting B-splines to dense meshes produced by laser scans [Krishnamurthy 96]. The residual of the fitting process is kept in a separate bump texture so that no fine detail is lost. The last method for modeling the skin layer consists in extracting it from underlying components when these exist.

One of the key features of human (and animal) skin is its natural smoothness. To automatically grant a certain smoothness to the rendered skin, mathematical surfaces that exhibit C1 or higher continuity are frequently chosen. For example, Shen and Thalmann model the human skin by fitting B-spline patches through the iso-value of a potential scalar field [Thalmann 96] while Forsey applies a hierarchical B-spline model [Forsey 91]. For generality, however, our approach does not settle on any particular surface type. Hence, what we mean by *skin mesh* in the following is the set of control points from which the geometric representation of the skin is derived. Best results are nonetheless expected to be achieved with hierarchical B-splines or subdivision surfaces.

A generic approach to skin deformation should take into account the different ways for producing a skin. Besides, to be fully functional as a modeling tool, it should also allow the designer to reshape certain areas at will. Like with the muscle layer, we do not actually consider the problem of skin modeling in a static posture bur rather concentrate on the issue of deformation. However, we may still want to provide a rough initial skin model by sampling the underlying layers. The next section describes how this can be done.

5.3 Skin Extraction From Underlying Components

Because the human limbs and trunk are roughly cylindrical, we can cast rays from the skeleton segments in a star-shaped manner and keep the outermost intersection points, as proposed by Shen and Thalmann [Thalmann 96]. However, unlike the afore-mentioned work, this sampling process needs to be performed in one pose posture only. If the pose posture is carefully chosen, most ambiguities concerning the skin regions are automatically resolved. To illustrate, let us consider the shoulder since it is a notoriously difficult area to sample.

Two problems must be faced when sampling the shoulder region. First, muscles belonging to different body parts are often in close proximity because the shoulder connects the arm to the

rest of the body. Hence, if the arm lies by the side, rays cast in the shoulder region are likely to intercept muscles belonging to the upper arm, upper torso and back, although some of these intersections will obviously be wrong. The solution proposed in [Thalmann 96] consists in grouping the primitives into body parts, so that intersections occurring too far from the origin of the ray can be culled. Grouping muscles is not needed in our system since we can produce muscle deformations for any posture and therefore choose a non-ambiguous pose posture. A typical pose posture is to abduct the shoulder by 90 degrees so as to outstretch the arm. Another problem that turns up is the concavity of the armpit. Once again, this problem can easily be solved since the shoulder must be sampled in one pose only. A quick, simple solution consists in interactively increasing the sampling rate in the region of the concavity until a correct profile curve of the armpit is obtained.

Sampling the muscle and skeleton layers produces a mesh that corresponds to the *écorché*[1]. It is easy to adjust the ray-casting process to simulate the deposit of fatty tissues on the muscles. For this purpose, intersection points along the rays can be displaced by a user-specified offset. Typically, the offset remains under 1 cm in the limbs and is set to higher values for the trunk. The resulting mesh provides a rough approximation of the character shape. Mesh smoothing and a few touch-ups in a modeling software are needed to get a visually-pleasing final result.

5.4 Deformation of Fatty Tissues

All organic tissues of the body undergo the effects of inertia and gravity. Thus fatty tissues hang somewhat loosely under the pull of gravity. So do muscles, but to a lesser extent, because they never are in a fully relaxed state [Richer 81]. On these grounds, we choose to concentrate dynamics effects mainly in the fat layer.

5.4.1 Mechanical model

Debunne *et al.* recently achieved real-time performance for simple virtual surgery simulation [Debunne 99] using a linear elasticity model derived from the Lamé equation. We adopt the same model, with minor variations, for simulating the fat layer. Although the model assumes geometric and physical linearity (i.e., infinitesimal deformations), it is acceptable for a computer graphics purpose. Note that this would be inadmissible for biomechanics as one generally considers that small deformations do not exceed a few percent. We briefly recall hereafter the theoretical background. For more details, refer to any good textbook on continuum dynamics.

The Lamé equation for a homogeneous, isotropic, linearly elastic material is given by:

$$\rho \boldsymbol{a} = \mu \Delta \boldsymbol{d} + (\mu + \lambda) \nabla (div(\boldsymbol{d})) + \boldsymbol{f_{ext}} \tag{5.1}$$

where ρ is the mass density, μ and λ are the *Lamé constants* that determine the behavior of the material (they can equivalently be expressed in terms of the more widespread *Young modu-*

1. An *écorché*, a common term in fine arts, denotes the three-dimensional representation of the human body with the layers of fat and skin removed.

lus and *Poisson coefficient*), $\boldsymbol{a}$ represents the acceleration of a small element of matter, $\boldsymbol{d}$ denotes its displacement, and $\boldsymbol{f_{ext}}$ is the set of external forces acting on it. Another way of looking at Eq. (5.1) is as the propagation of a longitudinal wave and a transversal one, whose velocities are respectively:

$$c_1 = \sqrt{\frac{\mu}{\rho}} \text{ and } c_2 = \sqrt{\frac{\lambda + 2\mu}{\rho}} \tag{5.2}$$

5.4.2 Numerical integration

As usual, we discretize Eq. (5.1) both in time and space. The time step for integrating Eq. (5.1) is chosen using Eq. (5.2) so that the waves propagate without missing discretization nodes, a well-known cause of divergence in numerical simulations. On account of numerical inaccuracies, the simulation may still diverge, however. Hence, we looked into the various techniques for circumventing mechanical instability as previously described in “Techniques for dealing with instability” on page 82.

Modal analysis and other techniques that lower the number of DOFs in the mechanical system were quickly ruled out. These methods get rid of high-frequency deformation modes, thus reducing the system’s instability. However, in doing so, they also remove most of the subtle effects that dynamics simulations produce. We then contemplated integrating Eq. (5.1) implicitly. Implicit integration schemes produce a large linear system that has to be solved for every integration step. Much work has already been devoted to accelerating the inversion of the sparse matrix resulting from the linear system (refer to Section 4.7.3 for more details). However, these works targeted elastic surfaces while we deal with elastic volumes. As the Hessian matrix is of size $9n^2$ for n discretization nodes, its inversion cannot be executed quickly. The sparsity of the matrix can nonetheless be exploited by a conjugate gradient solver with a reduced number of iterations [Baraff 98] [Volino 00]. However, Volino *et al.* [Volino 01] recently showed that a low number of iterations may create, in counterpart, severe artifacts in the dynamic simulation, thus reducing the benefit of using large time steps. Consequently, we resorted to adaptive high-order explicit schemes again.

Finally, artificial viscosity is also included in the model since we want to model a viscoelastic material. Besides, it grants additional stability to the simulation by up to an order of magnitude.

5.4.3 Approximation of differential operators

The differential operators in Eq. (5.1), i.e. the Laplacian and the gradient of divergence, are approximated by algebraic difference operators as in [Debunne 99]. One of the interesting features of the approximations is that they are intended to be resolution-independent. Hence, the model remains relatively well-behaved even at the surface where the discretization tends to be irregular.

5.4.4 Meshing and anchoring

The simulation takes as input any number of triangular meshes that make up an inner border

(representing the *écorché*) and an external border (the skin). The enclosed volume represents the fatty tissues. We begin by uniformly meshing the fat volume, with the voxel size under user control. Then, innermost nodes are automatically anchored to the inner border by projection (see Fig. 5.2). We parameterize the projection of a node onto the inner mesh by the barycentric coordinates of the triangle to which it belongs. In this way, providing the inner border keeps a fixed topology, the anchor moves naturally as the surface of the *écorché* deforms itself.

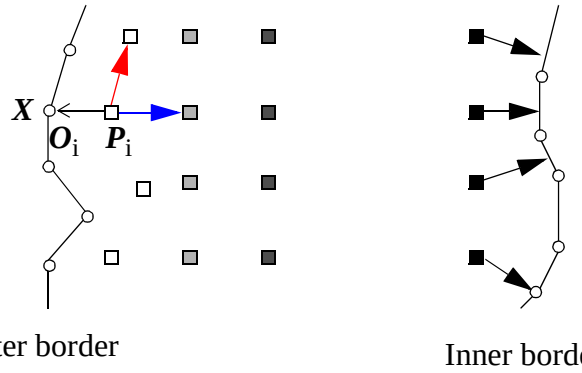


Figure 5.2 Anchoring skin vertices and deepest nodes.

Skin vertices, too, are automatically anchored to the k closest surface nodes (in our implementation, $k = 4$) using local frames for the attachments. This guarantees a smooth appearance of the skin, whatever the body's orientation and the amplitude of the local deformations. The position $\boldsymbol{X}$ of a skin vertex is expressed with respect to an attachment node $\boldsymbol{P_i}$ as follows:

$$\boldsymbol{X} = \boldsymbol{P}_i + M_i \boldsymbol{O}_i$$

The local frame M_i is a 3x3 matrix formed by taking the normalized vectors joining $\boldsymbol{P_i}$ to three other neighboring nodes (see Fig. 5.2). The deviation of the matrix from the singularity is measured for every possible triplet and we select the least singular. The position $\boldsymbol{X}$ of a skin vertex is ultimately given by the weighted average of the k local coordinates:

$$\boldsymbol{X} = \frac{\sum_{i \in \, links}^{k} w_i (\boldsymbol{P_i} + M_i \boldsymbol{O_i})}{\sum_{i \in \, links} w_i} \text{ where } w_i = \frac{1}{\|\boldsymbol{O_i}\|} \tag{5.3}$$

In places where the fat layer is too thin, skin vertices are automatically anchored to the inner border and neighboring voxels.

5.4.5 Comparison with other models

The final mechanical model exhibits a better robustness than the traditional mass-spring-damper network and an augmented realism. The computational cost is similar since deformations are computed using the displacements of neighboring nodes only. One of the key differences is that volume conservation is inherent to the formulation of the Lamé equation. In

theory, volume variation decreases to zero as λ goes to infinity. In practice, however, setting λ so that $\lambda > 100\mu$ suffices to keep volume variation below an acceptable threshold (see Fig. 5.3). On the contrary, volume conservation in mass-spring systems can only be enforced through the application of explicit, time-consuming constraints: soft constraints [Lee 95] partially enforce volume conservation while hard constraints that use Baumgarte stabilization [Witkin 90] preserve the global volume more accurately. In both cases, however, the enforcement of the constraint takes time as it "propagates" to inner nodes one iteration at a time. Furthermore, its application may dramatically alter the shape of the object, sometimes beyond recognition.

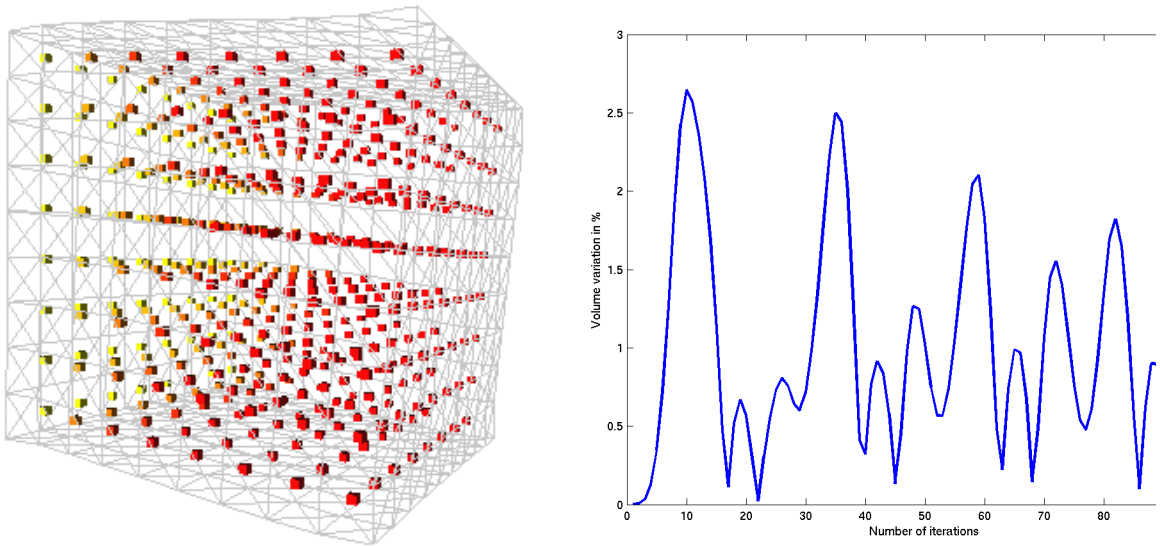


Figure 5.3 Volume variation upon deformation with our model. Left: A cube of matter is discretized into 512 nodes. The left-most nodes are constrained while the remaining nodes are subjected to gravity. The deformation of each node is encoded in pseudo-color. Right: volume variation over time with $\lambda = 150\mu$. The decay is due to the viscosity in the model.

In terms of realism and speed, the chosen model is closer to local explicit finite element models as those in [Cotin 99b] or [Debunne 01]. It must be noted, though, that explicit finite element models allow complex boundary shapes to be described by fewer elements than our finite difference model.

5.5 Skin Deformation

The mechanical model presented in the previous section allows to simulate inertially-induced oscillations and viscosity effects. Nevertheless, biological tissues rapidly converge to equilibrium under external forces, so it is hard to observe a creep behavior or inertia effects in many regions of the body [Hirota 01]. Furthermore, the mechanical model for deforming fatty tissues is computationally rather expensive since it works in 3D. Hence, we restrict its application to

areas where fatty tissues are most abundant e.g. the breast. In all the remaining zones, a faster geometric algorithm can be used for producing skin deformations.

When the *écorché* deforms itself under the influence of skeletal motion, we suggest to deform the skin mesh according to the following multi-step procedure:

- Rough vertex positioning on the surface of the limb or trunk.
- Displacement along the normal to the surface so as to simulate the action of the musculo-skeletal system.
- Creation of flexion wrinkles.

The first step functions to simulate the more or less strong adherence of the outer layers to the material beneath. In the second stage, the skin is sculpted by the underlying muscles and bones. In the last one, wrinkles could be created at the surface according to the joint flexion angles.

We now detail the first two steps. We have not simulated flexion wrinkles but since these form at fixed locations on the body in direct relation with the flexion/extension angles of joints, we feel that their simulation does not pose a major problem as long as medical accuracy is not needed.

5.5.1 Vertex positioning

Before applying muscle-induced deformations to the skin layer, we displace each skin vertex based on the skeletal posture. The goal of this step is two-fold:

- To ensure a smooth skin appearance including smooth deformation upon joint flexion
- To simulate the variable adherence of the outer layers to the material beneath

Beauty spots, hairs and other natural skin markers are quite useful for empirically determining skin motion along various directions. A close observation of their motion unveals the anisotropy of the skin. In particular, it is useful to decompose limb motion into a swing motion and an axial rotation. The swing-twist parameterization we use for 3-DOF joints does exactly that, so we do not need to expressly split 3-DOF joints into two sub-joints for separately handling each sort of rotational motion.

We now introduce a surface deformation model related to contour deformation and, to a lesser extent, to skinning. The skinning algorithm computes the position of a vertex as a linear combination:

$$\boldsymbol{P} = \sum_{i=1}^{n} w_i M_i \boldsymbol{P_i} \quad \text{with} \quad \sum_{i=1}^{n} w_i = 1 \tag{5.4}$$

where the orientations M_i are computed from the joint coordinate systems. The main flaw of Eq. (5.4) lies in the undesirable introduction of non-rotational terms in the deformation. This results in limbs that may shrink when twisted (see Fig. 5.4).

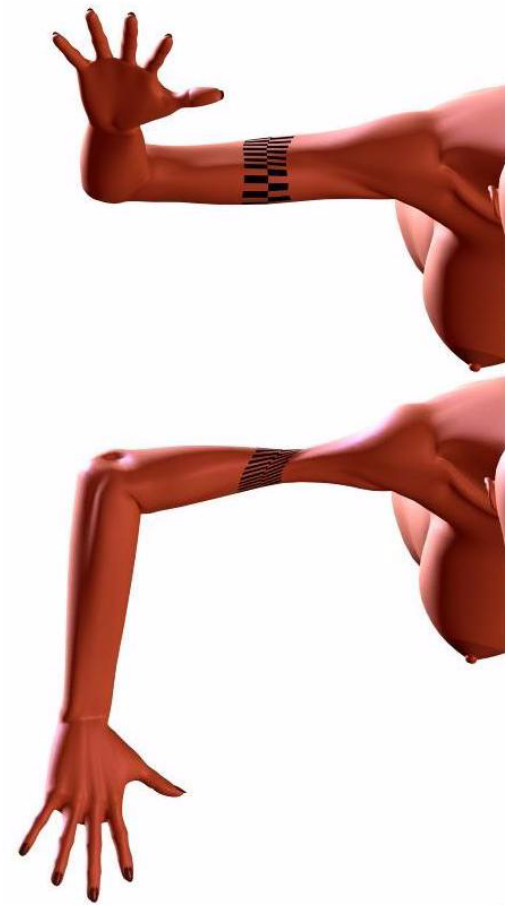


Figure 5.4 Unwanted scaling induced by the skinning algorithm (model courtesy of Thierry Michellod).

We could also deform the skin mesh by using the deformation scheme introduced for muscles (see “Muscle mesh” on page 69) provided that the orientations between which we interpolate do not differ by π or more. Thus, for a vertex $\boldsymbol{X}$ between two joints located at $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ and oriented by M_1 and M_2, the deformation can be expressed by the following equation:

$$\boldsymbol{X} = \begin{bmatrix} slerp(w, M_1, M_2) & 0 \\ w\boldsymbol{O_1} + (1-w)\boldsymbol{O_2} & 1 \end{bmatrix} \bar{\boldsymbol{X}}_{\boldsymbol{local}} \qquad (5.5)$$

In most places, using two joints is enough but in some regions, more joints may be needed to achieve better deformations. The arm provides such an example due to the slight offset between the radioulnar and elbow joints. Hence, skin vertices in the upper arm (resp. forearm) could be parameterized by the shoulder-radioulnar-elbow (resp. wrist-radioulnar-elbow) triangle. However, the *slerp* binary operator does not scale up easily to an *n*-ary version. A simple solution consists in uplifiting the rotational part of the original skinning algorithm into quaternion space. This is done by transforming every skin vertex into a local coordinate system constructed from the neighboring joints. The orientation is then found by linear interpolation of the quaternions that express the orientations at the *n* neighboring joints:

$$\boldsymbol{q} = \sum_{i=1}^{n} \lambda_i \boldsymbol{q_i} \text{ with } \sum_{i=1}^{n} \lambda_i = 1 \qquad (5.6)$$

A linear interpolation between two (resp. three) unit quaternions produces a straight line (resp. triangle) in 4D quaternion space. Obviously, the resulting geometry in R^4 dips below the surface of the unit sphere. Yet, since all quaternions on the 4D line through the origin correspond to the same rotation, the geometry can be projected back onto the unit sphere without altering the rotation. This is equivalently performed by renormalizing the quaternion resulting from the linear combination. However, the angular velocity does not remain constant[1] because the geometry takes a “shortcut” below the surface of the unit sphere [Dam 98]. Yet, this is not a real problem because angular acceleration can be compensated by carefully choosing the

1. This is the textbook classic Phong shading glitch, which generates anomalies in the shading when the normals are interpolated linearly and renormalized *post facto*.

weights λ_i.

5.5.2 Musculo-skeletal-induced deformation

Because of the fat layer, the skin mesh is usually positioned some distance away from the underlying components in the rest posture. We initially compute the signed distance between the skin mesh and the underlying layers. For each skin vertex, we trace a ray along the normal and compute the closest intersection with the bones and muscles (see Fig. 5.5). Note that because the skin mesh may not be extracted from the inner layers, it may penetrate the muscles and bones in some locations. Hence, the distance must be signed.

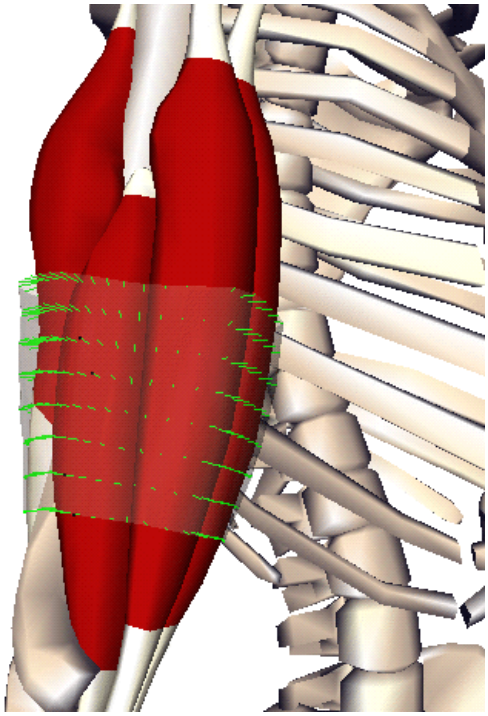


Figure 5.5 Rays are cast from the skin mesh to estimate the thickness of the fat and skin. Black dots indicate skin vertices too far away from the material beneath.

During subsequent animation, we try to maintain the initial distance from every skin vertex to the material beneath by casting a ray along the normal in the same way. Basically, the distance we try to maintain represents the thickness of the skin and fat layers combined. Recall that the skin and fat roughly behave as a monolithic layer as far as their motion is concerned. This implies that their combined thickness does not vary much over time (unless a normal stress is applied onto the surface e.g. when two body parts press against each other). Therefore, it is justified to try and preserve an initial distance throughout the animation.

In order to keep the ray-casting process within a reasonable time, we use hierarchical oriented bounding boxes for the bones since these are undeformable objects. Furthermore, each muscle is enclosed in a hierarchical axis-aligned bounding box that is recomputed anew whenever the muscle is deformed. This remains beneficial so long as the skin mesh is much denser than the muscle meshes.

Another way of looking at the set of distances is as a height field on a special manifold. Filtering this height field in the spatial and temporal domain is needed to smooth the skin mesh. In a first stage, we apply a spatial median filter so as to discard outliers in the height map. Unlike smoothing filters which restrict the frequencies of a signal, a median filter serves to detect and remove discontinuities. We choose a kernel that includes first and second-order neighbors because our skin mesh is fairly dense (to allow the bulging of tendons). Additionally, since the skin mesh may not be regular, the filter kernel is weighted. We compute the weight of any

neighbor using the inverse of its distance (along the edges) to the considered skin vertex. We then filter the height field in the temporal domain using the previous and next animation frames. Finally, the degree of smoothness of the skin mesh can be increased by a final smoothing filter, or alternately, cubic B-spline blending or a mesh subdivivision.

Our approach could be applied to other muscle representations. A similar approach with a metaball formulation is indeed employed in [Leclerq 01]. Unlike this previous work, however, we do not project skin vertices back onto the muscle layer but rather maintain an initial distance that represents the fat thickness. Various filters help to preserve the mesh smoothness.

5.6 Discussion

When the bone and muscle layers are explicitly modeled, deformation of the skin and fat layers is simulated in most previous works by anchoring skin vertices to underlying components via damped springs [Lee 95] [Ng-Thow-Hing 00] [Turner 93] [Wilhelms 97a] [Wilhelms 97b]. While this may be suitable for approximating the motion of animal skin for it is quite loose, it is less so in case of human skin. Besides, the volume of the fatty tissues may locally undergo signification changes unless explicit, time-consuming constraints are specified and enforced. Last but not least, the elasticity of the outer layers is strongly anisotropic. It follows that fat springs are not really appropriate for simulating fatty tissues. An improvement would consist in varying the stiffness of the fat springs as a function of the position on the body (although none of the works cited above do so). This, however, would have to be also coupled with anisotropic springs to minimize volume change[1].

Our approach solves some of the above issues in a simple fashion. The skinning procedure functions to ensure the variable adherence of the outer layers through a series of weights, whose specification is a standard procedure in computer animation [Alias 01]. The ray-casting process that follows insures a crude volume conservation. It is also independent of the muscle representation, as shown by the related work in [Leclerq 01].

1. A simple, elegant way to simulate anisotropic elasticity using volume mass-spring systems is exposed by Bourguignon and Cani in [Bourguignon 00].

Chapter 6

Results

In this chapter, we show the main results of this work. We first present our joint editor, then our interactive muscle deformation modeler, named *MuscleBuilder,* and finally show some examples of fat and skin deformation.

6.1 H-Anim Joint Editor

6.1.1 Architecture and functionalities

The joint editor is an independent graphical application that allows to specify, for each joint, the number of DOFs, the relative orientation via the anatomical matrix, the canonical position via the offset matrix and the limits. Technically, the application is based on a C++ library whose first version was developed by Paolo Baerlocher. The graphical user interface (GUI) is based on the portable FLTK toolkit while the visualization of the bones and stick figure is done using OpenInventor, a 3D hierarchy manager and viewer that is available on all major platforms.

The editor includes a certain number of features, some of which were suggested by the two end-users, Mireille Clavien and Stéphanie Noverraz. The main features are:

- Graphical manipulation of the frame that represents the anatomical/offset matrix. Additional menus allow to align any axis of the frame with the child or parent segment.
- Definition of joint limits e.g. spherical polygons.
- Copy/paste/mirror joints.
- Edit postures via three sliders that control either global Euler angles or the joint's internal parameters. Postures can be saved and reloaded at will. Keyframe animations can also be played to check limits more easily.

The output of the joint editor is stored in two files. The first one is the usual VRML file associated with every H-Anim character. It contains the hierarchy of joints, segments and sites and their respective positions in space. The information concerning the joints that go beyond the H-Anim standard are saved in the second file in ASCII form. This file specifies, for each joint, the

type, the anatomical and offset matrices and the limits.

6.1.2 Example of joint limits

Fig. 6.1 shows the concrete application of the shoulder joint limits. An invalid direction of the upper arm is projected back onto the spherical polygon while the outward twist is clamped to the maximal value defined by the outward twist height field.

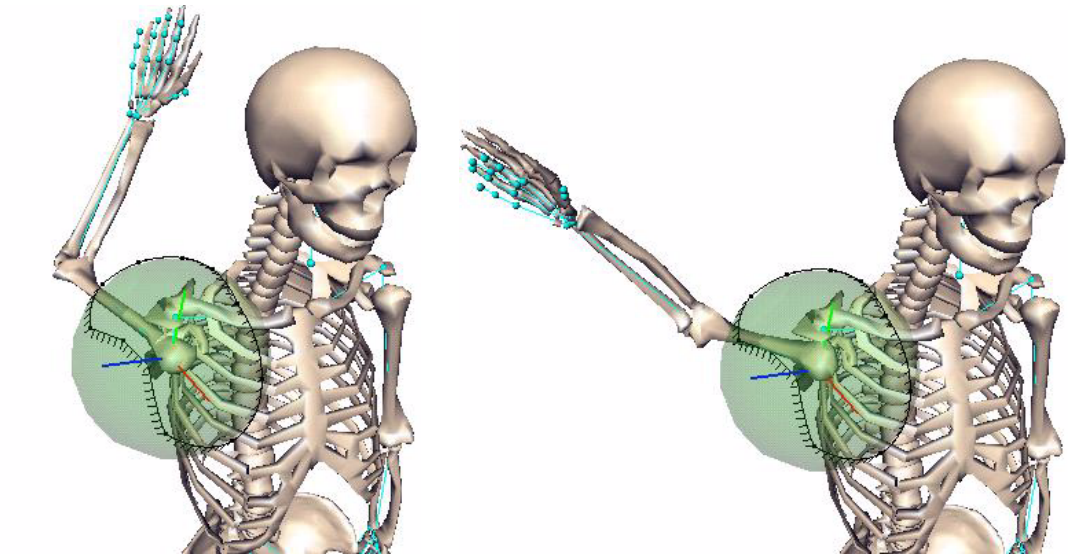


Figure 6.1 Shoulder joint limits. Left: No limits. Right: The current posture is restricted to the admissible space delimited by the spherical polygon and the twist height fields.

6.2 MuscleBuilder

6.2.1 Architecture and functionalities

MuscleBuilder is a an application that consists of a graphical user interface based on FLTK, a viewer based on Inventor and a deformation library written in the C++ language. The GUI was improved thanks to the many suggestions of the two end-users, Nicolas Elsig and Mireille Clavien. An example view of the GUI is shown in Fig. 6.2. The GUI is split into four main tabs: action line management, control of ellispsoidal primitives, mesh loading and skin deformation. The various muscle models are saved in an ASCII file with a simple, human-readable syntax. The final rendering of any deformation can be performed using Maya [Alias 01] through an associated C++ plug-in that interfaces the deformation library.

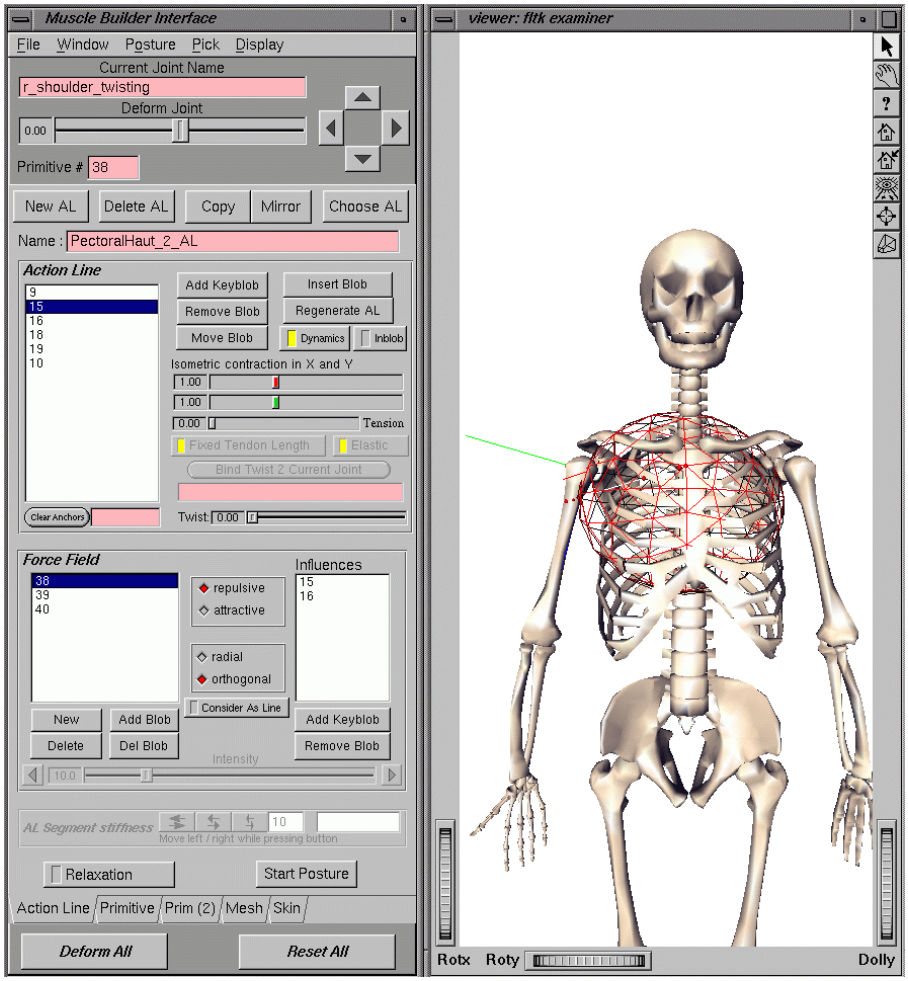


Figure 6.2 Interface of *MuscleBuilder*

6.2.2 Deformation of selected muscles

The biceps is modeled by two action lines, one for each belly. Fig. 6.3 shows an effect that is difficult to render with previous muscle models.The biceps is automatically pulled down when the ulna is rotated because the lower tendon wraps around the bone.

The abdominal muscle is modeled by two action lines (four lines when considering both the left and right parts). Transverse folds are automatically rendered when the backbone is flexed whereas, inversely, the muscle shape flattens when the column is extended, as in reality.

6.2.3 Examples of keyframe animation

We show three series of stills taken from keyframe animations rendered in Maya. In the first animation, the right leg of the character is raised forward (hip flexion), flexed (knee flexion), then drawn away (hip abduction), and finally twisted (hip twisting) while outstretched. The second keyframe is a captured backhand tennis motion. The elbow is flexed and rotated

Figure 6.3 The biceps is pulled down when the ulna is rotated because the lower tendon wraps around the bone.

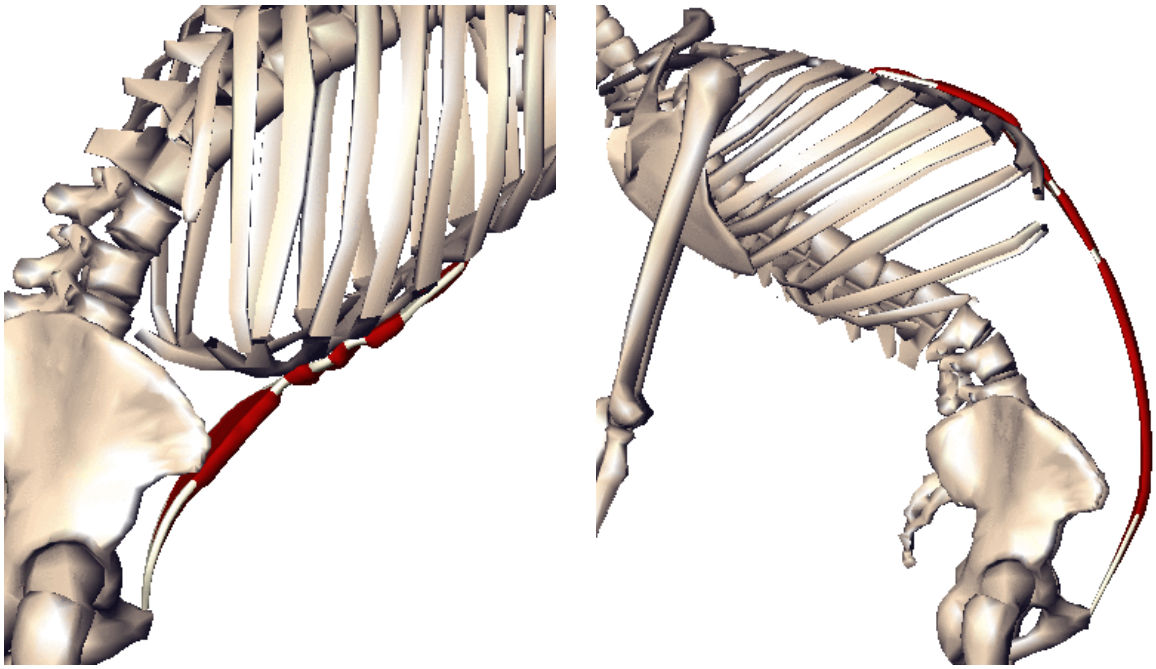


Figure 6.4 Deformation of the abdominal muscle. The shape automatically folds or flattens itself depending on the posture of the vertebral column.

inwards before the shot and brought forward and upwards while the elbow is extended after.

Figure 6.5 Leg motion. Upper left: hip flexion and knee flexion. Upper right and lower left: Hip is moreover abducted (moved away from the pelvis). Lower right: Inward rotation

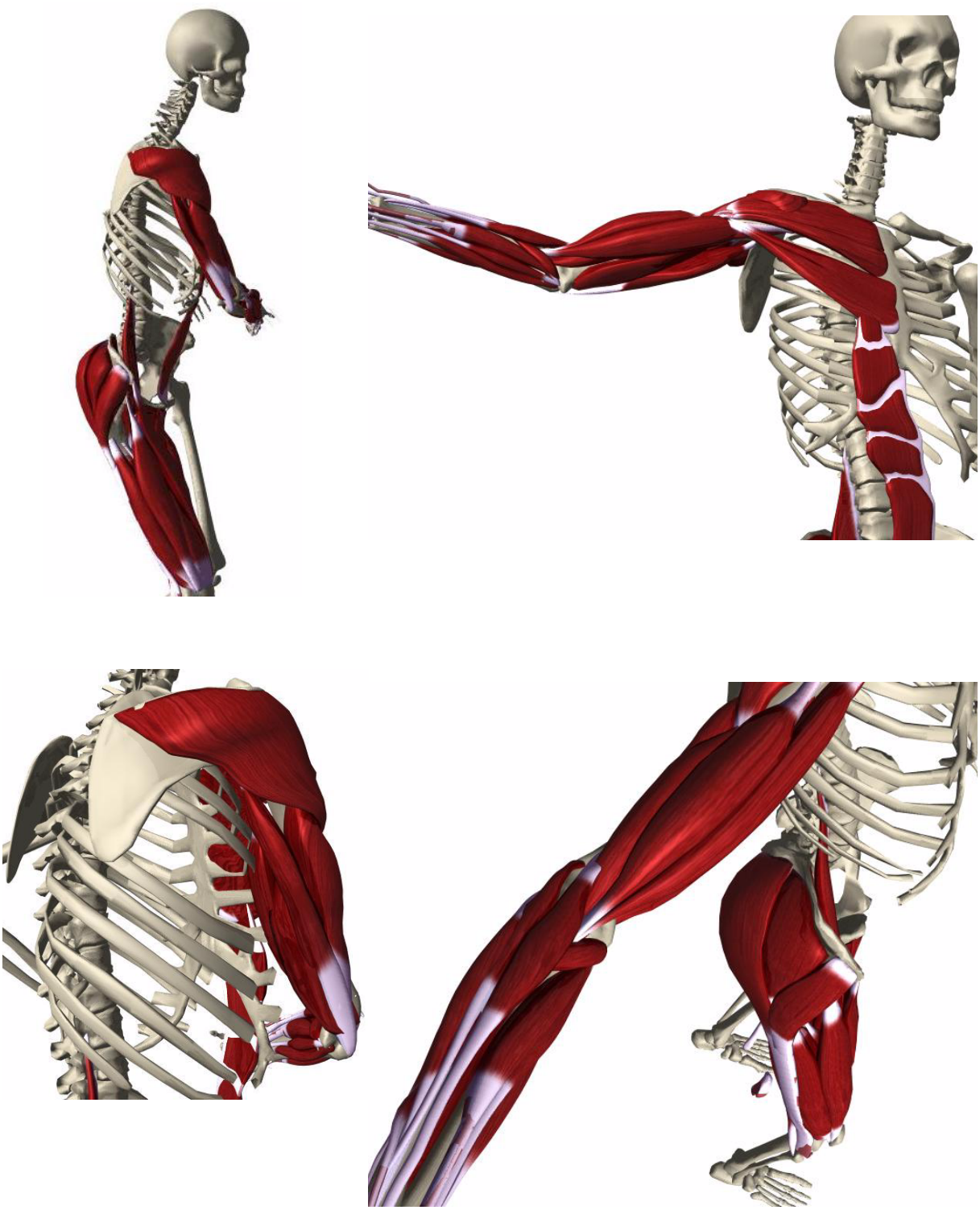


Figure 6.6 Backhand in tennis. Left: preparing the shot. Right: the ball has been hit.

Figure 6.7 Showing its muscles off.

The animations highlight two flaws in the skeleton layer: a “flying” scapula (see Fig. 6.6, upper right picture) and a rib cage that dissociates in some postures (see the abdominal and pectoral muscles that penetrate the ribs but do not penetrate the sternum in Fig. 6.7). Correcting these defects is left as future work.

6.2.4 Reusability of deformation models

As the human body exhibits a great symmetry, we provide an option for mirroring deformation models with respect to the sagittal plane. For this purpose, we only need to be able to symmetrize ellipsoidal primitives with respect to the sagittal plane. Furthermore, the action lines, which naturally parameterize the muscle, allow an easy, uniform scaling of the outer shape. Hence, one could easily apply a global scaling to the belly portions of the muscles in order to transform a skinny character into a more muscular person (or the reverse). At the same time, the action lines must be slightly displaced so as to avoid either the unwanted creation of gaps when scaling down, or important muscle interpenetration when scaling up. Preliminary results show that this simple approach already produces great results.

6.3 Deformation of the Female Breast

All parameters of the simulation are under the active control of the user via a GUI. Volume preservation of the fat and skin layers is naturally enforced for the Lamé parameters can directly encode it. Fig. 6.8 shows a simulation of a female breast subjected to gravity using 904 voxels out of which 254 are anchored to the inner border. The skin mesh contains approximately two thousand vertices and is intentionally shown untextured. On an Octane workstation with a R10000 processor, the voxelization takes a dozen second while the simulation reaches equilibrium in about two minutes that correspond to approximately one second of actual animation. The time step can be set as high as 0.001s without loss of stability. Constructing the local frames and updating the positions of the skin vertices using Eq. (5.3) takes a marginal share of the total simulation time.

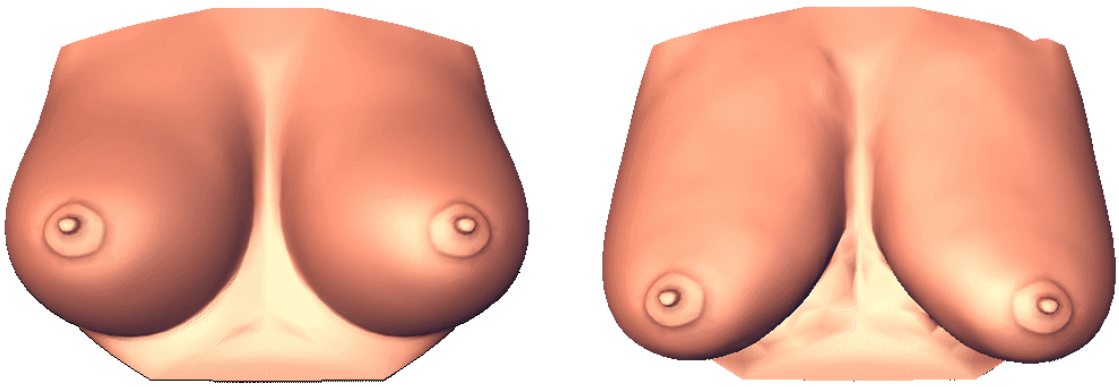


Figure 6.8 Simulation of the female breast subjected to gravity. Volume preservation is achieved by setting $\lambda = 100\mu$.

Introducing a variable voxel size would help to better approximate complex shapes. This would not introduce instability because the approximations of the differential operators in Eq. (5.1) can handle some irregularity [Debunne 99].

6.4 Examples of Skin Deformation

We show in Fig. 6.9 examples of surface deformation of the skin in which upper (resp. lower) arm vertices are controlled by the shoulder (resp. wrist), radioulnar and elbow joints.

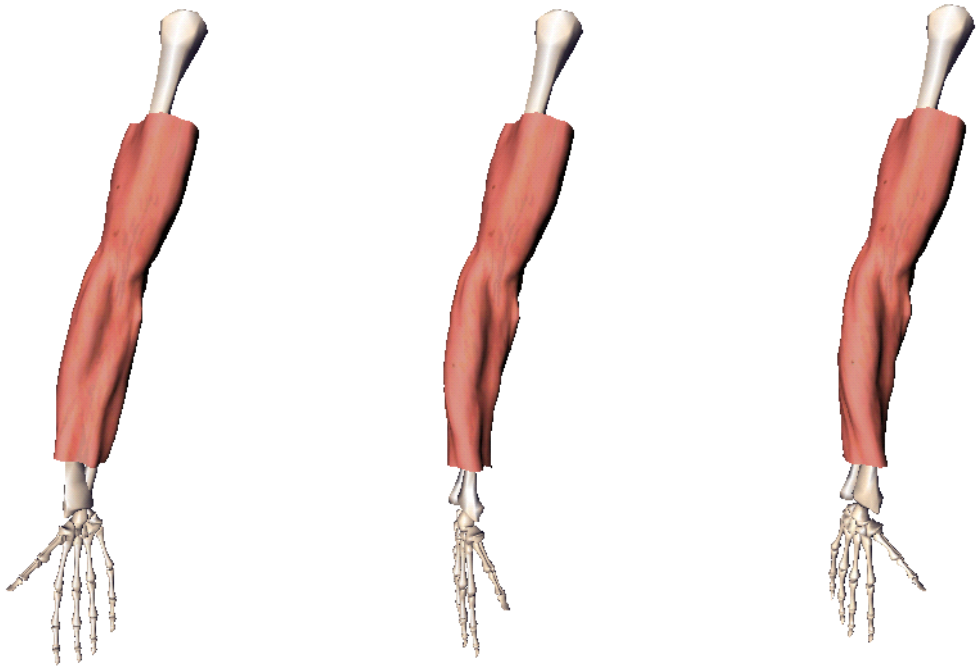


Figure 6.9 Linear interpolation of three quaternions during forearm twisting.

We finally show in Fig. 6.10 the effects of the muscle layer on the skin. In this animation, the elbow is flexed, which results in a contraction of the muscles of the upper arm, which in turns scultps the overlying skin.

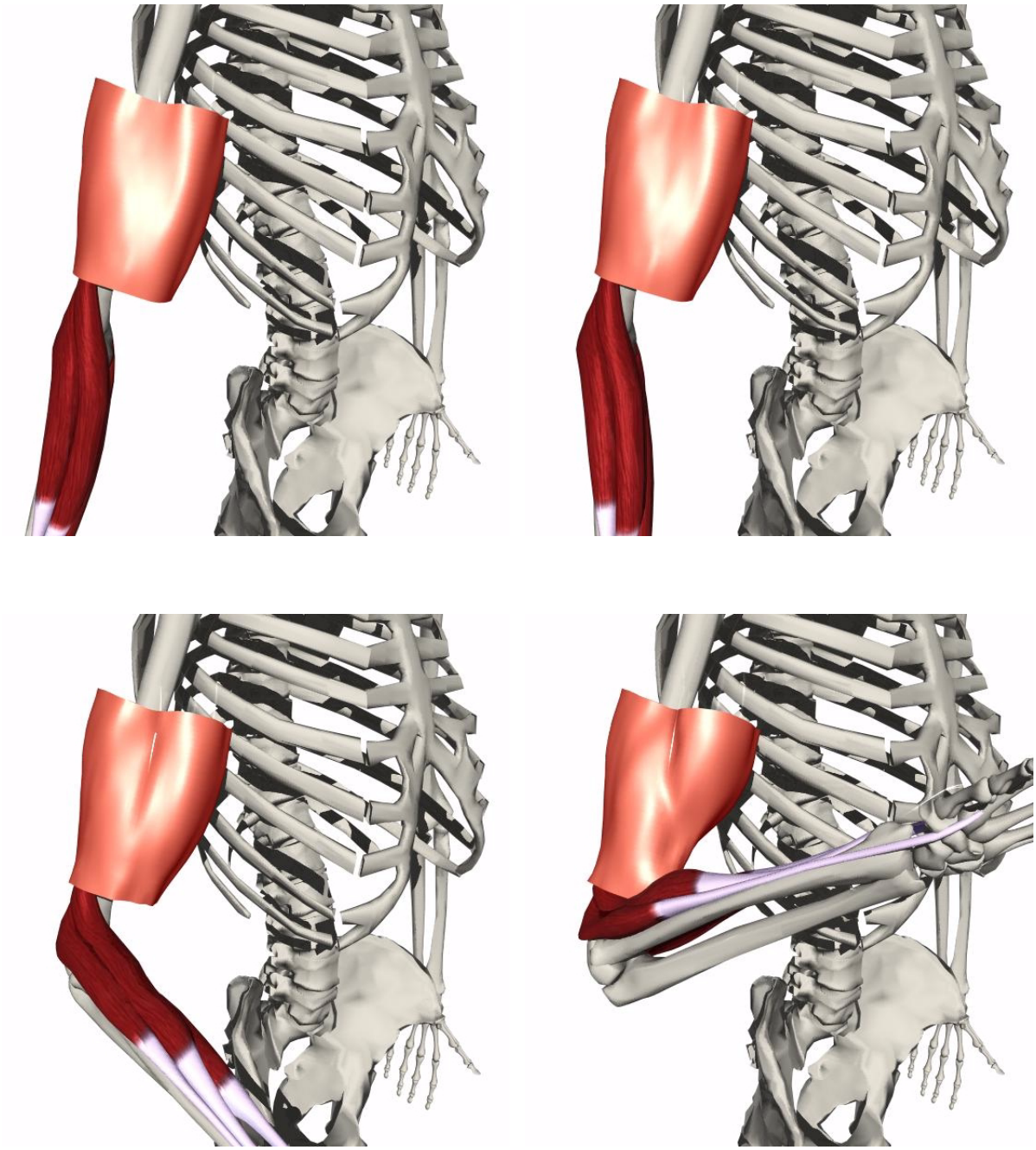


Figure 6.10 Muscle-induced skin deformation (upper arm muscles remain transparent to improve the visibility).

Chapter 7

Conclusion

7.1 Contributions

Our main contributions are the following:

- We have proposed a new multi-layered model founded on anatomy for producing realistic skin deformation in reasonable time.
- We have presented four joint models that approximate well the real human joints. Our ball-and-socket model allows to accurately express coupled limits for the swing motion and the axial rotation.
- We have exposed the benefits of the swing-and-twist parameterization for three-rotational DOF joints.
- We have introduced a sound, general, fast muscle model that reasonably approximates the human biology. The deformation of the muscle surface is completely driven by the muscle lines of action. Our approach unifies biomechanics and computer graphics since the action line can be used for computing both the muscle force and the deformation.
- Our muscle model facilitates the global parameterization of the character because the action lines act as a skeleton for the muscle shape. So, one could relatively easily transform a muscular character into a skinnier one or the reverse. Preliminary results show that a trivial scaling of the muscles bellies via a slider already produces great results.
- We have proposed a geometric algorithm for deforming the skin that can work with different muscle and skeleton models. Additionally, the body mesh retains a fixed topology during animation, which has tremendous benefits for other applications.

The main flaw of this work is the increased storage/communication cost of the model since all layers are explicitly represented.

7.2 Future topics

To conclude, we suggest a few directions for future research, starting from the innermost layer. As human body modeling is an infinitely complex topic, the present list is by no means

exhaustive.

- Animating the shoulder complex is a recurrent problem in computer animation. One of the difficulties lies in the motion of the scapula that is constrained by muscles to slide over the rib cage. Except for [Maurel 00] where a semi-automatic model based on inverse kinematics is proposed, the motion of the scapula is usually defined procedurally. As an alternative, one could try and use our muscle model to automatically constrain the scapula.
- We only handle the coupling of joint limits within a joint. Coupling of limits between different joints, as for example between the hip and the knee, should also be considered.
- The rib cage is a complex structure in the sense that certain ribs attach to the sternum by elastic bands of tissue thus allowing a certain degree of deformation. This behavior is not reflected in our model because each rib is anchored to a unique vertebral joint.
- Muscle deformation should be tailored to take into account the various pennations (orientation of muscle fibers with respect to the tendinous extremities) as there is evidence that the pennation directly influences the deformation [Lemos 01].
- Because action lines naturally parameterize muscles, the muscle layer could be fitted to different skin meshes more or less automatically. However, the speed of the fitting process is likely to be undermined by the use of polygonal meshes for representing the muscles. Hence, it would be worth investigating alternative representations like implicit surfaces for which a continuous distance function can be quickly evaluated.
- It would be most interesting to couple our geometric deformation system of the muscle layer with a biomechanics module for computing the forces needed to move the bones. The action lines could be directly used for determining the force and, thus, the level of activation of each muscle during movement.
- We have not dealt with the problem of collision detection and response between different body parts. At the bend of the elbow and the knee, the skin on either side of the joint often presses against each other when the joint is flexed enough.
- The simulation of people of varying age and corpulence is challenging. For example, simulating rolls of flesh is likely to call for new dynamics models to cope with the issues of collision and contact surfaces. Likewise, Langer’s lines, which characterize the anisotropic elasticity of skin and fat, would need a more thorough investigation.
- Veins contribute to the surface form to a small degree in many instances. However, their contribution can be more substantial is some areas. In the forearms, they form an elaborate network that can clearly be seen. Gravity also tends to reveal them (e.g. when the arm is kept raised) and should be taken into account.

Glossary

- **Abduction**

The action that draws a limb away from the median axis of the body e.g. abducting the shoulder raises the (upper) arm. The opposite movement is called adduction.

- **Adduction**

The action that brings a limb back (or even beyond) the midline of the body. The opposite movement is called abduction.

- **Anterior**

In front; towards the front.

- **Distal**

Away from the point of attachment; When comparing two locations in the body, the distal one is the farther from the center of the body.

- **Extension**

An unbending movement around a joint in a limb that increases the angle between the bones that meet at the joint; A backward raising of the arm or leg when extending the shoulder or hip. The opposite movement is called flexion.

- **Flexion**

A bending movement around a joint in a limb that decreases the angle between the bones that meet at the joint; A forward raising of the arm or leg when flexing the shoulder or hip. The opposite movement is called extension.

- **Posterior**

Behind; towards the rear.

- **Pronation**

Rotation of the hand and forearm so that the palm faces backwards or downwards. The opposite movement is called supination.

- **Proximal**

Next to the point of attachment; When comparing two locations in the body, the proximal one is the closer from the center of the body.

- **Radioulnar**

The radioulnar joint is a pivot joint that allows pronation and supination of the forearm. Proximally, the head of the radius rotates in a notch in the ulna, while distally, the radius rotates about the head of the ulna.

- **Sagittal**

Relating to the medial plane that divides the body into two symmetrical parts.

- **Supination**

Rotation of the hand and forearm so that the palm faces forward or upward and the radius lies parallel to the ulna. The opposite movement is called pronation.

- **Transverse**

Made at right angles to the anterior-posterior axis of the body.

Bibliography

[Alias 01] Maya Alias/Wavefront, "**Maya User Manual**", 2001.

[Ananova 00] Ananova, http://www.ananova.com.

[Badler 79] N.I. Badler, J. O'Rourke, H. Toltzis, **"A Spherical Representation of a Human Body for Visualizing Movement."** Proceedings of the IEEE LXVII/ 10 (October 1979), pp.1397-1403.

[Badler 93] N.I. Badler, C.B. Phillips, B.L. Webber, "**Simulating Humans, Computer Graphics Animation and Control**", New York, Oxford University Press, 1993.

[Baerlocher 01] P. Baerlocher, **"Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures",** Ph.D. Thesis, EPFL, April 2001.

[Baraff 98] D. Baraff, A. Witkin, **"Large Steps in Cloth Simulation"**, Computer Graphics (SIGGRAPH '98 Proceedings), pp.43-53.

[Beylot 96] P. Beylot, P. Gingins, P. Kalra, N. Thalmann, W. Maurel, D. Thalmann, J. Fasel, **"3D Interactive topological modeling using Visible Human Dataset"** Computer Graphics Forum (Eurographics '96 Proc.), 15(3), pp. 33-45, 1996.

[Blinn 82] J. Blinn, **"A Generalization of Algebraic Surface Drawing"**, *Transactions on Graphics*, Vol.1, pp.235-256, 1982.

[Bloomenthal 90] J. Bloomenthal, **"Calculation Of Reference Frames Along A Space Curve"**, In *Graphics Gems*, Andrew S. Glassner Ed., Academic Press, 1990.

[Bloomenthal 91] J. Bloomenthal, K. Shoemake, **"Convolution Surfaces"**, SIGGRAPH 91, pp.251-256.

[Bloomenthal 93] J. Bloomenthal, **"Hand Crafted"**, Siggraph Course Notes 25, 1993.

[Bloomenthal 97] J. Bloomenthal, editor. **"Introduction to Implicit Surfaces"**, Morgan Kaufmann, July 1997.

[Boissieux 00] L. Boissieux, G. Kiss, N. Magnenat-Thalmann, P. Kalra, "**Simulation of Skin Aging and Wrinkles with Cosmetics Insight"**, Computer Animation and Simulation '00, Interlake, August 2000, pp 15-27.

[Boulic 91] R. Boulic, O. Renaut, **"3D Hierarchies for Animation"**, *New Trends in Animation and Visualization*, JohnWiley and Sons, Chichester, UK, pp. 59-78, 1991.

[Bourguignon 00] D. Bourguignon, M-P. Cani, **"Controlling Anisotropy in Mass-Spring Systems**", Computer Animation and Simulation 2000, Interlaken, August 2000.

[Bro-Nielsen 96] M. Bro-Nielsen, S. Cotin, **"Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation"**, Computer Graphics Forum (Eurographics '96 Proceedings), 1996, pp.C57-C66.

[Cani-Gascuel 97] M.P. Cani-Gascuel, M. Desbrun, **"Animation of Deformable Models Using Implicit Surfaces"**, IEEE Transactions on Visualization and Computer Graphics, Vol. 3, No 1, mars 1997.

[Cani-Gascuel 98] M.P. Cani-Gascuel, **"Layered Deformable Models with Implicit Surfaces"**, Graphics Interface '98, Vancouver, Canada, 1998.

[Capin 95] T. Capin, I. Pandzic, N. Magnenat-Thalmann, D. Thalmann, **"Virtual Humans for Representing Participants in Immersive Virtual Environments"**, Proc. FIVE '95 Conference, 1995, London, UK.

[Carr 01] J. Carr, R. Beatson, J. Cherrie T. Mitchell, W. Fright, B. McCallum and T. Evans, **"Reconstruction and Representation of 3D Objects with Radial Basis Functions",** SIGGRAPH 2001, pp. 67-76, 2001.

[Chadwick 89] J. Chadwick, D. Haumann, R. Parent, **"Layered construction for deformable animated characters"**, Computer Graphics (SIGGRAPH '89 Proceedings), pp.243-252.

[Chen 92] D. Chen, D. Zeltzer, **"Pump it up: Computer animation of a biomechanically based model of muscle using the Finite Element Method"**, Computer Graphics (SIGGRAPH '92 Proceedings), pp.89-98.

[Cotin 99a] S. Cotin, H. Delingette, N. Ayache, **"Real-Time elastic Deformations of Soft Tissues For Sugery Simulation"**. IEEE Transactions on Visualization and Computer Graphics, **5**(1), pp. 62-73, 1999.

[Cotin 99b] S. Cotin, H. Delingette, N. Ayache, **"A Hybrid Elastic Model allowing Real-Time Cutting, Deformations and Force-Feddback for Surgery Training and Simulation"**, Proceedings of Computer Animation '99, Geneva, 1999.

[Craig 89] J.J. Craig, "**Introduction to Robotics: mechanics and control**", 2nd edition, Addison-Wesley, 1989.

[Dam 98] E. Dam, M. Koch, M. Lillholm, **"Quaternions, Interpolation and Animation"**, Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, July 1998.

[Desbrun 99] M. Desbrun, P. Shröder, A. Barr, **"Interactive Animation of Structured Deformable Objects"**, Proceedings of Graphics Interface,1999, pp. 1-8.

[Debunne 99] G. Debunne, M. Desbrun, A. Barr, M-P. Cani, **"Interactive multiresoution animation of deformable models"**, Computer Animation and Simulation' 99, Milano, September 1999.

[Debunne 01] G. Debunne, M. Desbrun, M-P. Cani, A. Barr, **"Dynamic Real-Time Deformations using Space and Time Adaptive Sampling"**, Computer Graphics (SIGGRAPH '01 Proceedings).

[Delp 00] S.L. Delp, J.P. Loan, **"A Computational Framework for Simulating and Analyzing Human and Animal Movement"**, IEEE Computing in Science & Engineering, Vol. 2, N. 5, Sept-Oct 2000, pp. 46-55

[DeRose 98] T. DeRose, M. Kass, T. Truong, **"Subdivision Surfaces in Character Animation"**, Computer Grpahics (SIGGRAPH '98 Proceedings), pp. 85-94, 1998.

[Dow 93] E. Dow, S. Semwal, **"A Framework for Modeling the Human Muscle and Bones Shapes"**, New Advances in Computer Aided Design & Computer Graphics, Zesheng Tang Eds., International Academic Publishers, pp. 110-113, 1993.

[Eberhardt 00] B. Eberhardt, O. Etzmuss, M. Hauth, **"Implicit-explicit Schemes for Fast Animation with Particles Systems",** Eurographics workshop on Computer Animation and Simulation, Interlaken, Switzerland, 2000.

[Engin 89] A.E. Engin, S.T. Tuemer, "**Three dimensional kinematic modelling of human shoulder complex - Part I: Physical model and determination of joint sinus cones**", *Journal of Biomechanical Engineering*, Vol. 111, pp. 107 - 112, 1989.

[Extempo 99] http://www.extempo.com/

[Farin 90] G. Farin, **"Curves and Surfaces for computer aided geometric design. A practical guide"**, Academic Press Limited, San Diego, USA, 2nd edition.

[Forsey 91] D. Forsey **"A Surface Model for Skeleton-Based Character Animation"**, Proc. Second Eurographics Workshop on Animation and Simulation (1991), pp. 155-170.

[Frisken 00] S. Frisken, R. Perry, A. Rockwood, T. Jones, **"Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics"**, Proc. Siggraph'00, pp.249-254, 2000.

[Frisken 01] S. Frisken, R. Perry, **"A Computationally Efficient Framework for Modeling Soft Body Impact"**, Siggraph'01 Conference Abstracts, 2001.

[Fung 81] Y.C. Fung **"Biomechanics: Mechanical Properties of Living Tissues"**, Springer-Verlag, 1981.

[Gascuel 91] M-P. Gascuel, A. Verroust, C. Puech, **"A Modeling System for Complex Deformable Bodies Suited to Animation and Collision Processing"**, The Journal of Visualization and Computer Animation,Vol. 2, pp.82-91,1991.

[Gascuel 93] M-P. Gascuel, **"An Implicit Formulation for Precise Contact Modeling between Flexible Solids"**, Computer Graphics (SIGGRAPH '93 Proceedings), pp. 313-320.

[Gourret 89] J.P. Gourret, N. Magnenat-Thalmann, D. Thalmann, **"Simulation of Object and Human Skin Deformations in a Grasping Task"**, Computer Graphics (SIGGRAPH '89 Proceedings), pp. 21-30.

[Grassia 98] F.S. Grassia, "**Practical Parameterization of Rotations using the Exponential Map**", *Journal of Graphics Tools*, Vol. 3, No. 3, pp. 29 - 48, 1998.

[Gray 00] H. Gray **"Anatomy of the Human Body"**, 20th edition. Philadelphia: Lea & Febiger, 2000. v.2.

[Grosso 89] M.R. Grosso, R.D. Quach, N.I. Badler, "**Anthropometry for Computer Animated Human Figures**", State-of-the-art in Computer Animation, *Proceedings of Computer Animation '89*, pp. 83 - 96, 1989.

[HAnim 98] **"Specification for a Standard Humanoid"**, Version 1.1., http://h-anim.org/Specifications/H-Anim1.1/

[Hanson 98] A.J. Hanson, **"Quaternion gauss maps and optimal framings of curves and surfaces"**, October 1998. Indiana University, Computer Science Department Technical Report 518 (October, 1998).

[Hart 94] J.C. Hart. **"Distance to an Ellipsoid"**. In *Graphics Gems* IV, Academic Press, 1994, pp. 113-120.

[Hauth 01] M. Hauth, O. Etzmuss, **"A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods"**, Proceedings of Eurographics, 2001.

[Henne 90] M. Henne, **"A Constraint-Based Skin Model For Human Figure Animation"**, Master's Thesis, University of California, Santa Cruz, June 1990.

[Herbison 78] D. Herbison-Evans. **"Nudes 2: A numeric utility displaying ellipsoid solids"**. SIGGRAPH '78 Conference Proceedings, pp. 354--356, August, 1978.

[Herda 01] L. Herda, R. Urtasun, P. Fua, A. Hanson, **"Automatic Determination of Shoulder Joint Limits using Quaternion Field Boundaries"**, submitted to Special Issue of IJRR, 2001.

[Hirota 01] G. Hirota, S. Fisher, A. State, C. Lee, H. Fuchs, **"An Implicit Finite Element Method for Elastic Solids in Contact"**, Proceedings of Computer Animation 01, Seoul, Korea, 2001.

[Houle 01] J. Houle, P. Poulin, **"Simplification and Real-time Smooth Transitions of Articulated Meshes"**, Proc. of Graphics Interface '01, Canada, 2001.

[James 99] D. James, D. Pai, **"ArtDefo : Accurate Real Time Deformable Objects"**, Computer Graphics (SIGGRAPH '99 Proceedings).

[Jensen 75] R. H. Jensen, 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*, v.8, pp. 103-110, 1975.

[Kalra 98] P. Kalra, N. Magnenat Thalmann, L. Moccozet, G. Sannier, A. Aubel, D. Thalmann, **"Real-Time Animation of Realistic Virtual Humans"**, *Computer Graphics and Applications*, Vol. 18, No. 5, 1988, pp. 42-56.

[Kass 93] M. Kass, Siggraph Course Notes on *Physically-based modeling*, 1993.

[Komatsu 88] K. Komatsu **"Human skin Model capable of natural shape variation"**, The Visual Computer, No. 3, 1988, pp. 265-271.

[Korein 85] J.U. Korein, "**A Geometric Investigation of Reach**", The MIT Press, Cambridge, 1985.

[Krishnamurthy 96] V. Krishnamurthy, M. Levoy, **"Fitting Smooth Surfaces to Dense Polygon Meshes"**, Computer Graphics (SIGGRAPH '96 Proceedings), pp.313-324.

[Kroemer 90] K.H. Kroemer, H.J. Kroemer, K.E. Kroemer-Elbert, **"Engineering Physiology,** Second Edition, Van Nostrand Reinhold, 1990.

[Lander 98] J. Lander, **"Skin Them Bones"**, Game Developer, May 1998, pp 11-16.

[Lanir 87] Y. Lanir, **"Skin Mechanics"**. In *Handbook of Bioengineering*, Ed. R. Skalak, McGraw Hill Book Compagny, 1987.

[Lasseter 87] J. Lasseter, **"Principles of Traditional Animation Applied to 3D Computer Animation"**, Computer Graphics, pp. 35-44, 21:4, July 1987 (SIGGRAPH 87).

[Leclerq 01] A. Leclercq, S. Akkouche, E. Galin, **"Mixing Triangle Meshes and Implicit Surfaces in Character Animation",** Animation and Simulation '01 (12th Eurographics Workshop Proceedings), Manchester, England, 2001, pp.37-47.

[Lee 95] Y. Lee, D. Terzopoulos, K. Waters, **"Realistic Modeling for Facial Animation"**, Computer Graphics (SIGGRAPH '95 Proceedings), pp.55-62.

[Lee 00] J. Lee, **"A Hierarchical Approach to Motion Analysis and Synthesis for Articulated Figures"**, Ph.D. Thesis, Department of Computer Science, KAIST, 2000.

[Lemos 01] R. Lemos, M. Epstein, W. Herzog, B. Wyvill, **"Realistic Skeletal Muscle Deformation using Finite Element Analysis"**, Proc. of SIBGRAPI 2001, Sao Paulo, Brazil, October 2001.

[Lewis 00] J.P. Lewis, M. Cordner, N. Fong, **"Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation"**, Computer Graphics (SIGGRAPH '2000 Proceedings), August 2000.

[Maciel 02] A. Maciel, L. Porcher-Nedel, C. Freitas, **"Anatomy-Based Based Joint Models for Virtual Human Skeletons"**, submitted to Computer Animation 2002.

[Magnenat-Thalmann 87] N. Magnenat-Thalmann, D. Thalmann, **"The Direction of Synthetic Actors in the Film Rendez-vous à Montréal"**, IEEE Computer Graphics and Applications, Vol. 7, No12, 1987, pp. 9-19.

[Magnenat-Thalmann 88] N. Magnenat-Thalmann, R. Laperriere, D. Thalmann, **"Joint-Dependent Local Deformations for Hand Animation and Object Grasping"**, Proceedings of Graphics Interface' 88, pp. 26-33.

[Maurel 98] W. Maurel, Y. Wu, N. Magnenat Thalmann, D. Thalmann, **"Biomechanical Models for Soft Tissue Simulation"**, Springer-Verlag, Berlin/Heidelberg 1998.

[Maurel 00] W. Maurel, D. Thalmann, "**Human Upper Limb Modeling including Scapulo-Thoracic Constraint and Joint Sinus Cones**", *Computers & Graphics*, Pergamon Press, Vol. 24, No. 2, pp. 203 - 218, 2000.

[Moccozet 96] L. Moccozet, **"Hand Modeling and Animation for Virtual Humans"**, PhD Thesis, No 442, University of Geneva, 1996.

[Molet 99] T. Molet, R. Boulic, D. Thalmann, "**Human Motion Capture Driven by Orientation Measurements**", *Presence*, MIT, Vol. 8, No. 2, pp.187 - 203, 1999.

[Monheit 91] G. Monheit, N. Badler, "**A Kinematic Model of the Human Spine and Torso**", *IEEE Computer Graphics & Applications*, Vol. 11, No. 2, pp. 29 - 38, March 1991.

[Moore 88] M. Moore, J. Wilhelms, "**Collision detection and Response for Computer Animation**", *Computer Graphics*, Vol. 22, No. 4, Aug. 1988.

[Nedel 98] L. Nedel, D.Thalmann, **"Real-TimeMuscle Deformations Using Mass-Spring Systems"**, Proc. CGI '98, IEEE Computer Society Press, 1998.

[Netter 87] F. H. Netter, **"Musculoskeletal System Part I: Anatomy, Physiology and Metabolic Disorders",** The Ciba Collection of Medical Illustrations, Volume 8, Ciba-Geigy Corp., 1987.

[Ng-Thow-Hing 00] V. Ng-Thow-Hing, **"Anatomically-Based Models for Physical and Geometric Reconstruction of Humans and Other Animals"**, Ph.D. Thesis, Department of Computer Science, University of Toronto, 2000.

[Pentland 89] A. Pentland, J. Williams, **"Good Vibrations: Modal Dynamics for Graphics and Animation"**, Computer Graphics (SIGGRAPH '89 Proceedings), pp. 215-222.

[Platt 88] J. Platt, A. Barr, **"Constraint Methods for Flexible Models"**, Computer Graphics (SIGGRAPH '88 Proceedings), pp. 279-288

[Porcher 98] L. Porcher-Nedel **"Anatomic Modeling of Human Bodies using Physically-Based Muscle Simulation"**, Ph.D. Thesis, No 1831, EPFL, 1998.

[Press 92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "**Numerical Recipes in C**", Second edition, Cambridge University Press, 1992.

[Richer 81] P. Richer, **"Traité d'anatomie artistique"**, Bibliothèque de l'image, (in French) 1996.

[Saladin 98] K. Saladin, **"Anatomy and Physiology: The Unity of Form and Function"**. McGraw-Hill Compagny, 1998.

[Scheepers 96] F. Scheepers, R. Parent, S. F. May, W. Carlson. **"A Procedural Approach to Modeling and Animating the Skeletal Support of the Upper Limb"**, Technical Report, OSU-ACCAD-1/96/TR1, The Ohio State University, 1996.

[Scheepers 97] F. Scheeppers, R.E. Parent, W.E. Carlson, S.F. May, "**Anatomy-Based Modeling of the Human Musculature**", *Computer Graphics* (SIGGRAPH '97 Proceedings), pp. 163 - 172, 1997.

[Sederberg 86] T. Sederberg, S. Parry, **"Free-From Deformation of Solid Geometric Models"**, Computer Graphics (SIGGRAPH '86 Proceedings), pp.151-160, 1986.

[Shoemake 85] K. Shoemake, **"Animating Rotation with Quaternion Curves"**, Computer Graphics (SIGGRAPH '85 Proceedings), pp. 245-254, 1985.

[Shoemake 94] K. Shoemake, **"Euler Angle Conversion"**, *Graphics Gems IV*, Paul S. Heckbert, ed.,Academic Press Professional, Toronto, 1994.

[Singh 00] K. Singh, E. Kokkevis, **"Skinning Characters using Surface-Oriented Free-Form Deformations"**, Proc. of Graphics Interface 2000, pp. 35-42.

[Sloan 01] P-P. Sloan, C. Rose, M. Cohen, **"Shape by example"**, 2001 Symposium on Interactive 3D Graphics, March, 2001.

[Sun 99] W. Sun, A. Hilton, R. Smith, J. Illingworth, **"Layered Animation of Captured Data"**, Animation and Simulation '99 (10th Eurographics Workshop Proceedings), Milano, Italy, 1999.

[Talbot 98] .D. Talbot, **"Accurate Characterization of Skin Deformations Using Range Data"**, Master's Thesis,University of Toronto, 1998.

[Terzopoulos 87] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, **"Elastically Deformable Models"**, Computer Graphics (SIGGRAPH '87 Proceedings), pp. 205-214.

[Thalmann 96] D. Thalmann, J. Shen, E. Chauvineau, **"Fast Realistic Human Body Deformations for Animation and VR Applications"**, Computer Graphics International'96, Pohang, Korea, June, 1996.

[Thomson 29] A. Thomson **"A handbook of anatomy for art students"**, Dover Publications, Fifth Edition, New York, 1929.

[Turner 93] R. Turner, D. Thalmann, **"The Elastic Surface Layer Model for Animated Character Construction**", Proc. Computer Graphics International '93, Lausanne, Switzerland, Springer-Verlag, Tokyo, pp. 399-412.

[Van Gelder 98] A. Van Gelder, **"Approximate Simulation of Elastic Membranes"**, Journal of Graphics tools, Vol. 3, No 2, 1998, pp. 21-42.

[Volino 94] P. Volino, N. Magnenat-Thalmann, **"Efficient Self-Collision Detection on Smoothly Discretized Surface Animations Using Geometrical Shape Regularity"**, Computer Graphics Forum (Eurographics '94 Proceedings), pp.155-166.

[Volino 95] P. Volino, M. Courchesne, N. Magnenat-Thalmann, **"Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects",** Computer Graphics (SIGGRAPH '95 Proceedings), pp. 137-144.

[Volino 00] P. Volino, N. Magnenat-Thalmann, **"Implementing Fast Cloth Simulation with Collision Response"**, Proceedings of Computer Graphics International 2000.

[Volino 01] P. Volino, N. Magnenat-Thalmann, **"Comparing Efficiency of Integration Methods for Cloth Simulation"**, Proceedings of Computer Graphics International, Hong-Kong, July 2001.

[Wang 98] X.G. Wang, F. Mazet, N. Maia, K. Voinot, J.P. Verriest, M. Fayet, "**Three-dimensional modelling of the motion range of axial rotation of the upper arm**", *Journal of biomechanics*, Vol. 31, No. 10, pp. 899 - 908, 1998.

[Watt 92] A. Watt, M. Watt, "**Advanced Animation and Rendering Techniques**", Addison-Wesley, ACM Press, 1992.

[Wilhelms 97a] J. Wilhelms, **"Animals with Anatomy",** IEEE Computer Graphics And Applications,Vol. 17, No. 3, pp. 22-30, May 1997.

[Wilhelms 97b] J. Wilhelms, A. Van Gelder, **"Anatomically Based Modeling"**, Computer Graphics (Proc. of SIGGRAPH'97), pp. 173-180, 1997.

[Witkin 90] A. Witkin, W. Welch **"Fast Animation and Control of Nonrigid Structures"**, Computer Graphics (SIGGRAPH '90 Proceedings), pp. 243-252.

[Wu 97] Y. Wu, P. Kalra, N. Magnenat-Thalmann, **"Physically-based Wrinkle Simulation & Skin Rendering"**, Computer Animation and Simulation '97 (8th Eurographics Workshop Proceedings), Budapest, Hungary, 1997, pp.69-80.

[Wyvill 98] B. Wyvill, A. Guy, E. Galin, **"The Blob Tree"**, *Journal of Implicit Surfaces*, Vol 3, 1998.

[Yoshomito 92] S. Yoshimito**, "Ballerinas Generated by a Personal Computer"**, The journal of Visualization and Computer Animation, Vol.3, pp.85-90, 1992.

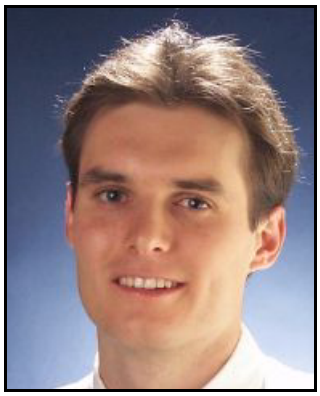
[Zajac 89] F. Zajac, **"Muscle and tendon: properties, models, scaling and application to biomechanics and motor control"**, In CRC Critical Reviews in Biomedical Engineering, v.17, pp. 359-411, CRC Press, 1989.

[Zhu 98] Q-H Zhu, Y. Chen, A. Kaufman, **"Real-time Biomechanically-based Muscle Volume Deformation using FEM"**, Computer Graphics Forum (Eurographics '98 Proceedings), pp.275-284.

[Zorin 96] D. Zorin, P. Schröder, W. Sweldens, **"Interpolating subdivision for meshes with arbitrary topology''**, Proc. of SIGGRAPH '96, pp. 189-192, 1996.

Curriculum Vitae

General information



Name	Amaury Aubel
Date of birth	25 July 1973 in Paris, France
Nationality	French
Mother tongues	French
Other languages	Good knowledge of English and German

Education

1989 - 1990

Baccalaureat C (scientific)

1992 - 1995

Engineering diploma in Computer Science

Institut d'Informatique d'Entreprise (I.I.E., France)

Diploma project: Standard Reporting on Dynamic Properties of Colored Petri Nets using Occurrence Graphs

Professional activities

1997 - 1998

French national service as an assistant with LIG-EPFL

1996 - 1997

Software engineer at the research center of Electricité de France. Contributed within a team of four to the conception and development in Java/VRML of an electronic virtual library.

Publications

- A. Aubel, D. Thalmann, “**Interactive Modeling of the Human Musculature”**, Proceedings of Computer Animation 2001, Seoul, Korea, November 2001.
- A. Aubel, D. Thalmann, **“Realistic Deformation of Human Body Shapes”**, 11-th Eurographics Workshop on Computer Animation and Simulation, Interlaken, Switzerland, August 2000.
- A. Aubel, R. Boulic, D. Thalmann, **“Real-Time Display of Virtual Humans: Levels of Detail and Impostors”**, IEEE Transactions on Circuits and Systems for Video Technology, 2000.
- A. Aubel, R. Boulic, D.Thalmann, **“Animated Impostors for Real-Time Display of Numerous Virtual Humans”**, Proc. Virtual Worlds 98, Paris, July 1998.
- P. Kalra, N. Magnenat Thalmann, L. Moccozet, G. Sannier, A. Aubel, D. Thalmann, **“Real-Time Animation of Realistic Virtual Humans”**, Computer Graphics and Applications, Vol. 18, No. 5, 1988, pp. 42-56.