A BIOMECHANICS-BASED ARTICULATION MODEL FOR MEDICAL APPLICATIONS

THÈSE N^{O} 3360 (2005)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut des systèmes informatiques et multimédias

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Anderson MACIEL

M.Sc. in Computer Science, Universidade Federal do Rio Grande do Sud, Brésil
de nationalités brésilienne et italienne

acceptée sur proposition du jury:

Prof. D. Thalmann, directeur de thèse
Prof. H. Delingette, rapporteur
Dr S. Ferguson, rapporteur
Prof. R. Hersch, rapporteur

Lausanne, EPFL
2005

To the first patient to ever be healed with the aid of a functional model of articulations.

"(...)
Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu,
Mas nele é que espelhou o céu."

(F. Pessoa)

Acknowledgements

My first thanks go to Professor Daniel Thalmann for the opportunity of working in his lab and the support, to Ronan Boulic for the supervision and advice, to Professor Nadia Thalmann for her leadership in CO-ME, and to Luciana Nedel for showing me the way to the VRlab.

To my family and friends in Brazil, thank you for your trust, understanding and love. To my teachers, professors and colleagues there, thanks for the parts of you echoing in the *ether* of this thesis.

To all my fellows in the VRlab and CVlab - the ones I barely knew and the ones I became friends with - whom in a way or another participated of this thesis or my stay in the lab, I'm sincerely grateful. Special thanks to Marcelo Kalmann for the priceless help in the beginning, and to Josiane Botarelli for the constant help. Amaury for the introduction in the lab, Benoît for the *résumé*, Helena for the review, Mireille for the ligaments, Pablo for the joints, Pascal for the support with tex, Schubert for the *resumo*. Great thanks also to my colleagues in MIRALab and in the project CO-ME for the team-work and mutual aid, in special to Sofiane Sarni, with whom I shared much pain and joy.

To all my students, who taught me more than they learned from me, in special to the ones who collaborated directly with this thesis by means of their diploma and semester projects, my very big thanks. Thanks also to my new friends in Switzerland and all over Europe. Your presence has been a big part of my feeling at home wherever I go.

Thank you Fabi for the unforgettable first two years in Switzerland. I still think I'd not have been able to go through all that alone.

Thanks also to the Swiss National Science Foundation for the financial support.

Abstract

Computer Graphics came into the medical world especially after the arrival of 3D medical imaging. Computer Graphics techniques are already integrated in the diagnosis procedure by means of the visual tridimensional analysis of computer tomography, magnetic resonance and even ultrasound data. The representations they provide, nevertheless, are static pictures of the patients' body, lacking in functional information. We believe that the next step in computer assisted diagnosis and surgery planning depends on the development of functional 3D models of human body. It is in this context that we propose a model of articulations based on biomechanics. Such model is able to simulate the joint functionality in order to allow for a number of medical applications. It was developed focusing on the following requirements: it must be at the same time simple enough to be implemented on computer, and realistic enough to allow for medical applications; it must be visual in order for applications to be able to explore the joint in a 3D simulation environment. Then, we propose to combine kinematical motion for the parts that can be considered as rigid, such as bones, and physical simulation of the soft tissues. We also deal with the interaction between the different elements of the joint, and for that we propose a specific contact management model.

Our kinematical skeleton is based on anatomy. Special considerations have been taken to include anatomical features like axis displacements, range of motion control, and joints coupling. Once a 3D model of the skeleton is built, it can be simulated by data coming from motion capture or can be specified by a specialist, a clinician for instance. Our deformation model is an extension of the classical mass-spring systems. A spherical volume is considered around mass points, and mechanical properties of real materials can be used to parameterize the model. Viscoelasticity, anisotropy and non-linearity of the tissues are simulated. We particularly proposed a method to configure the mass-spring matrix such that the objects behave according to a predefined Young's modulus. A contact management model is also proposed to deal with the geometric interactions between the elements inside the joint. After having tested several approaches, we proposed a new method for collision detection which measures in constant time the signed distance to the closest point for each point of two meshes subject to collide. We also proposed a method for collision response which acts directly on the surfaces geometry, in a way that the physical behavior relies on the propagation of reaction forces produced inside the tissue.

Finally, we proposed a 3D model of a joint combining the three elements: anatomical skeleton motion, biomechanical soft tissues deformation, and contact management. On the top of that we built a virtual hip joint and implemented a set of medical applications prototypes. Such applications allow for assessment of stress distribution on the articular surfaces, range of motion estimation based on ligament constraint, ligament elasticity estimation from clinically measured range of motion, and pre- and post-operative evaluation of stress distribution.

Although our model provides physicians with a number of useful variables for diagnosis and surgery planning, it should be improved for effective clinical use. Validation has been done partially. However, a global clinical validation is necessary. Patient specific data are still difficult to obtain, especially individualized mechanical properties of tissues. The characterization of material properties in our soft tissues model can also be improved by including control over the shear modulus.

Résumé

L'infographie a dans un premier temps été introduite dans le monde médical grâce à l'imagerie médicale tridimensionnelle. De nos jours, l'infographie fait partie du processus de décision par le biais notamment de l'analyse visuelle tridimensionnelle des tomographies, des images de résonnance magnétique et même des ultra-sons. Cependant, ces techniques fournissent uniquement des représentations statiques des patients et manquent d'informations fonctionnelles. Nous pensons que dans une prochaine étape, le diagnostic assisté par ordinateur et la planification de chirurgie dépendront du développement de modèles tridimensionnels ainsi que fonctionnels du corps humain. C'est dans cet esprit que nous proposons un modèle d'articulations basé sur la biomécanique. Un tel modèle est capable de simuler la fonctionalité des articulations pour permettre de nombreuses applications médicales. Nous l'avons développé en nous concentrant sur les exigences suivantes : il doit être à la fois assez simple pour être implémenté sur un ordinateur et assez réaliste pour permettre des applications médicales ; il doit être visuel de telle manière que l'utilisateur puisse explorer l'articulation dans un environnement de simulation tridimensionnelle. Ainsi, nous proposons de combiner le mouvement cinématique des parties pouvant être considérées comme rigides (les os), et la simulation physique des tissus mous (les ligaments et les cartilages). De plus, nous considérons les intéractions entre les différents éléments de l'articulation. Pour ce faire, nous proposons un modèle spécifique de gestion de contact.

Notre squelette cinématique est basé sur l'anatomie. Des précautions particulières ont été prises pour inclure des caractérisiques anatomiques commes le déplacement des axes, les limites articulaires ainsi que le couplage entre les articulations. Une fois le modèle tridimensionnel du squelette construit, il peut être animé en utilisant des données issues de la capture de mouvements ou spécifiées par un spécialiste. Notre modèle de déformation est une extension du système classique des masses-ressorts. Un volume sphérique est considéré autour des points de masse et les propriétés mécaniques des vrais matériaux peuvent être utilisées pour paramétrer le modèle. La visco-élasticité, l'anisotropie ainsi que la non-linéarité des tissus sont simulées. En particulier, nous proposons une méthode pour configurer la matrice masse-ressort de telle manière que les objets se comportent en suivant un module de Young prédéfini. Un modèle de gestion de contact est aussi proposé pour gérer les intéractions géométriques entre les éléments composant une articulation. Aprés avoir testé plusieurs approches, nous avons proposé une nouvelle méthode pour la détection de collisions qui mesure en temps constant la distance signée au point le plus proche pour chaque point de deux maillages susceptibles de rentrer en collision. Nous proposons également une méthode de réponse aux collisions qui agit directement sur la géométrie des surfaces afin que le comportement physique se base sur la propagation des forces de réaction produites à l'intérieur du tissu.

Enfin, nous avons proposé un modèle tridimensionnel d'une articulation combinant trois éléments : le mouvement anatomique du squelette, la déformation biomécanique des tissus mous et la gestion de contact. Sur cette base, nous avons construit une hanche virtuelle et nous avons implémenté un ensemble de prototypes d'applications médicales. De telles applications permettent l'estimation de la distribution des tensions sur les surfaces de l'articulation, l'estimation des limites articulaires basées sur les contraintes des ligaments, l'estimation de l'élasticité des ligaments à partir de limites articulaires mesurées cliniquement et d'évaluation pré- et post-opératoires de la distribution des tensions. Bien que notre modèle fournisse aux médecins d'importantes indications pour le diagnostic et la planification de chirurgie, il doit encore être amélioré pour une être utilisé cliniquement. Des validations partielles ont été effectuées mais une validation clinique globale serait encore nécessaire.

Resumo

A Computação Gráfica entrou no mundo médico principalmente depois da chegada das imagens médicas em 3D. Técnicas de computação gráfica já estão integradas ao processo de diagnóstico por meio da análise tridimensional de dados de tomografia computadorizada, de ressonância magnética e até mesmo de ultra-som. No entanto, as representações que eles fornecem são quadros estáticos do corpo de pacientes, carecendo de informação funcional. Acreditamos que o próximo passo em diagnóstico e planejamento de cirurgia auxiliados por computador seja dado em direção ao desenvolvimento de modelos tridimensionais funcionais do corpo humano. É nesse contexto que propomos um modelo de articulações baseado na biomecânica. Este modelo deve ser capaz de simular a funcionalidade da articulação de modo que possibilite o desenvolvimento de aplicações médicas. Ele foi desenvolvido tendo como base os seguintes requerimentos: simplicidade, para ser implementado em computador; realismo, para o seu uso em aplicações médicas; visualização, para que as aplicações possam explorar a articulação em um ambiente de simulação tridimensional. Para alcançar tais objetivos propusemos a combinação do movimento cinemático das partes que podem ser consideradas rígidas (ossos, por exemplo) com a simulação física dos tecidos moles (ligamento e cartilagem). A interação entre os diferentes elementos da articulação também foi considerada, e para isso propusemos um modelo de gerenciamento de contato adaptado ao nosso caso específico.

Nosso modelo de esqueleto cinemático é baseado em anatomia. Isso significa que considerações especiais foram tomadas para incluir características anatômicas, tais como: o deslocamento dos eixos de movimento, o controle da amplitude articular e a acoplagem de juntas. Logo, uma vez que um modelo 3D do esqueleto esteja construído, ele pode ser simulado a partir de dados de captura de movimento ou de movimentos especificados por um especialista (um clínico em ortopedia, por exemplo). Nosso modelo de deformação, por sua vez, é uma extensão dos clássicos sistemas massa-mola. Aqui, um volume esférico foi acrescentado em torno dos pontos de massa, e propriedades mecânicas dos materiais reais são utilizadas para parametrizar o modelo. Em especial, um método é proposto para configurar a matriz massa-mola de maneira que os objetos se comportem de acordo com um módulo de Young predefinido. Com isso, viscoelasticidade, anisotropia e não-linearidade dos tecidos são simuladas. Um modelo de contato também é proposto, este para lidar com as interações geométricas entre os elementos no interior da articulação. Depois de testar várias abordagens diferentes, nós propusemos um novo método para detecção de colisões que mede em tempo constante a distância ao ponto mais próximo para cada ponto de duas malhas poligonais sujeitas a colidir. Nós ainda apresentamos um método para resposta a colisões que atua diretamente na geometria das superfícies, deixando que o comportamento físico dependa da propagação das forças de reação produzidas dentro do tecido.

Por fim, propusemos um modelo de articulação 3D que combina os elementos de: movimento anatômico do esqueleto, deformação biomecânica dos tecidos moles e gerenciamento de contato. Com base nesse modelo, construímos um quadril virtual e implementamos protótipos de aplicações médicas, com o objetivo de validá-lo. Estas aplicações permitem a avaliação da distribuição das pressões e tensões sobre as superfícies articulares, a estimação da amplitude de movimentos baseada nas restrições ligamentares, a elasticidade de certos ligamentos segundo uma amplitude articular medida clinicamente, e a avaliação pré e pós-operativa da distribuição das pressões e tensões. Nosso modelo de articulação 3D é capaz de fornecer uma grande quantidade de informação útil aos médicos para diagnóstico e planejamento de cirurgia. No entanto, uma validação clínica se faz necessária antes que se possa efetivamente aplicá-lo a pacientes reais.

Contents

CHAPTER 1

Introduction

This thesis, despite its title, is not a work on Biomechanics or Medicine. It relies on biomechanical models and aims at medical applications, but remains a work in Computer Science. The focus on articulations is due to the fact that they constitute the basis of the human motion, and Computer Graphics, more specifically Virtual Humans modeling, are deeply involved in the study and reproduction of human motion and shape.

1.1 Context

The way clinicians proceed to diagnose pathologies in their patients today is a result of a broad evolution of the technics. The way they plan and execute a treatment as well. Many diagnosis methods are now based on images providing information that cannot be seen by the naked eye because they are in the interior of the patient's body. Image-based diagnosis started with X-rays after its discovery by Röntgen in 1895, and now a number of different techniques exist, among which: fluoroscopy, ultrasound, computer tomography and magnetic resonance.

Beside Physics, Computer Graphics became important for medical images specially with the use of 3D techniques. Great improvements of imaging and computer hardware allowed the development of new graphics methods, and increased the quality of the images. Consequently, a number of new applications using image-based 3D models of humans exist today. Diagnosis, surgery planning, and computer-aided surgery are examples.

Looking further, such image-based 3D models can still be improved. Let us consider that the transition from *blind diagnosis* to X-ray diagnosis was the first high technological jump. The second jump was the transition from 2D to 3D imaging. The third jump is coming next, and it will almost certainly be the transition from static images to functional models. Functional models of the human body would allow for a wide new palette of applications and would greatly improve the accuracy of diagnosis and treatment of a number of diseases.

Such functional models must be able to faithfully reproduce human physiology. However, due to the complexity of the human body, that is a very hard task, especially when it is known that everything in the body works in synergy, and the number of parameters is potentially enormous.

Nevertheless, we can assume some systems to be independent enough to be simulated separately. One which is specially interesting in Computer Graphics is the musculoskeletal system. Since the early ages of Virtual Humans modeling, it is the system we have been trying to reproduce for visual applications. Functional 3D models of the musculoskeletal system already exist, and are used in ergonomics, sports and character animation. Although some experiences have already been done, those models are not specifically focused on medical applications and are not yet clinically used. At the same time, clinical orthopedics and orthopedical surgery lack the information for diagnosis and treatment that a functional model of articulations could provide. False diagnosis and ineffective treatment still happen more often than we wish.

It is in this context that we propose this work.

1.2 Motivation and Objectives

Disease around articulations is one of the main causes of pain in Orthopedy. Pathologies such as hip arthritis reach virtually 100% of the population after a certain age. With the increase of the life expectance these last decades, existing treatments, such as joint replacement, are not effective any more. This and other musculoskeletal pathologies can be avoided or delayed in most of the cases if diagnosed early. On the other hand, intervention planning on biomechanical systems is still very primitive if compared to planning and design in construction and maintenance of mechanical systems. An analogy to computer-aided design (CAD), used in automobile or aeronautical industry, which could be called *computer-aided surgery design*, would be necessary.

Hence, early diagnosis by means of a functional analysis of a patient's virtual model coupled with surgical virtual prototype would greatly reduce pain occurrence and would increase the chances of success of the surgical treatment.

Motivated by such assertion, the aim of this work is to create a biomechanics-based 3D articulation model allowing the simulation of human joints for medical applications. This articulation model must be able to provide useful physical information despite the eventual simplifications it has in relation to the real articulation. Applications for which the model is suitable, as well as the ones for which it is not, must be identified. For that, we also aim at implementing a case study and assessing the model by means of test applications. Performance aspects as well as usability by the medical staff must be considered in the model conception. Finally, limitations and possible extensions must be discussed.

1.3 Approach

The ideal articulation model is the one which takes into account the mechanics of all the joint structures and their interactions, which can be seen as the real joint can be seen, and from inside as well, and which can be manipulated and modified geometrically in order to render a new behavior.

We will see, along the next chapters, that the existing models in Computer Graphics are far from being ideal. This happens because they are not focused on medical applications, but also because medical imaging hardware and computational resources and techniques are still limited when it comes to reproducing such an ideal model. Consequently, we must find a compromise.

Our approach proposes a hybrid model of the joint. Hybrid because part of it is merely kinematical, and the other part is based on biomechanical models of soft tissues. We assume that the geometry of bones, cartilages and ligaments are very important for the joint mechanics. Analogously, deformation of cartilages and ligaments are also very important; bone deformation can be neglected. Muscles, fat and skin are less important for most applications, and will not be considered in this work.

Our model relies on rigid kinematical bone motion, captured using dynamic MRI or optical motion capture, or yet defined by an expert. Models of soft tissue, opposingly, are fully physical. They are dynamically simulated, taking into account some important mechanical properties of the tissues, and the contact and load transmission between the different structures.

Such an approach must provide sufficient information to compute pressure and tension distribution all over the joint elements, as well as realistic visual feedback. It should also allow the analysis of the joint range of motions, particularly the detection of abnormal impingements.

1.4 Organization of this document

The structure of this document follows the methodology employed on the development of this work. Chapters 2, 3 and 4 bring our three conceptual models, respectively: the kinematical articulation model based on anatomy; the dynamic deformation model based on biomechanics; the contact model for collision detection and avoidance inside the joint. These three models are like pillars maintaining the medical applications on the top. They are combined together to compose what we called in the title of the thesis *a biomechanics-based articulation model for medical applications*.

Such articulation model is used to develop and evaluate four different medical applications, which are presented in chapter 5. A summary of contributions is presented in the conclusions (chapter 6).

Note that a global state of the art chapter is not present. Instead, each of the chapters describing a model contains a section bringing a literature review on its respective topic.

CHAPTER 2

Articulation model

2.1 Introduction

Both *joint* and *articulations* are terms denoting a biological or mathematical element connecting two or more parts of a skeleton. As stated in section 1.3, in this thesis we propose a hybrid approach to achieve a 3D articulation model suitable for medical applications in Computer Graphics. In this chapter, we detail the kinematical part of the model, while the biomechanical part is detailed in chapter 3.

One of the most important and more realistic approaches for human body modeling in Computer Graphics is the one in which the designers, like sculptors, take into account the anatomy under the skin to define the final shape of the body surface. Deformations and postures are defined rather manually, based on the designer's anatomic knowledge and artistic skills. However, the articulated motion still has a secondary importance for many applications, because the final result must *look real* and not actually be physically realistic. Consequently, most designers are satisfied with the simplified joint models based on Robotics often available. More recently, with the growing interest for medical applications based on human body models, the need for an articulation model including anatomy knowledge became a reality.

We present, in the next section, an Anatomy review highlighting the important features of the human joints anatomy often neglected. In section 2.3 we review the Computer Graphics literature around joints and articulated models. Next, in section 2.4, we introduce our anatomically based joint model, and in section 2.5 how we get the parameters to configure it.

2.2 Anatomy review

Mechanically, the human body can be briefly defined in terms of a skeleton, muscles, fat and skin. The skeleton is formed by around 200 bones connected to each other by articulations, and constitutes the basis for the shape of the human body surface [Kroemer et al., 1990]. A joint or articulation is then the union between two or more bones. The articulations are located at the bones extremities, where the participant bones are in contact with each other and motion may occur. Not all articulations are mobile; for those which are, motion happens around one or more axes, or just on a plane between the bones [Scheepers, 1996]. By convention of the medical sciences, all anatomical descriptions are done referring to the body in the posture called *anatomical position*, illustrated in figure 2.1. In that position, the body is erect, feet are joined and the arms are pending by the sides with the palms facing forward. In this work we followed the same convention.

This section presents the result of a general study on human joints. Some anatomic and biomechanical aspects are briefly described for the different articulation types, highlighting the points concerning their shape and functioning (motion).

2.2.1 Joints components

The joint of any mechanical device must be well lubricated such that its parts can move freely without damaging each other. In the human body, a special type of dense conjunctive tissue, known as articular cartilage or hyaline cartilage, provides a protector lubrication. A layer of this material, in average 1 to 7 mm thick, cover the bones extremities within the articulations presenting high degree of mobility, the so called diarthroses.

The articular cartilage has two main functions: first, it distributes the load of the joint onto a wider surface, reducing the stress on all contact points between the bones; second, it allows moving the joint with a minimum of friction and wear. Cartilage can reduce the maximum contact stress in an articulation of 50% or more. The lubrication provided by the articular cartilage is very efficient; the friction in the joint is about 17 to 33% of that of an ice skate with the same load [Hall, 1995].

Another important component of a joint is the articular fibrocartilage. As a complete or incomplete fibrocartilaginous disc, it can also be present between two articulated bony surfaces. Examples are the intervertebral discs (complete) and the knee menisci (incomplete) [Hall, 1995]. Some of the main functions of discs and menisci are:

- Absorption and distribution of loads
- Better fit between articular surfaces
- Limitation of the translation or sliding
- Protection of the joint periphery
- Lubrication

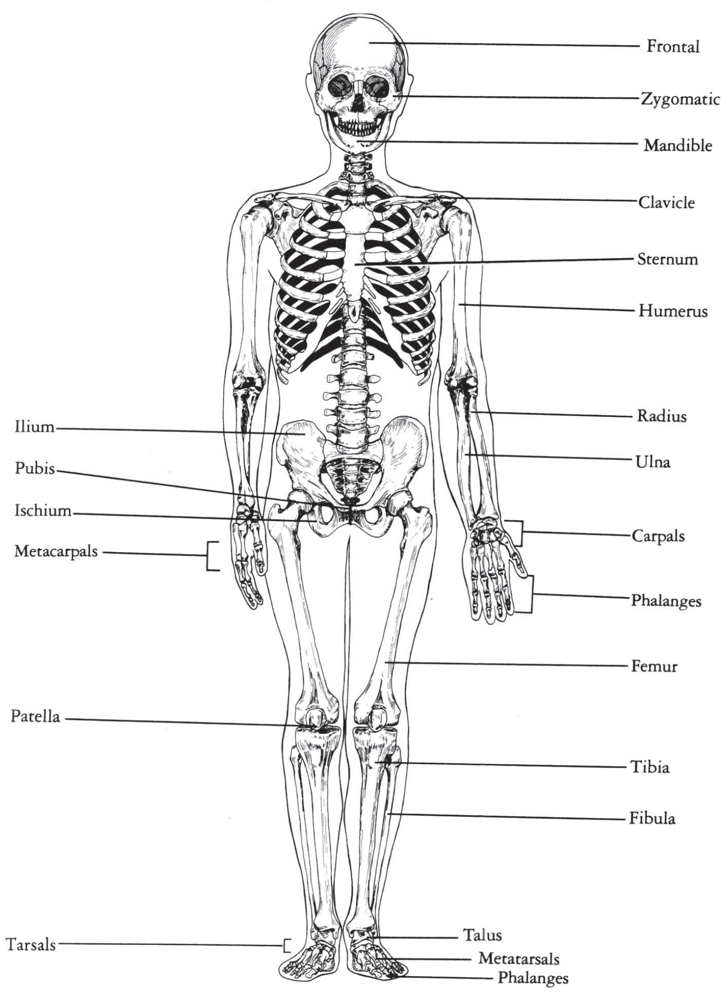


Figure 2.1: Anatomical position. Although on the image the right palm is not facing forward for better showing the metacarpus, in the anatomical position both palms are facing forward. Extracted from [Kreighbaum and Barthels, 1990].

Description	**Place**	**Type**	**DOF**
frontoparietal	head	sutura serrata	-
interparietal	head	schindylesis	-
sphenoethmoid	head	synchondrosis	-
between nasal bones	head	homogeneous sutura	-
dentoalveolar	head	gomphosis	-
atlantooccipital	neck/head	trochoid	1
temporomandibular	head	condylarthrosis	2
scapulo-humeral	shoulder	ball-and-socket	3
humero-ulnar	head	hinge	1
proximal radioulnar	forearm	trochoid/pivot	1
distal radioulnar	forearm	trochoid/pivot	1
radiocarpal	wrist	ellipsoid	2
carpometacarpal (thumb)	hand	saddle	2
carpometacarpal (others)	hand	arthrodial	6
metacarpophalangeal	hand	condylarthrosis (ellipsoid)	2
interphalangeal	hand	hinge	1
intervertebral	column	typical amphiarthrosis	3
costovertebral	thorax	arthrodial	1
costochondral	thorax	synchondrosis	-
chondrosternal	thorax	arthrodial	1
coxofemoral	hip	spheroid	3
femorotibial	knee	condylarthrosis	2
proximal tibiofibular	shin	arthrodial	3-plane
distal tibiofibular	shin	syndesmosis	-
calcaneotibial	ankle	hinge	1
metatarsophalangeal	foot	ellipsoid	2
interphalangeal	foot	hinge	1

Table 2.1: Listing of the human joints

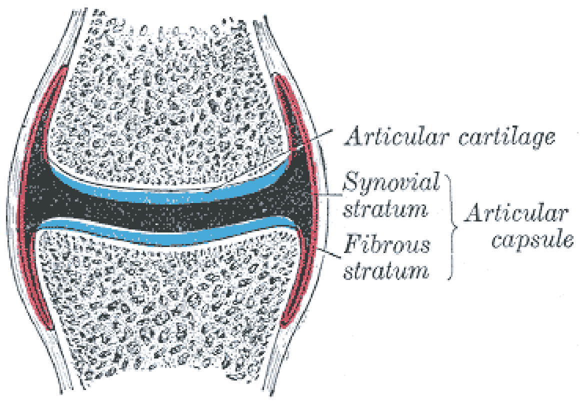


Figure 2.2: Schematic view of a synovial articulation. [Gray, 2000].

Beside the hyaline cartilage and the fibrocartilage, the joint capsule - also called synovial cavity - is an essential component for the joint physiology. Such capsule is a bag of conjunctive tissue involving the joint and hermetically closing it, as shown on figure 2.2. It is attached to the bones forming the joint, usually along the border of the joint cartilaginous surfaces. It is composed of one internal and one external layers. The internal surface is smooth and produce the synovial fluid, which function is lubricate the articular surfaces of hyaline cartilage easing their gliding. The external layer is a collagenous conjunctive tissue [Weineck and Dekornfeld, 1990].

2.2.2 Joints classification

Table 2.1 exhibits a complete listing of the human joints. Several classification systems are presented in Anatomy courses. They are typically based on the articular complexity, the number of degrees of freedom, the articular geometry, or on the motion capability [Hall, 1995]. As the goals of this work are related to the shape and motion presented by the joints, the classification shown here empathizes precisely those aspects. So, such is shown in figure 2.3, human joints can be split, according to their mobility, into three groups: synarthroses, amphiarthroses e diarthroses.

2.2.2.1 Synarthroses

This joint type is characterized by the union of two bones considered immobile because they do not present any specific type of movement. Despite this mobility restriction, these joints are very important because they respond to applied forces absorbing shocks. Examples of synarthroses are the sutures that keep the various skull bones hold together (figure 2.4).

a) **Synfibrosis.** It is a fibrous synarthrosis, it does not present any cartilage frame. Synfibroses can be classified into 6 different types:

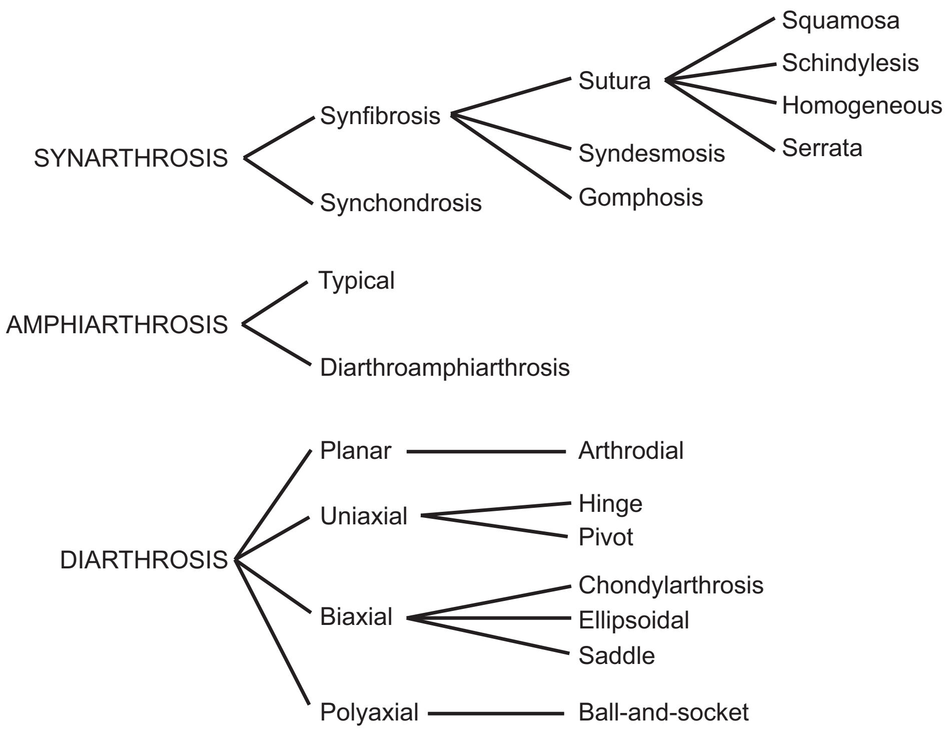


Figure 2.3: A classification for the types of human joints. Examples are given in table 2.1.

- **Sutura squamosa**: the bones bind like the scales of a fish. Ex: temporoparietal.
- **Sutura schindylesis**: a conic surface incases in a depression perfectly adapted to it. Ex: vomer-sphenoidal.
- **Homogeneous or plane sutura**: the contact is between two plane bony borders. Ex: between the nasal bones.
- **Sutura serrata**: the contacting surfaces look like the teeth of a fine saw. Ex: frontoparietal.
- **Syndesmosis**: great amount of fibrous tissue is interposed between the bony borders, which keep them joined and dramatically reduces the movement. Ex: tibiofibular distal.
- **Gomphosis**: a bony process fits into a socket. Ex: tooth-alveole, where the fibrous tissue interposed is called cement.

  A gomphosis is a special articulation, because the tooth is not bony tissue.

b) Synchondrosis. With this joints, the bones are separated by a thin layer of fibrocartilage. Examples are the sternocostal joints and the epiphysial discs (before ossification). Some authors consider the synchondrosis a type of amphiarthrosis.

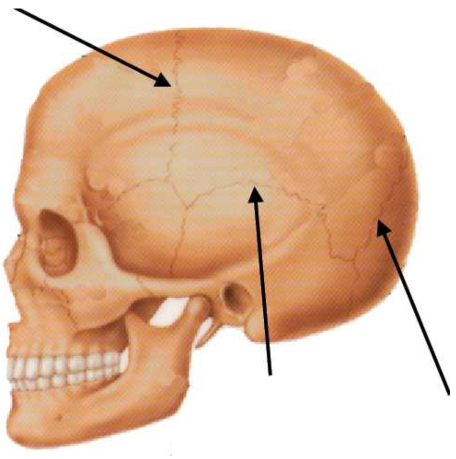


Figure 2.4: Examples of synarthroses: the sutures of the skull.

2.2.2.2 Amphiarthroses

Also called cartilaginous joints, the amphiarthroses have greater motion capability than the synarthroses, however, also present great motion constraints. This happens due to the presence of a disc or membrane of fibrocartilage in the link between the bones. Moreover, the ligaments involving a cartilaginous joint are relatively more rigid, which contributes on limiting their motion. On the other hand, this type of joint is excellent to absorb shocks, because the cartilaginous layer provides a good damping on the forces tending to approximate or separate the two bones.

The human body presents two types of amphiarthroses. With the *typical amphiarthroses*, only the fibrocartilaginous disc is placed between the bones. The joints between the human vertebral column are examples. With the diarthroamphiarthroses, also called symphyses, in addition to the fibrocartilaginous tissue there is also a kind of primitive articular cavity between the bones. The pubic symphysis is an example. Another example is the joint between the scapula and the clavicula, which are in turn separated by a cartilaginous membrane.

2.2.2.3 Diarthroses or synovial articulations

Joints of this type are characterized by the presence of the synovial capsule containing the synovial fluid. This fluid, together with the the cartilage caps which cover the bones extremities, aids on lubricating the joint, and consequently allow a wider motion capability. The synovial articulations represent the type of joint most commonly found on the study of the human motion.

Synovial articulations can be classified in several ways. One possible classification separates the joint in three groups:

a) Simple: with only two articular surfaces (ex: hip, ankle).

b) Composed: with three or more articular surfaces (ex: wrist).

c) **Complex:** with more than two articular surfaces and cartilaginous disc (knee) or fibrocartilage (clavicula).

Another classification, concerned by the shape and motion types presented, can place the diarthrodial joints into the following types:

a) **Planar, sliding or arthrodial:** Joints of this type allow very moderate translational and rotational movements (figure 2.5(a)). The articular surfaces are generally flat and small, and slide on each other. 6 DOF are permitted, however, all of them present a very low range of motion. There are examples of planar joints between the carpal bones (hand), and the tarsal and metatarsal bones (foot).

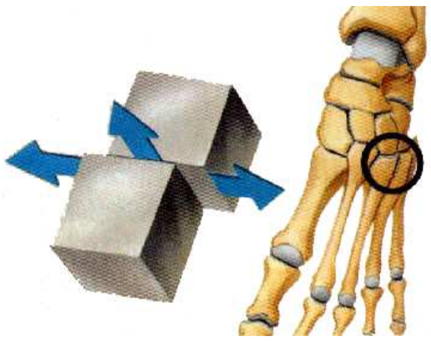
b) **Uniaxial:** Uniaxial joints are characterized by the presence of only one rotational motion axis (1 DOF). See figure 2.5(b).

- **Hinge, trochlearthrosis or ginglymus:** A joint of this type allow rotation around one axis perpendicular to the length of the bones involved. In a hinge joint, usually the extremity of one bone is concave while the other is convex. These complementary shapes are responsible for the hinge-like motion, limited to flexion/extension. Example: humeroulnar (elbow).
- **Pivot or trochoidal:** Pivot joints present angular motion around an axis parallel to the length of the bones involved, allowing one bone to turn around the other. Usually, one bone has a rather cylindrical head that twists within a notch on the other. Example: radioulnar (forearm).

c) **Biaxial:** These joints allow motion around 2 axes, characterizing 2 DOFs. Figure 2.6.

- **Condylarthroses:** Present two parallel rounded surfaces articulating with a plane surface. Because they have a motion very similar to the one of the ellipsoid joints, some classification may include them in the same group. Example: femorotibial (knee).
- **Ellipsoidal:** An egg-shaped head on one bone articulates with an also egg-shaped socket on the other. This class of joints is very similar and allow the same motion types as the ball-and-socket joint presented next, except rotation, avoided by the ellipsoidal shape of the articulating surfaces. Example: radiocarpal (wrist).
- **Saddle:** A saddle joint moves like an ellipsoidal one. The difference is in the joint shape. Each articulating extremity in a saddle joint has a double curvature, and when put together, the convex curve of one fits on the concave curve of the other. Example: carpometacarpal of the thumb (allows the thumb to move in opposition to the other fingers for grip).

d) **Polyaxial:** This type of joints allow motion around three axes, defining 3 DOF. The human polyaxial joints, also called spheroidal joints, enarthroses, or ball-and-socket joints, consist of a spherical head on one of the bones and a concave socket


(a) A planar joint

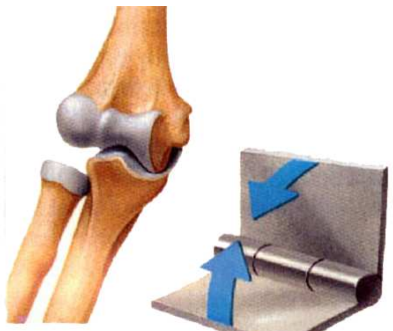
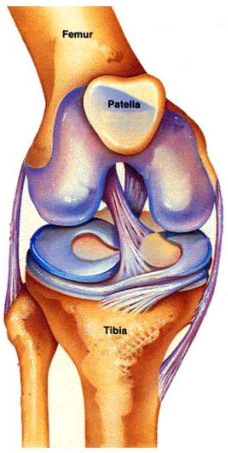
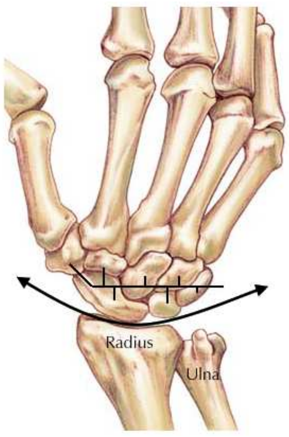

(b) A hinge joint

Figure 2.5: Examples of joints: (a) the planar joint between the tarsus and the metatarsus; (b) the hinge joint between the humerus and the forearm (elbow).



(a) Condylar



(b) Ellipsoid

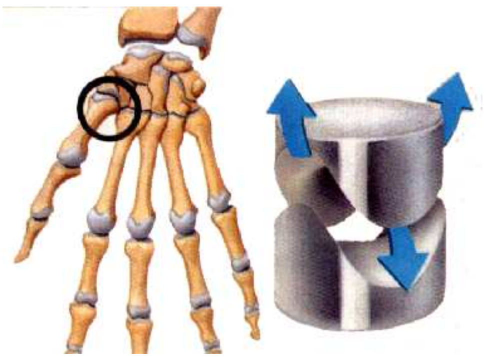

(c) Saddle

Figure 2.6: Examples of biaxial joints. (a) the knee, (b) the wrist, (c) the thumb.

on the other (figure 2.7). It is the most versatile type of joint, allowing wide motion on all planes, including flexion/extension, adduction/abduction, internal and external rotation. The range of motions depends a lot of the concave socket depth. A less deep socket, as in the shoulder, increases the range of motions. On the other hand, a deeper socket, as in the hip, increases the joint stability.

2.2.3 Factors of motion restriction

We have seen that the human body presents a large variety of motions distributed all over its articulations. But why some joints move differently of others? Which factors constrain the foot to not turn completely backwards, for example? Why olympic gymnastics athletes can perform movements that are impossible for most of the rather sedentary individuals? The search for answering those questions brings us to a series

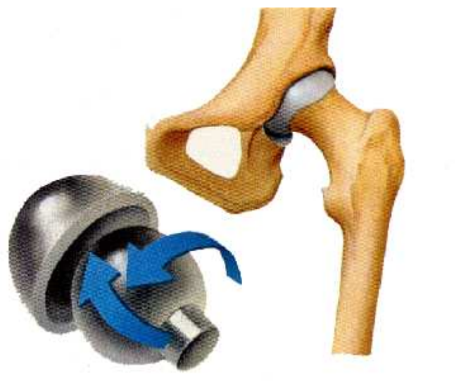


Figure 2.7: Example of ball-and-socket joint: the hip.

of hypotheses, all true, which work in synergy to determine the joint types and their range of motions.

2.2.3.1 Physiological factors

The first element of motion constraint is the **shape of the bones**. The spherical shape of the femoral head and its concave socket on the hip, for example, allow 3 DOFs motion, while the cylindrical shape of the elbow imposes a mechanical restriction allowing motion on one DOF only.

Another important factor on constraining motion is the presence of the **ligaments**, fibrous structures of low elasticity which push the bones against each other in order to keep them together around the locus they articulate. Ligaments are responsible, for example, for avoiding the interphalangeal joints of the fingers to be extended until de fingers touch the back of the hand. The **musculature** is also an important motion restriction factor. Strong muscles can increase power and stability of certain joints, and more than that can be important mechanical obstacles blocking some movements. An example is the elbow motion, which loses much of its range with athletes having a hyper developed brachial biceps. Another mechanical obstacle is **fat**. Obese people have much less motion capability around their joints than people with normal weight.

The presence of **cartilage** also influences directly on mobility. Among other examples, the thorax expansion during breathing show the important role of cartilage as a flexible structure of the body. It worth mention here that the babies have more flexibility than adults because their bones are not yet ossified, and are made of cartilage.

2.2.3.2 Other factors

In addition to the physiological constraints seen above, pathological cases, **lesions and contusions**, can also drastically modify the motion capability of some joints: a swollen finger does not flex as a normal one; if a scarred fracture, which usually presents a protuberance, is adjacent to a joint, it can influence its motion; when an impact causes a luxation, a ligament is often torn, reducing the joint stability, and after healing the joint will probably not have the same range of motion it had before. A specific case of pathology is the hip arthritis discussed further in this thesis.

Moreover, physical **training**, stretching, and physiotherapy can be applied to increase the range of motion of certain joints, improving the performance of athletes of different modalities.

2.3 State of the art on articulation models

A number of works has been published aiming at the creation of computer models of the human body and other animals. In those works, in a way or another, the joints are taken into account. Those works vary a lot in what they focus on and the way they approach modeling. Some prioritizes the skeleton, others put the muscles modeling as a first objective, yet others address the modeling of skin, hair and so on. Besides the focus, they also vary in their application. There are models created for Virtual Reality environments [Roehl, 1998], for entertainment in animation and games [Chadwick et al., 1989], for simulation of facial expressions [Guenter, 1989], for motion simulation in sports [Hodgins et al., 1995], applications in Ergonomics [Phillips and Badler, 1988], and many others. However, works approaching specifically the problem of anatomical realism of joints are rare in the Computer Graphics literature. We present in this section a study we have done aiming at surveying the state of the art in joint modeling. From such study we expected to detect strengths and limitations of existing models, and find indications to found the design of a new joint model based on anatomy. Next sub-section presents a rather old work, not in Computer Science, but in Robotics, a field which faced fairly earlier the articulations problem. In section 2.3.2, samples of the first significative works on human body modeling are presented. Section 2.3.3 comments briefly a series of works in Computer Graphics that, though not having joints as their main goal, for some reason had to consider and propose articulated hierarchical structures. Section 2.3.4 emphasize some works that in front of the complex task of anatomically representing all human joints, made the choice of delimiting the problem selecting specific groups of related joints to model them with a maximum of fidelity. Some commercial systems having functionalities closely related to articulations are presented in section 2.3.5. And to conclude, section 2.3.6 proposes a discussion and brings some conclusions from the observation of the works studied here, aiming at guiding the decisions made for the anatomically based joint model presented in section 2.4.

2.3.1 Model of Denavit-Hartenberg

In their pioneer work, which is still a reference in research on articulated structures, Denavit and Hartenberg described translational and rotational relationships between adjacent articulated parts for mechanical manipulators [Denavit and Hartenberg, 1955]. They used a matrix method establishing a coordinate system for each joint in the structure. Transformation matrices are created between those coordinate systems and

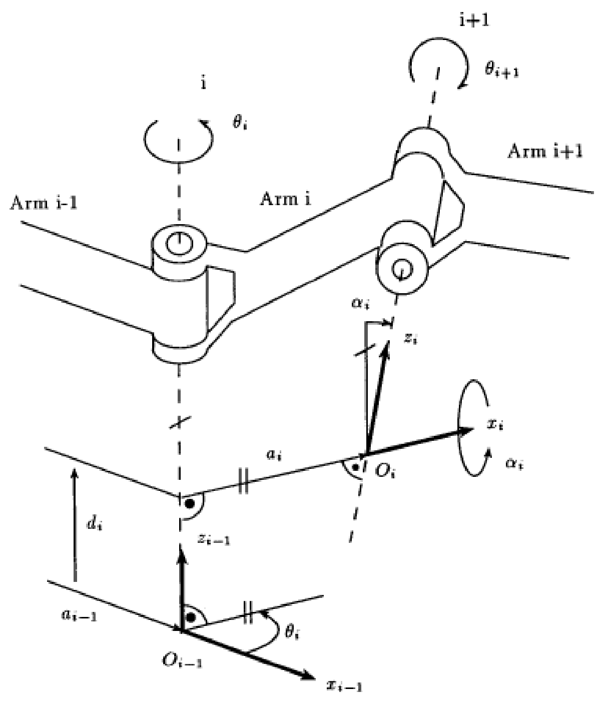


Figure 2.8: Schematic structure describing Denavit-Hartenberg hierarchy.

are associated to the respective articulations. See figure 2.8. Considering a joint i placed immediately below another joint $i-1$ in the hierarchical chain, the transformation matrices have basically 4 parameters: θ_i, rotation of an angle θ_i around the axis Z_{i-1}; d_i, translation of a distance d_i along the axis Z_{i-1}; a_i, translation of a distance a_i along the axis X_{i-1}; and α_i, clockwise rotation around the axis X_i of an angle α_i between the axes Z_i and $Z_i - i$ - where $(X,Y,Z)_{i-1}$ e $(X,Y,Z)_i$ represent the coordinate systems of the joints $i-1$ and i that are related by the parameters θ_i, d_i, a_i and α_i. Out of the 4 parameters mentioned, the matrix A (equations 2.1 and 2.2) is built, which represents the four homogeneous transformations relating the reference frame of the joint i and the reference frame of the joint $i-1$.

$$^{i-1}A_i = R(Z,\theta_i)T(0,0,d_i)T(\alpha_i,0,0)R(x,\alpha_i) \tag{2.1}$$

$$^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\theta_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

This articular hierarchy scheme, proposed by Denavit-Hartemberg (DH) for mechanical systems in Robotics, is efficient and not resource consuming for modeling diverse types of articulated structures, even those not belonging to Robotics. However, one can note that the articular topology of the human body presents features not covered by the model of DH. The DH notation has been developed to represent structures with joints presenting only one degree of freedom (DOF) each. Although it is possible to model joints with more than one DOF using a set of 1 DOF joints, this makes the structure more complex to manipulate and potentially more resource consuming. Another drawback, this one more critical, is the impossibility of representing structures with ramifications, like the human body. Moreover, the DH representation suppose one of the extremities of the articular chain to be fixed, which can limit even more its scope of applications.

2.3.2 Early models in Computer Graphics

The history of joint models in Computer Graphics mixes up with the history of human figure representations. The earliest models were aimed at applications on ergonomic analysis like the Landing Signal Officer, developed for Boeing in 1959 [Fetter, 1982]. The seven jointed *First Man*, used for the panel of a Boing 747, allowed pilot actions to be displayed by articulating the pelvis, neck, shoulders and elbows. Chrysler developed the *Cyberman*, based on 15 joints, for modeling human activity around a car [Blakeley, 1980]. The *Combiman* had a 35 internal-link skeletal system, and allowed to test how easily a human can reach objects in a cockpit.

Those and many others among the first models to represent human characters to appear in the Computer Graphics literature were the *stick-figures*, where the adjacent joint loci are connected by straight lines. Zeltzer introduced a skeleton representation with some advantages over the ambiguities and perception difficulties existing with stick-figures [Zeltzer, 1982]. Such model defines a language to describe the joints and the body segments connected by them. An interpreter for that language has been built that is able to generate two structures: a tree of transformations, where each node represents a primitive transformation on its child nodes; and a table of symbols with one entry for each joint. This table contains information about the degrees of freedom of the joint, amplitude constraints for the DOF, etc. For character animation with this representation, animation engines based on kinematics change the joint parameters along time, while a movement processor scans the transformation tree and computes the new positions and orientations for every joint. Within this model, all the joints may have up to 3 rotational DOFs parallel to the world reference frame.

Other systems at that time proposed using ellipsoids and other primitives to model body parts [Magnenat-Thalmann and Thalmann, 2004]. Korein and Badler presented techniques for motion generation on articulated structures, and considered their application on human body animation [Korein and Badler, 1982]. They also used a tree structure to model human characters. However, they represented the body segments as nodes and the joints as edges of the tree instead. Each segment is associated to a local reference frame; whenever a segment position is dependent of another segment, this relation is represented by an edge in the tree and corresponds to one joint. Some approximations have been done explicitly in this model: any joint with more than 1 DOF was transformed, by kinematical equivalence, into chains of joints with one DOF; complex structures, like the wrist and the shoulder, have first been simplified into 3

DOF joints, and later transformed into chains of three 1 DOF joints. In addition, even knowing the dependence existing among the positions and ranges of motions of the human joints, the authors assumed that each joint can move independently of any other, and the position of one joint does not affect the range of motions of another.

2.3.3 Generic models

[Boulic and Renault, 1991] developed the Hierarchy 3D (H3D), a hierarchy scheme for articulated human models. In this scheme, each joint represents only one translational or rotational DOF. Hence, to build a joint with 2 DOFs, like the human wrist for example, two joints are needed. The joints carry matrices with them which represent the relations of each joint with its parent in the hierarchy and with the root joint of the model, as well as its inverse and other important relations stored for a better performance of the editing and simulation processes.

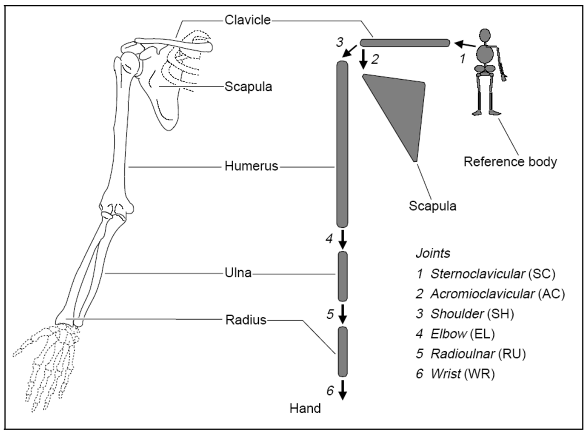
Jack is the software system developed in the University of Pennsylvania, which proposes to final users or developers an interface to handle complex articulated structures, and particularly human bodies in a 3D environment [Phillips and Badler, 1988]. In this system, the human body is represented by a set of rigid segments connected by joints. Each joint associates a segment to another by means of a local reference frame described in relation to the local reference frame of the parent segment. The joints can move with 6 DOF, allowing rotations and translations in the directions of the three cartesian axes, and the range of motion constraints are not part of the model. In the nineties, Jack started to be distributed as a commercial system, used mainly on applications in ergonomics. Since then, publications on techniques and algorithms used in the system practically disappeared, making difficult to keep track of its evolution.

In the axis-position joint (APJ) [Zeltzer and Sims, 1988], to each articulation, its position and the orientation of its reference frame in relation to the world-coordinate system have been associated. Moreover, every joint stores pointers to the segments connected by the joint. Thus, this notation allows building ramified articulated structures with several DOF. However, the joints are not hierarchically organized. The APJ notation has benn extended in [da Silva et al., 1997] and [da Silva, 1998] including three new parameters to the model such that the position and orientation of the joint be given in relation to its immediate superior in the hierarchy. In such hierarchical APJ, the position of each articulation is defined by means of the sequential bottom-up composition of the transformation matrices. That way, only the root joint is really free to be placed and oriented wherever in the space, while the remaining joints depend on that placement.

Wilhelms introduced a method to model any animal owning an articulated endoskeleton activated by muscles and covered by a flexible skin [Wilhelms, 1997]. The skeleton, much simplified, is formed by bones which geometry is composed of three ellipses - generally one more elongated, and the other two more spherical placed in the extremities of the former one. A tree hierarchy is utilized. Every group of bones and other tissues without relative motion is considered a segment of the body. The root segment is connected to the world reference frame by a 6 DOF joint and the other seg-

```
(define (the-arm-skeleton)
 (lambda
  (reference-skeleton)
   (model "clavicle" (ElevateDepress ProtractRetract)
    (SC-joint (ElevateDepress) (ProtractRetract))
    (clavicle)
    (separator
     (AC-joint (ElevateDepress) (ProtractRetract))
     (scapula))
    (model "humerus" (AbductAdduct FlexExtend Rotate)
     (SH-joint (AbductAdduct) (FlexExtend) (Rotate))
     (humerus)
     (model "ulna" (ElbowFlexExtend)
      (EL-joint (ElbowFlexExtend))
      (ulna)
      (model "radius" (PronateSupinate)
       (RU-joint (PronateSupinate))
       (radius)
       (model "hand" (FlexDorsiflex RabductUabduct)
        (WR-joint (FlexDorsiflex) (RabductUabduct))
        (hand)
)))))))
```

(a) AL example.



(b) Conceptual model.

Figure 2.9: Sample-code defining the arm hierarchy and its respective conceptual model (adapted from [Scheepers et al., 1997]). The description of the shoulder complex is not correct in this example because the scapula in in fact between the clavicula and the shoulder.

ments connected to each other by rotation DOFs with 3 DOF. Each segment (except the root segment) has a parent segment and zero or more child segments. In addition, each segment is described on its own local reference frame, and the geometric relations between all those segments are known and used during motion to visualize the model; the reference frame of the root segment is considered the reference frame of the whole animal. In that work, the author also presented an approach for muscle deformation and to build a surface to simulate the animal's skin; what is less relevant here.

Scheepers describes a model for the human skeleton in which the bones and articulations are created procedurally by means of a language: AL [Scheepers, 1996] [Scheepers et al., 1997]. The model is based on language productions that are utilized to place instances of bones and of different joint types in a textual hierarchical structure that, when evaluated, generates a geometrical representation of the skeleton in a specific position (see figure 2.9). In their implementation, a class - JOINTS - represents the articulations, containing information about position and orientation. The different articulation types, their DOFs and ranges of motion are determined by parametrization in the class constructor. There is a constructor for every specific type of joint.

The need for representing virtual human beings in VRML (virtual reality modeling language) brought the H-ANIM group to specify a standard for humanoids description [Roehl, 1998]. This standard is based on creating prototypes for the nodes representing the articulated human body structure - in the same way as cubes, cylinders and lights, among others, are basic prototypes defined in VRML for the construction of virtual environments. The central prototype of the articulated body is the *Joint*. Joints are defined as transformations establishing parent-child relationships between body segments, and store specific information about the joint, like a description and limit angles for each of its DOFs. By default, each joint presents 3 rotational DOFs, and to constrain it to 1 or 2 DOFs the practice is restricting the range of motion (angular

```
...
DEF hanim_l_shoulder Joint {  name "l_shoulder"
  center 0.167 1.36 -0.0518
  children    [
    DEF hanim_l_elbow Joint {  name "l_elbow"
      center 0.196 1.07 -0.0518
      children       [
        DEF hanim_l_wrist Joint {  name "l_wrist"
          center 0.213 0.811 -0.0338
          children    [
            DEF hanim_l_hand Segment {  name "l_hand"
              ...
            }
          ]
        }
        DEF hanim_l_forearm Segment {  name "l_forearm"
          ...
        }
      ]
    }
    DEF hanim_l_upperarm Segment {  name "l_upperarm"
      ...
    }
  ]
}
...
```

Figure 2.10: Example of joint hierarchy in VRML adapted from [Roehl, 1998].

limits) to zero. Figure 2.10 presents an example of structures used to represent a hierarchy of joints in VRML. Other node prototypes suggested are: *Segment*, which represents a body segment, containing one or more geometrical element, center of mass, momentum of inertia, etc.; *Site*, which represents a specific point and is used to define and end-effector for inverse kinematics; *Displacer*, which can be use to modify the shape of individual segments; *Humanoid*, which represents the complete body, allowing to store information about the virtual human, like their name, author and date, as well as providing a convenient way to move and place the humanoid in their environment.

Modeling and animating a virtual human can also be complex due to the high number of human joints. A wide range of models has been proposed for individual joints and specific regions (see section 2.3.4). The H-Anim standard, by imposing a uniform convention for joints orientations and skeleton topology, makes potentially viable the integration of heterogeneous joint models into a same system [Garchery et al., 2004]. [Babski, 2000] [Baerlocher, 2001] [Aubel, 2002] are examples.

2.3.4 Specific models

In a work extending the H3D, mentioned before, focusing on the human shoulder joint complex, Maurel and Thalmann presented a model based on restrictions for the displacement of the scapula over the thorax [Maurel and Thalmann, 2000]. In the real shoulder the articulations group forms a cycle, which makes a tree representation very little realistic. Regardless of that, to remove the cycle, many works in virtual humans propose to suppress the scapulothoracic functional joint. Hence, the scapula is more free to move and, unless a complex and costly animation control algorithm is developed, the

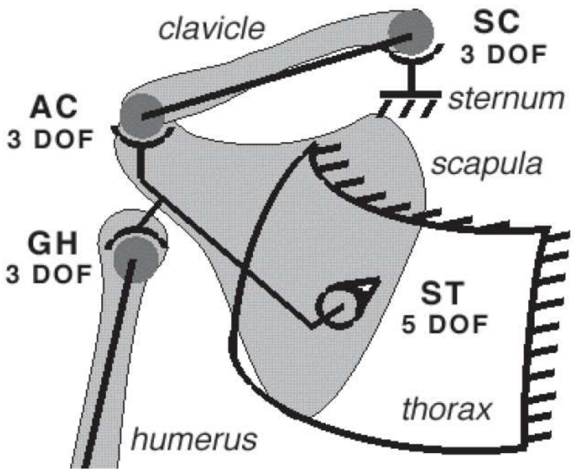


Figure 2.11: Shoulder kinematical model presented in [Maurel and Thalmann, 2000].

resulting motion is not realistic enough for many applications. Therefore, the authors developed a shoulder model in which each of the three bony joints has, 3 DOFs for rotation, and the scapula is bound to the thorax by means of a 5 DOFs joint, 3 for rotation and 2 for translations constrained to the surface of the thorax, see figure 2.11. Any motion is finally applied only to the scapulothoracic joint, being the other joints motions derived from the scapula position.

Monheit and Badler presented models specific for the vertebral column and the trunk, which were tested and visualized in the Jack system [Monheit and Badler, 1991]. Their work was based on research of the human vertebral column anatomy. Vertebrae and intervertebral discs dimensions, ranges and directions of movement, and rest positions for the column were obtained from medical measurements. Yet, the contribution of complex muscular groups and ligaments of the column have been considered on modeling, thus allowing to represent the traction and resistance forces existent around the column. The joints between each vertebrae pair have been modelled allowing 3 DOFs. For motion control, the column is seen as a black-box: with an initial state, input parameters, and an output state. Considering the still column, each vertebra has a current position defined on the local-coordinate systems of the joint connecting it to another vertebra. Adding the attributes of all joints, which are represented by 3D vectors, one can define the current position for the column as a whole, as well as the extreme positions it can achieve. Hence, when applying a movement on the model, the column target position must be given as a 3D vector, while the movement is defined as an interpolation between the current position and the limit positions or rest position for the column. The trunk is modelled as horizontal polyhedric slices, each of them corresponding to one vertebra and attached to it to follow its movement. [Garchery et al., 2004], in their work about H-Anim, also describe a spine model. They describe a technique to handle spine mapping to convert an animation between two H-Anim compliant skeletons with different number of vertebrae.

Ip et al. presented a hierarchical approach based on anatomy to model the human hand and animate its gestures [Ip et al., 2001]. Among their main contributions, they emphasize the possibility to generate realistic intermediate frames without using motion captors, and a gesturing data base relatively easy to generate and extend. The HACS (Hand Action Coding System) have been developed such that one can define gestures in terms of written text. With this system it is possible to synthesize anatomically correct intermediary motion from target positions for the hand. [Kurihara and Miyata, 2004] present techniques enabling skeleton driven deformation for a hand model. Their hand model is based on computer tomography (CT) images to reconstruct hand geometry in several poses. They also extract bones lengths and joints centers and angles from the CT images.

Dynamic simulation procedures and kinematic constraints have been used to compute the hand motion at each time step. For the hand modeling the authors considered several aspects of the human anatomy. Because the real hand is composed by bones connected by joints, and the movement of every bony segment is controlled by the contraction and relaxation of the muscles, other elements like fat, tendons and ligaments were not considered. The bones have been modelled as rigid segments of polygonal meshes, and the muscles as weightless expandable threads, which insertion points are fixed on the bones. The joints have been built and configured according to the parameters obtained from literature of Biomechanics around the human hand, i.e., the DOF and ranges of motions have been kept. However, they assumed that no isolated motion exist for the metacarpals (except for the thumb), a simplification justified by the relatively small range of motion of the carpus-metacarpal joint (CMC). Consequently, the 16 bony segments presented in the model are connected by 5 metacarpal-phalangeal (MCP) which present 2 DOF, 1 CMC of the thumb, also with 2 DOF and 9 interphalangeal (IP), these last with 1 DOF each.

2.3.5 Commercial products

There are systems for modeling and simulation of human bodies which have been developed as commercial software products, out of the academic world. Then, detailed descriptions of such systems cannot be found in the literature. Nevertheless, based on the specifications given by the developers and the experience using them, we selected two among the most important in the context of this work to describe here.

The SIMM (Software for Musculoskeletal Modeling) developed by Musculographics Inc. is mainly destined to users with problems involving Biomechanics, like physiotherapists and physics education professionals or researchers. This system allows creating and simulating articulated models of the human skeleton where, beside the joints, bones and muscles can be included. For that, SIMM offers the following tools: Plot Maker, Muscle Editor, Joint Editor, Wrap Editor e Deform Editor. *Plot Maker* allows inserting properties of muscular actuators, including force, momentum, and length of muscles and tendons. In the *Muscle Editor* the user has access to parameters describing the muscle geometry and topology, being able to place them on the bones surfaces. *Joint Editor*, the most interesting in the context of this work, enables the graphical manipulation of the joints kinematics. It allows defining up to 6 DOFs for one joint, 3 for rotation and 3 for translation. These DOFs, in fact, are implicit and cannot be directly

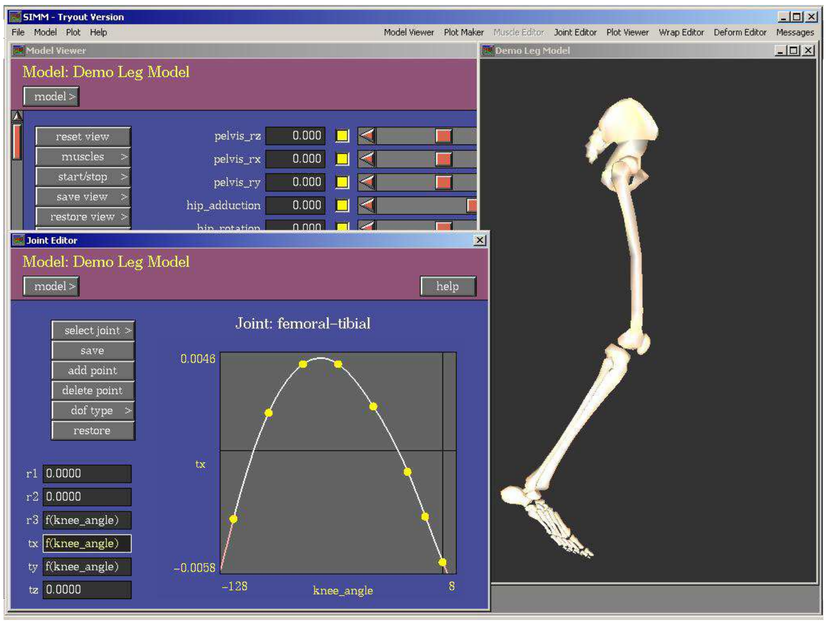


Figure 2.12: Screen shot from SIMM showing the *Joint Editor*.

manipulated during simulation. As it can be seen on the inferior window at the left in figure 2.12, the DOFs are associated to functions that keep them subordinated to the actual model variables: an angle of this or of another joint. Or yet, to constants, meaning that the joint does not have this DOF. Hence, we can say that the rotation axis representing the knee flexion (R1) will move as a function of *knee_flex*, a variable created especially for the flexion angle of the knee itself; but also, we can say that R1 will move as a function of *elbow_flex*, a variable storing the flexion angle of the elbow. The functions mentioned, respect a graphic relating the two variables (figure 2.12). Other tools within SIMM are the *Wrap Editor* and the *Deform Editor*, respectively responsible for allowing the creation of muscles that shrink and extend, and the specification of parameters for deformation of segments usually considered rigid, like bones.

Another commercial system strongly based on the articulations and destined for animation of many types of characters - not only human forms - is Character Studio, an extension for the 3D Studio Max modeling and animation system by Autodesk Inc. and Discreet Logic Inc. Although this system is not specifically directed for anatomically realistic models, it has the potential to build, refine and animate skeletons and skin. Under the implementation of several animation techniques (key-frame animation, kinematics, dynamics, motion capture, etc.), including intelligent mechanisms for crowds and flexible skin control, there is an articulated skeleton. Most part of the system being unknown because it is proprietary software, it is difficult to uncover details of its articular system implementation. Even though, some important points can be mentioned. One of them is the use of floating bones, a strategy that keeps some bones disconnected of the body hierarchy such that they can be easily handled in facial animation or breathing simulation. Another point is the possibility of fine tuning the joint gliding. In which concerns associating muscles with the joints, Character Studio also offer some tools. The muscular motion can be associated to the motion of bones and joints, in a way that they will influence muscles control, weight and power.

2.3.6 Discussion

In this section, we have shown that computer models of the human skeleton present important limitations when compared to real articulations. It is possible to find works in Computer Graphics modeling all fundamental types of human joints [Scheepers et al., 1997], but in opposition to what happens in the real skeleton, they are almost all modelled as rotations around one or more fixed axes, or translations on the 3 cartesian directions. The study presented in section 2.2 show that in real human skeletons the motion axes can change position as they rotate, and that dependencies exist between different axes causing the range of motion of an axis to be constrained or modified by the current position and orientation of others.

The articulated models described in this chapter rise important trade-offs: models are very much simplified in comparison to real joints, which results in loss of anatomic fidelity [Phillips and Badler, 1988] [R. Boulic and Werner, 1994] [Unuma et al., 1995] [Scheepers, 1996] [Wilhelms, 1997] [Roehl, 1998] [da Silva, 1998]; or include that complexity in the model, but are too much specific, once they treat only the very particular case of a restrict group of articulations [Monheit and Badler, 1991] [Maurel and Thalmann, 2000] [Ip et al., 2001]. Moreover, as the simplification is abandoned to achieve more anatomical realism, the performance become a critical factor. If, on the other hand, as the models are simplified, the complexity of the algorithms necessary for motion specification and control increases.

2.4 Our anatomy-based joint conceptual model

In this section we introduce a model of articulations for virtual humans based on anatomy, which is one of the contributions of this work [Maciel et al., 2002]. This model have been elaborated based on crossing the information we obtained from the study of human joints of section 2.2, and the survey on existing models in Computer Graphics of section 2.3. Designing this model, we searched to minimize the loss of realism present in most of the existent models ([Phillips and Badler, 1988], [R. Boulic and Werner, 1994], [Unuma et al., 1995], [Scheepers, 1996], [Wilhelms, 1997], [Roehl, 1998], [da Silva, 1998]), always keeping in mind the need to save processing power to other more complex tasks, like deformation and collision detection. The realism level must allow for medical applications but some simplifying assumptions are still acceptable. Moreover, the way a body is configured and how motion is specified for the articulated model can be very complex, what is to be avoided. For that reason, we also tried to encapsulate lower level details of the motion complexity. Object oriented modeling techniques were employed in order to obtain a layered model with relatively simple interfaces for the model parametrization.

The goal of this model is to represent the human body, so the Body is the higher level structure of the model. Bodies are composed of a set of joints hierarchically organized. Each joint, in turn, represents a relation between two adjacent parts of the body, and can be modified along time according to its degrees of freedom. Geometrical objects can be associated to articulations to represent bones, muscles, or any other structure of the anatomy. Those objects strictly depend of the joint, and have their position modified according to the changes made on the relation represented by the joint. Hence, this model should allow building graphic systems for simulation of anatomically realistic human bodies.

2.4.1 Joints

The bones of the human body are linked to each other in one or more points of their surfaces. Each of those links, with all their adjacent structures, is called a joint, or articulation [Gray, 2000]. In the model proposed here, joints are capital structures. Each joint is able to describe a positional relation between any two adjacent segments of a body, and allows that any modifications to those relations determine relative motion between the two segments. Such motion can happen in several different manners, according to figure 2.13:

- rotation around one axis (a);
- combination of rotations around two or three axes (b);
- translations in one, two, or in the three cartesian directions (c);
- rotations associated to translations (d);
- sliding axes during any of the rotations above (e).

Each joint represents a set o motion possibilities, or in other words, a set of degrees of freedom (DOF). A degree of freedom of a joint is what determines the existence of a movement and how such movement performs. A joint with 2 DOFs can perform two types of movement, for example: flexion/extension and adduction/abduction. In the model proposed here, a DOF means more than that, it has also an important role as a conceptual element of the model; this matter is detailed in section 2.4.2.

In order to build a human body from a set of joints, it is also necessary to determine its topology. We adopted the tree topology - which is very close to the one of the human articular system - and established relations between parent and child joints with homogeneous 4x4 matrices, which we call LIM (Local Instance Matrix). Consequently, every joint is defined on the reference frame of its parent joint, except the root joint, which is defined on the world-coordinate system (WCS), or on any other system exterior to the skeleton (depending on the application). Whenever it is necessary to obtain the position and orientation of a joint in relation to the WCS - for visualization, for example - successive multiplications of the LIMs from the root to the current joint can be done, as in the equation 2.3:

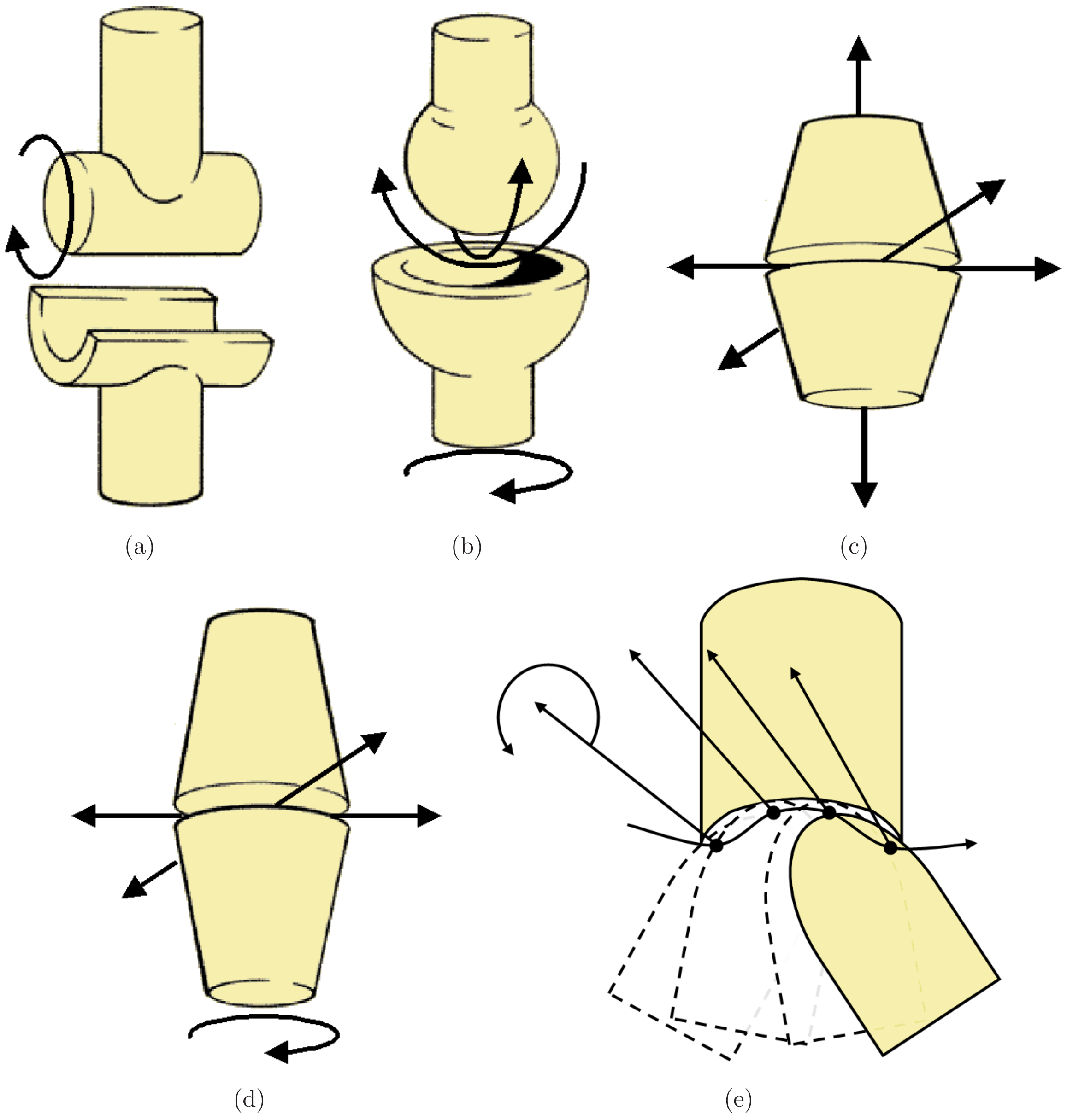


Figure 2.13: Possibilities for the articulated motion.

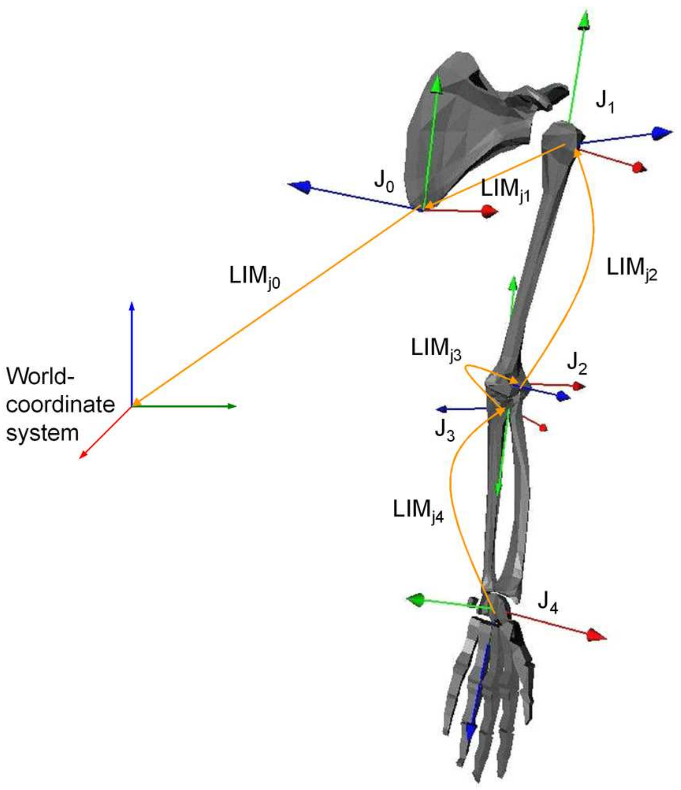


Figure 2.14: Example of hierarchical articular representation.

$$M_{[J_n \rightarrow WCS]} = (LIM_{J_n} \times (LIM_{J_{n-1}} \times \cdots \times (LIM_{J_2} \times (LIM_{J_1} \times LIM_{J_0})) \cdots)) \quad (2.3)$$

In equation 2.3, J_0 represents the root-joint and J_n represents any joint of the body. M is the homogeneous 4x4 matrix determining the position and orientation of J_n in the world-coordinate system, and is obtained multiplying the LIMs of all joints in the branch of the tree between J_0 and J_n. Figure 2.14 illustrates the role of the LIMs in the case of the human arm articulated hierarchy. The orthogonal axes in red, green and blue (RGB) represent the relation defined by the LIM of each joint, while the orange arrows determine the relation parent-child between each pair of joints. The joint J_0 is the root of the body and is placed on the WCS according to the relation represented by LIM_{j0}. The joint J_1 is child of J_0, being thus placed in relation to J_0 according to LIM_{j1}. Analogously, J_2, J_3 and J_4 are instantiated in relation to their respective parent joints following the relation expressed by their LIMs. The joints themselves do not have a geometric representation. They are important to define mathematical relations between parts of the body, but it is necessary to define other structures to represent bones, cartilages, etc. In our model, those structures are represented by an abstraction we call *Biostructure*. Biostructures can be specialized into very complex objects to represent shape and behavior of biological structures. However, in this chapter, what is most important is that they also consist of geometric information in the form of polygonal meshes that can be associated to any joint of the tree composing the body. Each biostructure also has a LIM serving to associate it to a joint and define its position and orientation in the frame of the joint. Accordingly, structures represented by biostructures change position and orientation as the joint moves. When global coordinates of a biostructure are needed - for collision detection or display, for example - they can be obtained multiplying the LIMs from the root joint until the joint in which the biostructure is, and finally multiplying also the LIM of the biostructure. Such process generates the GIM (global instance matrix) of the biostructure:

$$GIM_{B_m} = M_{[J_n \to WCS]} \times LIM_{B_m} \tag{2.4}$$

In equation 2.4, B_m is any biostructure, GIM_{B_m} is the matrix representing B_m in the WCS, matrix M is the same of equation 2.3, and LIM_{B_m} represents B_m in the local frame of J_n.

2.4.1.1 Different types of joints

Section 2.2 showed that there is a class of joints in the human body which do not perform specific motion and are considered immobile, like the skull sutures. So, once the model we propose focuses on motion, the synarthroses will not be considered. For the amphiarthroses, we have chosen to use the same type of joint we use for the diarthroses, because they are very similar in which concerns motion. Specific constraints imposed by cartilage discs can be modelled as range of motion control and sliding axes. In consequence, this model focus on the diarthroses motion, which is considered sufficient to simulate practically all human skeleton motions with great anatomical realism.

Four joint types were considered necessary to model a virtual human body appropriate to meet the goals of this thesis. Three of them are rotational joints and one is planar joint combined with rotations.

a) **Representing ginglymi and trochoides.** The human skeleton presents two basic types of joints with one degree of freedom (1 DOF). A ginglymus joint, also called hinge-joint due to its shape, has a motion axis laying approximatively orthogonal to the longest axis of the bones it connects. A trochoid joint, in turn, has the motion axis parallel to the length of the bones, and is thus also called pivot-joint. As the only difference between the two types is an axis orientation, we considered sufficient to model only one type of *uniaxial* joint. With such 1 DOF joint, we represent ginglymi and trochoides just changing the axis orientation

Representing saddle-, ellipsoid and condyloid joints. These three types of human joints have the same two degrees of freedom (2 DOFs). They all present similar motion, even if they have some differences in bones shape around the joint. Consequently, as this model does not relate the bones geometry with the joint motion, but only the constraints imposed by geometry, we grouped all 2 DOFs joint into only one *biaxial* joint type without realism loss.

Representing enarthrosis. The human skeleton presents two occurrences of enarthroses, or ball-and-socket joints, as these 3 DOF joints are mostly known: one is the shoulder, the other is the hip. Both allow the same motions of flexion/extension, adduction/abduction and rotation, differing basically in their ranges. So, we consider in this model only one type of 3 DOF joint that we called *polyaxial.*

Representing planar joints. Beside the joint types already mentioned, which have rotational movements, there are some joints in the human skeleton presenting translational movements. This type of joints, called planar joints, can be found between the bones of the hands and feet, at the clavicula, in the lower leg - where the fibula accomplish an up and down movement during gait - among others. The translation motion in these joints usually present a very reduced range, but it can be carried out in any of the three directions: a bone slides on the surface of another (two directions); a bone comes apart of another depending on the pressure exerted by the muscles or external objects (another direction). In this kind of joint, rotations are often associated with translations. Then, such that all joints of this type can be modelled, and to increase the flexibility to the potential applications, we included in the model a type of *planar* joint allowing for 6 DOF, three rotational and three translational.

When we mentioned flexibility, we referred to the use of such joint as the root for an articulated body. The planar joint can then be used to instantiate the body on a more global coordinate system in any position or orientation.

2.4.2 Dof as a conceptual element of the model

The expression degrees of freedom, often abbreviated by DOF, has originally and still is used to designate the motion capability of certain mobile systems, like mechanical robots. However, in this work, the concept of DOF can be wider, and to differentiate of the original one we decided to write *Dof* when talking about a conceptual element of our model instead of the original DOF. So here, Dof stands for the element of a joint serving to describe each of the individual movements present in the joint.

Among the existent articulation models, two distinct approaches are especially relevant in describing geometrical relation between two body segments. One of them considers only joints of 3 rotational DOFs ([Unuma et al., 1995] [Wilhelms, 1997]). The other considers that each joint has only one DOF; when it is necessary to represent joint of 2 DOFs, one must use a sequence of 2 joints of 1 DOF ([Korein and Badler, 1982] [R. Boulic and Werner, 1994]). With the first approach, when making rotations on two axes, one must be careful with the order in which they are done. Rotating on x and then y does not result the same as rotating on y and then x. With the second approach this is a minor problem because we know in advance which of the motions is subordinated to the other. But on the other hand, using several elements to model only one joint makes more complex the motion control procedure.

In our approach, each joint can have a number of Dofs, but as already explained before, a Dof in small caps means here a conceptual element of the model. Basically, to each Dof correspond: a motion axis (represented by a vector); its range of motion (angular or distance limits); its comfort range; its rest position; and its current posi-

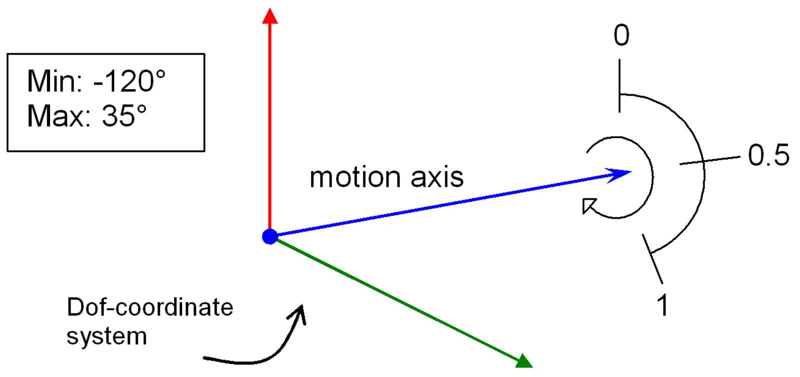


Figure 2.15: Example of *Dof*.

tion. However, such that different orientations of the motion axis around itself can be described, each Dof is also contemplated with two other orthogonal axes, characterizing thus the Dof reference frame. When one wishes to flex a joint, they send to it a new normalized flexion parameter (a number between 0.0 and 1.0) that is used to determine the new angle for the flexion Dof or the joint. That is done interpolating between the angular limits. This normalized parameter is useful because it is not always possible to reach the angle one wishes. Each Dof has a set of four of those limits, two inferior and two superior. Within each of these two limit pairs, the *limit* indicates the extreme angle that can be reached, and the *comfort limit* indicates the extreme angle one can bare before the posture becomes uncomfortable. In addition, the same normalized parameter is also used afterwards to parameterize the sliding curves (section 2.4.3).

As shown on figure 2.15, when the flexion/extension Dof is completely extended (parameter = 0.0) its angle in relation to the parent-coordinate system is -120°C. As the flexion/extension parameter is increased, it is mapped to its respective angle, in a way that when the parameter is 0.5 the angle will be -42.5°, and when it is 1.0 the angle will reach its limit of 35°.

2.4.3 Sliding curves

The human joints study presented in section 2.2 shows that the joints rarely move around stationary axes. In fact, as the rotation performs around an axis, a displacement of this axis also occurs. This displacement, often relatively small but essential for anatomic realism, can be described by a curve on the space.

In the Computer Graphics literature, many solutions can be found for curve modeling. They go from polynomial representations until parametric representations like Splines, Bézier, Catmull-rom, etc. [Foley et al., 1990]. We observed such representations and the crossed information with the curve shapes found in the anatomy in order to choose an adequate representation in the context of this work. Figure 2.10 shows the sliding curve for the flexion/extension axis of the human knee, the joint presenting the largest axis sliding. Although the figure brings only a 2D lateral view, we can conclude that a simple parametric representation in 3D, like a Bézier with 4 control

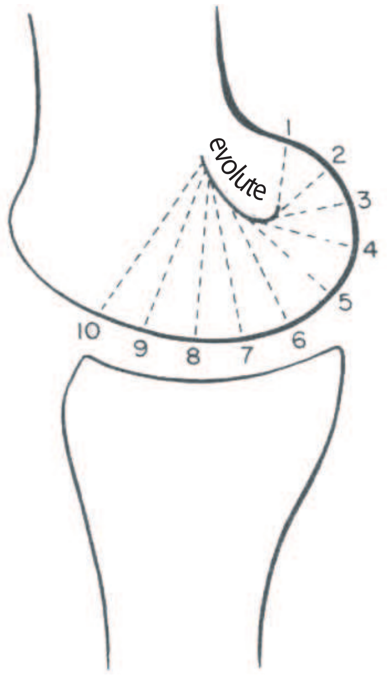


Figure 2.16: Schematic knee joint showing its *evolute* or sliding curve.

points, satisfies the requirements of the model being proposed here. Nevertheless, we believe that a future work aiming at the study of sliding axes can bring more precise results.

To assure that the axis will slide on the curve during the joint motion we use the following strategy: every time the parameter defining the angular position of a given axis is modified, the same parameter is given to the respective sliding curve as its normalized parameter; from that, the point on the curve corresponding to the given parameter is calculated, and the axis is translated to such point. In the example of figure 2.17, when the flexion/extension Dof is completely extended (parameter = 0.0), its axis will be placed on A on the curve, and its angle will be -120°. As the flexion/extension parameter is increased, the axis position changes to another point on the curve, coming closer to point B, until to reach it when the parameter is equal to 1.0. At the same time, the parameter is being mapped to its respective angle, such that when it is equal to 1.0 the angle reaches it limit of 35°.

2.4.4 Dofs as joint components

We showed in section 2.4.1 that a LIM of a joint describes a coordinate system placed and oriented in relation to the parent joint-coordinate system. Because of that, moving a joint means modifying this reference system, and consequently, modifying its LIM.

To perform the necessary modifications on the LIM of a joint we use its Dofs. As we have also shown, the joints are composed by Dofs, and each Dof is responsible for each motion type carried out by the joint. Each Dof has also a LIM, which describes its local-coordinate system placed and oriented in relation to the reference frame of the

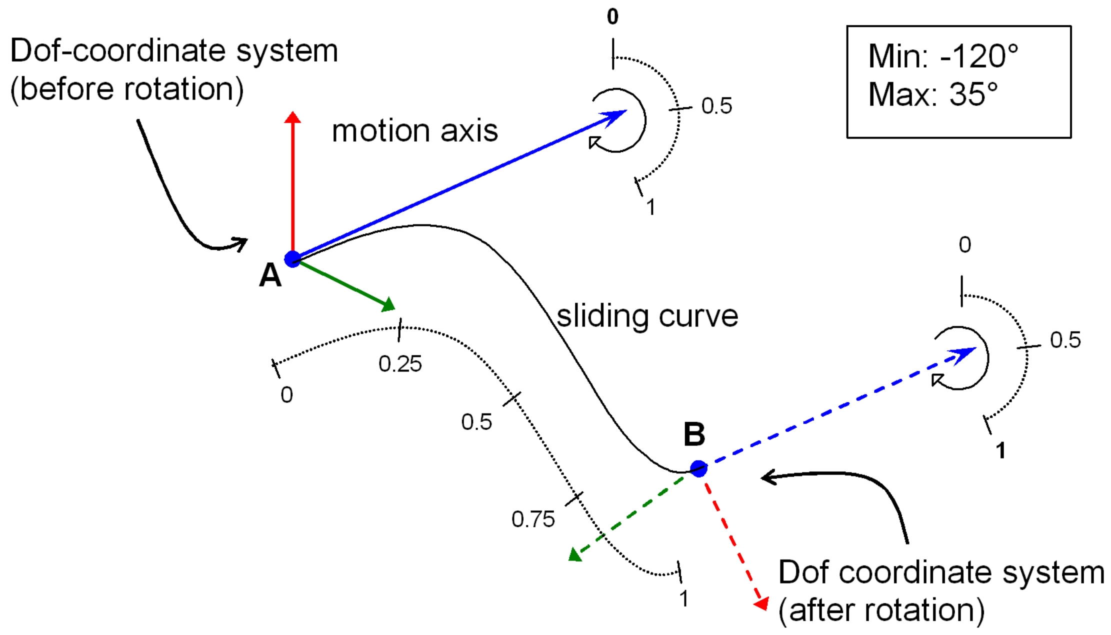


Figure 2.17: Example of sliding curve.

Dof immediately superior in the hierarchy, or to the parent joint for the case of the first Dof. Hence, the LIM of a joint is actually a combination made out of its Dofs LIMs, and must be recalculated whenever some motion happens in one or more Dofs. Equation 2.5 shows how the LIM of a joint j is obtained.

$$LIM_j = LIM_{Dof_{0_j}} \times LIM_{Dof_{1_j}} \times \cdots \times LIM_{Dof_{n_j}} \tag{2.5}$$

Figure 2.18 presents an example of articulated body architecture where the relations between the elements are shown, especially the Dofs role in determining relations between joints. J_0 is the root joint, its LIM determines how it is placed and oriented in relation to the world-coordinate system. Its Dofs are not shown, but it has a child joint, J_1. J_1 is instantiated in relation to J_0 according to LIM_{j_1}. This matrix (LIM_{j_1}) is obtained as in equation 2.5 by composing the LIMs of all j_1 Dofs. Finally, the LIM of the *Biostructure* represents its instantiation parameters in relation to J_1; LIMs of J_0, J_1 and the Biostructure are combined to produce GIM_{bio}.

2.4.5 Coupled movements

One of the greatest problems on articulations modeling is related to the difficulty for human beings to perform isolated motions. Usually, researchers in Biomechanics try to isolate the motion of each articulation and of each of its DOFs to simplify the motion extraction or measuring. In consequence, most of the range of motion tables found in medical or biomechanical literature present individual values for motion axis. However, we know that the human body works in synergy, and that the range of motion in one DOF can be, temporarily, modified by position changing on other DOFs of the same

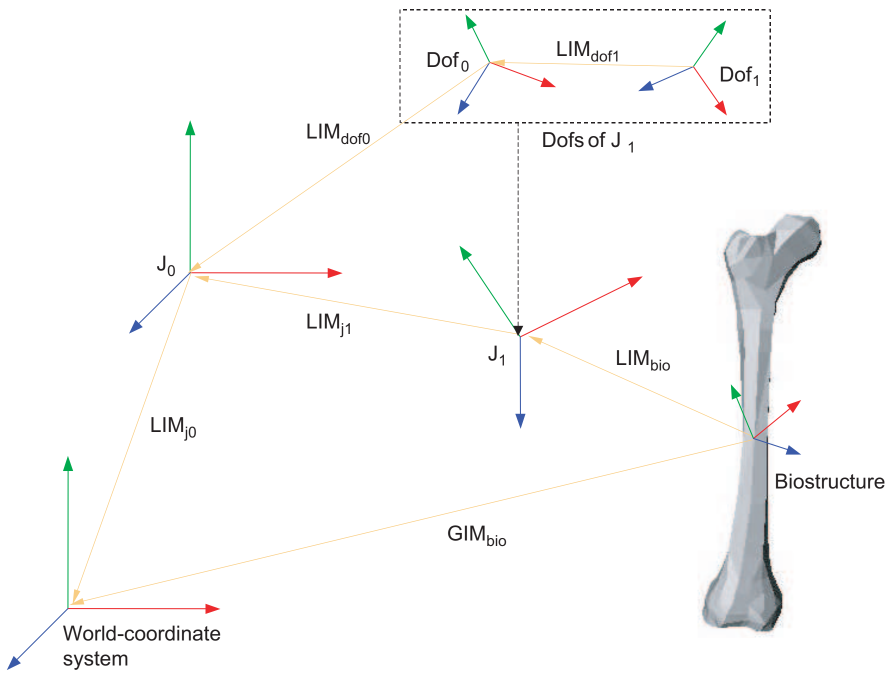


Figure 2.18: Joints are a combination of Dofs.

joint or even DOFs of other joints. Yet, in Computer Graphics we could not find works really focused on the problem of range relationship between different joints. Usually, to produce natural appearing motion, the works in the area propose complex animation control algorithms.

We propose to use a strategy allowing to relate every Dof of a virtual body to a list of other Dofs which exert influence on it. Even if motion amplitude relations can vary from one person to another, such strategy allows modeling some general low level mechanical constraints in a lower level of the model, permitting the use of simpler control motion algorithms in the higher levels. Two functions are associated to each *influencer* Dof, one to represent its effect on the minimal angle of the *influenced* Dof, and another on the maximum angle. Figure 2.19 shows a schema of the modifier, the structure used to represent amplitude relations in our model.

When Dof_n gets a new motion parameter, it has to verify its max and min limits such that it can calculate its new angular position. If it has a modifier, the modifier will provide it with the current max and min values. The calculation of those values is done by the modifier as follows: it reads the position of Dof_x and use it as input (x) for the curve *min* related to Dof_x, obtaining thus (y) as output, the minimal value that Dof_x allows to Dof_n. Following the same procedure to Dof_y, and to any other Dof in the list of influencers of Dof_n, the minimum value allowed for Dof_n is obtained. If this value is greater than the normal minimum of Dof_n, it will become its current minimum. Everything happens analogously for the maximum angle.

An example of application where this strategy is useful is the motion between fingers. When the fingers are extended, it is possible to execute the motion of adduc-

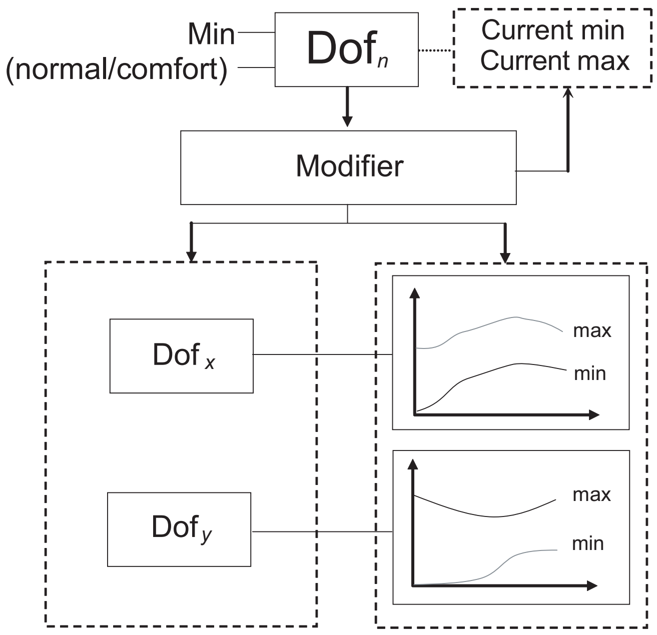


Figure 2.19: Range modifiers.

tion/abduction (bending the index finger from side to side, like to say *no*). With the finger in flexed position at about 90°, such movement becomes extremely limited and many times impossible. In this example, the flexion Dof of the finger exerts a dramatic influence onto the range of motion of the adduction Dof of the same finger.

2.4.6 Model design

In order to better understand the conceptual anatomy-based joint model we are presenting in this section, and to see it from the computer system point of view, we show now the results of our design using the Unified Modeling Language (UML) [Booch et al., 1998].

One of the characteristics demanded for an articular model is simplicity on setting up and on motion specification. Hence, we attempted to encapsulate the inherent complexity of representing human joints using principles of object oriented design. Figure 2.20 presents a class diagram in which the principal classes of the model and their associations are shown.

Once this model has been conceived for medical applications, it seemed normal to start the description by the class *Patient*. A *Patient* is associated to a *Joint* corresponding to the root of the joints hierarchy of the patient. Each *Joint* has a list of child-joints, a parent-joint, a list of *Dofs*, a LIM, and a list of *Biostructures*. However, a *Joint* is an abstract entity and cannot be instantiated. Instead, more specific classes (*UniaxialJoint*, *BiaxialJoint*, *PolyaxialJoint*, *PlanarJoint*) have been created, and inherit features of the class *Joint*. *Biostructure* is also an abstract class and, by means of its specializations, can represent different biological structures, like bones, cartilages, ligaments or muscles. It is useful because, depending on the subclass, a *Biostructure*

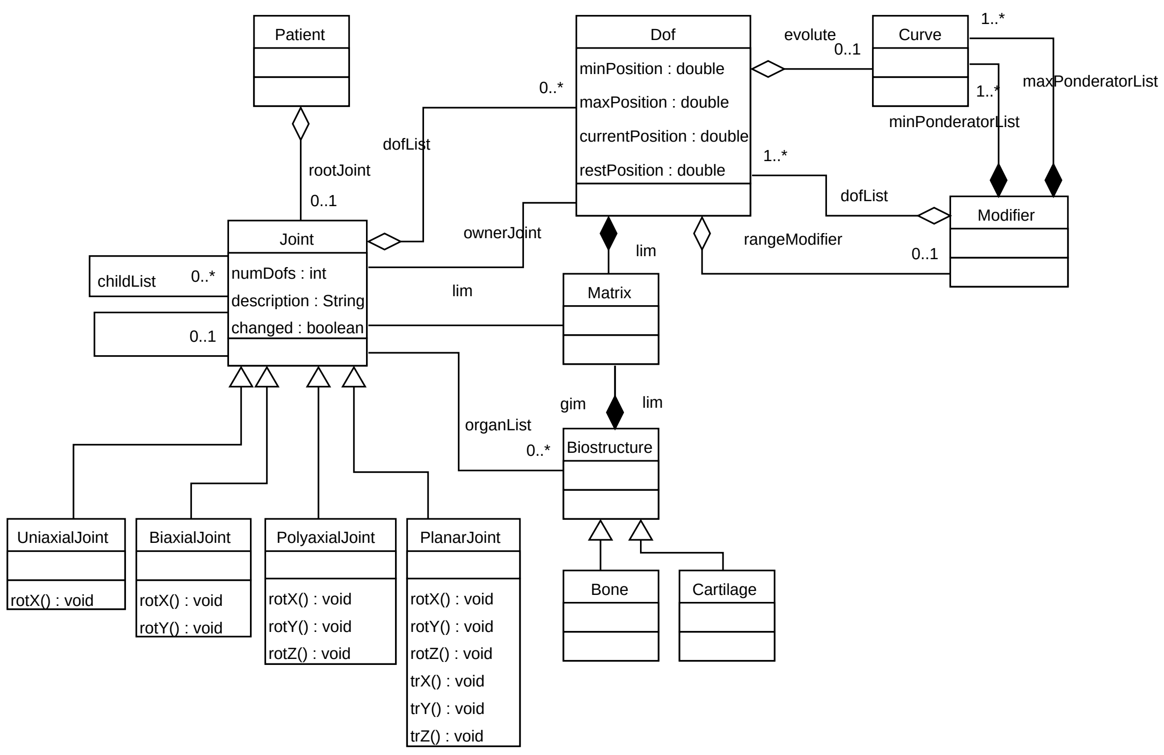


Figure 2.20: Simplified class diagram for the joint model in UML.

can be deformable or not, be represented geometrically by polygons, polyhedra, or any other geometrical representation with their different attributes and functions. A *Dof* has its ranges of motion, a vector defining its motion axis and an evolute *Curve* for axis sliding. Moreover, each Dof has a range *Modifier*, which in turn has a list of *Dofs* and two lists of *Curves* used to establish their influence on the *Dof* owning this modifier.

2.4.7 An example of articulated structure

We present here an example of use for the model presented in this chapter on human body representation. The structures necessary to represent the body are schematically instantiated in figure 2.21. In the figure, the left human leg is shown in parallel with a schema showing how it is expressed in the model. The grey circles represent the *Joints*, while the smaller and colored circles represent the *Dofs* - indicated according to their movement by the letters *R* (rotation) and *T* (translation), and the axes x, y and z. A thicker arrow indicates the presence of a *Modifier* between the two knee *Dofs*, and small curves inscribed in dashed circles - associated to some *Dofs* - underline the presence of a sliding curve. Dashed arrows relate the joints schema with with the leg image, where the position and orientation corresponding to each *Dof* are represented by a set of colored axes, according to the colors distributed do the *Dofs*.

The root joint of this body has been put between the two hip bones, and is considered to be the reference frame for the whole body. Such joint has 6 Dofs, allowing

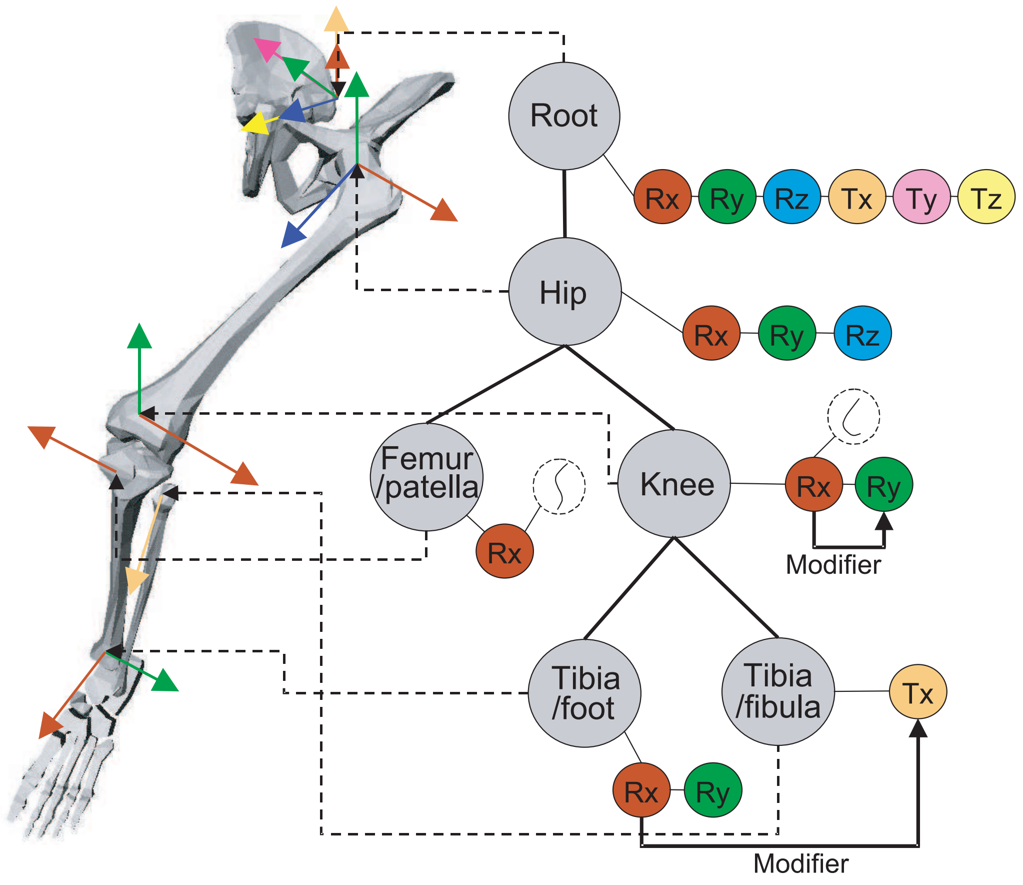


Figure 2.21: Example of modeling a body part.

rotations and translations in all directions. Expressed in the root-coordinate system are the two iliac bones - represented geometrically by polygonal meshes - and a child joint, the Hip, defining the motion of the femur on the ilium. The Hip has 3 rotational Dofs, and the *Biostructure* representing the femur is associated to it. This joint has two child-joints. The first, femoropatellar, has 1 Dof and a sliding curve. It represents the motion of the patella on the femoral condyles during the knee flexion. The second, femorotibial, is the knee itself, a joint with two Dofs that is responsible for the motion of the tibia in relation to the femur. The first Dof allows for flexion/extension motion, and holds curve for sliding axis. The second Dof provides the axial rotation of the knee, a movement which the range of motion is dramatically constrained by the status of the knee flexion/extension; a *Modifier* is applied to represent this constraint. The knee joint has two child-joints. The first, tibia/foot, has two rotational Dofs, one for flexion/extension and the other for adduction/abduction of the foot on the tibia. The second, fibula/tibia, has one translational Dof. It is responsible for the up-and-down displacement of the fibula in relation to the tibia when the foot is flexed.

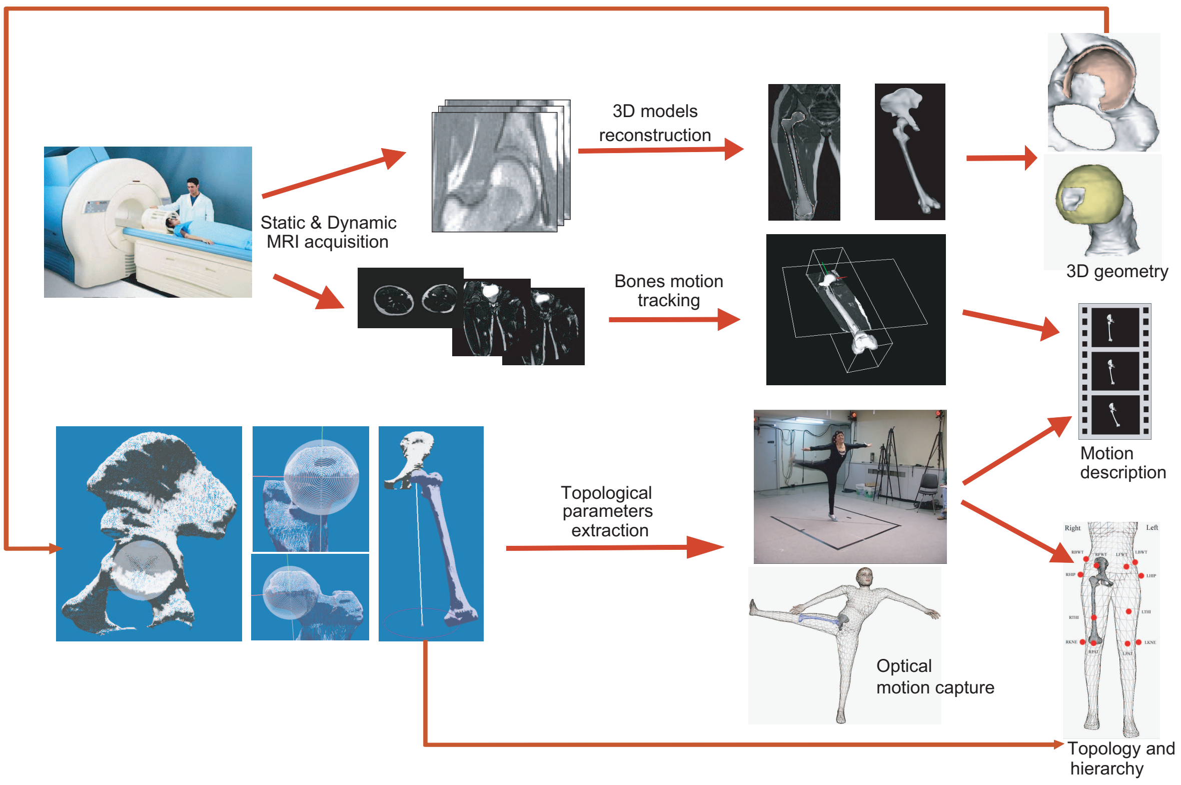


Figure 2.22: Automatic data input for the anatomy-based joint model.

2.5 Data model

The conceptual model presented in section 2.4 defines data relationship and state changes to simulate articulated motion. However, the model depends of several parameters that must be specified in some way. Those parameters should ideally be measured on the body of the person one wants to simulate, and in a very simplified approach could be input manually by an expert.

As said in chapter 1, this thesis work has been carried out in the context of a larger project where a group of researchers is working in parallel on closely related and complementary problems. So, the data provided by other research can and have been used to setup the model. Still, to keep independence, the model allows to apply a manually defined set of parameters.

The input data supply for the model is described in figure 2.22. Essentially, magnetic resonance images (MRI) and optical motion capture (MoCap) allow to extract anatomical features and motion information from a real person. The acquired data is treated and become useful information for the anatomically based kinematical model, like: joint center and anatomical axes location, center displacement, length of segments, range of motion, motion specification, etc. More detailed information about the data model can be seen in chapter 5 for the hip case study.

2.6 Conclusion

The model presented in this chapter is a proposal for a simplified and computer suitable mathematical description of the complex mechanics of human joints. It has been, nevertheless, conceived taking care of including the main peculiarities of human anatomy often neglected on human modeling in Computer Graphics. It brings solutions for the sliding axes problem, and also the inter-influence of different joints and their ranges of motion. Besides, the problem of the articulated body hierarchy has been analyzed, and a new solution was proposed - using Dofs - for the problem of modifying the transformation matrices during motion transparently during motion.

The design of the model has been oriented by the concern of making it computationally viable. Then, important elements of the anatomy, like ligaments and cartilages, do not explicitly appear in the kinematical model, their presence being here simulated by rules and restrictions. A deformation model for ligaments and cartilages is considered in chapter 3. It is used then to define motion constraints for the kinematical model and to calculate physical parameters for medical applications.

Further, the object oriented design used here emphasize the encapsulation of complex parts of the model, facilitating the model implementation into a human bodies modeling and simulation graphics system.

CHAPTER 3

Deformation Model

3.1 Introduction

The difference between rigid and deformable objects is that the deformable ones allow internal position change, i.e., the relative positions of two or more different points of the same object change. Four basic types of deformation can be cited: plastic, elastic, viscoelastic and fracture.

In plastic deformation, the object does not return to its original shape after cessation of force application. The deformation can be said to be definitive. Elastic deformation, opposingly, means that a relation between force and deformation exists. The object recover its original shape when the force application is interrupted. Viscoelastic objects, in turn, deform elastically but the relation between force and deformation is time-dependent. Details on viscoelastic properties are in section 3.3.1. Finally, fracture supposes an irreversible topology change; an object is broken into two or more objects.

Biological tissues can endure the four types of deformation, but in physiological situations those tissues are usually viscoelastic. One wishing to model bio-tissues should take into account their very complex mechanical behavior. Nevertheless, if the goal is to implement such model in a 3D computer system, they should search a compromise between physical realism and viability. The system applications will determine the model minimal acceptable accuracy, and the resources available must be handled such that we make the most of them. In the specific case of this thesis, the intended applications are in Medicine, more specifically, diagnosis and treatment planning.

In this chapter, we present a conceptual model of soft tissues for simulation of deformable biological tissue. We start by reviewing the existent deformation models and crossing that information with the requirements our model must comply with. Then we identify limitations and propose solutions. Finally we propose an implementation of the model and discuss computer issues. Next chapter brings a case study on our articulation model based on bio-tissues mechanics.

3.2 State of the art

Approaches for modeling object deformation range from non-physical methods (purely geometrical) to methods based on continuum mechanics, which account for the effects of material properties, external forces, and environmental constraints or object deformation [Gibson and Mirtich, 1997]. Historically, deformation modeling in graphics systems follows a timeline parallel to a computer power line. In a first phase they were mostly non-real-time (batch). They were used for creation and editing of complex curves, surfaces and solids. Applications existed in computer-aided design and image analysis. In a second phase, real-time (interactive) applications started to be in the focus. In this phase, faster machines allowed interactive design, games, virtual sculpting and even virtual surgery. However, up to there, the modeling goal was to produce credible deformation, not physically realistic deformation. The physically correct systems were still batch simulation systems, and were only rarely graphical. More recently, new advances in hardware resources allowed physically correct graphical simulation at interactive rates. Researchers still have to find the good compromise between realistic modeling and interactive speeds, but new applications, like computer-aided surgery, are now possible.

In this section, we present some of the most important non-physically based methods, and then focus more on the existing physically based approaches. Among them, we give more attention to the two widest used approaches: mass-spring systems and finite element methods.

3.2.1 Non-physically based methods

Twenty years ago, the most traditional methods for computer graphics modeling were kinematic, i.e., models were either stationary or subjected to motion according to prescribed trajectories. Since the work of [Terzopoulos et al., 1987], the physics based methods (section 3.2.2) became popular and now are used for many applications. However, the non-physically based methods have also been evolving. They are still widely used, and there are various applications to which they fit better than physically based models, especially when a high level of geometric control is required, like in geometric design and character animation.

Non-physically based methods are usually based heuristic geometric techniques or use a sort of simplified physical principles to achieve a reality-looking effect. They apply a kind of clay approach, allowing a user to deform the space containing the objects or manipulating operators like bend and twist onto primitives or abstract data like in [Barr, 1984].

The remaining of this section brings an overview on the principal non-physically based deformation methods.

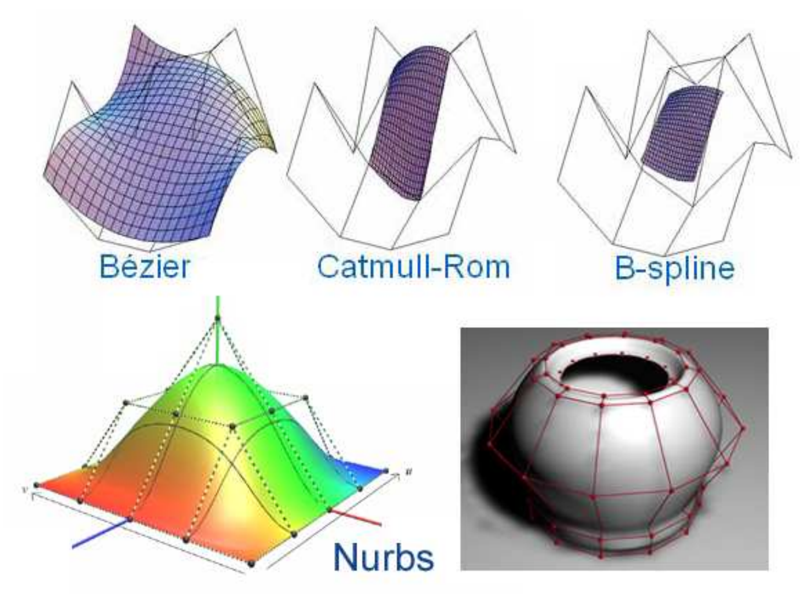


Figure 3.1: Examples of splines (adapted from several sources).

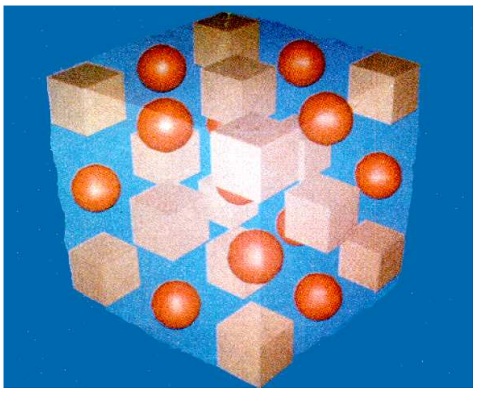
3.2.1.1 Splines

Mainly applied in the field of computer aided geometric design, spline techniques are a tool for creating and interpolating curves and surfaces as well as for modifying and refining these objects when needed. The spline technique is based on the representation of both planar and 3D curves and surfaces by a set of control points also called landmarks (details in [Bartels et al., 1987]). See figure 3.1 for examples of splines.

The first spline technique to appear in computer graphics literature was the *Bézier* spline, based on *Bézier* curves. Bézier curves are widely used in computer graphics to model smooth curves. The curve is completely contained in the convex hull of its control points, and then the points can be graphically displayed and used to manipulate the curve intuitively. Affine transformations such as translation, scaling and rotation can be applied on the curve by applying the respective transform on the control points of the curve. Bézier splines are then sets of low order Bézier curves patched together (obeying certain smoothness conditions) to represent more complex shapes.

Basis splines (B-splines) are a generalization of Bézier curves that can be further generalized to NURBS. They depend on the k nearest control points at any point t. Combining B-splines allow creating B-spline surfaces.

Non-uniform rational basic splines (NURBS) are also a generalization of Bézier-splines, with the primary difference being the weighting of the control points which makes them rational (non-rational B-splines are a special case of rational B-splines) [Piegl and Tiller, 1995].


(a) Undeformed plastic

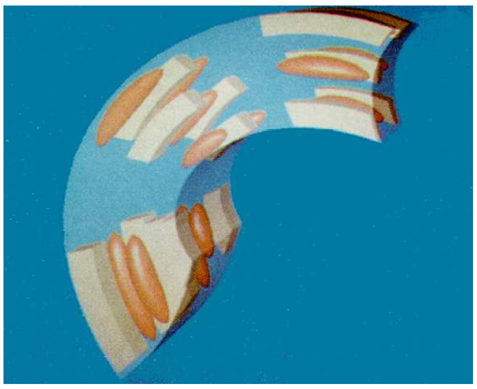
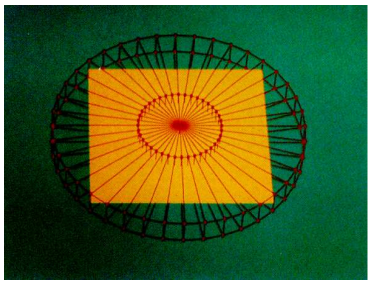
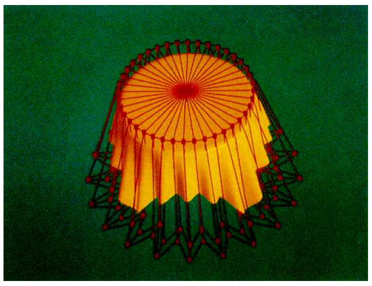

(b) Deformed plastic

Figure 3.2: A physical analogy for free-form deformation. A parallelepiped of transparent plastic material with objects embedded; deforming the plastic block causes the objects to deform (extracted from [Sederberg and Parry, 1986]).


(a) A cylindrical lattice around a planar surface


(b) The deformed lattice and the deformed surface

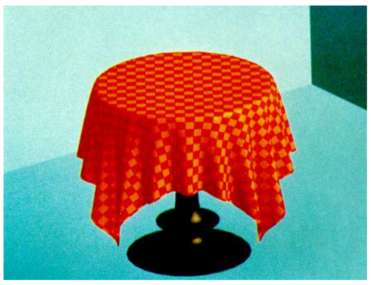

(c) A tablecloth

Figure 3.3: A non-parallelepipedical FFD (extracted from [Coquillart, 1990]).

3.2.1.2 Free-form deformation

FFD is the acronym for free-form deformation. This technique was first introduced by [Sederberg and Parry, 1986], and can be applied on both CSG[1] and B-rep. It consists of deforming the space embedding the objects. A good physical analogy to explain how it works is to consider a parallelepiped of a transparent plastic material in which one or more objects are embedded, like in figure 3.2. When the plastic parallelepiped is deformed, the objects inside deform in consequence.

Mathematically, a local-coordinate system and a grid of control points is distributed on the parallelepiped. The control points are used as coefficients of a trivariate tensor product Bernstein polynomial (like in [Sederberg and Parry, 1986]) or another polynomial base, like tensor product B-splines, for example. Then, the deformed position of any point is found first computing its coordinates on the local system, and finally evaluating the polynomial equation.

Extensions have been proposed to overcome limitations of classical FFD. [Coquil-

[1]Constructive Solid Geometry

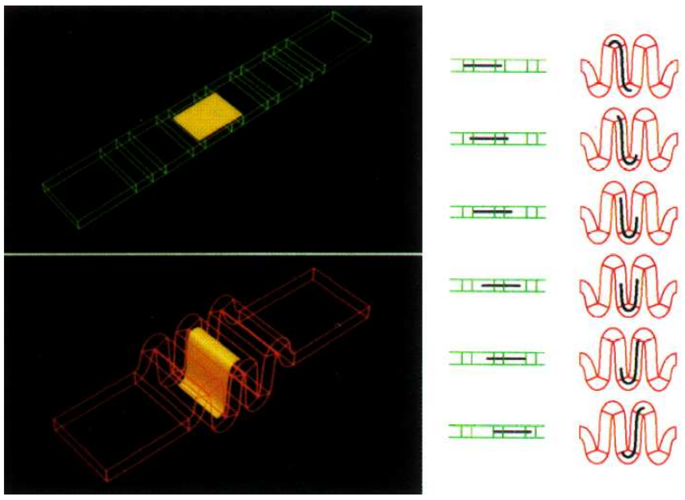


Figure 3.4: Animating a paper sheet (adapted from [Coquillart and Jancéne, 1991]).

lart, 1990] proposed an extension allowing to design arbitrarily shaped bumps and to bend surfaces along arbitrary shaped curves. It is based on non-parallelepidedical FFD lattices, like cylindrical or spherical lattices (see figure 3.3. In [Coquillart and Jancéne, 1991], the authors introduce the animated free-form deformation (AFFD). Extending the classical FFD to animation, the technique allows interactive animation of deformable objects (see figure 3.4). The result is that one can produce complex animation by simple key-frame of the FFD control points, for example. [Chang and Rockwood, 1994] apply repeatedly affine transformations in space. Such transformations are performed on a user-defined curve, a Bézier for instance, which plays the role of an FFD lattice. In addition to the reduced FFD, they also generalize the *de Casteljau* algorithm for curve evaluation. [Moccozet and Thalmann, 1997] presented a generalized method for free-form deformation that combines the traditional FFD model with techniques of scattered data interpolation based on Delaunay and Drichlet-Voronoi diagrams. They applied the method to model deformations around a human hand model.

3.2.1.3 Other techniques

Splines and FFD based techniques are the most important in non-physically based deformation. However, other techniques exist. The metaballs model is based on implicit surfaces typically defined from simple building block functions called primitives. The final object is constructed by blending the primitives, and as they move and deform, the blended surface is updated. See figure 3.5 for an example.

In the basic formulation proposed by [Blinn, 1982], some blobs are defined by a set of points $P_i(x_i, y_i, z_i)$ where each point represents the source of a potential field. Each source is defined by a field function $F_i(x, y, z)$ that maps $\mathbb{R}^3$ to $\mathbb{R}$ (or a subset of $\mathbb{R}$). At a given point $P(x, y, z)$ of the euclidian space, the fields of all the sources are computed and added together, leading to a global field function $F(x, y, z)$. [Wilhelms, 1994] used implicit surfaces to simulate muscles, and [Turner, 1995] developed a system to

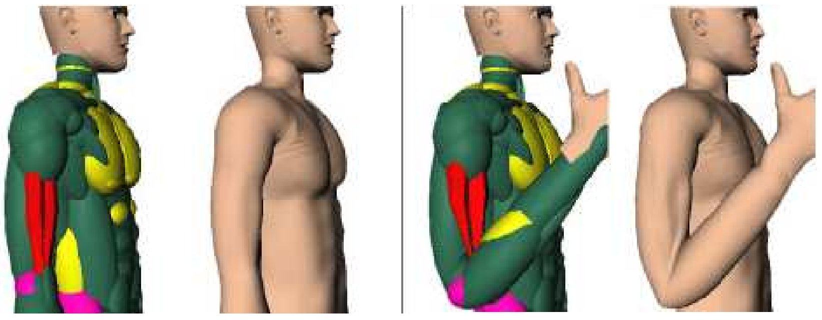


Figure 3.5: An example of metaballs modeling (adapted from [Nedel and Thalmann, 1998]).

construct and animate 3D characters based on the elastic surface layer model; muscles were modelled as deformable implicit surfaces (spheres, cylinders and super-ellipses).

Besides metaballs, we can also mention other techniques like the energy constraints method [Witkin et al., 1987] and the active contour models (Snakes) [Kass et al., 1988].

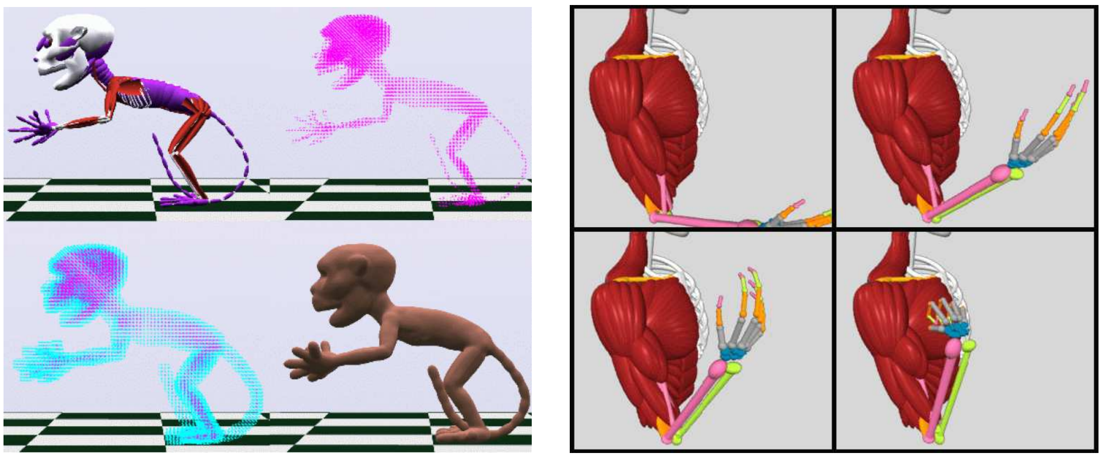
[Scheepers et al., 1997] and [Wilhelms, 1997] applied ellipsoids to model animal and human musculature. It is a natural choice because an ellipsoid approximates fairly well the appearance of a fusiform muscle. Ellipsoids can then be scaled on one of their axis depending on the skeleton motion, and muscle volume can be controlled compensating on the other axes the displacement on the first one (see figure 3.6). More recently, [Müller et al., 2005] presented a meshless deformation approach based on shape matching. The approach is very much geometrically motivated, which characterizes a non-physically based approach, even if the objects are embedded in a rather physical environment. Objects topology is not taken into account, they being seen as point clouds. The main idea is to replace the energies and forces of a physical method by respectively geometric constraints and distances of current positions to goal positions. Goal positions are determined by a shape matching with an undeformed rest state of the point cloud. Because the points are already driven to well defined locations, no explicit integration is necessary. The approach is particularly interesting for games (see figure 3.7).

3.2.2 Physically based methods

Non-physically based methods are useful in many cases, but for applications which demand realistic simulation of deformable physical bodies, the best choice is consistent physical modeling. This invariably means numerically solving the partial differential equations (PDEs) that govern the evolving shape of the deformable objects and their motion through space.

Then, the major problem on physical modeling is that:

- the physical phenomena to be simulated are extremely complex;
- solving the intrinsic PDEs requires considerable computational resources.



(a) [Wilhelms, 1997] (b) [Scheepers et al., 1997]

Figure 3.6: Muscles modelled with ellipsoids.

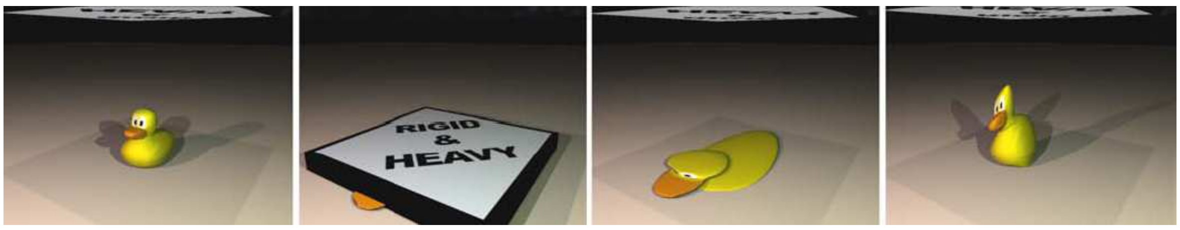


Figure 3.7: Meshless deformations. In an extreme squeezing situation the model remains stable, showing its ability to recover from a highly deformed situation (adapted from [Müller et al., 2005]).

Two answers can be given to that:

- finding an adequate simplified model of the given problem covering the observations relevant to the target application;
- applying efficient numerical techniques for solving the PDEs.

A number of approaches has been developed both on model simplification and on efficient numerical techniques. [Terzopoulos et al., 1987] and [Hégron and Arnaldi, 1992] present a comprehensive overview of the mechanical formalism for such physical models. In Computer Graphics, the physically based deformation methods mostly applied are: the mass-spring systems, the finite differences method, the finite elements method, the boundary elements method, and the particle systems.

The remaining of this section brings an overview on those methods and discuss some works derived from their basic principles.

3.2.2.1 Mass-spring systems

Mass-spring systems, also called mass-spring-damper (MSD), are a physically based technique that has been widely and effectively used in Computer Graphics for modeling deformable objects. An object is modelled as a collection of point masses connected by springs in a lattice structure 3.8. MSD can be seen as a simplified model of particle interaction, since physical bodies consist of discrete sub-elements (atoms and molecules). Springs connecting point masses exert forces on neighboring points when a mass is displaced from its rest position [Gibson and Mirtich, 1997]. For each point, applying the fundamental laws, we can write a vectorial equation:

$$m\ddot{\vec{x}} = \vec{F} \tag{3.1}$$

where m is the mass of a point, $\vec{x}$ is its position and $\vec{F}$ is the sum of all forces acting on it.

Applying equation 3.1 on all the points leads to a differential system of ordinary equations that can be solved explicitly using various algorithms.

One of the first works exploiting mass-spring systems for 3D simulation of deformable bodies was [Miller, 1988]. He dynamically simulated worms and snakes locomotion by controlling tensions inside the mass-spring lattice (see figure 3.9). He already dealt with many of the issues around this approach, like stiffness control, numerical instability, collision detection and response.

Mass-spring systems have been widely used in facial animation [Parke and Waters, 1996]. [Terzopoulos and Waters, 1990] used a three-layers mesh of mass points associated to three anatomically distinct layers of facial tissue (dermis, subcutaneous fatty tissue, and muscle). To improve realism, [Lee et al., 1995] added further constraints to prevent penetrations between soft tissues and bone. In biomechanical modeling, mass-spring systems were used by [Nedel and Thalmann, 1998] to simulate muscle deformation. Muscles were represented at two levels: action lines and muscle shape. This shape

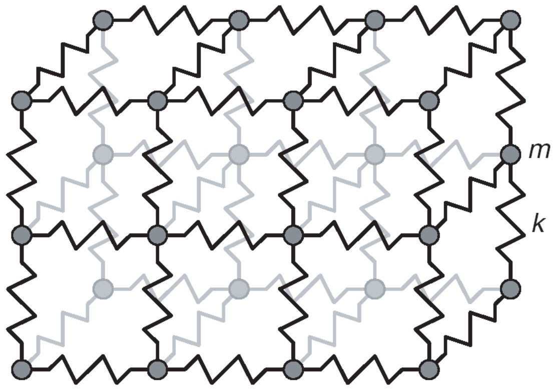


Figure 3.8: Mass-spring model (extracted from [Gibson and Mirtich, 1997]).

Figure 3.9: Eric the Dynamic Worm (adapted from [Miller, 1988]).

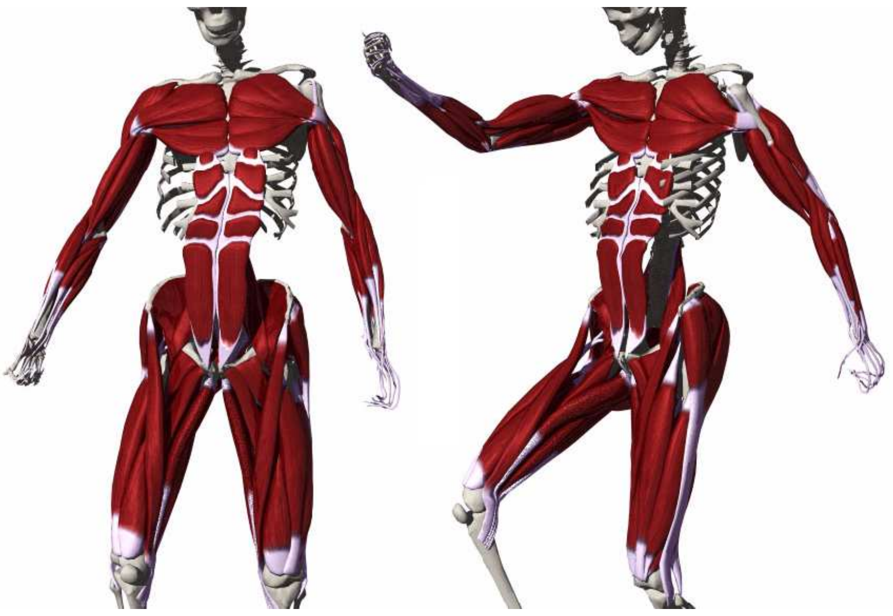


Figure 3.10: Examples of virtual humans with mass-spring modelled muscles (adapted from [Aubel and Thalmann, 2001]).

was deformed using a mass-spring mesh. In order to control the volume of muscles during deformation and smooth out mesh discontinuities, *angular springs* were introduced. These springs differ from other springs by the way they are attached. In that work, emphasis was put on interaction rather than simulation quality. [Aubel and Thalmann, 2000] and [Aubel and Thalmann, 2001] used a similar approach with a multi-layered model based on physiological and anatomical considerations. [Bourguignon and Cani, 2000] proposed a model offering control of the isotropy or anisotropy of elastic material. The basic idea of their approach is to let the user define, everywhere in the object, mechanical characteristics of the material along a given number of axes corresponding to orientation of interest. All internal forces will be acting along these axes instead of acting along the mesh edges. Figure 3.10 brings examples of mass-spring based virtual humans.

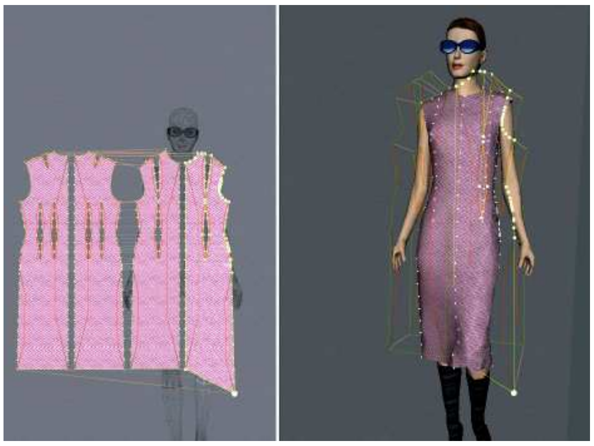
Mass-spring systems have also been used for cloth motion. [Lafleur et al., 1991] applied the method described in [Terzopoulos and Fleischer, 1988] combined with with collision avoidance to model cloths. [Carignan et al., 1992] animated simple cloths like a skirt according to the actor's motion in a physical environment (figure 3.11). [Provot, 1995] also developed a physical modeling technique based on the classic mass-spring model. He introduced constraints based on the rate of deformation to solve the problem of excessive deformation when simulating a hanging cloth. [Volino et al., 1995] present techniques for simulating the motion and the deformation of cloth, fabrics or, more generally, deformable surfaces. In particular, they consider difficult situations with respect to deformations and collisions, like wrinkled fabric falling on the ground. [Baraff and Witkin, 1998] describes a cloth simulation system that can take large timesteps without numerical instability. The method uses implicit integration to enforce constraints on individual cloth particles. A modified conjugate gradient method is used to

Figure 3.11: Simple cloths simulated with mass-springs (adapted from [Carignan et al., 1992]).

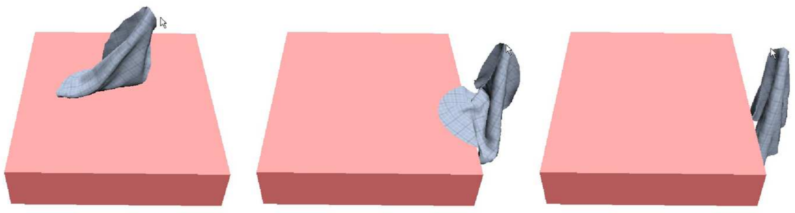
Figure 3.12: Fast simulation of cloths with mass-springs (adapted from [Baraff and Witkin, 1998]).

solve a sparse linear system at each time step. This results on a system significantly faster than the previous systems for cloth simulation (figure 3.12). More recently, [Vassilev et al., 2001], [Fuhrmann et al., 2003], [Volino et al., 2005], and others presented interactive or real-time approaches for clothing (figure 3.13).

In surgical simulation, [Promayon et al., 1996] presented a mass-spring model to simulate respiratory movements for an educational tool. [Brown et al., 2001] present algorithms for animating deformable objects in real-time for surgical training. It focuses on computing the deformation of an object subject to external forces and detecting collisions among deformable and rigid objects. To achieve realtime performance, the proposed algorithms take advantage of the facts that most deformations are local, human-body tissues are well damped, and motions of surgical instruments are relatively slow. They have been integrated into a virtual-reality system for simulating the suturing of small blood vessels (see figure 3.14). [Teschner et al., 2000]present a method using



(a)



(b)

Figure 3.13: Physics-based interactive cloth simulation (adapted from [Volino et al., 2005] (a), and [Fuhrmann et al., 2003] (b)).

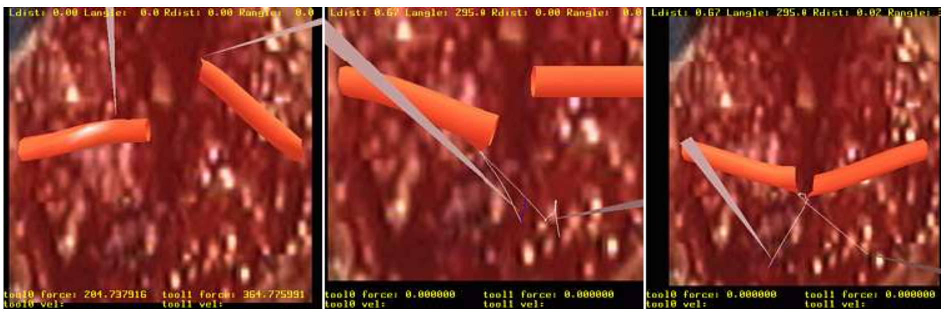


Figure 3.14: Surgery simulation for medical training (adapted from [Brown et al., 2001]).

mass-spring to model patient individual soft tissue for craniofacial surgical simulation. They compute nonlinear deformation and take into account the incompressibility of bio tissues. The idea is to compute the rest position of the deformed soft tissue instead of simulating the dynamic behavior, and thus achieving better efficiency and robustness.

Some works on mass-springs focus less on the application and more on the fundamental problems of this technique. [Hutchinson et al., 1996], for example, propose a method to adaptively refine mass-spring models according to precision requirement. It allows to reduce the number of mass-nodes to be processed globally by starting the simulation with a relatively coarse mesh and refining only when needed. Improving compromise between efficiency and realism was also the goal of [Teschner et al., 2004]. They simulate deformable tetrahedral and surface meshes elastically and plastically incorporating efficient ways for volume and surface area preservation (figure 3.15). Their method handles materials ranging from stiff to fluid-like behavior with computational efficiency similar to simple mass-spring. [Joukhadar et al., 1997], in turn, propose a method to determine the parameters of a mass-spring systems such that the mechanical behavior can be tuned to provide realistic behavior. Their method is based on a genetic algorithm approach. They first used a numerical method to distribute the masses along the volume, and then, given the desired behavior of an object represented by a set of constraints (position and orientation in function of time, maximum deformation, velocity, acceleration), they use the genetical analogy to evolve the set of parameters toward the satisfaction of the constraints.

Concluding, we can say that mass-spring models are easy to construct, and both interactive and real-time simulations of mass-spring systems are possible even with desktop systems. Another well-known advantage is their ability to handle both large displacements and large deformations.

However, mass-spring systems have some drawbacks. Since the model is tuned through its spring constants, good values for these constants are not always easy to derive from measured material properties. Furthermore, it is difficult to express certain constraints (like incompressibility and anisotropy) in a natural way. Another problem occurs when spring constants are large. Such large constants are used to model nearly rigid objects, or model non-penetration between deformable objects. This problem is referred as ”stiffness”. Stiff systems are problematic because of their poor stability,

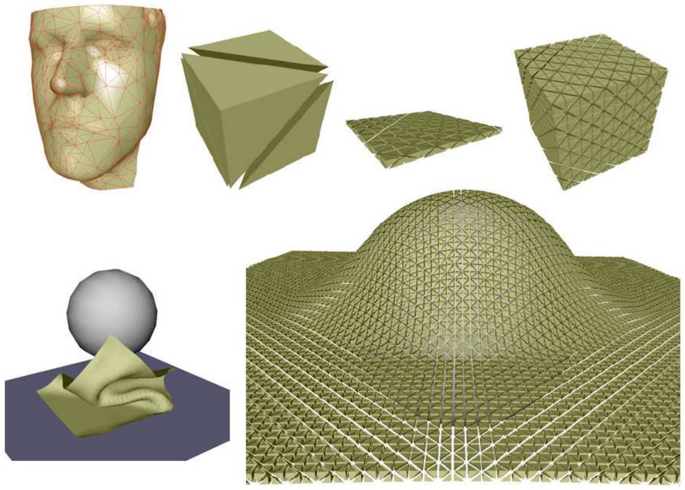


Figure 3.15: General tetrahedral and triangular meshes for efficiency tests (adapted from [Teschner et al., 2004]).

which requires small time steps for numerical integration resulting in slow simulation [Gibson and Mirtich, 1997].

3.2.2.2 Finite elements method

Whereas mass-spring models start with a discrete object model, more accurate physical models consider deformable objects as a continuum: solid bodies with mass and energies distributed throughout. Though models can be discrete or continuous, the method used for solving it is discrete. Finite element method is used to find an approximation for a continuous function that satisfies some equilibrium expression. In FEM, the continuum (object) is divided into elements joined at discrete node points. A function that solves the equilibrium equation must then be found for each element.

The basic steps in using FEM to compute object deformations are [Gibson and Mirtich, 1997]:

1. Derive an equilibrium equation from the potential energy equation of the deformable system in terms of material displacement over the continuum.

2. Select the appropriate finite elements (triangles, boxes, tetrahedra, etc.) and corresponding interpolation functions for the problem. Subdivide the object into elements.

3. For each element, re-express the components of the equilibrium equation in terms of the interpolation function and the element's node displacements.

4. Combine the set of equilibrium equations for all the elements in the object into a single system. Solve the system for the node displacement over the whole object.

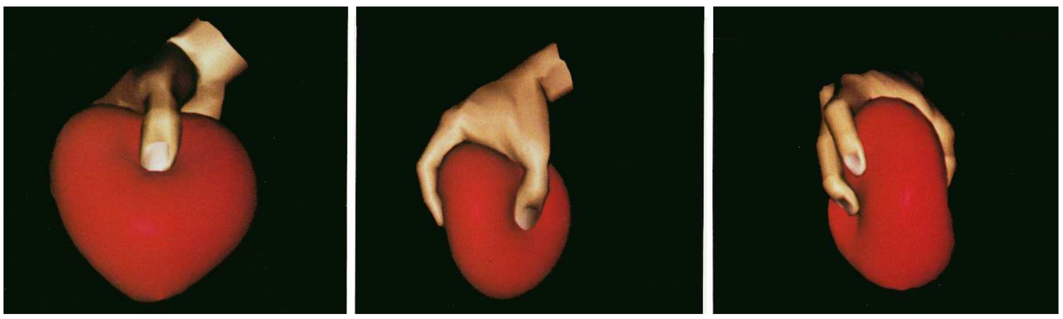


Figure 3.16: Grasping of an FEM ball submitted to internal pressure seen from various viewpoints (adapted from [Gourret et al., 1989]).

5. Use the node displacements and the interpolation functions of a particular element to calculate displacements or other quantities of interest (such as internal stress or strain) for points within the element.

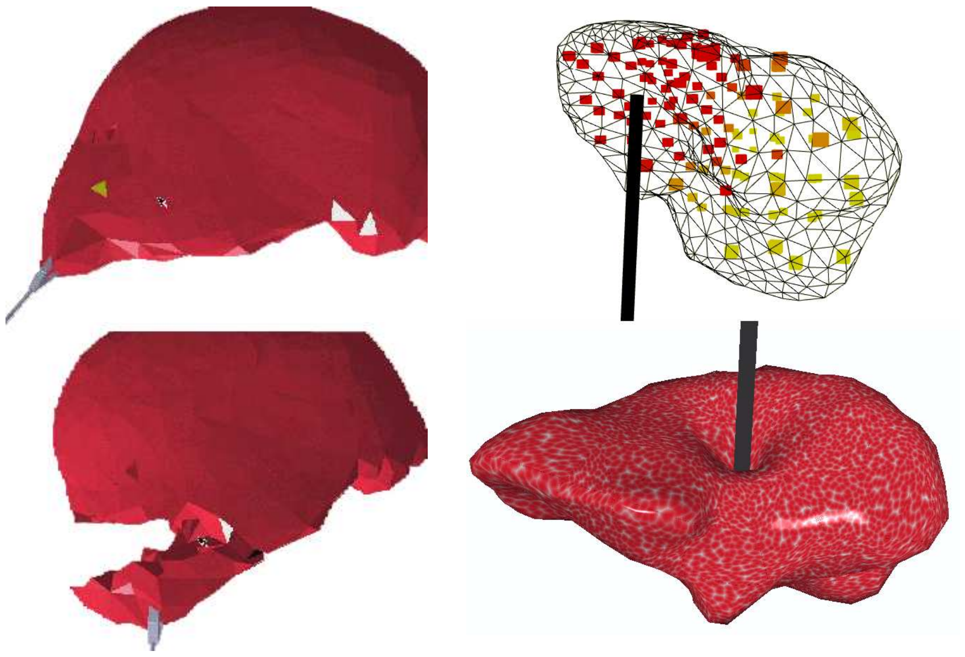
FEM has been widely used in soft tissues modeling. [Gourret et al., 1989] first generated tissue deformation by applying the engineering Finite Element Modeling technique. A human hand grasping an elastic ball is shown as an example (see figure 3.16). A linear constitutive law for the flesh tissue is 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:

$$K\mathbf{u} = f \tag{3.2}$$

where K is the stiffness matrix and $\mathbf{u}$ the displacement vector. Boundary conditions can be imposed by assigning fixed values to some components of $\mathbf{u}$. Theoretically, equation 3.2 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 large deformations.

[Chen and Zeltzer, 1992] proposed the use of FEM for biomechanically simulating skeletal muscle. They first demonstrated the suitability of the model for both visual applications and biomechanical analysis. They have tried to validate their model using videos of living muscles, what could be done only qualitatively. In front of the enormous difficulties to perform quantitative validation, they propose the use or MRI as a future work. Another specific work on muscles is [hong Zhu et al., 1998]. The authors presented a biomechanically based finite elements muscle model allowing prediction of muscle deformation. They utilize a voxel mesh reconstructed from CT and MRI, that is used for both FEM simulation and volume visualization. They simulate dynamically and statically a linear elastic model, and can achieve real-time on lwer resolution meshes.

To achieve real-time deformation, reducing computing time is necessary. [Bro-Nielsen and Cotin, 1996] studied this problem by using a condensation technique. With this method, the computation time required for the deformation of a volumetric model



(a) [Stéphane Cotin, 1998] (b) [Gilles Debunne and Cani, 1999]

Figure 3.17: Models of the liver from different approaches.

can be reduced to the computation time of a model only involving the surface nodes of the mesh. Finite element methods provide a more realistic simulation than mass-spring methods but are computationally less efficient. In addition, the linear elastic theory used to derive the potential energy equation assumes small deformation of the object, which is true for materials such as metal. However, for soft biological material, objects dimensions can deform in large proportions so that the small deformation assumption no longer holds. Because of this change, the amount of computation required at each time is greatly increased. [Roth et al., 1998] opposingly focus on the highest possible accuracy instead of speed. The extend the usual linear elasticity approaches towards incompressibility and nonlinear behavior. On the top of their model they propose a facial surgery simulation application using the Visible Human Data Set. They call attention to the fact that case studies on individual patient models are necessary to develop a prototype medical application that can be cross evaluated with actual surgical procedures.

[Stéphane Cotin, 1998] describe a surgery simulator prototype. They focus on the interaction of the rigid surgical tools with the soft biological tissues. They first modelled soft tissues with a linear elastic FEM that does not allow cuts. Then, to allow cuts, they proposed another method based on mass-spring-like model, which in turn has to be limited to a few nodes to run in realtime. They finally combine the two approaches into a hybrid one allowing real-time deformation and cutting on large enough anatomical structures, like the liver (figure 3.17(a)).

[Cotin et al., 1999] focus on a specific application: surgery simulation for training. Then, they focused on a very detailed geometrical model, leaving the physical

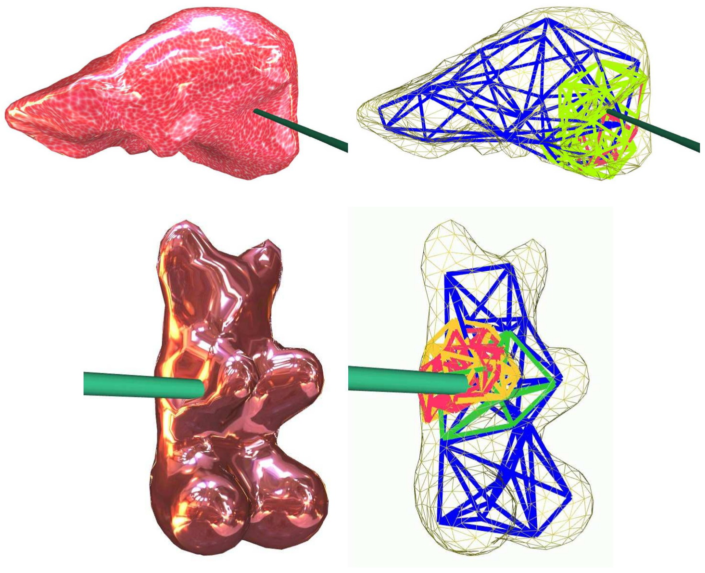


Figure 3.18: Space and time adaptive sampling. The approach makes use of a local refinement technique to ensure high physical fidelity while bounding the global computing load to guarantee real-time animations (adapted from [Debunne et al., 2001]).

accuracy in a second plan. In fact, they use a FEM for precomputing elementary deformations, that are later used as assumptions to simplify the online simulation. They also integrated a force-feedback device to improve the realism of a virtual laparoscopy simulation. They qualitatively evaluated their application with surgeons specializing in laparoscopic surgery, who reported sensations very close to reality. [Debunne et al., 2000] present a new method for computing dynamic simulations exploiting a set of 3D meshes of different resolutions for the same object. The method is closely related to FEM, though it does not solve a global system matrix. Instead, the force computation for each node is local and involves only the neighboring nodes. According to deformation extent or other criteria, the method switches between levels of detail, concentrating the available computing time only where needed. [Wu et al., 2001] also aimed at real time FEM using a hierarchy of meshes of different resolutions. However, they used dynamic progressive meshes for mesh adaptation and a nonlinear finite elements formulation. In [Debunne et al., 2001], the principal innovation is the use in addition to the spatial levels of detail also temporal levels of detail. This, which they called a space and time adaptive level of detail, was used in combination with a large displacement strain tensor formulation. To solve the system, explicit FEM was used where each element is solved independently through a local approximation, which reduces computational time (example in figure 3.18).

[Hirota, 2001] used FEM in simulation of mechanical contact between nonlinearly elastic objects. The mechanical system used as a case study was the human leg, more precisely, the Visible Human right knee joint and some of its surrounding bones, muscles, tendons and skin (see figure 3.19. The approach relied on a novel penalty finite element formulation based on the concept of material depth to compute skin, tendons

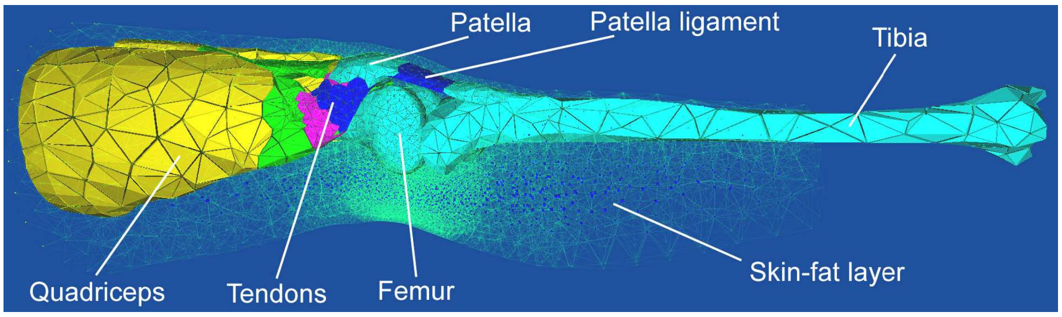


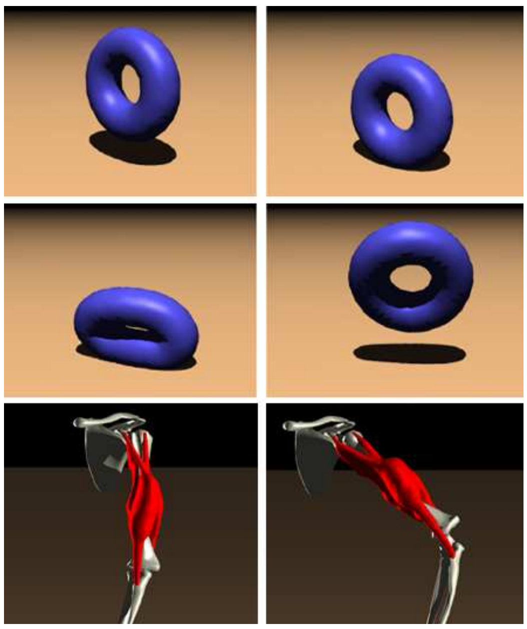
Figure 3.19: The constituent parts of the leg finite elements model derived from the Visible Human database (extracted from [Hirota, 2001]).

and muscles deformation. By linearly interpolating pre-computed material depths at node points of the finite element mesh, contact forces can be analytically integrated over contacting regions without raising computational cost. The algorithm was implemented as a part of an implicit finite element program for dynamic, quasi-static and static analysis of nonlinear viscoelastic solids. High nonlinearity and anisotropy typically observed for biological materials were also supported. [Lemos et al., 2001] also dealt with geometric non-linearity induced by large deformations. Although, the non-linear FEM solver in [Lemos et al., 2001] uses eight-node brick-like elements instead of tetrahedra. They apply their non-linear FEM solver for simulation of leg muscles of a cat. Both [Hirota, 2001] and [Lemos et al., 2001] the equation of motion is linearized for each integration step takin Newton-Raphson steps.

[Picinbono et al., 2003] presented new extensions to the biomechanical model of the INRIA's virtual hepatic surgery prototype. The model now takes into account anisotropic behavior and incompressibility properties of bio-tissues. They also solver the problem of rotational invariance, and optimized the model to compute nonlinear forces only for mesh parts undergoing large displacements. [Hauth et al., 2003] focused more on the material description, and used a conventional FEM to solve the system.

[Teran et al., 2003] present the finite volume model (FVM) as a geometrically intuitive alternative for FEM. It is said to be as simple as mass-spring to implement and understand, and as powerful as FEM for both Biomechanics and Graphics, even if heuristic methods are used for force calculation. The main advantage to mass-spring, and which brings FVM closer to FEM, seem to be the use of an arbitrary constitutive model. The advantage to FEM is the very reduced algorithmic complexity, and consequently the calculation time. [Irving et al., 2004] propose a technique to achieve robust simulation of large deformations, including situations involving degeneracy, complex geometries, anisotropy and plasticity. They modify the constitutive model of an FEM, by diagonalizing the deformation mapping prior to computing forces, in a way that it behaves robustly even for inverted elements. The performances, nevertheless ar far to be high. For the smallest meshes it takes 10 to 20 seconds per frame, and for the largest, about 20 minutes per frame. Figure 3.20 puts some similar results from [Teran et al., 2003] and [Irving et al., 2004] face-to-face.

On a more game-directed realtime approach, [Müller et al., 2004] propose a method


(a) [Teran et al., 2003]

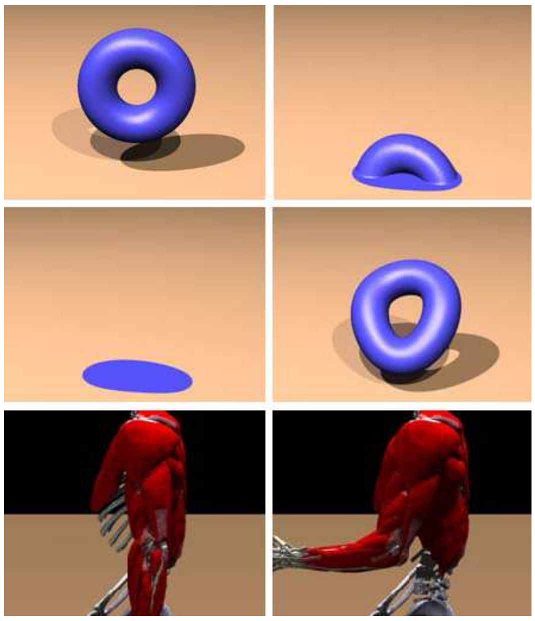

(b) [Irving et al., 2004]

Figure 3.20: Similar problems dealt with by quite different robust approaches. The models in (a) are realtime; the ones in (b) are biomechanically accurate.

for physically-based animation of objects defined by their surfaces only. They voxelize the objects into a cubic finite elements model. The actual surface elements are finally deformed according to the displacement of their respective cubes. Besides deformation, they also show successful examples of fracture (figure 3.21).

3.2.2.3 Other techniques

The finite differences method (FDM) is historically the first true discretization technique for solving partial differential equations. The general approach with FDM is replacing the continuous derivatives within the given boundary value problem with finite difference approximations on a grid of mesh points that spans the domain of

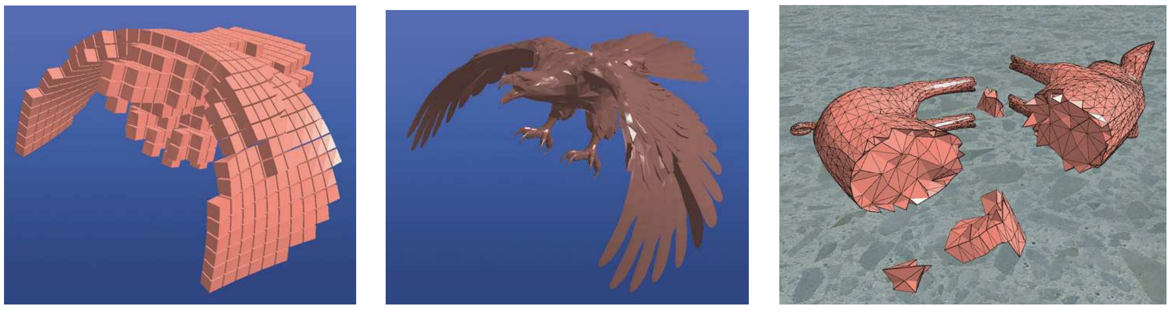


Figure 3.21: Deformable eagle FEM cube mesh (a) and surface mesh (b). A fractured pig model (c). (adapted from [Müller et al., 2004]).

interest. Consequently, the differential operator is approximated by an algebraic operator, and the resulting system of equations can be solved by a number of standard techniques. The FDM achieves efficiency and accuracy when the geometry of the problem is regular, like the cubic grids given by pixels or voxels from 2D or 3D images respectively. However, the discretization of objects with complex geometry becomes extremely dense, requiring too much computational resources. [Sarti et al., 1999] applied FDM on craniofacial surgery planning.

A general principle for solving the boundary value problem given by the partial differential equation and the boundary conditions consists in bringing the differential problem into an integral form. For a certain class of problems, the resulting integration over the whole domain of interest can be substituted by the integration over the boundary. Consequently, only the boundary of the domain has to be discretized, which means that:

- the dimension of the resulting system of equations is significantly smaller;
- the problem of volumetric mesh generation becomes redundant.

In [Beskos, 1989], the boundary elements method (BEM) is described for static and dynamic problems of continuous mechanics. Unfortunately, the volume integrals in the BEM can be completely eliminated only if the material ie homogeneous and no volumetric forces are given. This is generally not the case in soft tissue modeling.

Example applications using BEM are in [Monserrat et al., 1999] and [James and Pai, 1999]. [James and Pai, 1999] presented a fast method for simulating linear elastic deformable objects in interactive applications using BEM. They have shown that it is useful in graphics applications because of the accuracy and simplicity derived from its boundary-only nature. Specifically, for the small localized changes in boundary conditions which are typical of user interaction, the algorithm is linear in complexity. Examples from their ArtDefo system are shown in figure 3.22. [James and Pai, 2003] use multi-resolution Green's function methods to achieve interactive simulation with BEM. They build on Capacity Matrix Algorithm approaches for simulating precomputed Green's function models associated with discrete approximations of linear elliptic partial differential equations, such as for elastostatic objects. The enhancements presented extend the complexity of objects that can be interactively simulated.

Particle systems is in fact a computer graphics technique often used to simulate certain fuzzy phenomena, which are otherwise very hard to reproduce with conventional rendering techniques. Examples of such phenomena include fire, explosions, smoke, flowing water, sparks, falling leaves, clouds, fog, snow and dust. However, particle systems can also be effective to model deformable objects. [Ganovelli et al., 2000] proposed a particle system for modeling human organs. The defend the idea that the topological information necessary for other techniques is a great source of time consuming problems. With a model without topology, like particle systems, cuts and lacerations can be simulated with better performance. [Hieber et al., 2004] present a particles model for the simulation of human organs mechanical behavior in their work on computer aided surgery. They modelled kidney- and liver-like materials as

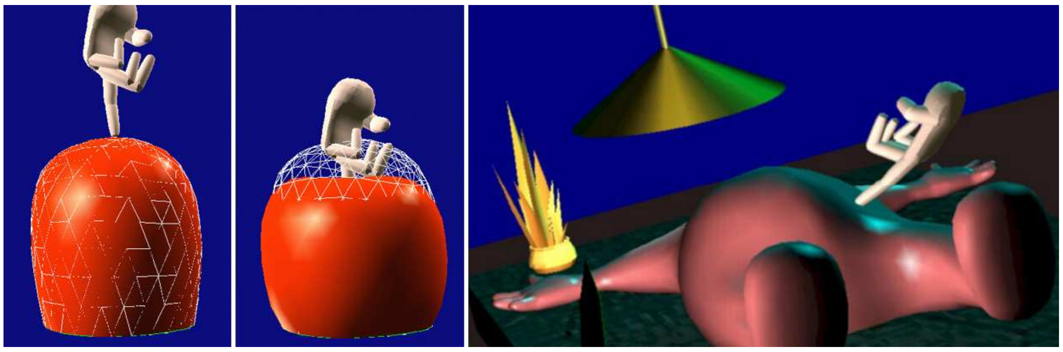


Figure 3.22: Examples of BEM deformation from ArtDefo. At left the detail of the compressibility test, and at right an interactive simulation (adapted from [James and Pai, 1999]).

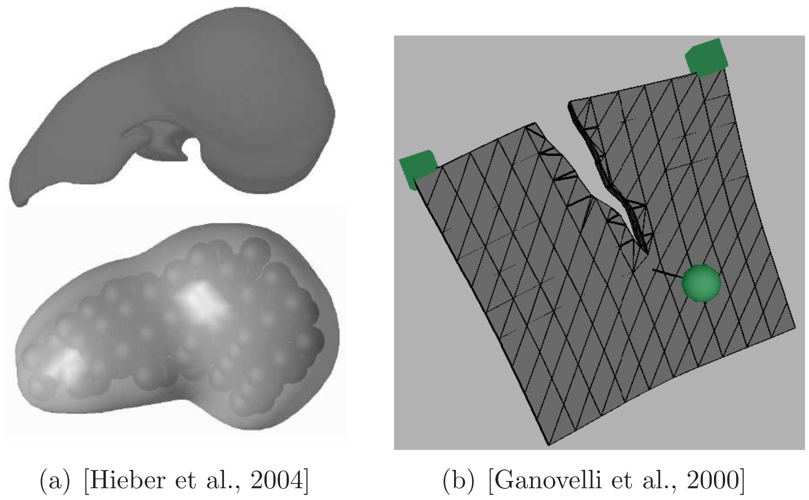


Figure 3.23: Examples of solids modelled as particle systems for surgery simulation.

linear viscoelastic solids. The constitutive equations for the material behavior were discretized using a particle approach based on the Smoothed Particle Hydrodynamics method. Figure 3.23 illustrates the works cited. [Gilles Debunne and Cani, 1999] show a case study on the liver (figure 3.17(b)). They achieve interactive rates using space-time adaptive resolution while integrating the PDE of their particle system with simple differential operators.

[Müller et al., 2002] have developed new techniques to achieve realtime animation of deformable objects that can be adapted on both the classical FEM and mass-spring approaches. They propose warping the constant stiffness matrix of the system along a rotation field. With that, they can handle large rotational deformations as fast as the linear approaches without the well known visual artifacts of such approaches. An example is shown in figure 3.24. [Pauly et al., 2004] introduce the concept of quasi-rigid objects, which are intermediate between rigid and deformable objects and combines their benefits. It also presents a contact handler model that fills the gap between methods to handle contact of rigid bodies and methods to handle contact of

(a) [Hieber et al., 2004] (b) [Ganovelli et al., 2000]

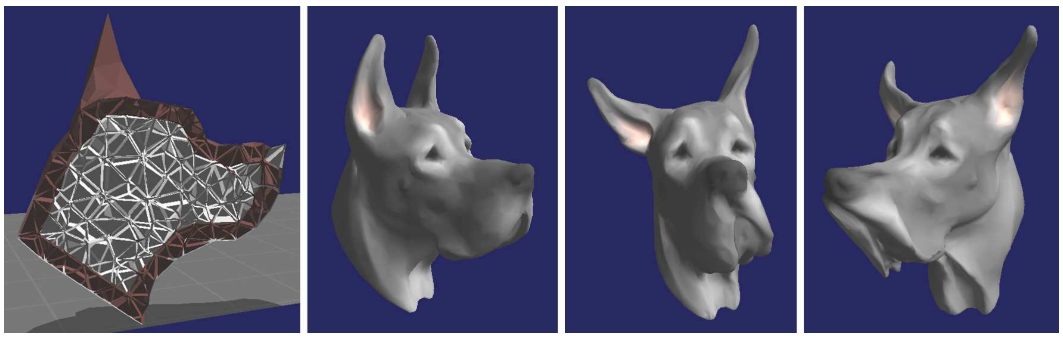


Figure 3.24: The great dane's soft skin is animated in realtime during skull movement (adapted from [Müller et al., 2002]).

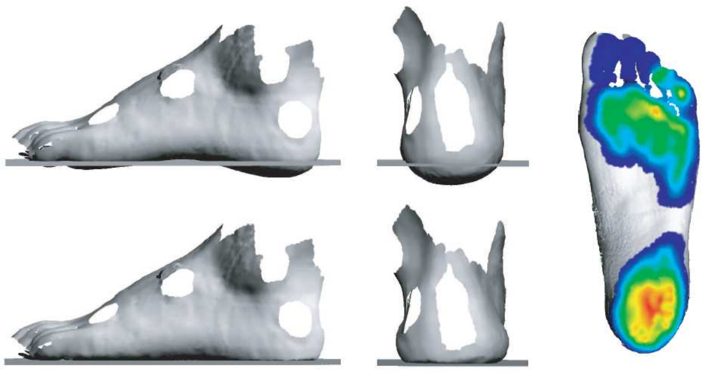


Figure 3.25: Example of quasi-rigid model. Contact of a deformable human foot with a rigid ground plane. Color mapping denotes normal traction from zero (blue) to maximum (red) (adapted from [Pauly et al., 2004]).

deformable solids (see figure 3.25).

3.3 Bio-tissues properties

Anthropometry is the science and practice of measuring the human body and its parts. Measure the properties of biological materials and tissues is also Anthropometry's task. Biomechanics, in turn, studies the structure and function of biological systems using the methods of mechanics [Fung, 1972]. These and other sciences have been studying the composition and behavior of bones, cartilages and ligaments for many years. However, although we know much about these tissues, newer and better measurement techniques continuously change the available data concerning their mechanical properties [Fung, 1993]. This section introduces the terms and procedures involving biological materials, and identifies consensual or average physical data regarding the tissues relevant in the study of the human hip joint capsule behavior. In this group we include: skeletal bones, articular cartilage, and ligaments.

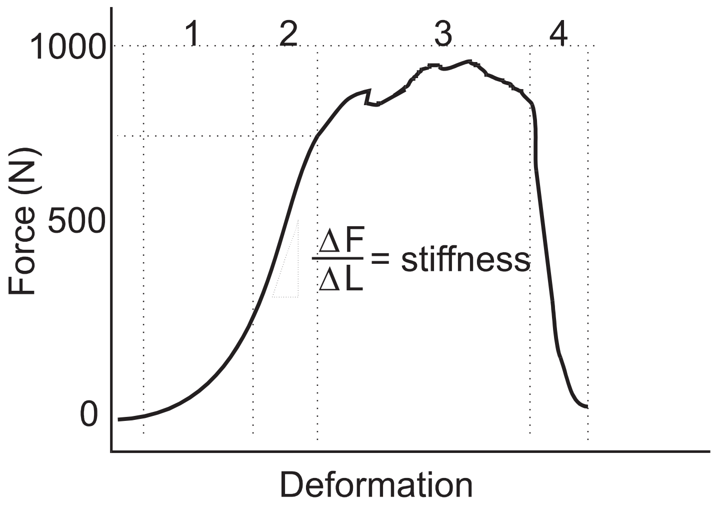


Figure 3.26: The 4 regions of a standard stress-strain curve obtained from a test with ligamentous tissue. Non-linear response in the region 1 is due to the straightening of the crimp pattern. In the region 2 the crimp is lost and further deformation stretches the collagen fibers themselves. Microstructural damage occurs in the region 3. Further stretching, in region 4, causes progressive fiber disruption and ultimately complete ligament rupture [Hawkins, 2002]

3.3.1 Terms and definitions

- Stress-Strain
  Stress is defined as a pressure or a tension exerted on an object. It is calculated dividing the force by its area of application. Strain, in turn, is the amount of deformation, and is calculated dividing the change of length by the initial length. See equations 3.3 an 3.4. Hence, a tissue's material properties may be obtained from force-elongation test data dividing the recorded force by the original cross-sectional area to give stress, and by dividing the variation of the specimen length by its original length to give strain. It allows constructing a stress-strain diagram that approximates the material's stress-strain behavior independently of the sample dimensions. Stress-strain, also called force-elongation, curves are typically described in terms of four regions. These four regions are illustrated in the ligament test of figure 3.26.

$$\sigma = \frac{\vec{F}}{A} \tag{3.3}$$

$$\varepsilon = \frac{\Delta l}{l_0} \tag{3.4}$$

- Structural vs. Material Properties
  A biological tissue is often described in terms of its structural and material properties. Structural properties characterize the tissue in its intact form. Important

structural properties are represented by a relationship between force and deformation, and must be understood in order to predict how a tissue will behave in-vivo. Material properties characterize the behavior of the material comprising the tissue and to a first approximation are independent of the size of the tissue. The material properties are usually expressed in terms of the stress-strain relationship of the material. Structural and material properties curves are similar in appearance, differing only by a scaling factor.

- Stiffness (Elasticity)
  The stiffness of a material represents the materials ability to resist to deformation. Stiffness is commonly characterized by the slope of the linear region of a stress-strain curve, also referred to as Young's Modulus. To describe the slope of other regions of the stress-strain curve a Tangent Modulus is often defined. If a Tangent Modulus is defined it should have associated with it a strain value or a range of strains. There can be different modules depending on the loading conditions (e.g. shear modulus, compression modulus). The larger the stiffness, the greater the force required to cause a given deformation. If the stress in a material is directly proportional to the strain for all strains, the material is called a *Hookean* material.

- Anisotropy and Non-homogeneity
  Ideal materials are isotropic and homogeneous. A material is called isotropic when its properties are the same in each of three coordinate axes (x,y,z). Tensile and compressive properties may be different, but each respective property must be the same in three directions. A material is said to be homogeneous if it is made of the same material throughout. Biological tissues are, instead, anisotropic and non-homogeneous.

- Viscoelastic Properties
  Biological tissues are viscoelastic materials. It means that their behavior is time and history dependent. A number of different behaviors characterizes a material as viscoelastic. They are: stress-relaxation, creep, strain rate sensitivity, and hysteresis.

  Force-relaxation (or stress-relaxation) is a phenomenon that occurs in a tissue stretched and held at a fixed length. Over time the force present within the tissue continually declines. Force-relaxation is strain rate sensitive. In general, the higher the strain rate, the larger the peak force and subsequently the greater the magnitude of the force-relaxation.

  In contrast to stress-relaxation, that occurs when a tissue's length is held fixed, **creep** occurs when a constant force is applied across the tissue. When subjected to a constant tensile force, a tissue elongates with time. The general shape of the displacement-time curve depends on the past loading history (e.g. peak force, loading rate).

  Another time-dependent property is **strain rate sensitivity**. Different tissues show different sensitivities to strain rate. For example, there may be little difference in the stress-strain behavior of ligaments subjected to tensile tests varying in strain rate over three decades while bone properties may change considerably.

Besides, the *loading* and *unloading* curves obtained from a force-deformation test of biological tissues, for example, do not follow the same path. The difference in the calculated area under the loading and unloading curves is termed the area of **hysteresis** and represents the energy lost due to internal friction in the material. The amount of energy liberated or absorbed during a tensile test is defined as the integral of the force and the displacement.

- Viscosity
  The viscosity of a fluid is a measure of the fluid's resistance to flow. Viscosity of water is used as reference to calculate other fluids viscosity, and is considered to be 1. The capsule of diarthrodial joints is normally filled with a fluid of viscosity 10 called synovial fluid. This fluid helps to reduce friction and wear of articulating surfaces. Just for comparison, the viscosity of olive oil, for example, is 84 [Hawkins, 2002].

- Coefficient of Friction

  The coefficient of friction is that fraction of the force transmitted across two bearing surfaces that must be used to initiate movement (s -static friction) or keep the surfaces moving (d -dynamic friction). The static coefficient of friction between two surfaces is always greater than the dynamic coefficient of friction. Joints of the human body are well designed to reduce the coefficient of friction between articulating surfaces. See the section 4 that is dedicated to cartilage.

- Testing procedures

  Structural and material properties of biological tissues are usually determined through some form of mechanical testing (e.g. tensile tests, compressive tests, bending and torsion tests). Customized workstations utilizing force transducers, clamps, and an actuator to control the distance between clamps are commonplace. Commercial systems are also available and vary in design depending on the type of tissue being studied (e.g. macroscopic vs. microscopic, hard tissue vs. soft tissue etc.) and the type of loading rates required. *Instron* [Instron] and *MTS* [MTS] are the two most common suppliers of mechanical testing systems. Most systems allow either force control or length control. See pictures in figure 3.27. Mechanical testing of tissue in-vivo is very difficult and hence not commonly performed. Some of the techniques that have been utilized include: 1) buckle transducers to monitor tendon and ligament forces, 2) telemetered pressure sensors to measure joint contact pressure, and 3) strain gauges to quantify bone and ligament strain. More general procedures include the works of Nava [Nava et al., 2003] and Valtorta [Valtorta and Mazza, 2004] which used respectively aspiration and torsional resonance tests in-vivo to determine bio-tissues properties. Some non-invasive approaches have also been employed. Ultrasound techniques have been used to detect changes in the speed of sound in different tissues and these changes have been correlated with the tissue's elastic properties.

Various imaging techniques have also been used to quantify tissue geometry and deformation [Hawkins, 2002].

3.3.2 Properties of materials

In this section, a brief presentation of the material properties of the main tissues involved in a joint is done. Muscles and tissues of superior layers will not be discussed here because of their active role (muscles produce force) and their less significant influence on the joint range of motion. So, we see the properties of bone, cartilage and ligament.

3.3.2.1 Bone

Bone is identified as either cancellous (also referred to as trabecular or spongy) or cortical (also referred to as compact), see figure 3.28. Cortical bone is roughly 4 times the mass of cancellous one in any long bone. The basic material comprising cancellous and compact bone appear identical, thus the distinction between the two is the degree of porosity and the organization. The porosity of cortical bone ranges from 5 to 30% while cancellous bone porosity ranges from 30 to 90%. Bone porosity is not fixed and can change in response to altered loading, disease, and aging.

Cancellous bone is actually extremely anisotropic and inhomogeneous. Cortical bone, on the other hand, is approximately linear elastic, transversely isotropic and relatively homogenous. The material properties of bone are generally determined using mechanical testing procedures; however ultrasonic and MRI techniques have also been employed [Kallel and Ophir, 2000]. Force-deformation (structural properties) or stress-strain (material properties) curves can be determined by means of such tests. However, the properties of bone and most biological tissues depend on the freshness of the tissue. These properties can change within a matter of minutes if allowed to dry out. Cortical bone, for example, has an ultimate strain of around 1.2% when wet and about 0.4% if the water content is not maintained. Thus, it is very important to keep bone specimens wet during testing. Bone shows a linear range in which the stress increases in proportion to the strain. The slope of this region is defined as Young's Modulus or the Elastic Modulus. An illustration of the material properties of bone relative to other materials is shown in figure 3.29.

Material properties of the two types of bone differ. Cortical bone has an elasticity (Young's) modulus of 12 GPa (longitudinally) and 13.4 GPa (transversely), while cancellous has an elasticity of 0.39 GPa. Cortical bone is stiffer than cancellous bone. It can sustain greater stress but less strain before failure. Cancellous bone can sustain strains of 75% before failing in-vivo, but cortical bone will fracture if the strain exceeds 2%. Cancellous bone has a greater capacity to store energy compared to compact bone. Children's bones tend to absorb more energy before failure compared to adults (as much as 45% more). Children's bones are weaker, but more compliant (children's bones can be 68% as stiff as adult bone).

Figure 3.27: Examples of mechanical testing systems [Instron][MTS].

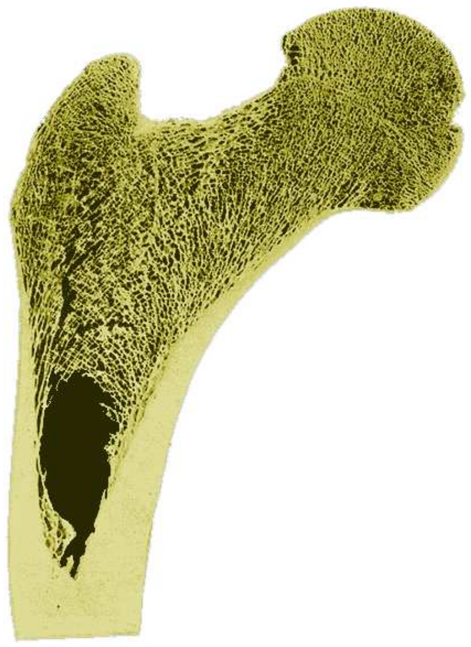


Figure 3.28: Frontal longitudinal midsection of the femur showing the bone internal structure with its different layers (modified from [Gray, 2000]).

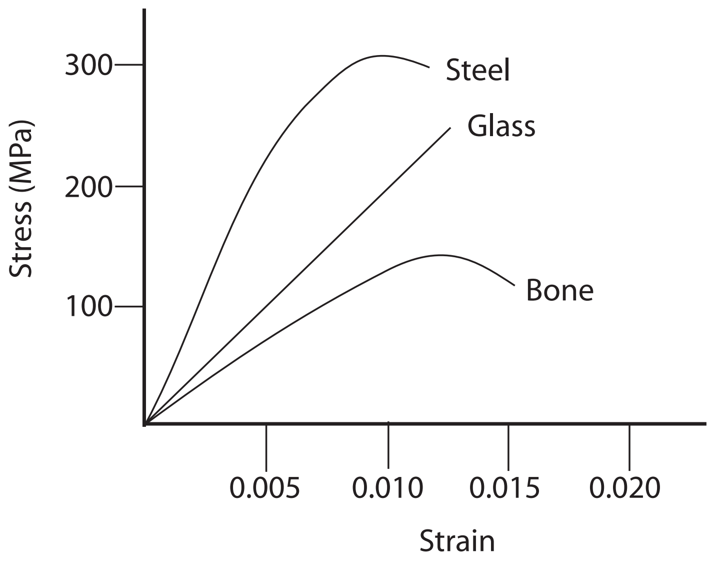


Figure 3.29: Comparative of stress-strain curves of bone with other materials (adapted from [Hawkins, 2002]).

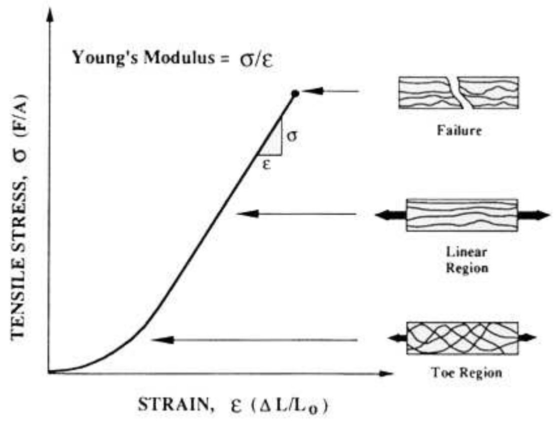


Figure 3.30: Cartilage stress-strain curve. Young's modulus is defined by the slope of the linear region (adapted from [Mow et al., 1991]).

3.3.2.2 Cartilage

Articular cartilage, also called hyaline cartilage, is made of a multiphasic material with two major phases: a fluid phase composed by water (68-85%) and electrolytes, and a solid phase composed by collagen fibrils (primarily type II collagen) (10-20%), proteoglycans and other glycoproteins (5-10%), and the chondrocytes (cartilaginous cells) [Mow and Hayes, 1997]. The cartilaginous tissue is extremely well adapted to glide. Its coefficient of friction is several times smaller than the one between the ice and an ice skate. There are electrostatic attractions between the positive charges along the collagen molecules and the negative charges that exist along the proteoglycan molecules. Hydrostatic forces also exist as forces are applied to cartilage and the fluid tries to move throughout the tissue. It is the combined effect of all these interactions that give rise to the mechanical properties of the material. Like bone, the properties of cartilage are anisotropic. The anisotropy results in part from the structural variations. Because of its structure cartilage is rather porous allowing fluid to move in and out of the tissue. When the tissue is subjected to a compressive stress, fluid flows out of the tissue. Fluid returns when the stress is removed. The mechanical properties of cartilage change with its fluid content, thus making it important to know the stress-strain history of the tissue to predict its load carrying capacity. The material properties also change with pathology. The compressive aggregate modulus for human articular cartilage correlates in an inverse manner with the water content and in a direct manner with proteoglycan content per wet weight. There is no correlation with the collagen content thus suggesting that proteoglycans are responsible for the tissue compressive stiffness. Some relevant data for cartilage mechanics compiled from several sources are 0.08 to 2.3 MPa for Young's modulus, and 0.08 to 0.41 for Poisson's ratio [Korhonen et al., 2002] [Nieminen MT, 2004]. Figure 3.30 shows the typical cartilage stress-strain curve.

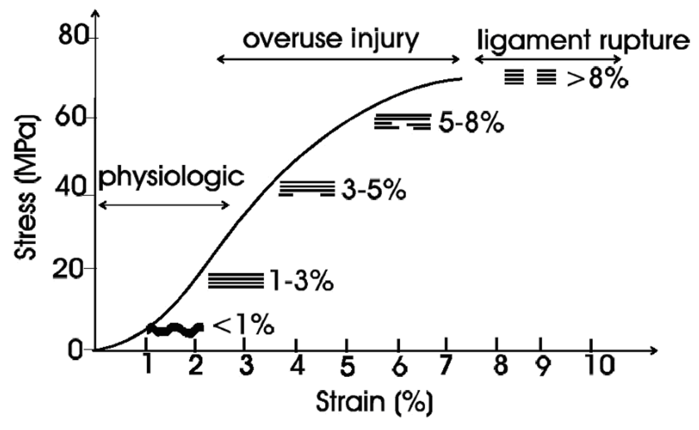


Figure 3.31: Ligament stress-strain relationship (adapted from [Butler et al., 1984]).

3.3.2.3 Ligament

Although significant advances have been made in the biology, biochemistry, and mechanics of soft tissue biomechanics, there is limited information available on in-vivo tissue mechanical characteristics and behavior. Without accurate values of such in-vivo information, extrapolations from animal and human insitu bone-ligament-bone testing to the function of intact human ligaments can not be made confidently. Currently, we know that ligaments are composite, anisotropic structures exhibiting non-linear time and history-dependent viscoelastic properties. Described in this section are the mechanical behavior of ligamentous tissue, the physiological origin of this behavior, and the implications of such properties to ligament function during normal joint motion. As seen in section 3.3.2.1, the force-elongation curve represents rather structural properties of the tissues. Material properties, in turn, are more generally expressed in terms of a stress-strain relationship. See for stress-strain relationship of ligament.

Ligaments have characteristics of strain rate sensitivity, stress relaxation, creep, and hysteresis. They exhibit significant time- and history-dependent viscoelastic properties. Time-dependent behavior means that, during daily activities, ligaments are subjected to a variety of load conditions that affect their mechanical properties. For example, they become softer and less resistant after some minutes of running, returning to normal hardness when the exercise is interrupted. History-dependency, in turn, means that frequent intense activities will change the tissue properties in a medium term basis. For example, the ligaments of an athlete, after 6 months of daily training, will become softer and thus more adapted to the intense exercise, even when he is not training. In the same way, if the activities are interrupted for some months, the ligament properties will go back to normal levels. Ligaments are also temperature and age sensitive.

Mechanical data reported in literature present a very wide range. They depend of the ligament being tested, the specimen condition, the force applied and many other factors. Thus, they can give a general idea of magnitude but are not applicable for definitive conclusion on a specific case. Young's modulus from several sources vary from 1.0 to 4.8 MPa at 0% of failure strain, and from 76 to 285 MPa at 80% of failure strain. Ultimate strength is reported to be between 2 to 6 Mpa for the hip ligaments,

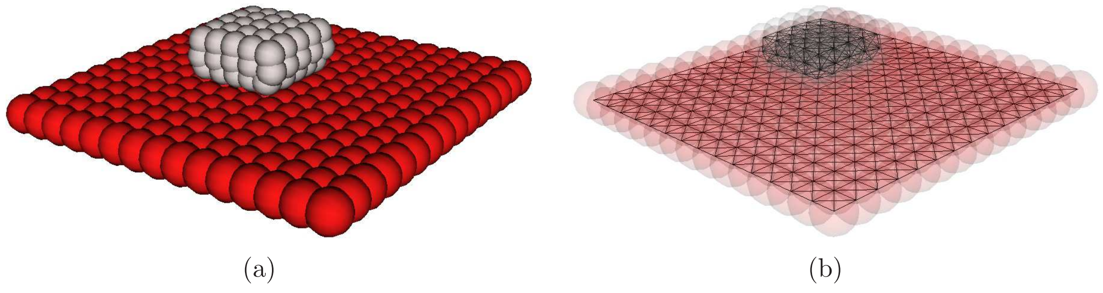


Figure 3.32: Samples of deformable tissue in contact. The grey object has just fallen on the red one. In (a) they appear shaded; in (b) with transparency to show spring connections.

but it also depend of structural features. Poisson's ratio vary from 1.1 to 2.0 [Hewitt et al., 2001] [Hewitt et al., 2002] [Stewart et al., 2002] [Weiss et al.].

3.4 Our molecular model

Our approach to model soft biological tissues is presented in this section (figure 3.32). The idea was inspired in [Jansson and Vergeest, 2002] that was applied in computer aided design for interactive shape modeling and analysis. Their work is founded on a generalized mass-spring model - which they call molecular model - where mass points are, in fact, spherical mass regions called molecules. This approach is an analogy of the relationship of the real matter molecules with they neighborhood in a piece of material. Elastic forces are then established between molecules by a spring-like connection. Starting from that, we extended the use of the spherical volume and adapted the model for dynamic simulation of biological structures.

In [Jansson and Vergeest, 2002], generality and simplicity were more important than metric correctness. This because it is known that with a classical mass-spring approach (see section 3.2.2.1, setting up the parameters of the model is not trivial. This problem is still more intricate when the behavior of a complex real material is intended to be faithfully represented. That is the case in the present work. We wish that properties of materials will be taken into account to define the stiffness of the spring-like connections. Then, much of the novelty of our model is in characterizing physical properties of materials within a mass-spring based approach [Maciel et al., 2003a].

In this section, after introducing the basic aspects of the molecular model we explain how we configure the model parameters in order to control the material elasticity. We also describe the method we used to discretize the volumes, how we cover the bulk of spheres with a smooth skin, we discuss numerical integration issues, and finally illustrate a case we implemented using a haptic device for force feedback.

3.4.1 The force model

The model is described by two sets of elements: E, a set of spherical elements (molecules), and C, a set of connections between the elements in E (equation 3.5).

$$E = \{e_1, e_2, \cdots e_n\}; C = \{C_{e_1}, C_{e_2}, \cdots C_{e_n}\}; C_e = \{C_1, C_2, \cdots C_n\} \tag{3.5}$$

The model behavior is determined by the forces produced on each element of E by each connection of C and some external forces.

$$\vec{F_e} = \vec{F_G} + \vec{F_L} + \vec{F_C} + \vec{F}_{collisions} \tag{3.6}$$

$$\begin{aligned} \vec{F_G} &= m_e \vec{g} & (3.7) \\ \vec{F_L} &= -\Pi r_e^2 \rho |\vec{V_e}|^2 \frac{\vec{V_e}}{|\vec{V_e}|} & (3.8) \\ \vec{F_C} &= \vec{F_b} + \vec{F_d} + \vec{F_f} & (3.9) \end{aligned}$$

where:

$\vec{F_G}$: gravity (m_e is the mass of e and g is the gravity acceleration);

$\vec{F_L}$: ambient viscous friction (r is the radius, ρ is the medium density and V is the velocity);

$\vec{F_C}$: connection forces, see equation 3.10.

The force from connections is given by:

$$\vec{F_C} = \vec{F_b} + \vec{F_d} + \vec{F_f} \tag{3.10}$$

$$\begin{aligned} \vec{F_b} &= \sum_{i=0}^{|C_e|} -k_c(|\vec{P_e} - \vec{P_p}| - l_c) \frac{\vec{P_e} - \vec{P_p}}{|\vec{P_e} - \vec{P_p}|} & (3.11) \\ \vec{F_d} &= \sum_{i=0}^{|C_e|} -b_c(\vec{V_\parallel}) & (3.12) \\ \vec{F_f} &= \sum_{i=0}^{|C_e|} -|\mu_e \vec{F_N}| \frac{\vec{V_\perp}}{|\vec{V_\perp}|} & (3.13) \end{aligned}$$

$$\vec{V_\parallel} = \frac{(\vec{V_e} - \vec{V_p}) \cdot (\vec{P_e} - \vec{P_p})}{|\vec{P_e} - \vec{P_p}|^2} (\vec{P_e} - \vec{P_p}) \tag{3.14}$$

$$\vec{F_N} = \vec{F_b} + \vec{F_d} \tag{3.15}$$

$$\vec{V_\perp} = (\vec{V_e} - \vec{V_p}) - \vec{V_\parallel} \tag{3.16}$$

where:

$\vec{F_b}$: elasticity (k_c is the spring Hooke's constant, l_c is the spring elongation, and $\vec{P_e}$ and $\vec{P_p}$ are the positions of the elements involved with connection c);

$\vec{F_d}$: internal damping (b_c is the damping coefficient, $\vec{P}$ and $\vec{V}$ are respectively the positions and velocities of the elements involved with connection c);

$\vec{F_f}$: sliding friction (μ_e is the friction constant for the element and $\vec{\vec{F_N}}$ is the force normal to friction direction).

3.4.2 Setting up springs stiffness

The rheological standard to define the elasticity of a material is Young's modulus (section 3.3.1). Young's modulus is a property of a material, not of an object. So it is independent of the object's shape. However, when you discretize an object by a set of springs, the stiffness of every spring must be proportional to the fraction of the volume of the object it represents. Towards a generic method, able to calculate all spring constants in a system where the only information we have is the object's geometry, springs topology and Young's modulus of the material, we tested a number of hypotheses. These tests have been performed on an implementation of our model where objects were deformed due to a pressure force - a homogenously distributed force on one face - applied onto them. To verify the correspondence between the deformation obtained with our model and the deformation the object should undergo with respect to its Young's modulus we used the following relation:

$$E = \frac{F \cdot l_0}{\Delta l \cdot A} \tag{3.17}$$

where E is the Young's modulus, F is the applied force, Δl is the object elongation variation, l_0 is the length of the object in rest condition, and A is the cross-sectional area of the object. So, we started by the simplest linear topology case and went through many variations until the generic random topology. An evaluative comparison is also presented.

3.4.2.1 Linear case

In the linear case, where all springs produce force in only one direction and you want to measure elasticity in the same direction, you can easily calculate the constant k of the springs directly from the geometrical information of the object and the Young's modulus of its material. This is done by using the equation:

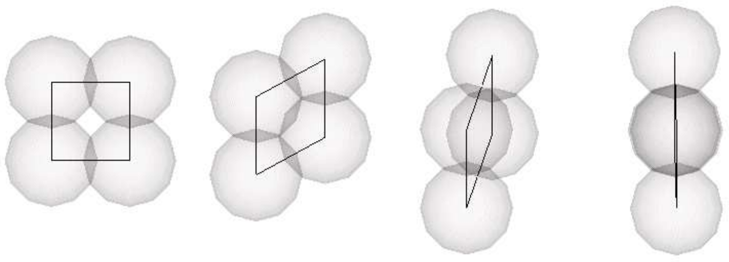


Figure 3.33: Problem of the linear case. These four situations are stable states with a rectangular topology. These and many other different relative positions between spheres are allowed for the same elongation of springs.

$$k = \frac{EA}{l_0} \tag{3.18}$$

where k is the spring constant to be calculated.

The problem of this very specific case is that its rectangular topology does not allow the construction of stable objects in two or three dimensions. Many solutions are valid as stable state (total elastic force equal to *zero*) and the situations of figure 3.33 may take place.

To overtake the limitations of the linear case in terms of types of objects that can be represented, two possibilities were considered. The first using tetrahedral discretization and the second including diagonal springs to the rectangular topology. Both methods allow to obtain stable objects. However, using tetrahedral meshes incurs in loss of generality, and diagonal springs can make the number of springs to grow up to $(n^2 - n)/2$, where n is the number of molecules. This causes problems that we discuss later on in this work.

3.4.2.2 Tetrahedral regions

One possibility to reach object stability is creating a tetrahedral mesh of springs. With this method, besides getting stable objects, a method exists to calculate springs stiffness.

[Gelder, 1998] presented a method to calculate stiffness for elastic edges of triangular meshes. His approach is based on the area of the/both triangles formed by the edge. He also proposed the extension of this approach to 3D, where volumes of neighboring tetrahedra are used to calculate the k for an edge. Figure 3.34 shows the tetrahedron T_e and its edge c to which we want to calculate k. k is calculated as shown in equation 3.19.

$$k_c = \sum_e \frac{E \cdot vol(T_e)}{|c|^2} \tag{3.19}$$

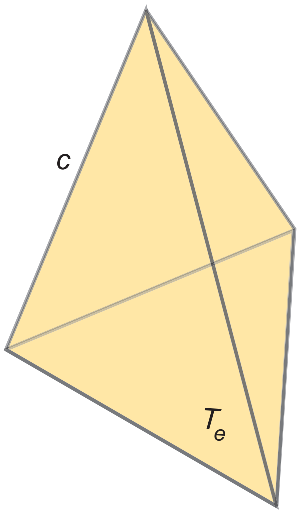


Figure 3.34: A typical tetrahedron.

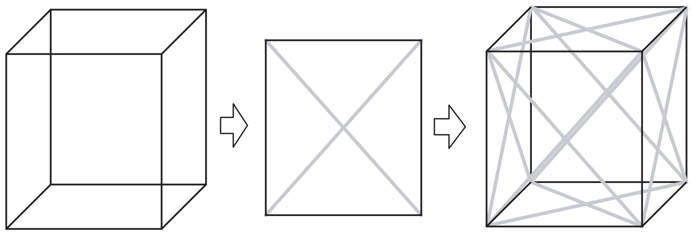


Figure 3.35: Diagonals of rectangles.

The main drawback of this method is that it only works if the mesh of springs has a tetrahedral topology. We did not implement this case because we make the choice of a general spherical discretization. Nevertheless, we can consider the use an intermediary tetrahedral mesh as a future work.

3.4.2.3 Diagonals on the planes

Avoiding tetrahedral meshes, our first try in order to reach object stability was creating diagonal spring connections (also called shear springs) in every rectangle of a rectangular mesh of springs (figure 3.35. It offered the desired result in terms of stability but created difficulties in terms of computing spring constants.

If we use the standard linear method (section 3.4.2.1) to compute spring constants, the diagonal springs will produce an extra force to linear movement that will causes elongation of the whole object to be shorter than it should. The object will become abnormally stiffer, and to avoid this some method should be developed to reduce springs k's.

The straightforward solution is to divide all linearly calculated k's by a constant C. We tested arbitrary values to C and concluded that even for the same topology, when

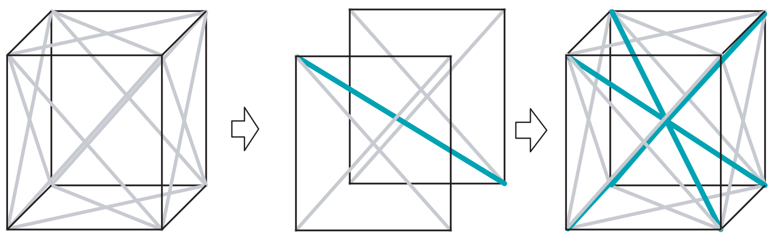


Figure 3.36: Diagonals of the parallelepiped.

other parameters change, such constant must change to preserve same elasticity. To find a value for C seem to be a problem as complex as finding k values. In addition, the situation is even more difficult to control when topology changes. These aspects are analyzed in next section (section 3.4.2.4).

3.4.2.4 Diagonals on space

We will see later in section 3.4.4 that the number of connections a molecule has depends on its radius and its distance to other molecules. So, if we change distances or radii, diagonals of the parallelepiped are created in addition to rectangles ones. In this case, other values to C must be used to provide correct elasticity. Once we cannot work manually to find which constant best fulfils each new topology we have to deal with, we worked on a method that is supposed to automatically calculate new k's depending on diagonal's angles. The idea is to compensate the contributions of each spring weighting by the angles between them.

Bringing the problem to 2D to make explanation easier, we see that force produced in vertical direction is increased by diagonals springs depending on the angles between them. As we have right triangles in this topology, the relationship between a vertical connection and a diagonal one is proportional to cosine and the final k is given by the relation:

$$k_f = k_0 \cos_0 + k_1 \cos_1 + \cdots + k_n \cos_n \tag{3.20}$$

(3.13) where k_0 is the connection to which we are calculating k, and k_1 to k_n are the connections that share a extremity with k_0. $\cos_i$ is the cosine of the angle between connections 0 and i.

Considering a homogeneous object, all k_i are equal. So, from this and equation 3.20 we deduce:

$$k_0 = \frac{k_f}{1 + \cos_1 + \cdots + \cos_n} \tag{3.21}$$

This approximates k_0 of the actual k value we are looking for. However, angles change along simulation and should be recalculated for every iteration, which causes increase of computing cost. Even more tricky, the right triangles deform along simulation becoming arbitrary triangles, which reduces the precision of this method for large deformations.

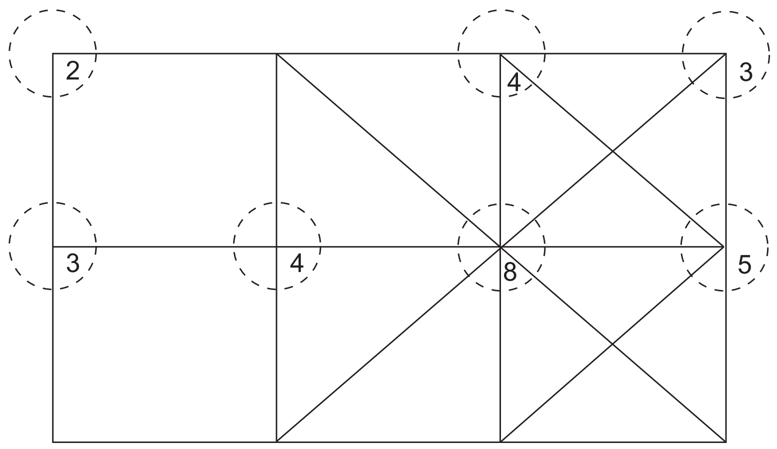


Figure 3.37: Different 2D situations and their numbers of connections.

3.4.2.5 Statistical method

As we have just seen, during deformation the molecules may change their positions in a way that diagonals angles are no longer right angles. Besides, arbitrary starting topologies may happen where some angles are not right angles from the beginning. Trying to cover these cases we developed a novel method to calculate spring constants. It is a statistical method inspired on general concepts of Quantum Mechanics that heuristically estimates k's reductions from the number of connections around a molecule. As the number of connections of a molecule increases, as small is the portion of the object's volume that each connection represents, and smaller must be its spring constant. So, though we do not calculate exactly the volume represented by a connection, we can probabilistically guess which topological case we have, just counting how many connections share a molecule. Figure 3.37 illustrates some typical 2D situations.

We considered the 3D case and stated the algorithm 1 to calculate Hooke's constant for every connection of an object. This algorithm should present better results when larger and larger numbers of molecules are involved. It was expected to happen because, statistically, as bigger the sampling as smaller the error. However, practically, because of the border effects, the relation between the number of connections on the surface and inside the object also plays an important role.

3.4.2.6 Iterative method

The division by a constant approach mentioned earlier (section 3.4.2.3) can be extended by automatically calculating the value of C according to the obtained deformation. The elongation of an object can be applied in equation 3.17 to verify, for any given known force, what is the effective Young's modulus (E) of the object [Maciel et al., 2003b].

We use any of the methods presented up to here to find a configuration of springs that gives an E relatively close to what is required. Then we initialize C with 1.0 for that configuration, and use $\frac{k}{C}$ every time a k is used in the force equations. After, we start a simulation applying a known force on the object, and at each time step, we

Algorithm 1 Calculating k based on number of connections.

```
for all connections $c_i$ between the molecules $c_{i_1}$ and $c_{i_2}$ do
  N1 = number of connections of molecule $c_{i_1}$
  N2 = number of connections of molecule $c_{i_2}$
  D1 = estimated number of different directions among the connections of molecule $c_{i_1}$
  D2 = estimated number of different directions among the connections of molecule $c_{i_2}$
  if $N1 < 8$ then
    D1 = N1
  else
    if ($N1 >= 9$) AND ($N1 < 12$) then
      D1 = N1 - 1
    else
      if ($N1 >= 13$) AND ($N1 < 20$) then
        D1 = N1 - 4
      else
        if ($N1 >= 20$) then
          D1 = N1 / 2
        endif
      endif
    endif
  endif
  if $N2 < 8$ then
    D2 = N2
  else
    if ($N2 >= 9$) AND ($N2 < 12$) then
      D2 = N2 - 1
    else
      if ($N2 >= 13$) AND ($N2 < 20$) then
        D2 = N2 - 4
      else
        if ($N2 >= 20$) then
          D2 = N2 / 2
        endif
      endif
    endif
  endif
  area1 = ( 2 * cross-sectional area ) / D1
  area2 = ( 2 * cross-sectional area ) / D2
  $k_{c_i}$ = ( ( $E_{c_{i_1}}$ * area1 ) + ( $E_{c_{i_2}}$ * area1 ) ) / ( 2 * nominal_dist$_{c_i}$)
end for
```

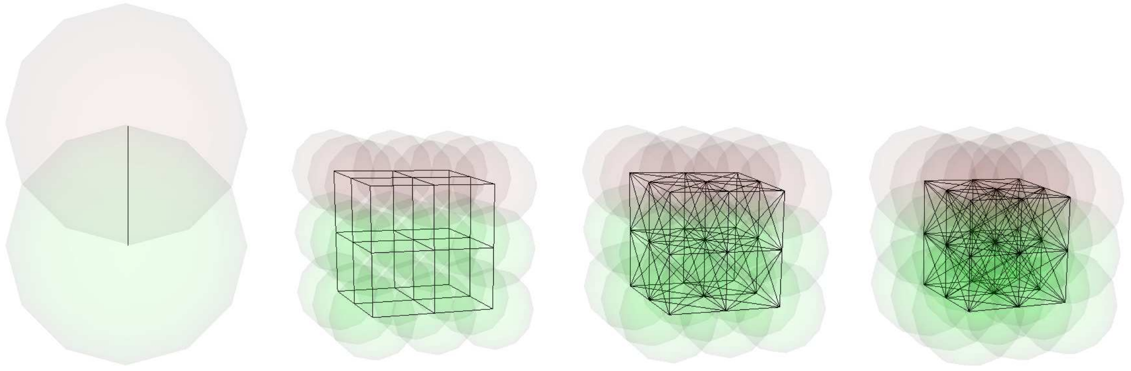

(a) 2 mol./1 con. (b) 27 mol./54 con. (c) 27 mol./126 con. (d) 27 mol./158 con.

Figure 3.38: The different scenarios being studied.

apply equation 3.17 to verify the momentary E. We then increase or decrease the value of C to make the material harder if the measured E value is lesser than required, or softer if it is greater. After a number of simulation steps, which depends on how close the initialization was of the required E, we find a value for C that gives the exact E we were looking for.

Note that, because the stress-strain relationship is not linear for the material, the obtained C cannot guarantee a valid E at any strain. However, using for the test a force compatible with the typical load the material bear in the real world will give correct behavior when similar loads are applied on the model.

3.4.2.7 Analysis and comparison

A number of experiences have been performed to test the validity of the hypotheses of sections 3.4.2.1 to 3.4.2.7. Four deformable objects have been created which represent the same volume in the space (figure 3.38). Though they do not have exactly the same volume, the regions considered here have exactly the same vertical length and nearly the same cross sectional area. The vertical length of the considered volume in the initial state is equal to $15mm$ and its cross-sectional area is around 9 cm^2. The superior extremity of the objects is fixed and a tension force of $1N$ is applied on the inferior extremity. The elasticity of the material is specified as 5000 N/m^2 $(= 5\ kPa)$.

For each object we computed all methods to calculate k. Table 3.1 compares the results obtained for Young's modulus when each method is used for calculation of k. The method based on tetrahedral regions has not been implemented here.

Later on, to inspect the behavior of the methods when the number of molecules grows drastically, we created the situation of figure 3.39 where 1000 molecules and 7560 connections represent a volume of approximately 1 m^3. The vertical length of the considered volume in the initial state is equal to 86 cm and its cross-sectional area is 1 m^2. One extremity of the object is fixed and a tension force of 1000 N is applied on the other one. The elasticity of the material is specified as 10^5 N/m^2 $(= 0.1\ MPa)$. The obtained results are in table 3.2.

Method	**Young's Modulus (N/m^2) / Error (%)**							
	(a)		(b)		(c)		(d)	
Linear case	**4'994**	**0.12**	**4'996**	**0.08**	12'465	149.30	20'678	313.56
Statistical	10'004	100.08	6'098	21.96	3'327	27.26	**4'904**	**1.92**
Angles	**4'997**	**0.06**	**5'000**	**0.00**	**4'916**	**1.68**	5'903	18.06
Angles (update)	**4'999**	**0.02**	**4'996**	**0.08**	11'550	131.00	18'621	272.42
Iterative	**5'000**	**0.00**	**5'000**	**0.00**	**5'000**	**0.00**	**5'000**	**0.00**

Table 3.1: Resultant elasticity obtained with the different methods for the 4 cases of figure 3.38. Values in bold indicate they are correct or present a small acceptable error.

Method	**Elongation (mm)**	**Young's Modulus (N/m^2)**	**Error (%)**
Linear case	2.30	373'913	274
Statistical method	11.38	75'544	24
Angles	5.89	146'010	46
$k/3.75$	8.60	100'000	0
Iterative	8.60	100'000	0

Table 3.2: Tension tests results for a cube of elasticity expected to be 10^5 N/m^2.

We also compared our results with data from a finite elements analysis (FEM) performed by our fellows in the Institute of Mechanical Systems of the ETH Zurich. For comparison, we used the same 1 m^3 model and applied a number of different loads while tracking displacement of chosen points. The position of the points tracked corresponded in both models with an error inferior to 1%. Figure 3.40 shows a visual comparison after the application of the same shear force of 5000 N on the two models.

3.4.3 Numerical integration issues

In numerical analysis, numerical integration constitutes a broad family of algorithms for calculating the numerical value of a definite integral, and by extension, the term is also used to describe numerical algorithms for solving differential equations, which is the case in this work. What is required is a numerical solution to derive how the state of a system changes over a given timestep. Many differential equations cannot be solved analytically, in which case we have to satisfy ourselves with an approximation to the solution. The algorithms briefly discussed here can be used to compute such an approximation.

Numerical integration can be done explicitly or implicitly, and several methods exist in both sides. Let us see an example. Consider the linear system:

$$\dot{x} = Ax + Bu \tag{3.22}$$

For the purposes of simulation, it can be discretized in at least two ways:

$$\dot{x} \approx \frac{x_{i+1} - x_i}{\Delta t} \tag{3.23}$$

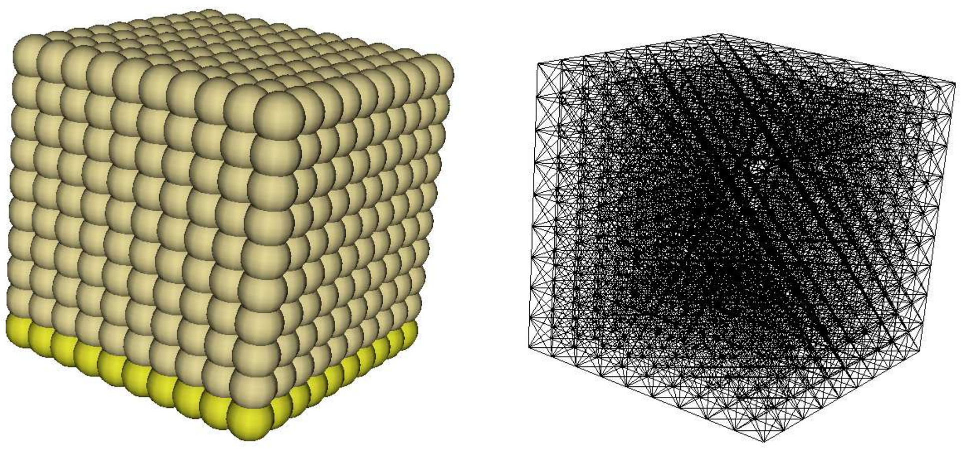
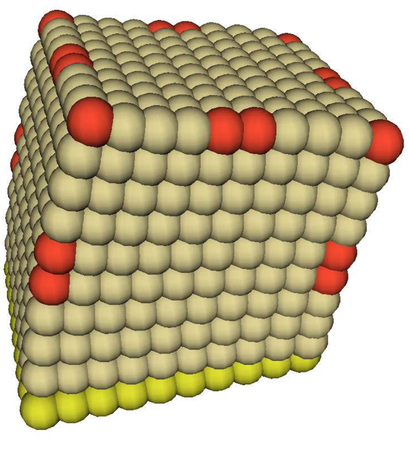
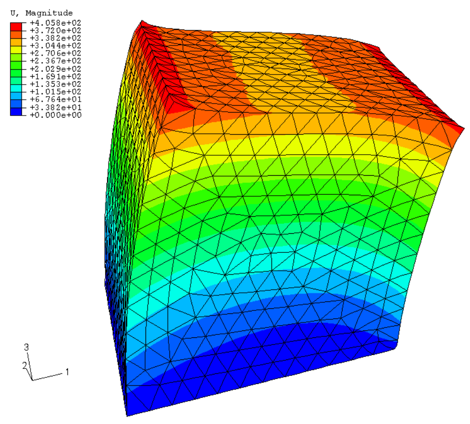


Figure 3.39: Increased number of molecules (1000) and connections (7560).



(a) Molecular model after force application.



(b) FEM model after force application.

Figure 3.40: Comparing with FEM analysis. The same shear force of 5000 N is applied on both our model and an FEM. The deformations correspond visually and also numerically. (FEM courtesy of IMES-ETHZ)

$$\dot{x} \approx \frac{x_i - x_{i-1}}{\Delta t} \tag{3.24}$$

Equation 3.23 gives rise to the forward Euler or explicit integration scheme:

$$x_{i+1} = x_i + \Delta t[Ax_i + B_i u] \tag{3.25}$$

and equation 3.24 gives rise to the backward Euler or implicit integration scheme:

$$x_i = x_{i-1} + \Delta t[Ax_i + B_i u] \tag{3.26}$$

which for implementation purposes must be rewritten as:

$$x_i = [I - \Delta t A]^{-1} x_{i-1} + \Delta t B_i u \tag{3.27}$$

The explicit method gives simple implementation whereas the implicit method requires matrix inversion. However, the explicit method is only stable if:

$$\Delta t < \frac{2}{|\lambda|} \tag{3.28}$$

where $\tau$is the largest eigenvalue of A. If this largest eigenvalue is real so $\lambda = \frac{1}{\tau}$where τ is the smallest system time constant:

$$\Delta t < 2\tau \tag{3.29}$$

If the system is stiff, which means it contains at least one small time constant relative to the dominant time constants, Euler integration is not feasible due to the very small sample interval required. In contrast, the implicit method is stable.

We worked only with explicit methods because dynamically simulating models similar to our molecules system has shown to be very slow with implicit integration. We first tested explicit Euler, and then also tested Runge-Kutta of order 4 method (RK4), that is reputed to be much more stable, for comparison (see section 4.3.3.2). Runge-Kutta methods use a weighted sum of the values of $f(x,t)$, evaluated at the start of each step and at various points across the integration step. These are the formulas of a Runge-Kutta of fourth-order method:

$$k_1 = f(x_i, t_i) \tag{3.30}$$

$$k_2 = f(x_i + \left(\frac{\Delta t}{2}\right) k_1, t_i + \frac{\Delta t}{2}) \tag{3.31}$$

$$k_3 = f(x_i + \left(\frac{\Delta t}{2}\right) k_2, t_i + \frac{\Delta t}{2}) \tag{3.32}$$

$$k_4 = f(x_i + \Delta t k_3, t_i + \Delta t) \tag{3.33}$$

$$x_{i+1} = x_i + \left(\frac{\Delta t}{6}\right)(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.34}$$

Because of all those steps, RK4 is slower to compute than Euler. However it usually achieve stability using larger timesteps, which makes the global computation time inferior than using Euler. In practice, we noticed that our Euler integration is sometimes stable with larger timesteps than our RK4 one, which makes it much faster. It is especially true when the dimensions of our models are very small in terms of metric values, i.e., models of a few millimeters are more stable with Euler, and models of several meters with RK4.

3.4.4 Discretization

Up to now we showed examples of molecular models that are very simple: cube and parallelepiped. Placing the molecules in those objects is straight forward; we did it with a simple 3D marching grid algorithm.

However, the model is intended to be used to represent more complex shaped objects, like biological structures of the human body. For that, we propose to use a 3D mesh representing the volume as input, and discretize the volume into a number of spherical regions.

[del Pobil and Serna, 1995] presented a work in robotics where a spatial representation was needed for collision detection. In chapter 2, they study the possibilities of using spheres as primitive for object representation. They discuss several approaches and hypotheses around the theme. [Shimada and Gossard, 1995] present a physically based mesh generation algorithm using bubbles as regular element shapes. They applied attractive and repulsive forces to decide where to place the bubbles, and an adaptive population control mechanism to decide how many bubbles are necessary. [Mello and Cavalcanti, 2000] used crystal lattices as templates for creating background meshes. They affirm that applying the appropriate template results in meshes with better quality than simply using a regular grid. They present examples in 2 and 3 dimensions always evaluating the minimum dihedral angle [2] obtained. Their background meshes could be used to spheres distribution placing them on the mesh vertices. A more recent and interesting method for sphere distribution is presented in [Bradshaw and O'Sullivan, 2004]. They used a medial axis approximation algorithm to construct a sphere-tree that can be refined, for example, to assure total coverage.

In **our method**, we apply a homogeneous distribution of spheres. To do that, we decided to explore an alternative making use of the force equations of our deformation model to find an equilibrium state to the molecules distribution.

Basically, it works in two steps:

1. Given a 3D mesh and two input parameters (spheres radius and preferred distance between sphere centers), generate a 3D grid. Then place a sphere at each

[2]The angle at which two adjacent faces meet. For example, the dihedral angle of a cube is 90°.

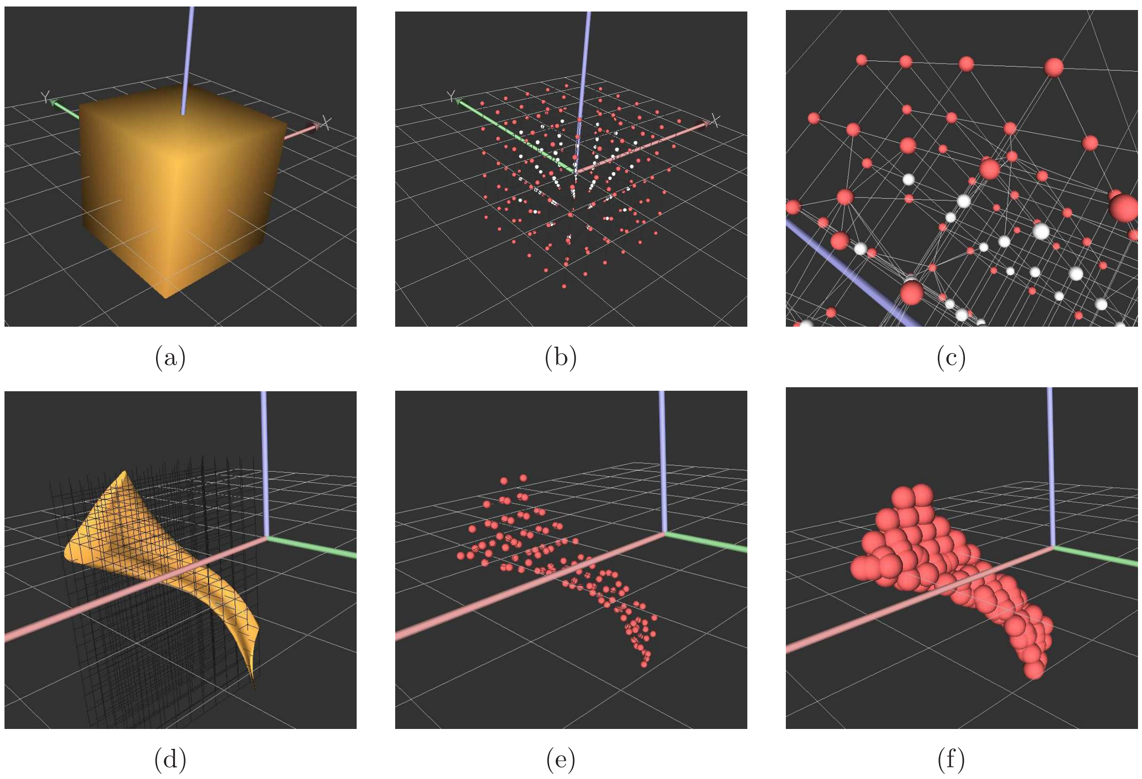


Figure 3.41: Example of discretization. On the top row, a simple cube is *spherized*. On the bottom row, the same method is applied to a model of a human hip joint ligament.

intersection of the grid, and also at each intersection of the grid with the mesh surface.

2. Generate elastic springs between all spheres touching each other ($dist(s_1, s_2) < 2r$). Springs will have different lengths, but their stiffness and nominal length are all the same. Start a simulation considering the spheres on the surface as fixed. The internal spheres will move according to the stress on their connecting springs such that after simulation they stop in an equilibrium position.

The positions after simulation are saved as definitive. As we do not take into account the internal structure of the corresponding real objects on defining the placement of our mass nodes, when we want to model specific structural features of real objects we assign different material properties to different nodes depending on their positions.

A future extension that worths to test is adapting the sphere sizes to the shape. Subdividing big spheres into a set of smaller ones can improve the precision on representing small details of an irregular surface, analogous to [Amrani et al., 2000]. Another extension, which should improve homogeneity, would be to allow the spheres on the surface to slide constrained on the surface during the simulation of the item 2 above.

3.4.5 Skinning model

The model as it has been presented up to now describes objects as bulks of spherical elements (molecules). As consequence, their surfaces are not smooth. One approach to obtain a smooth surface would be drastically increasing the number of molecules, analogously to real macro objects. However, it would make the deformation methods inviably slow.

Then, we tried to get inspiration from literature on skin modeling to cover our molecules with a kind of smooth *skin*. Nowadays, the most widespread technique for skin deformation consists in assigning to each vertex of the polyhedral mesh, which represents the skin, a number of bones with appropriate weights [Lander, 1998]. During animation, a vertex mesh is thus moved based on transformations of the underlying skeleton. This process is often called *skinning*. A number of works discuss of propose skinning methods [Singh and Kokkevis, 2000] [Houle and Poulin, 2001] [Aubel, 2002]. Commercial modeling packages, e.g. Maya [Ali, 2001], also rely on the skinning technique.

The solution we propose is to cover the bulk of molecules with a polygonal mesh. We did it in two different ways. We first used an implicit surface considering the molecules as metaballs, and we also did the inverse approach, starting from an existing mesh and discretizing it (see section 3.4.4).

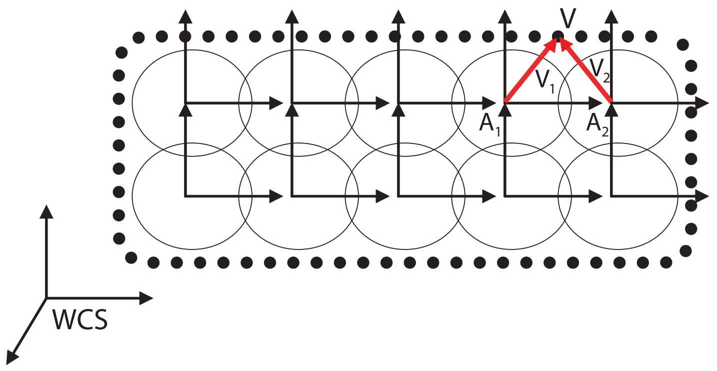
The remaining problem to be solved was to connect the surface mesh to the physical model such that the surface deforms accordingly. Regenerating the implicit surface has shown to be too slow to be done at each simulation step. So, we decided to deform the surface in a somewhat free-form-like deformation style, but connecting the surface vertices to the model in a way that their new positions could be directly derived from the positions and orientations of the objects molecules.

For that, we utilize an anchoring system, associating the vertices positions to the position and orientation of one or more molecules underneath. The initial situation is represented in figure 3.42. The global positions of all molecules and surface vertices are given. Then, the algorithm is composed of the following steps:

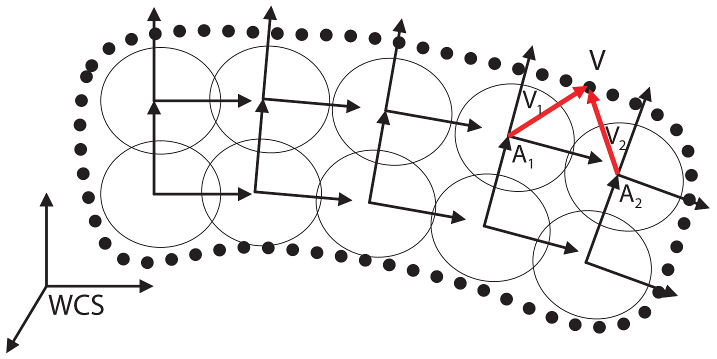
1. For each molecule, create a local-coordinate system centered at the molecule position. Construct the 3 axes from the 3 most perpendicular vectors starting on the center and going to the neighboring molecules centers.

2. For each surface vertex, calculate its positions relative to the local-coordinate systems of the n closest molecules (red arrows in figure 3.42). These positions will remain constant throughout the whole simulation. We refer to this as "anchoring"the surface vertices to the molecules. After initialization, a global vertex position can only be obtained combining the anchor local-coordinate system and the local coordinates of the vertex. In the example of figure 3.42, $n = 2$, meaning that V has two anchors and two calculated global positions. Those two positions are blended into one through weighted interpolation.

3. At each simulation step, the local-coordinate systems of the molecules are recalculated using the new positions of the neighbor molecules for the axes. Because of



(a)



(b)

Figure 3.42: Illustration of the skinning algorithm. The local coordinates V_1 and V_2 of a surface vertex V (red arrows) to its anchoring molecules A_1 and A_2 are constant after initialization. The local-coordinate systems of the molecules, in turn, can move and rotate when the object deforms.

that, the anchors local-coordinate systems may move and rotate. Consequently, the result of the combination of the anchor local-coordinate system (updated) and the local coordinates of the vertices (fixed) gives new global positions for the vertices. In practice, the surface follows the deformation of the molecular model.

The reason for using n anchors instead of one is to prevent the formation of discontinuities on the surface. This is best understood by looking at figure 3.43. The images (a) and (b) represent the surface of a deformed object using only one anchor. Orientation differences between neighboring molecules induce into a saw-tooth effect on the surface at the region between neighbor vertices anchored to two different molecules. To avoid this side effect we need to soften the transition from one anchor to another.

As already mentioned in the item 2 of the algorithm above, the n global positions (one for each anchor) calculated for each vertex must be blended into one. A blending function, which calculates a weighted average of all these positions, is used to compute

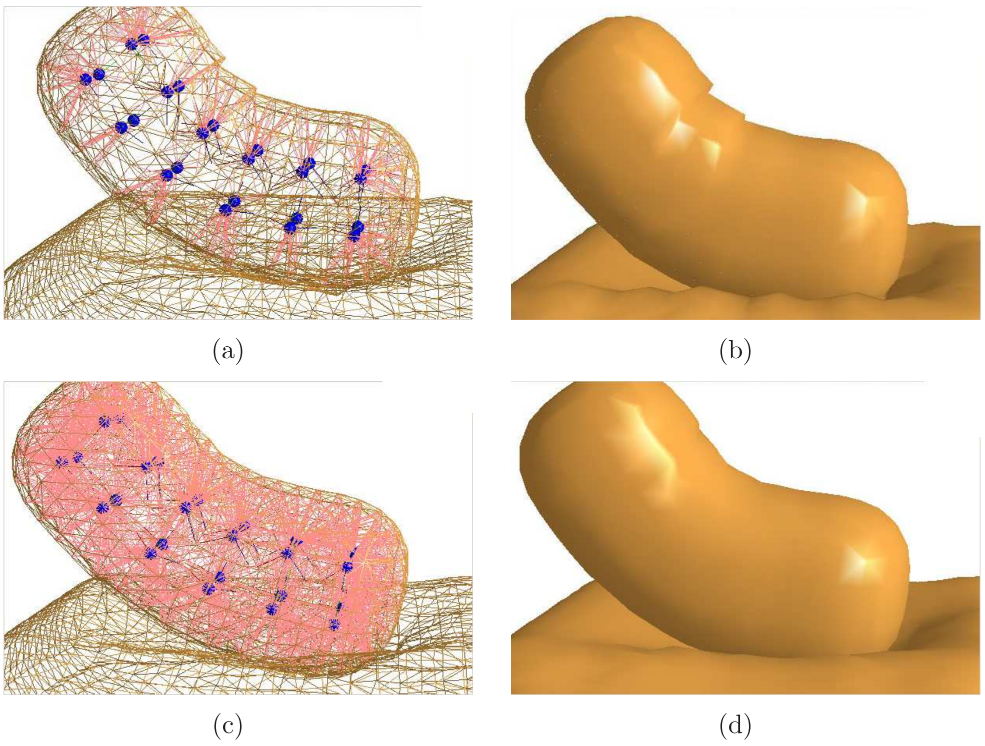


Figure 3.43: Skin deformation using only one anchor (upper row) and using four anchors (lower row) per skin vertex. With only one anchor the border where the anchors of the vertices change can clearly be seen (upper right). The connections between the anchors and the skin vertices are shown in red on the transparent view (left column).

the final global position. The weights corresponding to the different anchors contributions are inversely proportional to the square of the distance between the anchor and the vertex, i.e. closer anchors impose a greater weight.

We noticed that increasing the for most of our models increasing n to values greater than 4 does not improve surface smoothness.

3.4.6 A haptic application

In Virtual Reality, haptic techniques like force-feedback have been used to improve the immersion sensation. A number of electromechanical systems are available nowadays allowing to return meaningful force or vibration back to VR users. We developed an application aiming at simulating simple interactions on deformable elements and returning to the user the reaction forces from the virtual objets being touched [Maciel et al., 2004]. With such application, the user well equipped with a multi-finger force-feedback device, and a hand motion capture device, sees a model of their hand and models of 3D objects on the screen, and can interact with them in real-time (see figure 3.44).

The main application integrates the force-feedback device (figure 3.45(a)), the motion capture device (figure 3.45(b)), the viewer and the deformation model.

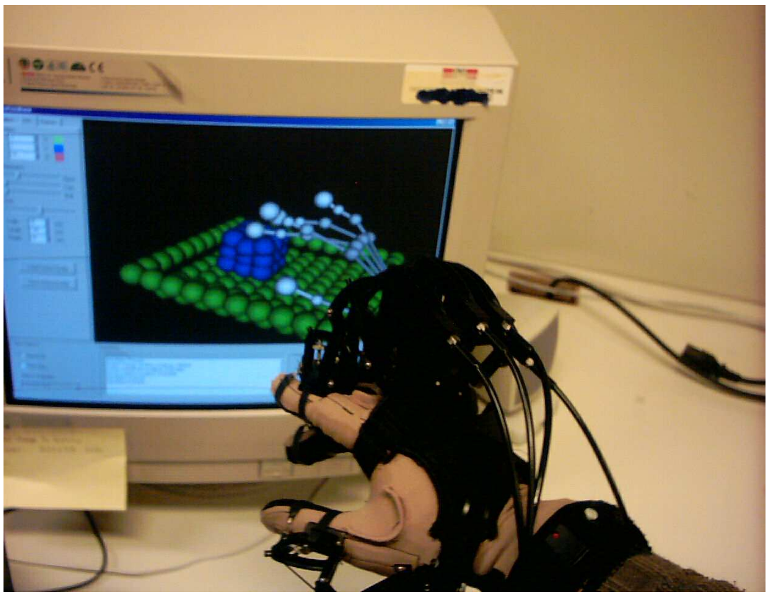
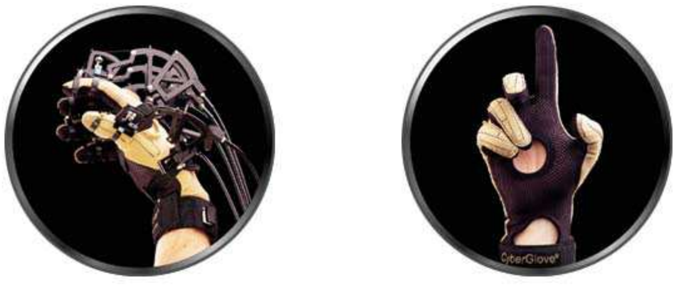


Figure 3.44: A user's hand equipped with haptic devices and the application screen.



(a) Cybergrasp (b) Cyberglove

Figure 3.45: External peripherals for force-feedback (a) and hand motion capture (b).

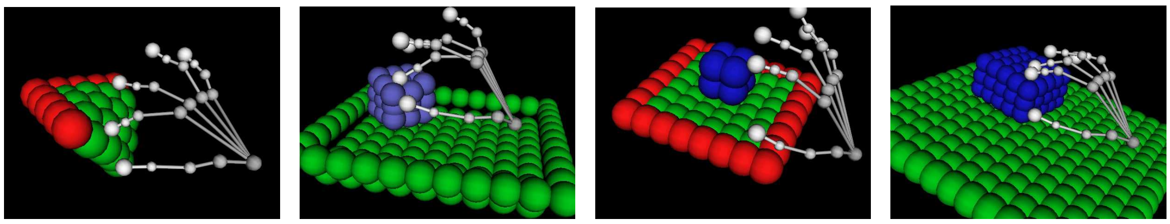


Figure 3.46: Different scenes for performance test.

	Test a)	Test b)	Test c)	Test d)
Main thread	35	19	15	7
Physical simulation thread	131	90	79	38
Force-feedback update thread	403	424	430	440

Table 3.3: Update frequencies (Hz) for the 4 scenes of figure 3.46.

The goal of the system is to provide the users with the ability to feel the forces in a sufficiently realistic way. They must be able to recognize the type of material they are touching, analogously to when they handle a real material. Taking into account the limitations of the peripherals being used, though, a real sensation cannot be completely reproduced, and the material recognition can be done only comparatively.

3.4.6.1 Performance test

To insure real-time interaction, we had to limit the application to use a limited number of very simple objects. We used two different machines, one to control the external devices and another for simulation and display. The different threads are synchronized and priorities are set according to the frequencies necessary to reach good results. The force-feedback device, for example, required the force to be updated at a minimum o 400 Hz, while the viewer can be kept at 15 Hz and the physical simulation depends of the scene complexity.

The application performance has been tested on a *bi-Xeon 1700 MHz* with RAM 1 gigabyte. Figure 3.46 shows the 4 scenes we tested and table 3.3 brings the results obtained. The 4 scenes are described below:

a) A membrane with 16 deformable molecules attached to a set of 9 fixed molecules.

b) A cube containing 27 deformable molecules and a platform of 121 fixed molecules.

c) A cube of 8 deformable molecules, as well as a membrane of 49 deformable molecules sustained by a frame of 28 fixed molecules.

d) A parallelepiped of 75 deformable molecules and a platform of 225 fixed molecules.

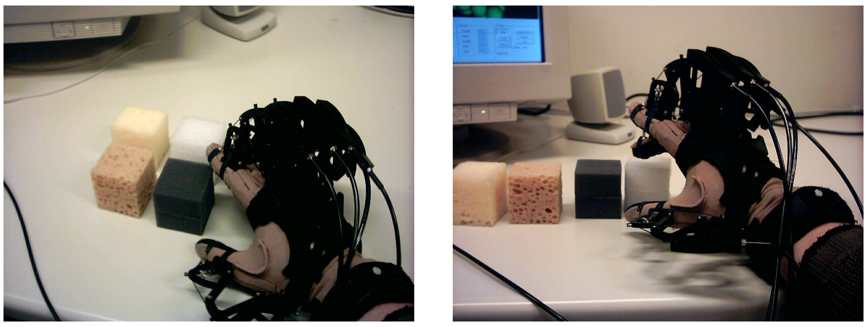


Figure 3.47: Typical haptic test situations.

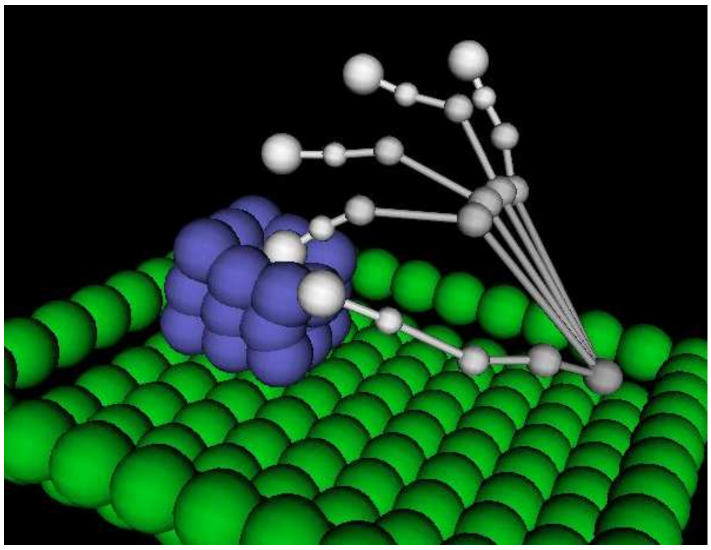


Figure 3.48: Screenshot from the haptic application.

3.4.6.2 Perception test

In order to test the force-feedback sensation given by our application, we realized an experience with some students and researchers of our laboratory. They were required to put 4 real deformable objects of different stiffness in correspondence with 4 virtual objects.

The real objects are cubes of 5 x 5 *cm* made of different materials. Their virtual analogs are deformable cubes composed of 27 molecules (3x3x3). They were first calibrated modifying their Young's modules to correspond to the 4 real objects. Figure 3.47 shows typical test situations and figure 3.48 brings a screenshot taken from a test.

We performed the test with 8 different people. 6 of them have found the correct correspondence between real and virtual objects. Concerning the other 2 people, one inverted the two softer objects, and the other inverted the two softer and the two stiffer. The graph in figure 3.49 summarizes the results.

These are interesting results because most of the users have recognized the different elasticities correctly. In addition, the errors were inversions of objects with very similar

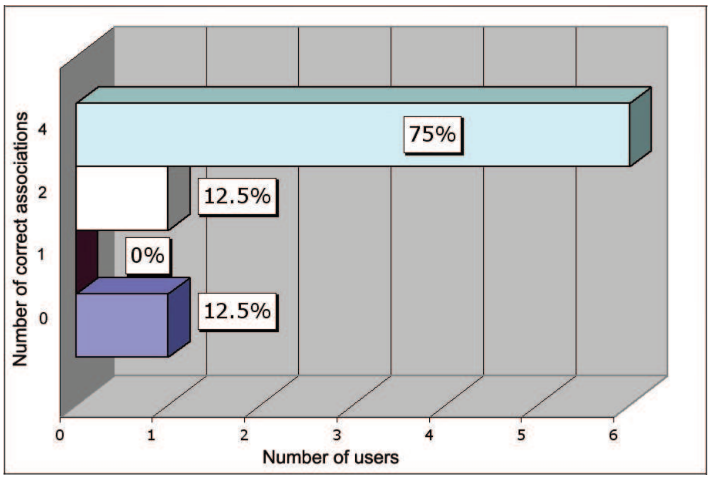


Figure 3.49: Perception test results. 75% of the volunteers found the 4 correspondences, 12.5% only 2, and 12.5% none.

elasticities.

We also noticed that the visual feedback influence the users sensation. It seems that the visual deformation of the objects gives information about their rigidity. We performed a few blind tests and observed that it is more difficult to recognize the stiffness differences. Nevertheless, more controlled tests would be necessary to draw a definitive conclusion.

CHAPTER 4

Contact model

4.1 Introduction

Although our knowledge of the laws of Physics has evolved along time, the Newton's law of non-penetration, which states that two bodies cannot occupy the same place in space within the same time interval is still not refuted, at least in which concerns the *opinion* of most people. A physical constraint avoids objects of our world to penetrate each other, and we are used to that. It is essentially because of that, that the problem of collision detection came into the Computer Graphics problematic, because virtual objects are also expected not to penetrate each other.

People are also used to diverse reactions the objects undergo when submitted to conditions toward the violation of the non-penetration law. Some objects bounce on each other, some deform, some others break; they generally produce sound when colliding, sometimes heat. In Computer Graphics, modeling virtual objects reproducing such phenomena is a big issue on increasing the realism of virtual worlds.

Sometimes, physical objects in collision remain in collision due to an equilibrium of the antagonist forces acting on them. The period of time the objects remain in collision is what we call *contact*. Permanent contact is a tricky problem to simulate in computer because of numerical instability, and that particularly when the contacting surfaces are large and the antagonist forces vary along the surfaces and along time. Fast solutions exist for rigid, smooth, parametric surfaces in contact [Kry and Pai, 2003], but continuous contact between polygonal, irregular, deformable objects still remains an open issue.

In the next section, we see how the problems of collision detection, response and contact have been treated in Computer Graphics and Virtual Reality. We review the advantages and disadvantages of each method, how they adapt to different kinds of objects, and which situations are still difficult to model and simulate in the computer.

Following, the problematic of contact inside a human articulation is presented, and methods for collision treatment are proposed to deal with that specific case.

4.2 State of the art

4.2.1 Collision detection review

There are many ways of approaching a classification for collision detection methods. The problem is multidimensional and many different classifications can be presented depending on what one is interested to highlight. Some methods are exclusively applicable to rigid bodies while others can be also extended to deformable bodies. Collision detection methods also depend on the technique used for objects modeling. Methods made for polygonal objects cannot be applied on objects constructed by CSG, for example. In addition, some methods for collision detection are intimately related to collision avoidance methods, some are able to detect self-collisions, others are not. Besides, like very often in Computer Science, many of the existent methods for collision detection can be classified as a classical method or as an optimization, improvement or extension based on one or more classical methods.

Therefore, we decided to orient our survey more in the direction of the fundamental principles explored by each method, i.e., we grouped them according to the concept they are based on. We also tried to put in evidence the elements related to the goals of the present thesis. For more detailed surveys exploring other dimensions of taxonomy of collision detection, refer to [Lin and Gottschalk, 1998], [Jiménez et al., 2001] and [Teschner et al., 2004].

4.2.1.1 Basic methods

Basically, detecting collisions means checking the geometry of the target objects for interpenetration. Mathematics and Geometry give enough tools for that, what is usually called static interference test. They can be split into two types:

- **Distance calculation**. Distance calculation algorithms start by determining the closest elements of each object. Such elements are in most cases vertices, edges and faces. Then, their euclidian distance can be calculated and a negative distance indicates a collision. Different techniques for distance calculation are proposed in [Gilbert et al., 1988] [Dobkin and Kirkpatrick, 1990] [Lin and Canny, 1991] [Joukhadar, 1996] [Mirtich, 1998].

- **Intersection tests**. In this case, an intersection between two primitives indicates the occurrence of a collision. Intersection test between spheres is the quickest and simplest. Given two spheres A and B, an intersection exists if:

$$|p_A - p_B| < r_A + r_B \tag{4.1}$$

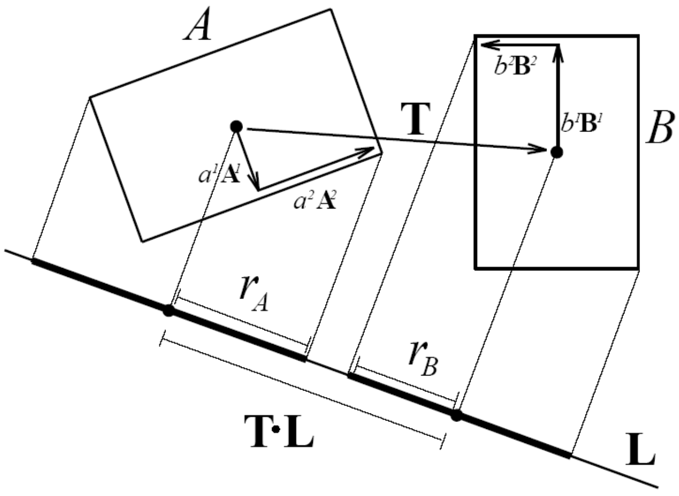


Figure 4.1: **L** is a separating axis for the bounding boxes A and B because A and B become disjoint intervals under projection onto **L** (extracted from [Gottschalk et al., 1996].

> Bounding boxes intersection test is particularly useful in BVHs (section 4.2.1.2). They can be tested using the separating axis test [Gottschalk, 1996] (see figure 4.1). Intersection test between triangles can be done efficiently with the algorithm of [Möller, 1997]. Triangle-triangle test are useful because most 3D polygonal representations are triangle meshes.

However, in practice, detecting collisions also means doing that with a limited amount of computing resources, in a limited time interval, and for a potentially huge number of primitives. Such primitives are often simulated dynamically, which generates a great number of static interference situations to be tested. Methods to reduce complexity of collision detection are presented in the following subsections.

4.2.1.2 Methods based on bounding volume hierarchies

Bounding volume hierarchy (BVH) methods subdivide the elements of a scene into hierarchically organized spatial cells or bounding volumes (BV) to accelerate interference computations. Testing interference with bounding volumes is typically less complex than testing interference with the objects they bound. Complex tests are then avoided because when BVs do not collide, the objects they contain - other BVs or primitives - do not collide either.

Generation, traversal and update are the three main phases present in BVH methods. The hierarchies are usually **generated** for a scene at initialization phase using a top-down strategy, even if some works can be found exploring bottom-up and insertion techniques to build similar hierarchies in other domains, like [Goldsmith and Salmon, 1987]. It is done in a way that the tree can be consulted as fast as possible during simulation. The set of primitives of the geometrical model is recursively split into BVs until a threshold is reached. For example, when the BV contains only one primitive.

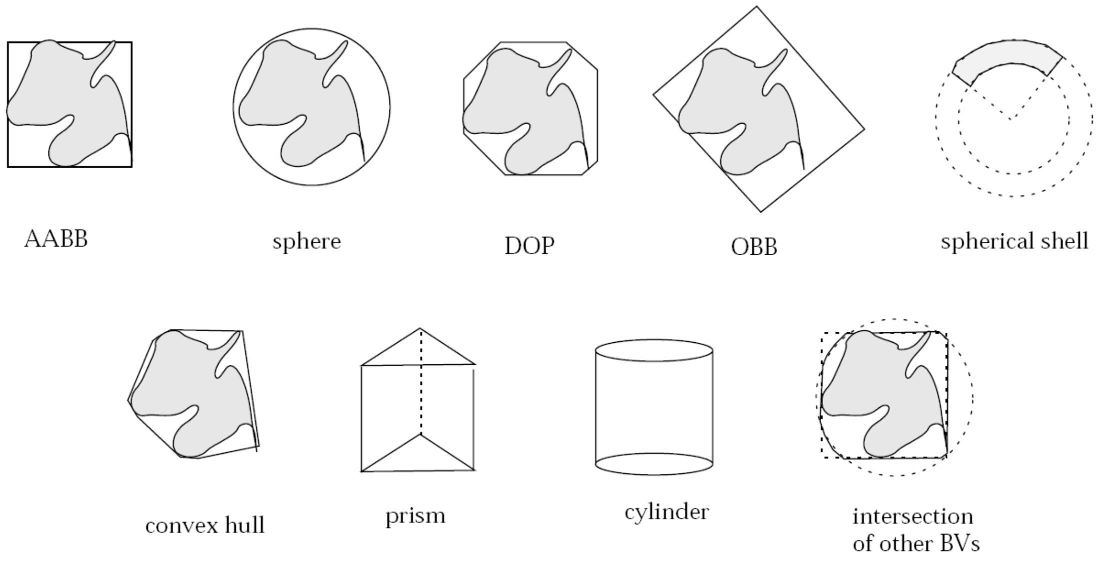


Figure 4.2: Some of the most used volume shapes in bounding volume hierarchies. [Teschner et al., 2004]

Primitives are finally placed on the leaves of the tree while the internal nodes are BVs.

The tree analysis, or **traversal**, is done top-down, from the root to the leaves, testing the nodes for overlap and following recursively the branches of which the parent node collides.

BVH methods perform better on rigid objects, because an update phase is not necessary. But they have also been much applied on deformable objects. When non-rigid transformations occur, like with deformable objects, an **update** phase is required. It consists of rebuilding completely the BVH or refitting it to the scene when necessary. If the choice is for rebuilding, the optimization approaches try to avoid rebuilding at every time step [Mezger et al., 2003] and/or avoid rebuilding the whole tree at every rebuild [Larsson and Akenine-Möller, 2001]. In terms of refitting, one method is having one version of the BVH or several versions of the nodes in a same BVH for each key position of the objects. Then, at traversal time interpolate between adjacent nodes or tree versions. It was applied with morphing targets by [Larsson and Akenine-Möller, 2003]. Accelerated update methods are the principal research thread on BVH applied to deformable objects, especially on applications like clothing simulation, but also on surgery simulation and other medical applications.

Different BVH methods use different bounding volumes (figure 4.2), which may incur in different techniques to generate, traverse and update the BV tree, as well as being more or less adapted to different environment types. Octrees have been used to build bounding box hierarchies around elements of polygonal models belonging to their convex hull and around its concavities [Ponamgi et al., 1997]. Hierarchies of spheres [Quinlan, 1994] [Palmer and Grimsdale, 1995] [Hubbard, 1995] or spherical shells [Bonner and Kelley, 1988] [Krishnan et al., 1998] have been used for bounding volumes at different resolution levels, often using inner and outer bounds [Jiménez et al., 2001]. In [James and Pai, 2004], the authors introduced the bounded deformation trees (BD-Trees) applied to deformable objects. BD-Trees perform similarly to sphere tree methods for rigid objects with some differences like the update method that is rather output-sensitive than hierarchical. Despite its several limitations, the method has proven to be well adapted to reduced deformable models.

Axis-aligned bounding boxes (AABB) simply consider the limits of an object in terms of its x, y and z coordinates to determine its bounding boxes; which is thus aligned to the axes. That has the advantage that the intersection test does not need to project the bounding boxes into the world reference frame several times, which makes this process faster. In addition, the generation and update of the tree is relatively fast, and is not memory consuming [van den Bergen, 1997]. It makes AABB suitable to detect collisions also between deformable objects, like in [Hughes et al., 1996]. Self collisions between deformable objects are treated in [Volino and Magnenat-Thalmann, 1994] and [Volino and Magnenat-Thalmann, 1995].

In environments with many objects placed in a quite disordered way, oriented bounding boxes (OBB) perform better than AABB because they fit the objects tighter, incurring in fewer false hits. It seems not to be the case for the example object on figure 4.2, but it is true for shapes which the dimensions are very different. Theoretically, overlaps with OBBs are determined performing 15 axis projection tests (about 200 arithmetic operations) [Gottschalk et al., 1996]. Another work using OBBs is [Garcia-Alonso et al., 1994]. Later, in [Hirota, 2001], a OBB-like BVH with tetrahedra as leafs was used to detect collisions on a FEM-based model of the lower limbs. The collision detection method is associated with the collision response to deal with self-collision and permanent contact on large areas.

Discrete orientation polytopes (DOPs) were chosen in [Held et al., 1995] and [Klosowski et al., 1998] as bounding volumes to reach a compromise between the poor tightness of AABBs and bounding spheres, and the relatively high cost of overlap test and update associated with OBBs and convex hulls. More often called k-DOPs, this representation uses convex *polytopes*[1] whose facets are determined by halfspaces coming from a small fixed set of k orientations (see figure 4.3). More recently, k-DOPs were used in [Mezger et al., 2003] for cloth simulation. Observe that 6-DOPs are equivalent to AABBs.

Regarding the *arity* of the trees, for rigid bodies binary trees are commonly chosen. However, for deformable objects, 4-ary trees and 8-ary trees have shown better performance. Two causes are in the origin of this fact: the recursion depth in query time is lower, which is less memory consuming; fewer nodes need to be updated, reducing the total update cost [Teschner et al., 2004].

One variation that worth mention for BVH is not using polygons as leafs of the tree, but a probability of them to exist in a BV instead [Klein and Zachmann, 2003]. It can be classified as a stochastic or probabilistic method but we believe it is closer to the hierarchical ones.

[1]The term polytope is a generalization to any dimension of polygon and polyhedron, and is used for a variety of related mathematical concepts.

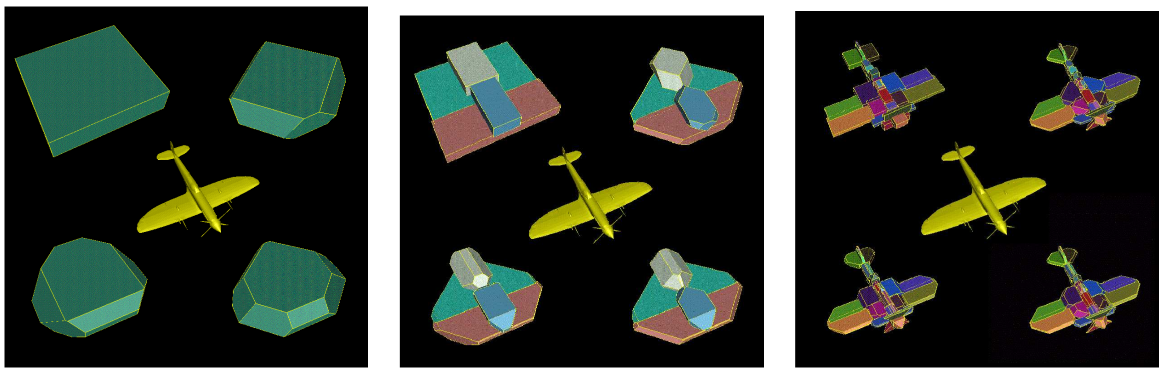

(a) Level 0 (b) Level 2 (c) Level 5

Figure 4.3: Example of k-DOPs. The images show selected levels of a bounding-volume tree for a model of an aircraft. For comparison purposes, the approximations achieved by means of 6-DOPs, 14-DOPs, 18-DOPs, and 26-DOPs are displayed for each level [Held, 2004].

4.2.1.3 Spatial partitioning representations

This approach has similarities with bounding volume representations (section 4.2.1.2) - which are also called object partitioning representations - such that sometimes they are both classified under the name of hierarchical volume bounding. However, the idea here is subdividing the space not the objects. Dividing the space occupied by the objects, it is necessary to check contact only between objects or object elements that are located nearby or in the same subspace.

Using space decomposition in a hierarchical manner can further speed up the collision detection process. Octree-like structures [Bandi and Thalmann, 1995] [Hamada and Hori, 1996], BSP-trees [Naylor et al., 1990], brep-indices [Bouma and George Vanecek, 1991], tetrahedral meshes [Klosowski et al., 1998], regular grids [Garcia-Alonso et al., 1994] and spatial hashing [Turk, 1990] [Ganovelli et al., 2000] are examples of spatial partitioning representations. Octrees have been much used. They recursively partition the space into octants. Then, octants are labelled *empty*, *full* or *mixed*, standing respectively for: non occupied by any part of any object; completely occupied by one object; partially occupied. It allows to avoid checking the octants labelled as *empty*, and the representation must be refined only when pairs of octants in which one *full* is contacting one *mixed* or two *mixed* are contacting. Although the cube is the standard octree primitive, we can find works on spherical octrees and other less diffused representations.

Binary space partitioning trees (BSP) are also very much used [Melax, 2000]. In this case, the space is recursively cut by hyperplanes. With BSP trees, in opposition to octrees, partitioning is not restricted to be axis-aligned. It allows using a simpler procedure when rigid transformations like orientation changes occur. Transformations are then applied to each hyperplane, without rebuilding the whole representation.

A third structure we want to highlight here is spatial hashing, because it has recently been used for collision detection with deformable models [Teschner et al., 2003].

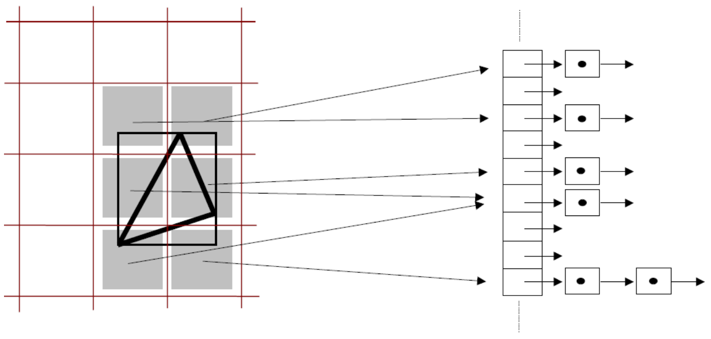


Figure 4.4: Spatial hashing. Hash values are computed for all grid cells covered by the AABB of a tetrahedron. The tetrahedron is checked for intersection with all vertices found at these hash indices [Teschner et al., 2003].

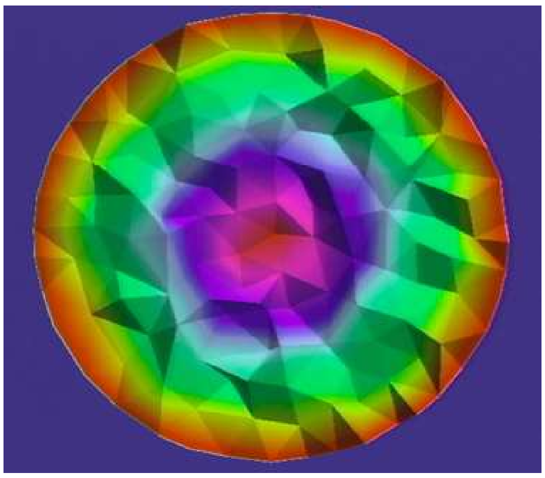
The idea of the algorithm is subdividing the space using a relatively fine grid. And, instead of using complex and costly data structures to keep track of the cells, indexing them using a hash function that maps 3D cells to a hash table (see figure 4.4). In [Teschner et al., 2003], the authors propose a spatial hashing method to detect collisions between deformable objects composed of tetrahedra. Their experiments showed that an environment up to 20 k tetrahedra can be processed in real-time.

4.2.1.4 Inexact probabilistic methods

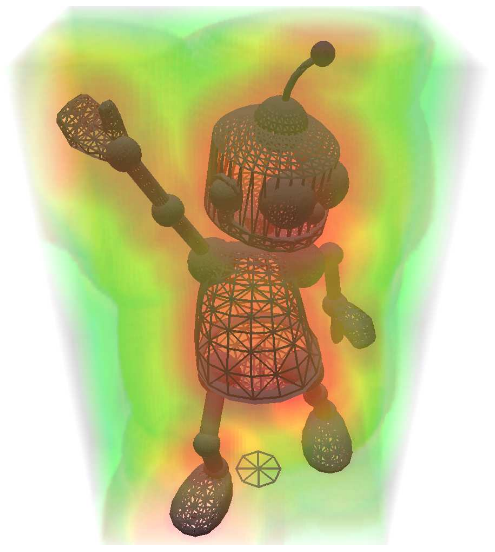
Polygonal modelled objects are inexact by definition, because even with a very large number of very small polygons, one can only approximate the shape of most objects. Based on that assertion, and in the fact that physically correct and physically plausible simulations are perceived by humans in the same way, some methods for collision detection use heuristical approaches to improve performance of collision processors.

The most basic approach using probability to reduce number of collision checks would be randomly selecting a small set of features to be tested. As consequence, the quality of detection would also be random. To better drive the chances toward checking elements actually prone to intersect, authors like [Lin and Canny, 1992], [Lin, 1993], [Lin and Manocha, 1995] and [Raghupathi et al., 2004] propose solutions considering temporal and spatial coherence. Temporal coherence is a property of dynamic environments allowing to assume that, if the motion between two timesteps is small, the geometric relationship may only differ slightly from one step to the previous one. Spatial coherence, in turn, permits to suppose that, if a given point has a high probability to collide, any point close to it has a similar probability, and one of them can be excluded of the interference checking. By means of the parameters used to quantify time and space coherence, it is possible to find the best compromise between the quality of the collision detection and performance of the applications.

The proximities method we introduce in section 4.3.1.3, makes also use of temporal and spatial coherence.



(a) The internal distance field of a sphere (adapted from [Fisher and Lin, 2001]).



(b) 3D Distance Field of Hugo Model (extracted from [Sud et al., 2004]). Distance to the surface is color coded, increasing from red to green to blue.

Figure 4.5: Examples of distance fields.

4.2.1.5 Techniques using distance fields

Distance fields or *distance volumes* are not a specific collision detection technique. They have been used to a number of applications like morphing, volumetric modeling, motion planning and fire animation [Teschner et al., 2004]. A distance field is a scalar field that specifies the minimum distance to a shape, where the distance may be signed to distinguish between the inside and outside of the shape. As simple examples, consider the distance field of the unit sphere S in $\mathbb{R}^3$ given by $h(x) = 1 - (x^2 + y^2 + z^2)^{\frac{1}{2}}$, in which h is the Euclidean signed distance from S, or $h(x) = 1 - (x^2 + y^2 + z^2)$, in which h is the algebraic signed distance from S, or $h(x) = (1 - (x^2 + y^2 + z^2))^2$, in which h is an unsigned distance from S [Frisken et al., 2000]. Figure 4.5 shows examples of distance fields.

The distance field is an effective representation of shape and can provide very precise information for collision detection. However, regularly sampled distance fields, like uniform 3D grids, have huge memory requirements and limited resolution when representing objects with fine details, even when the fine detail occupies only a small fraction of the volume. To overcome this limitation, many strategies have been proposed. Adaptively sampled distance fields (ADFs) [Frisken et al., 2000] use a BVH, typically an Octree, for adaptive detail-directed sampling, with high sampling rates in regions where the distance field contains fine detail and low sampling rates where the field varies smoothly. Other BVH, like BSP-trees can be used with the advantage of a reduced memory consumption. However, the construction of some structures can be computationally expensive. [Fisher and Lin, 2001] present a geometric technique, which first precomputes internal distance fields for each object, deforms each field on the fly, and then later utilizes them for enforcing the non-penetration constraints based on a penalty method. By taking advantage of precomputed distance fields that deform as the finite element mesh deforms, their algorithm enables efficient computation of penalty forces and their derivatives, and yields a versatile and robust contact resolution algorithm. Similarly, in [Hirota, 2001] the concept of *material depth* is used as an approximation of the distance field for the interior of the objects.

Updates of a distance field are a common bottleneck of all methods. That is a big issue when the objects are deformable and the field must be updated. Many techniques have been presented proposing efficient approaches for field generation. We can mention methods based on Voronoi diagrams [Kenneth E. Hoff et al., 1999], and propagation methods computed near the triangulated surfaces. Nevertheless, faster methods are not very accurate, while sufficiently accurate methods are not interactive when dealing with deformable objects. Despite the experiments based on graphics hardware [Vassilev et al., 2001] and other image-based techniques [Kenneth E. Hoff et al., 2001] to accelerate these updates, for interactive applications, only interactions between deformable objects with a rigid object can be carried out.

4.2.1.6 Image-based methods

Image-based collision detection or image-space techniques are approaches processing projections of objects to accelerate collision tests.

One of the first examples of image-based approaches is [Shinya and Forgue, 1991]. They first rasterize projections of the objects and calculate the z-values analogously to the z-buffer visibility algorithm. Z-values for each pixel are compared to determine overlapping in the z-direction. Their method, as well as the one of [Baciu et al., 1999], work only with convex objects, does not consider self-collisions, and was applied for rigid objects only. However, as they do not require any preprocessing, image-based methods are suitable for environments with deformable objects. Later, non-convex objects [Heidelberger et al., 2003] and self-collisions [Heidelberger et al., 2004] have been contemplated. And more recently, image-space techniques have been implemented in graphics hardware [Govindaraju et al., 2003] [Teschner et al., 2004]. [Heidelberger et al., 2004] also optimized the use of layered depth images (LDI), introduced by [Shade et al., 1998], for collision detection. The main drawback seem to be the dependence of image-based methods to the image-space resolution. Collisions can be missed between small polygons, and it is complex to obtain precise information for collision response.

4.2.1.7 GPU-based methods

Algorithms based on the graphics processing unity (GPU) have recently become very popular with the increasing adaptability of the graphics hardware. Implementations specific for the graphics hardware are being developed in diverse areas and applications. It is particularly interesting for collision detection because of the proximity of the image-based methods (section 4.2.1.6) with the GPU. Sometimes, image-based and GPU-based methods are still classified in the same group.

Works being used to perform interference and proximity computations include [Knott and Pai, 2003] [Heidelberger et al., 2003] [Heidelberger et al., 2004] [Govindaraju et al., 2003] [Govindaraju et al., 2005]. They usually do not need preprocessing and consequently are well adapted to dynamically deforming objects. Some of them can detect self-collisions, but may be limited to process closed objects, and have difficulties with some types of contact configurations.

When the algorithm depends on the graphics buffer resolution, which is often the case with image-based techniques (on GPU or CPU), it can miss collisions due to sampling errors. However, we can also imagine traditional collision detection techniques based on polygons to be implemented on the graphics hardware. A data structure adaptation would be necessary, but they could take advantage of the GPU parallelism to boost the performance without that resolution issue.

4.2.2 Collision response review

4.2.2.1 Penalty method

The physical interpretation of the penalty method is a rubber band that attracts the physical state to the subspace *g(x)=0*. The penalty method adds a quadratic energy term that penalizes violations of constraints [Platt and Barr, 1988]. Thus, it converts a constrained optimization problem:

$$\textit{minimize } f(x) \textit{ such that } g(x) = 0 \tag{4.2}$$

to an unconstrained problem:

$$\textit{minimize } f(x) + kg(x)^2 \textit{ as } k \rightarrow \infty \tag{4.3}$$

where deviation from the constraint is penalized; that is, in the new problem, satisfaction on the constraint is encouraged, but not strictly enforced. In equation 4.3, *kg(x)*^{2} is called the penalty function. The idea of the method is that as k grows larger, potential solutions for x must make $kg(x)^2$ smaller to minimize equation 4.3. And as k goes to infinity, the solution of equation 4.3 must satisfy $g(x) = 0$ while minimizing $f(x)$. The method presents a theoretically firm basis, but, in practice, it is not a very robust numerical method. This occurs because as k grows, equation 4.3 can become very poorly conditioned and difficult to solve [Baraff, 1993]. Let us see how the penalty method can be applied to calculate collision and contact forces. Suppose that two objects A and B collide. The penalty method allows objects in the simulation to penetrate each other. Upon penetration, a temporary spring is attached between the contact points. This spring compresses over a very short time and applies equal and opposite forces to each body so that they will separate. As objects interpenetrate, the forces generated increase with the penetration distance. Thus, the penalty force can be written as:

$$\vec{F} = -k(x(t) - p(x(t))) \tag{4.4}$$

Here, $\vec{F}$ is the force applied against each object, $p(x(t))$ denote the closest point on the surface to $x(t)$, and k is the spring constant which is a measure of the spring stiffness.

Often, there is friction or other dissipation in the spring [Barzel 92]. A damping term can be added to help to limit oscillations that springs induce in a model. Thus, the force equation can be then written as:

$$\vec{F} = -(k_s(\|x\| - x_0) + k_d\vec{v})\frac{x}{\|x\|} \tag{4.5}$$

where x is the difference vector from the fixed point to the surface point, $\vec{v}$ is the velocity, x_0 is the initial length of the spring and k_s is the spring constant and k_d is the damping coefficient. The penalty method has been employed in a vast number of simulations in Computer Graphics and Robotics [Terzopoulos et al., 1987] [Platt and Barr, 1988] [Moore and Wilhelms, 1988] [McKenna and Zeltzer, 1990] [Joukhadar et al., 1998] [Desbrun et al., 1999] [Jansson and Vergeest, 2000] [Hirota, 2001] aiming to enforce non-interpenetration constraints. Their applications include the simulation of deformable bodies, cloth, and articulated rigid bodies. Moore and Wilhelms [Moore and Wilhelms, 1988] pioneered the method by introducing penalty forces to prevent bodies in resting contact from penetrating. The main drawback of this method is finding penalty constants that are effective for different objects in different environments. The choice of these constants interacts with the choice of the integration time step, because to keep penetration to a minimum, the penalty constants need to be set as high as possible, but this imply large contact forces and these contact forces demand small integration time steps. Consequently, it is computationally expensive, with stiffer spring needing smaller time steps to solve the resulting equations accurately. Despite their limitations, the penalty method presents advantages as computational simplicity, facility of incorporating static friction models, and ability to simulate a variety of surface characteristics.

4.2.2.2 Constraint-based method

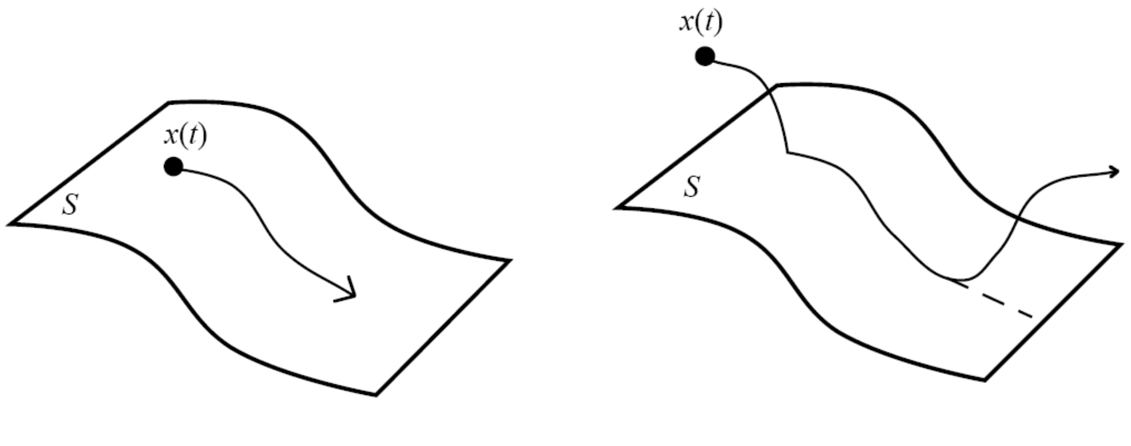
Constraints are used to describe interactions between objects, which often occur through physical contact. Constraint-based method computes constraint forces that are designed to cancel any external acceleration that would result in interpenetration. This method results in simulations where interpenetration is completely eliminated. However, it is required solving nonlinear systems of equations [Baraff, 1989]. Suppose that S is a surface in the three-space ($R3$) and P is a particle. The particle's position p at time t is a function $p(t)$ and an external force acts on the particle at time t. Now suppose that the P is constrained to always remain on S. The constraint on the particle can be expressed by

$$C(p(t)) = 0 \tag{4.6}$$

where C is a scalar function that models the surface S implicitly. This is an example of an equality-constrained problem (see figure 4.6(a)). However, the particle P can be constrained to lie either on or "above"S, then this constraint can be written as

$$C(p(t)) \geq 0 \tag{4.7}$$

Here we have the example of an inequality-constrained dynamics problem (see figure 4.6(b)).



(a) $C(p(t)) = 0$ (b) $C(p(t)) \geq 0$

Figure 4.6: A particle and a constraint surface. The particle in (a), constrained by $C(x(t)) = 0$, must remain on the surface S. To satisfy the constraint, the particle in (b) may lie on or "above"the surface S, but may not move "below"the surface. [Baraff, 1993].

Consider again that we introduce a constraint force $\vec{F_c}(t)$ into the inequality-constrained problem. Since $\vec{F_c}(t)$ acts in a direction normal to the contact surface at the contact point, we can write

$$\vec{F_c}(t) = f(t)\hat{n}(t) \tag{4.8}$$

where $\hat{n}(t)$ is the surface normal at the point $p(t)$, f is an unknown scalar and must satisfy

$$f(t) \geq 0 \ . \tag{4.9}$$

Constraint-based method can be used to solve the problem of resting contact, i.e., objects are in resting contact, but are not effectively *colliding*. For a single contact point, the scalar f (Eq. 4.8) is easily computed and must satisfy the following conditions:

1. $f(t) \geq 0$, since the constraint force must be repulsive; that is, it can push bodies apart, but never pull.

2. Eq. 4.8 must be strong enough to prevent the objects of being pushed towards each other.

3. When the objects begin to separate, $\vec{F_c}(t) = 0$. This condition is written as $f(t)a = 0$ which ensures that if $a > 0, f(t) = 0$. Here a denotes the acceleration.

As a and f are linearly related, we can write

$$a = cf(t) + d \tag{4.10}$$

where c and d are variables of the system. Using equation 4.10 we can say that f must satisfy the conditions

$$f \geq 0 \ , \ cf(t) + d \geq 0 \ and \ f(t)(cf(t) + d) = 0 \ . \tag{4.11}$$

Now consider a system of frictionless bodies contacting at n different points. For each contact point there will be some force normal to the surface at the contact point. For that reason, it is necessary to calculate contact force magnitude, $f(t)$, at each contact point.

Thus, for each contact point p_i between two bodies we have a relative acceleration a_i and a contact force magnitude f_i at time t. We will represent the collection of all a_i by the vector a, and the collection of all f_i is a vector f [Baraff, 1989], in this manner we can write

$$a = Af + b \tag{4.12}$$

where A represents the masses and contact geometries of the bodies and b represents the external and inertial forces [Baraff, 1994]. The matrix A and the vector b are determined from the known configuration of the system. At each contact point the same conditions as in Eq. 4.11 must be satisfied, yielding the system

$$f \geq 0 \ , \ a \geq 0 \ for \ f_i a_i = 0 \ for \ 1 \leq i \leq n \ . \tag{4.13}$$

Therefore, using the Eq. 4.12 we can rewrite the conditions on f_i as

$$f \geq 0 \ , \ a = Af + b \geq 0 \ and \ f^T a = f^T(Af + b) \ . \tag{4.14}$$

The Eq. 4.14 is known as a linear complementary problem (LCP). A detailed explanation of LCP can be found in [Cottle et al., 1992]. Furthermore, [Baraff, 1994] salients that the conditions described in Eq. 4.14 can be considered as a quadratic program (QP) that is, a vector f that satisfies those conditions is a solution to the QP

$$\min_f f^T(Af + b) \ subject \ to \ \begin{cases} Af + b \geq 0 \\ f \geq 0 \end{cases} \tag{4.15}$$

We can find simulation methods that have used LCP to calculate forces between contacting rigid bodies in [Baraff, 1989][Baraff, 1994][Pang and Trinkle, 1996][Faure, 1996][Popovic' et al., 2000]. According to [Baraff, 1989], solving for the accelerations in the contact points and substituting the result into the constraint equations results in a system of equations which can be used to compute the contact forces. So, linear programming techniques are used to formulate and heuristically solve a system of inequality and equality constraints on the forces. This system must assure that the contact forces will prevent the interpenetration and satisfy the Newton's laws. When there is no friction, the LCP is convex and solutions can be computed using algorithms that run in worst case exponential time but expected polynomial time in the number of contacts. Friction can be incorporated to the algorithms by modifying or adding constraints to the LCP. In [Baraff, 1994] an algorithm for computing the contact forces between objects with static and dynamic friction is presented. This algorithm is an adaptation of the one described by Dantzig, which is related to pivoting methods for solving linear and quadratic programming. Dantzig's algorithm for solving LCP is described in [Cottle et al., 1992]. Baraff's algorithm presents the followings problems:

Convergence. It was not proven with friction and in this case we have no guarantee that the algorithm would terminate;

Control of computation time. An unpredictable number of iterations (unless the contacts are frictionless) is necessary to maintain the previously computed values for the forces and accelerations within the correct bounds.

Faure presented a method to compute resting contact forces based on energy transfer between the bodies in contact that satisfies both the conservation of energy and the inequality constraints [Faure, 1996]. The first iteration of the algorithm consists of a global dynamic solution involving inertia and external forces that satisfy the conservation of energy. The subsequent iterations consist of global redistributions of energy through the solids. This method simultaneously handles both static and sliding friction, like the approach presented in [Baraff, 1994], and avoid the problems of convergence and computation time. The former is proven in the frictionless case, and the later occurs when either desired precision or an allowed computation time is reached.

4.2.2.3 Impulse-based method

In Physics, when we apply a force on an object, we also exert an impulse on it. Suppose that two objects are in contact at the point P_c at time t_c, and they have a relative velocity towards each other. Unless the relative velocity is abruptly changed, interpenetration will immediately occur after time t_c. Thus, we apply an impulse, which will instantaneously change the velocities of the two objects. An impulse $\vec{J}$ is the product of a force $\vec{F}$ by the time interval Δt that $\vec{F}$ acts on an object,

$$\vec{J} = \vec{F}\Delta t. \tag{4.16}$$

If we apply an impulse $\vec{J}$ to a rigid body with mass m, then the change in linear velocity v of the body is

$$\Delta v = \frac{\vec{J}}{m}. \tag{4.17}$$

Let us see how an impulse is treated in impulse-based dynamic simulation of rigid bodies. Impulse-based method for dynamic simulation was pioneered by Hahn [Hahn, 1988] and extended by Mirtich and Canny [Mirtich and Canny, 1994]. The central idea of this approach is to model all contacts between objects through a series of impulses [Mirtich and Canny, 1994]. It is based on the treatment of contacts as momentary collisions, where two objects are separated by applying a brief impulsive force. When

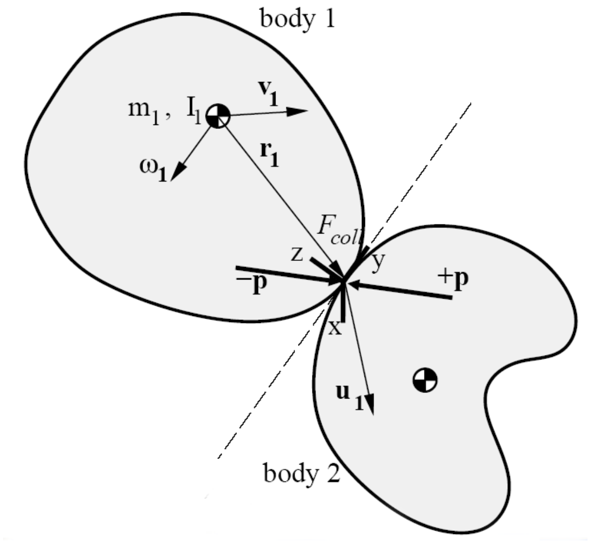


Figure 4.7: Collision response between two bodies [Mirtich and Canny, 1994].

a collision is detected between a pair of objects, a collision impulse will be calculated. This impulse must prevent interpenetration and obeys certain physical laws relating to friction and energy restitution. The collision response is calculated as an impulse p which is applied on one object and, from Newton's third law, an impulse $-p$ is applied on the other object (see figure 4.7).

The frictional force is dependent on the relative sliding velocity of the objects in contact, and this velocity is not constant during a collision. Because of that, the dynamics of the object must be analyzed during the collision to compute the collision impulse. So, at each time frame the initial relative velocity between the two objects, $\vec{u}$, is computed from

$$\vec{u} = \vec{u}_1 - \vec{u}_2 \tag{4.18}$$

where $\vec{u}_1$ and $\vec{u}_2$ are the absolute velocities of the objects 1 and 2 respectively. These velocities are calculated as

$$\vec{u}_i = \vec{v}_1 + \vec{w}_i \times R_i \tag{4.19}$$

Here $\vec{v}$ is the linear velocity of center of mass, w_i is the angular velocity of the rigid body around the center of mass, and R is the position relative to the center of mass. Then $\vec{u}$ is projected to the collision frame. If u_z is non-negative, no action needs to be taken, because the objects are not in contact with each other; if u_z is negative, a collision impulse must be applied to prevent interpenetration. Then, equation 4.20 is used for numerical integration:

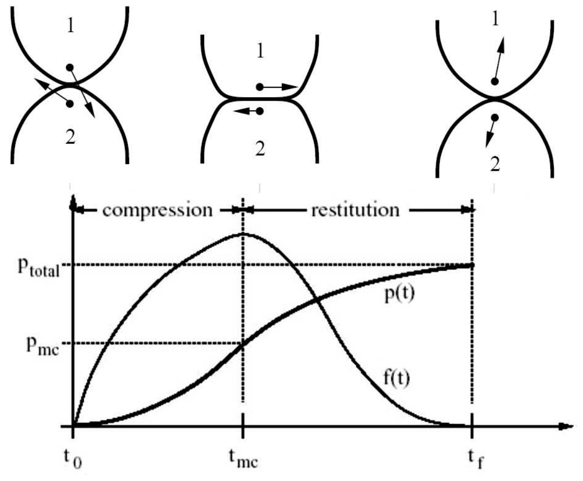


Figure 4.8: Phases of a collision [Mirtich and Canny, 1994].

$$\begin{bmatrix} u'_x \\ u'_y \\ u'_z \end{bmatrix} = M \begin{bmatrix} -\mu \frac{u_x}{\sqrt{{u_x}^2+{u_y}^2}} \\ -\mu \frac{u_y}{\sqrt{{u_x}^2+{u_y}^2}} \\ 1 \end{bmatrix} \tag{4.20}$$

where M is a matrix dependent only upon the masses and mass matrices of the colliding bodies, and the location of the contact point relative to their centers of mass. μ is the coefficient of friction.

During integration, u_z will increase until it reaches zero. When this occurs, the point of maximum compression is reached. As it can be seen in figure 4.8, this point is the limit between compression and restitution phase in a collision process, where $f(t)$ and $p(t)$ are respectively the force and total impulse delivered at time t in the collision.

At this point, the integration variable p_z is the impulse that has been applied. Based on the Poisson's hypothesis, p_z multiplied by $(1+e)$ gives the terminating value for p_z. Here e is the coefficient of restitution that can range from 0 to 1. A value of 0 means that practically all energy is lost (the objects do not separate after collision; plastic collision), and value of 1 means that no energy is lost (elastic collision) [Mirtich and Canny, 1994]. So, we continue the integration until the termination value $p_z(1+e)$ in order to find the final value for u that is the change in the contact point velocity of an object over the course of the collision. The final value of the relative velocity Δu is used to calculate the impulse by reversing equation 4.20, that is,

$$p = M^{-1}\Delta u. \tag{4.21}$$

With this calculated impulse, the positions and orientations of all the objects are recalculated by assuming ballistic trajectories. Hahn presented a method to calculate collision impulse with friction at a single point. This method models sliding and rolling contacts using impact equations. The contact forces are not computed explicitly, but occur only as time averages of reaction impulses [Hahn, 1988]. Mirtich and Cany ex-

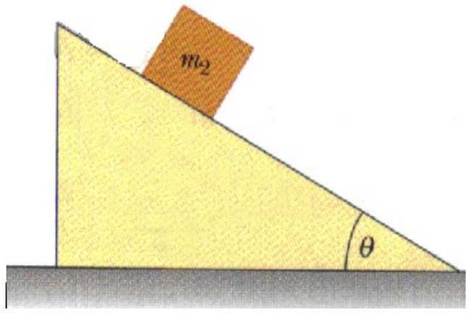


Figure 4.9: A box resting on an inclined plane.

tended the applicability of Hahn's method to resting contacts, and gave a more unified treatment for multiple objects in contact and a fully general treatment of frictional collisions. The interaction between objects is modeled as a series of tiny micro-collisions that are frequent collisions between objects in continuous contact, for example, a box resting on a floor. The effect of a micro-collision is to reverse the motion direction of the object [Mirtich and Canny, 1994]. This work, in turn, has been extended in [Mirtich and Canny, 1995] to support generalized articulated bodies, but the constraints equations are still fundamentally based on equation of motion for simple rigid body systems. The major disadvantage of this method is its inability to efficiently handle simultaneous and persistent contacts. Consider for example a box resting on an inclined plane (see Figure 4.9). Under impulse-based simulation, the box gradually slides downward. This happens because constant micro-collisions are bouncing the box off the surface at a high frequency. This means that the frictional forces that would normally prevent the box from sliding are only acting intermittently and the box slides, regardless of the properties of the surface.

Nevertheless, impulse-based method presents advantages like simplicity and robustness in comparison with constraint-based method, real-time speed and physical accuracy [Mirtich and Canny, 1994].

4.2.2.4 Comparison

We have presented the main methods for calculating forces between two objects in resting contact or collision. We analyze these methods from a critical point of view and present the advantages and disadvantages of each of them which offered us the guidelines to select a method suitable to our case.

Penalty forces are usually computed as elastic forces that depend on the interpenetration between objects. The main problem with penalty method is that it causes instabilities or unwanted vibration. This can happen if the stiffness of contacts is too high, or the force update rate is not high enough. Besides, penalty for rigid bodies are often computational expensive, give only approximate results and may require adjustments for different simulation conditions. In particular, the differential equations that arise using penalty methods may be "stiff"and require an excessive number of time-steps during simulation to obtain accurate results. Additionally, the correctness

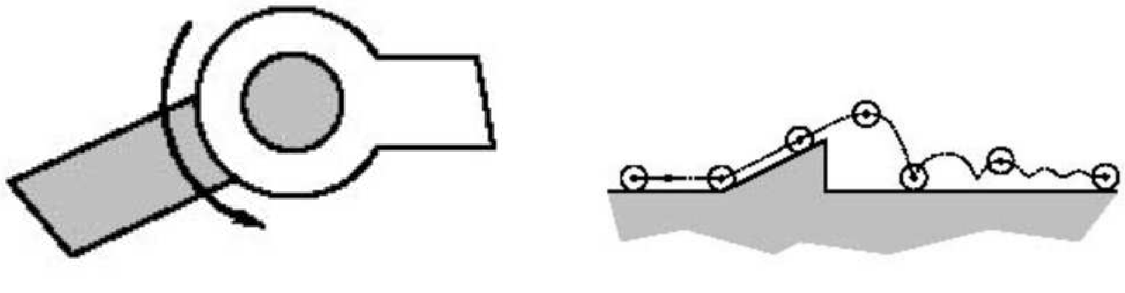
	Penalty Method	**Constraint-based Method**	**Impulse-based Method**
Type of Objects	Rigid and deformable	Rigid, deformable and articulated	Only rigid
Concept and implementation complexity	Simple	Complex	Medium
Computational cost	High	Low	Medium
Number of time-steps required	High	Low	Low to medium
Contact types supported	Problems with stiff contacts	Problems when contact modes change frequently	Problems with resting contacts
Parallel computation	Possible	Complicated and difficult	Potential
Physical accuracy	Depends on time discretization	Accurate in most cases	Accurate
Accuracy verification	Very difficult	Easy	Easy

Table 4.1: Comparative analysis of the different methods

of the simulation is very difficult to verify. In their defense, penalty method for rigid bodies presents computational simplicity and effectiveness. Add to this, the ease of incorporating static friction models and the ability to simulate qualitatively a variety of surface characteristics. This method for rigid bodies is easily extendible for flexible bodies.

In contrast, **constraint-based method** is based on finding exact contacts between the rigid bodies. It gives exact answers and produce differential equations that require far fewer time steps during simulation. The correctness of simulation when using constraint-based method is easily provable because it is directly based on the laws of Newtonian dynamics. Unlike penalty method, this method is computationally more efficient unless the collision is very gentle. In this case, the penalty method is more adequate. However, constraint-based method is much more complex to derive and implement. Besides, the computation is too complicated, the assumptions of perfectly rigid bodies interacting without friction are too restrictive and it must declare each contact to be a resting contact or colliding contact.

Unlike constraint-based method, in the **impulse-based method** non-penetration constraints do not exist because the collision is responsible for enforcing separation between two bodies in collision. Besides, it is conceptually and algorithmically simpler. The impulse-based method works well on systems of bodies where the contacts are changing rapidly, but it has difficulty to adequately simulate frequent and prolonged contact. In contrast, the penalty method is a poor choice for simulating brief rigid


(a) A suitable situation for constraint-based method.

(b) A suitable situation for impulse-based method.

Figure 4.10: The choice of the best collision treatment method depends on contact types.

body collision, which demand high spring constants, but provides an efficient and flexible qualitative model of prolonged contacts. Finally, there are simulations in which constraint-based method is more appropriate than impulse-based method. Consider a hinge joint (see Figure 4.10(a)). We can model this hinge by micro-collisions between the hinge pin and its sheath. However, due to the enormous amount of collision detection and resolution that would be necessary to model this contact, the simulation would be too slow. On the other hand, the constraint-based method is not well suited to situations as shown in Figure 4.10(b). Under constraint-based simulation, the constraints change when the ball begins travelling up the ramp, leaves the ramp and settles into a roll along the ground. All these occurrences must be detected and processed and new equations of motion for the system must be derived at every transition [Mirtich and Canny, 1995].

4.3 Our contact model

In the context of this work we are particularly interested on collisions taking place around a human articulation. Such environment can influence a lot the way the collisions happen and the methods that are more appropriated to detect and respond to them. We highlight two features particularly important: *permanent contact*; *layered topology.*

To provide good stability to the human articulations, their internal structures are assembled in a way that at least two of them are in **permanent contact**. The contact between the surfaces of the acetabular cup and the femoral head cartilages is an example. Also for good stability, there is a permanent stress inside the joint - which the distribution change when the joint moves - provoked by stressed ligaments, muscular actuation and charges due to interaction with objects of the world. At the same time, the classical problems of collision detection remain present. An example is that not all surfaces that can be in contact at some given posture are in contact during the whole motion.

The terms used in human anatomy (refer to section 5.2) to name body parts and define topological relations also apply within a joint, meaning that starting from the joint center to the body surface we know for each element that it is always on or under another, or between two others (analogously with proximal, medial and distal in Anatomy). This topological knowledge always use the center of the joint as reference, avoiding orientation ambiguity, and it is what we decided to call **layered topology**.

We have implemented and evaluated several approaches for both detection and response to collisions. In the next two sections we present the methods we tested and the results we obtained in terms of accuracy and performance for both collision detection and response. In the list of methods presented, general approaches come first and then the methods become more and more joint specific, which reflects our research methodology. Note that although the problem of detecting collisions is very different of the one of reacting to a collision situation, some of the methods proposed for detection only make sense when combined with some of the response methods, and vice-versa.

4.3.1 Collision detection

Collision detection is the art of being aware of the collisions as soon as they take place. Other methods exist which have the goal of predicting a collision before it takes place. Yet others exist which the goal is finding the exact instant a collision happens. In this work, a collision is considered and must be detected in time t if any pair or objects sharing one or more points in space exist at that moment.

4.3.1.1 Sphere penetration

A collision between spheres is one of the most straightforward situations to be detected. Every sphere corresponds to a molecule in the deformation model (section 3.4) and is described as a central position and a radius, both obtained from the molecule. After optimizing the object-to-object test by a bounding box test, we simply compare the distances between centers of candidate spheres with the sum of their radius. If the distance between the centers of the two spheres is smaller than the sum of their radius they collide; otherwise they do not (figure 4.11). The exceeding distance is used later in a penalty function to clear penetration (see section 4.3.2.1).

4.3.1.2 Mesh intersection

Our meshes are all composed of triangles, so computing mesh intersections means testing all triangles for interpenetration. It is not difficult to determine geometrically if two triangles in R_3 penetrate or not. However the algorithmic complexity for two meshes A and B is $O(n^2)$ in the worse case, where n is the number of triangles of the bigger mesh. When many meshes are involved the complexity increases even more. A number of methods exist and have been published to optimize the collision test by reducing the number of intersection tests [Lin, 1993] [Cohen et al., 1995] [Klosowski et al., 1998] [Li and Chen, 1998] [James and Pai, 2004]. Some of them are implemented

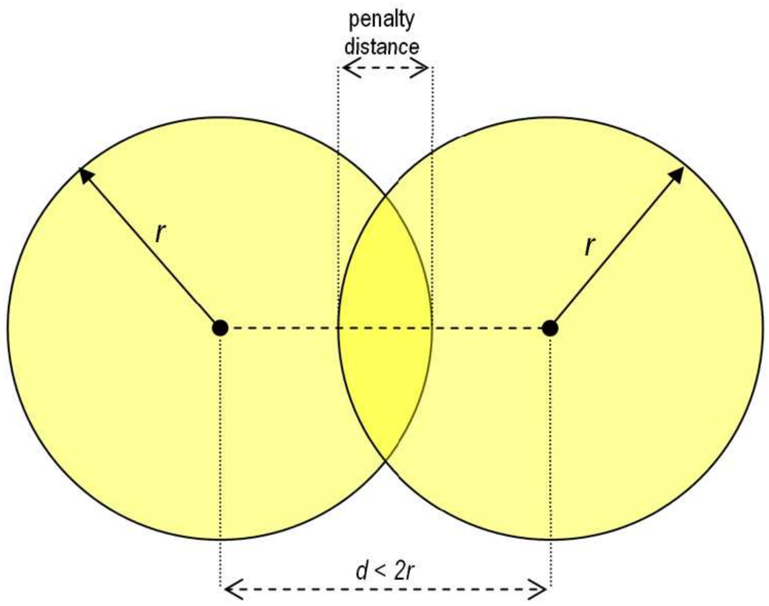


Figure 4.11: Two spheres colliding.

in the open library V-Collide [Hudson et al., 1997].

As a first approach we tried to detect mesh collisions was using V-Collide collision detection library, developed at the University of North Carolina by the *GAMMA* team. V-Collide is a "n-body"processor which works with polygon soups. It uses a fast n-body algorithm to decide which pairs of objects are potentially in contact, and then for each potential contact pair it calls functions of another library (RAPID [Gottschalk et al., 1996]) to determine whether the objects actually collide. RAPID is the acronym for "Robust and Accurate Polygon Interference Detection". It is a library that is applicable to polygon soups and operates only with triangles. While V-Collide identifies which objects potentially collide, RAPID processes these objects two by two to find triangle intersections. RAPID is based on two algorithms. The first algorithm uses a top-down decomposition technique to build a hierarchy of Oriented Bounding Boxes (OBBs) of an input polygon soup model. The second one realizes collision tests among OBB pairs. These tests consist of verifying whether two high-level OBBs overlap; if they overlap, then the algorithm verifies the overlapping of lower level OBBs, otherwise, the two objects do not collide and the algorithm ends. Finally, RAPID returns a list of contact pairs, where each contact pair is formed by a triangle taken from each of the objects; self-collisions are neglected.

Once we have obtained the colliding pairs from V-Collide/RAPID, we compare the positions of triangles vertices and use the penetration distances obtained with a penalty method to eliminate penetration (see section 4.3.2.1).

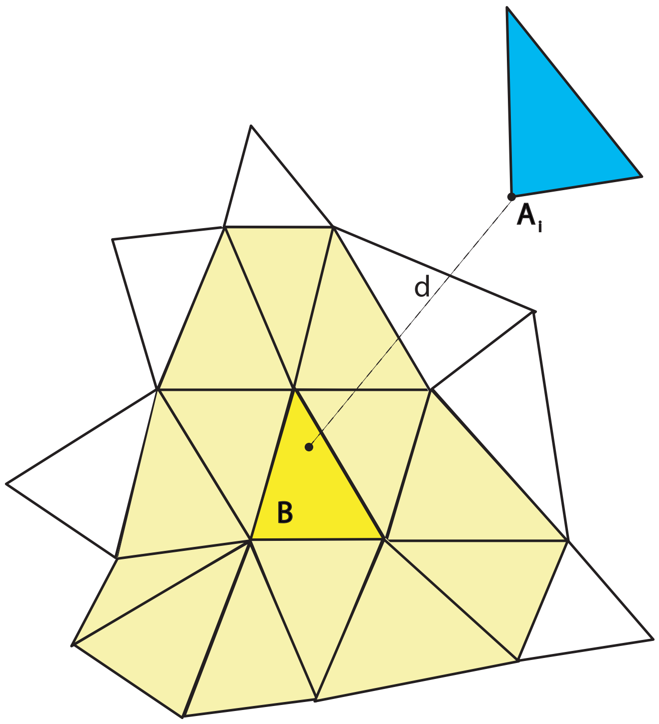


Figure 4.12: Example of a proximity. The vertex A_i is projected onto the closest triangle in B.

4.3.1.3 Proximity structure

We introduce here a technique for collision detection based on spatiotemporal coherence. It relies on a table we call proximity structure, which keeps the information of proximity between two objects, i.e., to each feature of one object, it knows which feature of the other object is the closest. Figure 4.12 illustrates a proximity.

The method has two parts. A time consuming pre-computed part initializes the proximity structure and, during simulation, another part is responsible to efficiently update the proximity structure. It is explained here considering its application to objects described as triangle meshes.

At **initialization**, an $O(n^2)$ algorithm exhaustively searches for each vertex of the object A, which triangle of the object B is the closest to it. In other words, it finds the triangle of B which the Voronoi region contains the vertex of A. Once the triangle is found, the closest point on the triangle surface to the vertex in A is calculated, and thus we obtain the distance between the vertex and the triangle. The proximity structure stores triples vertex-triangle-distance, and every time a distance is negative a collision can be signalized.

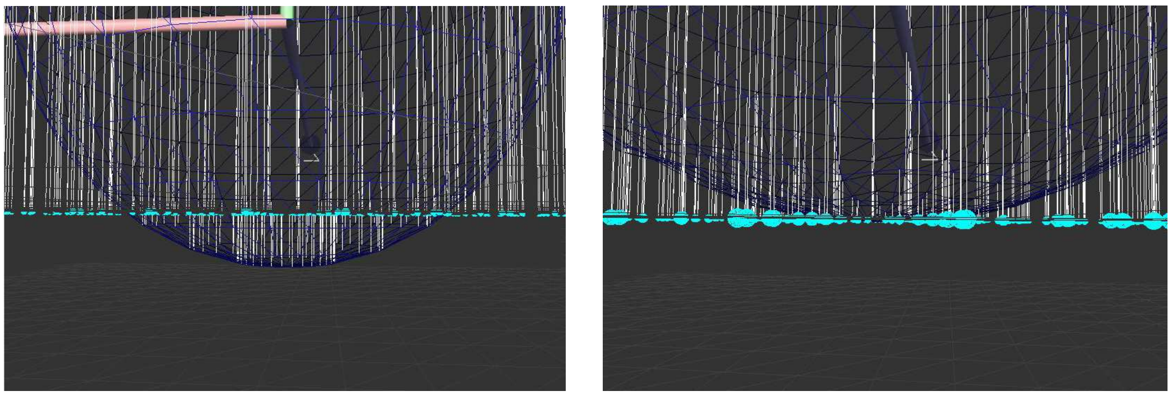
During simulation, the position between the two objects eventually changes, and their proximity relations too. Then, the proximity structure must be **updated**. Assuming that the relative motion is small within a simulation timestep, only the neighboring triangles (light yellow in figure 4.12) to the one in the proximity are tested for distance with the vertex. If one of them is closer, it replaces the former in the proximity structure. In consequence, the algorithm complexity goes down to $O(n)$ in the worse case. For clearness, we could say the complexity is $O(Cn)$, where the constant C is the average number of neighboring triangles, and n is the number of vertices susceptible to collide.

The distance test, both for the initialization and for the update, must be very precise because the distance is necessary afterwards in the collision response method. We compute distance vertex-triangle like shown in the algorithm 2.

Algorithm 2 Calculating distance vertex-triangle for proximities initialization and update.

```
Given a triangular face $F$ and a vertex $V$, and being $F_i$ a vertex of the face and $\vec{n}$
the normal of the face.
// compute the distance of V to the plane of F
$\vec{a} = V - F_i$
$distPlane = \vec{n} \cdot \vec{a}$
// compute the projection of V on the plane of F
$V' = V - (\vec{n} * distPlane)$
if $V'$ is inside $F$ then
  return $distPlane$
else
  // compute a vector on each edge of the face
  $\vec{ab} = F_1 - F_0$
  $\vec{bc} = F_2 - F_1$
  $\vec{ca} = F_0 - F_2$
  // project the point on the plane onto the lines formed by the edges of F
  $pointOnAB = F_0 + (\vec{ab} * (\overrightarrow{pointPlane - F_0}) \cdot \vec{ab})$
  $pointOnBC = F_1 + (\vec{bc} * (\overrightarrow{pointPlane - F_1}) \cdot \vec{bc})$
  $pointOnCA = F_2 + (\vec{ca} * (\overrightarrow{pointPlane - F_2}) \cdot \vec{ca})$
  // if the projections are out of the edges project them onto the vertices of F (only
  pointOnAB is shown here, do the same for BC and CA)
  if $(|F_0 - pointOnAB| > |F_0 - F_1|)$ or $(|F_1 - pointOnAB| > |F_0 - F_1|)$ then
    if $|F_0 - pointOnAB| < |F_1 - pointOnAB|$ then
      $pointOnAB = F_0$
    else
      $pointOnAB = F_1$
    endif
  endif
  // the closest projected point gives the distance
  dist = min(distance(V,pointOnAB), distance(V,pointOnBC), dis-
  tance(V,pointOnCA))
  return dist
endif
```

The proximity structure allows us to detect collisions and consequently avoid penetration for different kinds of scenes and objects. It is, nevertheless, more efficient than



(a) Before detection and avoidance (b) After detection and avoidance

Figure 4.13: Proximity structure test-case. A sphere interacting with another solid object before and after application of collision avoidance. The lines start from the vertices of the sphere and finish on the small spheres representing the closest point to the vertex.

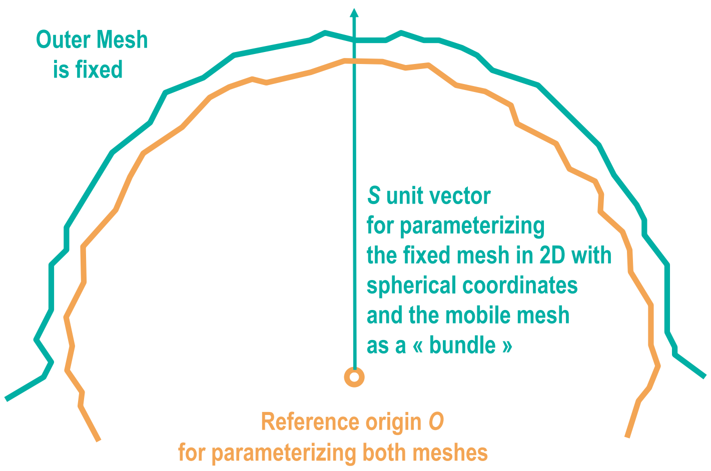


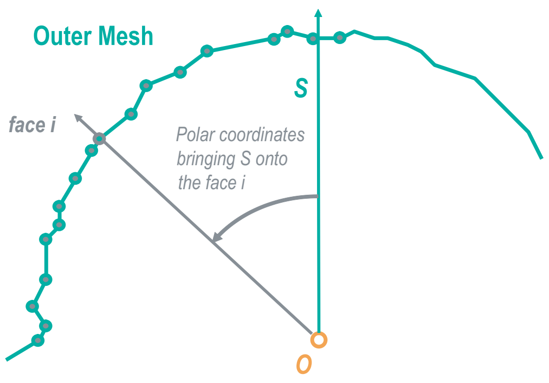
Figure 4.14: 2D conceptual view of the spherical sliding collision detection.

other techniques presented in section 4.2 when there is permanent contact along large areas of the objects involved. Figure 4.13 shows proximities in a test-case before and after collision detection.

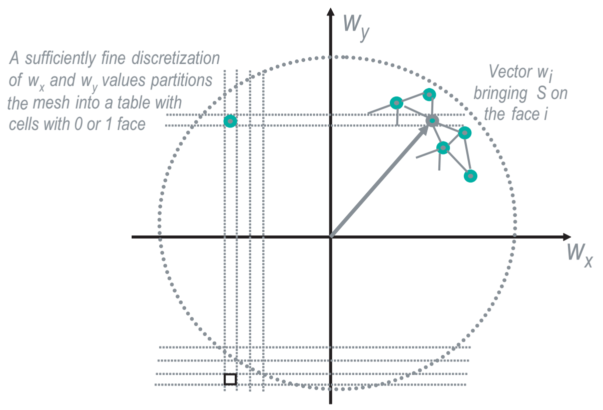
4.3.1.4 Spherical sliding

In this section we introduce our spherical sliding collision detection method, which measures in constant time the signed distance to the closest point for each point of two meshes. The meshes are assumed to be of quasi spherical shape or placed in layered topology, and be sliding against each other (see figure 4.14).

The method is based on a 2D hash table which the indexing is defined by the two angles (spherical coordinates, see equations 4.23 and 4.24) defining a vector starting at the origin of the local frame and passing through the center of a triangle of the mesh



(a) Parametrization of the vertex i with its spherical coordinates of radius 1.


W_y
A sufficiently fine discretization of w_x and w_y values partitions the mesh into a table with cells with 0 or 1 face
Vector w_i bringing S on the face i
W_x

(b) Parametrization of the 3D fixed mesh as a 2D mesh discretized into a table for finding the facing or closest neighbor in constant time.

Figure 4.15: Generating the hash table for spherical sliding collision detection.

considered fixed. Then, each element of the table stores a pointer to the triangle, and the two angles allow to access a unique element. To detect a collision, a vector is created starting on the same center but passing through the vertices of the mesh considered mobile. Then, the angles of that vector are calculated and used for a lookup on the hash table to find the triangle with which the vertex may collide. Finally, a point-to-point distance test determines if there is a collision. The hash function is presented in equation 4.22.

$$(row, col) = \left(\theta \frac{hashSize - 1}{2\Pi} + \frac{hashSize - 1}{2}, \phi \frac{hashSize - 1}{\Pi}\right) \tag{4.22}$$

$$\theta_i = \tan^{-1}\left(\frac{y_i}{x_i}\right) \tag{4.23}$$

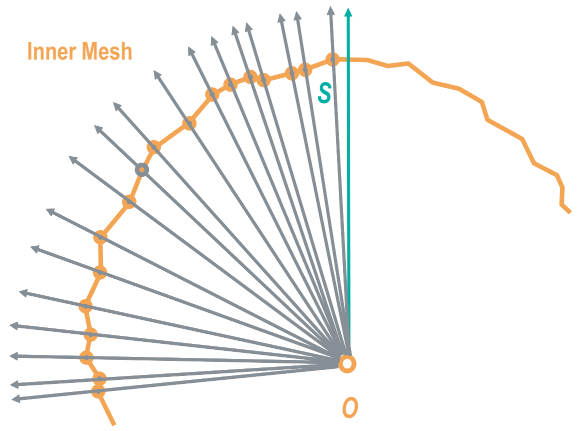
$$\phi_i = \cos^{-1}\left(\frac{z_i}{r_{sphere}}\right) \tag{4.24}$$

To **create** the hash table, every triangle of the fixed mesh is visited. For each of them, its spherical coordinates of radius 1 are used as hash key for the lookup table. Then, the corresponding position of the table is assigned to that triangle (figure 4.15(b)). The positions not assigned to triangles remain empty. The algorithm 3 summarizes the construction of the hash table.

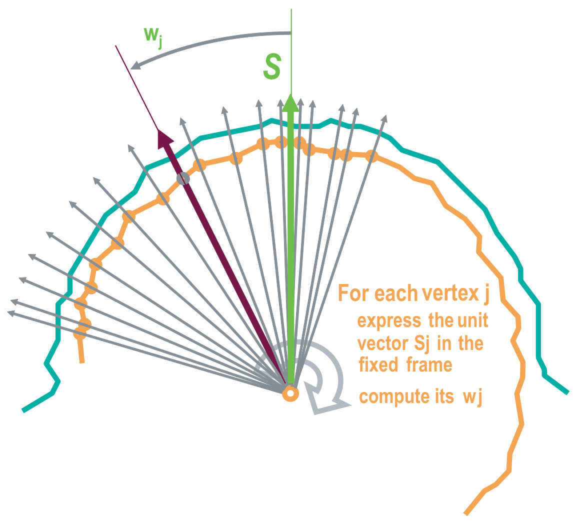
For **collision detection**, the mobile mesh is parameterized as a bundle of unit vectors passing through each vertex j, and expressed in the local coordinate system of the mobile mesh (figure 4.16(a)). After motion, the spherical coordinates of each w_j are calculated in the fixed frame (figure 4.16(b)). They are then applied in the hash function (equation 4.22) to obtain row and col. The triangle found in that position of the hash table is assumed to be facing the vertex j. The geodesic distance between the

Algorithm 3 Parametrization of the 3D fixed mesh as a 2D mesh discretized into a hash table.

```
Be HASH a bidimensional array of pointers to faces
**for all** faces $f_i$ of the fixed mesh **do**
    find the vector $w_i$ that passes through the face
    calculate the spherical coordinates of $w_i$ like in equations 4.23 and 4.24
    apply the spherical coordinates on equation 4.22 to find the hash indices (row, col)
    store a pointer to $f_i$ at HASH[row,col]
**end for**
```



(a) Mobile mesh parameterized as a bundle of unit vectors passing through each vertex j.



(b) Computing w_j in the fixed frame after motion. Spherical coordinates of w_j are then used as input for hashing.

Figure 4.16: Detecting collisions exploiting the spherical sliding hash table for finding which elements are facing in the two meshes.

cell center and the involved elements can be retrieved for making decision regarding collision. The precise distance can be computed if necessary for the simulation (collision response). The algorithm 4 summarizes how we lookup the hash table for collision detection.

The hash table does not need to be updated at simulation time if we assume that the fixed mesh is rigid or deforms only in the directions not far from the surface normal.

Algorithm 4 Detecting collisions with spherical sliding by lookups on the hash table.

```
Be HASH a bidimensional array of pointers to faces initialized as in algorithm 3
for all vertices $v_j$ of the mobile mesh do
    find the vector $w_j$ that passes through the vertex
    calculate the spherical coordinates of $w_j$ like in equations 4.23 and 4.24
    apply the spherical coordinates on equation 4.22 to find the hash indices (row, col)
    retrieve the geodesic distance between the cell center and face at $HASH[row, col]$ $(\Rightarrow dist_f)$
    retrieve the geodesic distance between the cell center and $v_j$ $(\Rightarrow dist_v)$
    if $dist_v > dist_f$ then
        report collision
    endif
end for
```

The idea of this method arose when we were working on collisions within a virtual hip joint. The name *spherical sliding* comes from the ball-and-socket shape of that joint. However, it is not a solution only for that specific case. It also contemplates a whole family of similar problems. The only requirement is that a layered topology persists between the two meshes in relation to a central point fixed in the local-coordinate frame of one of them all along the motion. Figure 4.17 illustrates some examples of suitable situations.

4.3.2 Collision response

Collision response is the art of modifying colliding objects such that they cease colliding. In this work, we address that by deforming and/or displacing the objects whenever a collision is detected.

4.3.2.1 Penalty method

As seen in section 4.2.2.1 penalty methods basically rely on an instantaneous repulsion spring placed in every collision location to avoid penetration. In our implementation, we tested penalty methods with two different detection approaches: sphere penetration (section 4.3.1.1) and mesh intersection (section 4.3.1.2).

Once collision has been detected we need to produce a collision response, that is, to calculate the forces acting on the objects that collide. As usual with penalty methods, in this work a very small penetration between the two objects is allowed, and

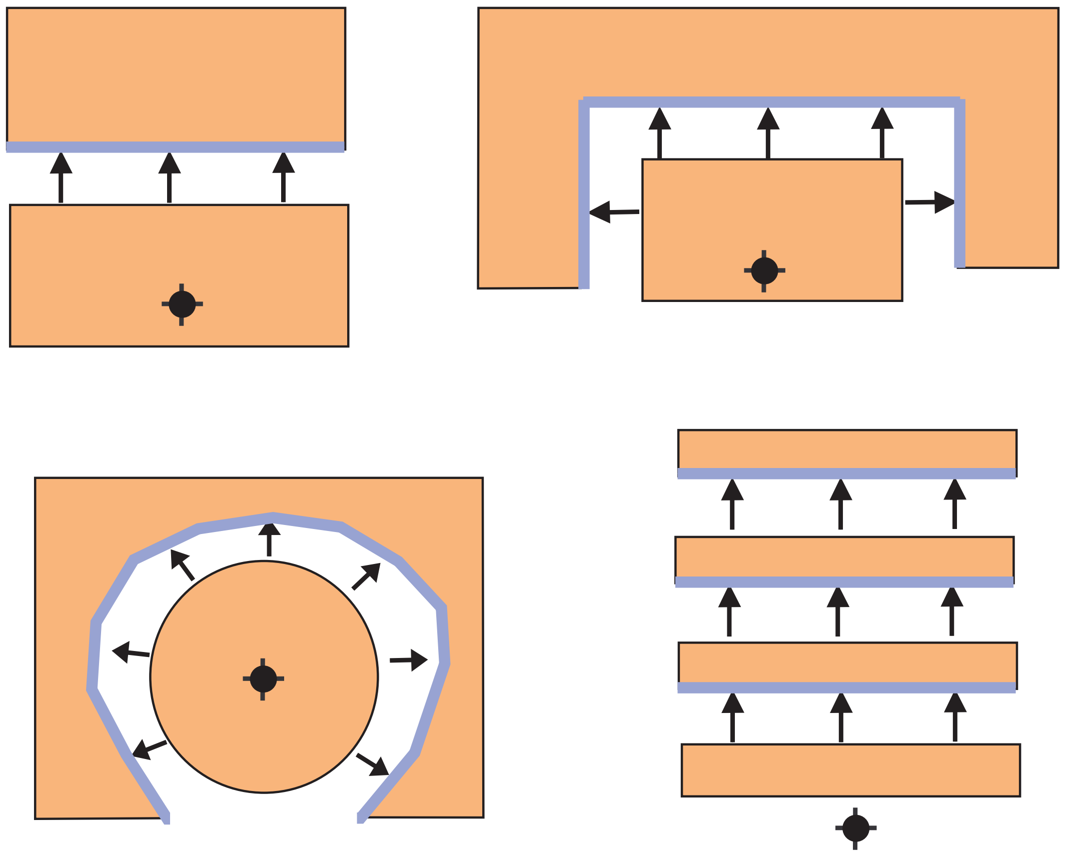


Figure 4.17: Cross-sections of typical situations suitable for collision detection with spherical sliding. For the fixed objects, only the faces represented in blue are taken into account.

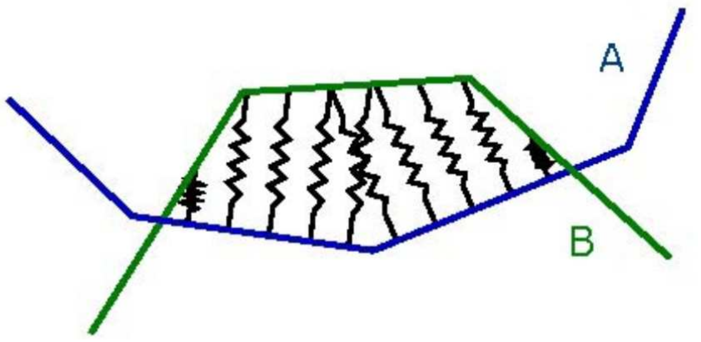


Figure 4.18: Springs inserted between the objects.

a separation force is caused by that penetration. This force tries to prevent further penetration and to separate the colliding bodies. The following rules are taken into account to compute this force:

- If object A does not penetrate any other object, the force acting on the object A is *zero*;
- If object A and B penetrate, then forces must be created to act on the objects A and B. These forces are applied at the point of contact on each object;
- The magnitude of the force is proportional to the penetration depth of A and B. This depth is the minimum (translational) distance required to separate two colliding objects;
- The direction of the forces is the normal vector to the surface in each object.

The penetration depth is incorporated into a penalty-based formulation to enforce the non-penetration constraint between two deformable objects. Essentially, penalty method models contacts by placing a spring at each contact point between the two contacting bodies (see Figure 4.18). Interpenetration is allowed between the objects at a contact point, and the amount of interpenetration is used to introduce restoring or *penalty* force that acts between the objects, pushing them apart.

We do not introduce the spring into the model as a body, but as a force that acts on the objects it attaches. A spring always pulls or pushes along its own length, thus yielding a collinear force pair. The two forces are based in the Newton's third law ("every action has an equal and opposite reaction") being consistent with many natural sources, such as elastic springs, gravitational attraction, etc. The approaches for estimating penetration depth and penalty force (or spring force) are as follows.

1. **Penetration depth estimation.**

   For the case when spheres are used for collision detection, the penetration depth is straightforwardly estimated subtracting the distance between the two sphere centers from the sum of their two radii. See figure 4.11 and equation 4.25.

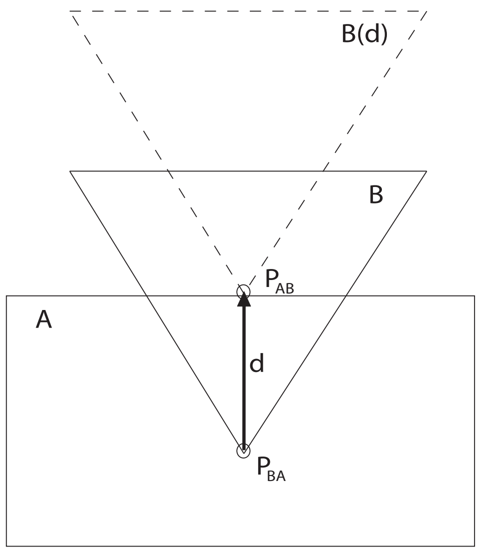


Figure 4.19: Collision geometry.

$$d = (r_{s_i} + r_{s_j}) - \|\overrightarrow{c_{s_i} - c_{s_j}}\| \tag{4.25}$$

When collisions are detected between meshes instead (refer to figure 4.19), the penetration depth d is the length of the shortest displacement that can cause the separation of the meshes A and B. Assume that A and B collide at time t and that the separation force in direction d is applied to the object B at the point P_{BA}; it pushes out the object B from A. The opposite force in direction $-d$ is applied to the object A at the point P_{AB}, and pushes A away from B.

The pair V-Collide/RAPID used to detect contacts does not provide distance between objects. Hence, to estimate penetration depth, all the contacts are analyzed such that we can compute an approximate penetration depth which is necessary for calculating the non-penetration force. We have a list of pairs of faces (f_i, f_j) from each object i and j involved in the collision. Imagine the face f_j represented in Figure 4.20 as the face in which a vertex P of the face f_i collided. For calculating penetration depth d, that here is the distance of the face f_i to face f_j, we project the point P onto the plane generated by the face f_j. Point P' is the projection of this point on the plane. Thus, we can calculate the distance between P' and P, and we obtain an approximate value for penetration depth.

The projection plane was defined from the following equations

$$\vec{w}_1 = \frac{\vec{CD}}{\|\vec{CD}\|} \cdot \|\vec{CP}\| \cdot (\vec{CP} \cdot \vec{CD}) \tag{4.26}$$

$$\vec{w}_2 = \frac{\vec{CE}}{\|\vec{CE}\|} \cdot \|\vec{CP}\| \cdot (\vec{CP} \cdot \vec{CE}) \tag{4.27}$$

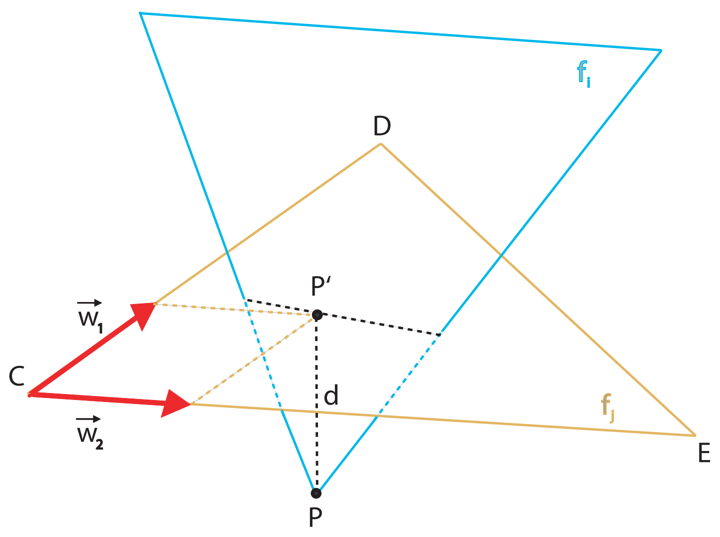


Figure 4.20: Approximate penetration depth estimation in a collision situation.

Then, using the equations 4.26 and 4.27, we find the projection of the point P'_{BA} on this plane.

$$P' = \vec{w}_1 + \vec{w}_2 + C, \tag{4.28}$$

and the distance between these points is calculated from

$$d = P - P' \ . \tag{4.29}$$

In the situation where two or three vertices of one face are penetrating the other object, we calculate the penetration distance for both vertices and then choose the greatest as the estimated penetration depth.

2. **Penalty force.**

   Once we have the penetration depth we calculate the non-penetration force using a penalty method (section 4.2.2.1). The penalty force model considered here assumes that the non-penetration force depends on the penetration depth d as follows:

$$\vec{F}_{np} = \begin{cases} -kd\vec{n} & \text{if } d > 0 \\ 0 & \text{if } d \leq 0 \end{cases} \tag{4.30}$$

   where k is a positive constant (called penalty coefficient) and $\vec{n}$ is the non-penetration force direction. Physically k corresponds to a stiff spring, temporarily placed between the objects during the collision. In this work the value k is calculated using the following equation

$$k = m \cdot K \tag{4.31}$$

where m is lightest object mass and K is a positive constant that was chosen from the analysis of the object behavior over the course of a simulation. Using the equations 4.30 and 4.31, we calculate $\vec{F}_{np}$ for each pair of colliding face. Thus, we write the non-penetration force for the face f_i as

$$\vec{F}_{np_i} = -kd\hat{n}_j \tag{4.32}$$

where $\hat{n}_j$ is the normal vector to the face f_j. Analogously but using the direction of the normal vector to face f_i instead, we calculate $\vec{F}_{np_j}$ for the face f_j.

Given those equations, we propose the algorithms 5 and 6 to calculate non-penetration forces for the mesh-to-mesh case and the sphere-to-sphere case respectively.

Algorithm 5 Calculating Non-Penetration Force for Mesh Collision

```
compute the spring constant $k$ from equation 4.31.
**for all** pairs of contacting face $(f_i, f_j)$ **do**
    compute the penetration depth according to item 1 of this section.
    compute the non-penetration force for the face $f_i$ using the Eq. 4.32.
    compute the non-penetration force for the face $f_j$.
**end for**
```

Algorithm 6 Calculating Non-Penetration Force for Spheres Collision

```
**for all** pairs of contacting spheres $(s_i, s_j)$ **do**
    compute the spring constant $k$ from the elasticity of the molecule material
    compute the penetration depth according to item 1 of this section.
    compute the non-penetration force for the sphere $s_i$ using the Eq. 4.32.
    compute the non-penetration force for the sphere $s_j$.
**end for**
```

Afterwards, all non-penetration forces are given as input to the deformation engine that will calculate itself the correct deformation and motion of the objects in virtual environment.

4.3.2.2 Position correction

With our soft tissues model, force and deformation are correlated in a way that we can always see the problem from two different points of view: a force that causes a deformation; a deformation that produces force. In this context, we can approach the collision response problem by not only calculating forces to separate colliding objects (as presented in section 4.3.2.1), but also by acting directly on the geometry of the colliding objects, deforming their surfaces such that they stop colliding. In this section, we describe how we correct the position of the points on the surfaces of two colliding objects to avoid penetration.

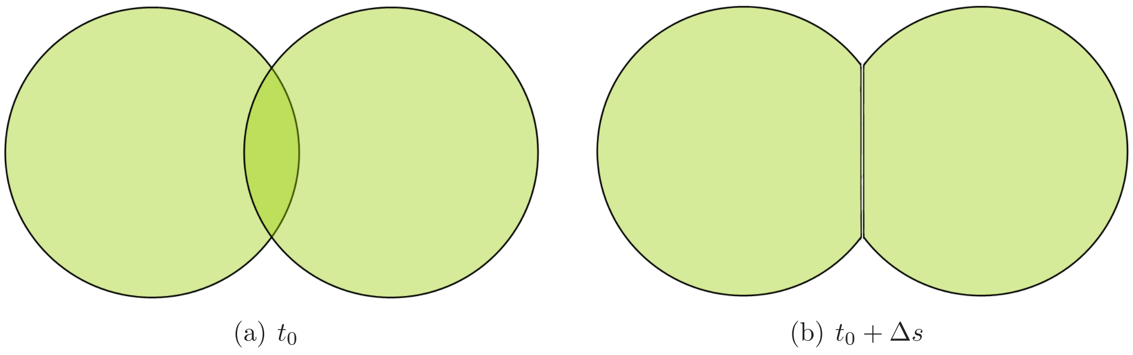


Figure 4.21: The effect of the geometrical response in the moment of the interpenetration detection (a) and just after the application of the method (b).

The method is applied after an interpenetration is detected between two meshes (sections 4.3.1.2, 4.3.1.3 and 4.3.1.4). Depending on the detection method used, pairs triangle-triangle or vertex-triangle are returned. When using the method of section 4.3.1.2, for every pair of colliding triangles, 6 pairs vertex-triangle are formed covering thus all combinations. Then, for every pair vertex-triangle, the distance is computed by parallel projecting the vertex onto the triangle. The vector going from the vertex to its projection is then used to calculate a displacement vector for the vertex. The displacement should be such that it puts the vertex in a position of non-collision but at the same time physically coherent, i.e., on the surface of its opposing triangle. In our implementation, this vector is oriented according to the normal of the respective triangle and its modulus is half of the penetration distance when both objects are deformable, and the total penetration distance in the case one is rigid. In the both-deformable case, as the vertices in both sides are displaced of half of the distance, the interpenetration is eliminated. In the one-rigid case, the rigid object is not modified because in our model the movement of rigid bodies is controlled kinematically; just the deformable has the position of its vertices modified. Figure 4.21 shows a 2D schema illustrating the effect of applying the method on two colliding spheres.

However, although the method assures that the objects surfaces do not penetrate, at the first sight it seems that it compromises their geometrical integrity and the physical correctness of the reaction. In section 3.4.5 we describe how we update the surface wrapping the mass-spring lattice of our deformation model. It is based on an anchoring system, where the vertices are anchored to a set of molecules and are constrained to follow them when they move or deform. In that way, the vertices of the boundary mesh are not really free such that we can apply a displacement vector here to prevent them to penetrate other objects. What we do, in fact, to change the vertices positions, is changing the positions of their anchor molecules. Displacement vectors are then required to be calculated for each molecule serving as an anchor. This is done by combining of the displacement vectors of the vertices anchored to each anchor-molecule. Be three sets:

$$A = \{a_0, a_1, \cdots, a_n\}; V = \{v_0, v_1, \cdots, v_m\}; D = \{d_0, d_1, \cdots, d_m\} \tag{4.33}$$

where A is the set of anchor-molecules positions, V is the set of vertices, and D is the set of displacement vectors for V. Then:

$$a'_i = a_i + \sum_{j=0}^{m} D_{v_j} \omega_{v_j} \tag{4.34}$$

where a'_i is the position of a_i after the displacement and ω_{v_j} is a normalized weight given by the inverse of the square of the distance between the vertex v_j and the molecule a_i in the form:

$$\omega_{v_j} = \frac{1}{dist^2_{v_j} \sum \frac{1}{dist^2_{v_j}}} \tag{4.35}$$

After the update of the anchors, the vertices positions are automatically updated by the skinning model, and the mechanical spring-like connections of the anchors will eventually be stressed. The stress on the springs produces a chain reaction pushing the objects against each other on the contacting surfaces, and pushing or pulling the other molecules, those that are not anchors, in a way that the energy of collision produced initially by the surface geometrical change influences the object as a whole.

The practice, hence, showed that the reaction carried out by the molecules stressed by the displacement they also bear, propagates the initial vertices displacement as a force within the object that is coherent with the energy liberated by the collision, which gives a physically correct behavior.

4.3.2.3 Geometrical response with velocity correction

The position correction method presented in section 4.3.2.2 proposes a geometrical approach to eliminate penetration. That method has proven to be effective, however, in practice it needs very small timesteps for the simulation to converge. This happens because a reaction to a collision only occurs after some penetration is allowed, an error which is not negligible when timesteps are large. In this section, we propose an extension that looks one step forward in time, and tunes the velocity of a vertex before it collides. Every time the current velocity potentially brings the vertex into a collision situation in the next step, it is adapted to bring the vertex close to but not inside the other object, damping the impact. This makes the system less stiff and more adequate to converge, as the repulsion forces are thus applied more smoothly.

Volino presented a similar idea in his thesis [Volino, 1998]. However, his colliding objects being typically garments, he had different constraints, and had to correct accelerations as well as apply some adjustments after numerical integration to correctly decrease the local kinetic energy and ensure dissipative collision response.

Our method checks if the distance between two features A and B is negative (collision). If it is, the features positions are corrected to the midpoint between them, and their velocities are inverted. If not, the method calculates the relative position of A and B in the time $t + \Delta s$. If they collide at that time, a velocity correction vector is calculated to modify the velocities of the two features at the time t such that they arrive at a coincident but non-penetrating position in time $t + \Delta s$. The algorithm 7 gives more details; refer to figure 4.22 for an illustrative example.

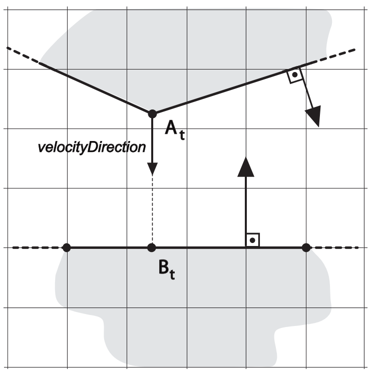
Algorithm 7 Avoid penetration geometrically combining position and velocity corrections.

```
if dist(A, B) > 0 then
  // A and B do not penetrate in time $t$, calculate positions in $t + \Delta s$
  velocityDirection= $\frac{A-B}{|A-B|}$
  relativeVelocity = velocityDirection$*((V_A - V_B)\cdot$ velocityDirection)
  $A_{t+1}$=$A$ + (relativeVelocity$*\Delta s$)
  if dist($A_{t+1}$, B) > 0 then
    // A and B do penetrate in time $t + \Delta s$, calculate velocity correction
    velocityCorrection = relativeVelocity$-((B - A) * \frac{1}{\Delta s})$
    $V_A$ -= velocityCorrection / 2.0
    $V_B$ += velocityCorrection / 2.0
  endif
else
  // A and B do penetrate in time $t$, calculate position correction and invert velocities
  positionCorrection = B - A
  A += positionCorrection / 2.0
  B -= positionCorrection / 2.0
  $V_A = -V_A$
  $V_B = -V_B$
endif
```

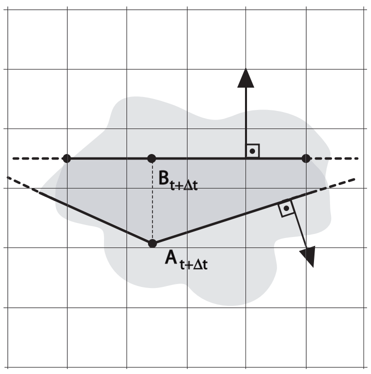
4.3.3 Methods comparison and evaluation

We implemented the techniques described in section 4.3.1 for collision detection and section 4.3.2 for collision response. We tested them combining different pairs of detection-response methods. Some combinations need very small timesteps to converge, others converge better but allow too much penetration, some are very slow even with relatively large timesteps, others simply do not work well with large triangles or sharp angles. We selected some combinations that gave good visual results in terms of penetration avoidance and physical behavior to explore the range of input parameters. We tested them all on a typical situation of permanent contact on large areas, and the results are presented in this section. Note that the computing times displayed are not optimal. They are only comparatively significative.

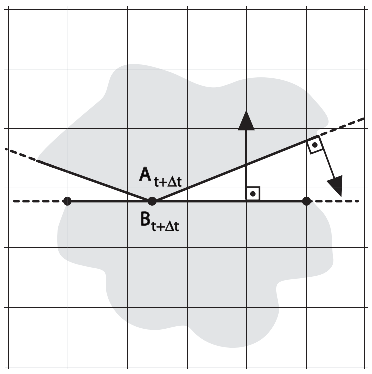
The situation we have chosen is that of the contact between the two cartilage caps of the human hip joint. A 3D hip has been modelled based on the joint model and the deformation model presented respectively in the chapters 2 and 3. On that model, we applied rotational motion around two axes simultaneously, describing a flexion and an internal rotation of the thigh. Then, the femoral head and the acetabular cartilages



(a) Situation at t.

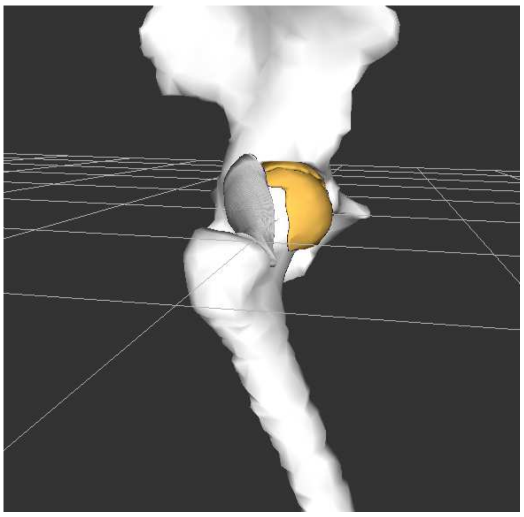


(b) Situation estimated at $t+\Delta s$ without collision response.

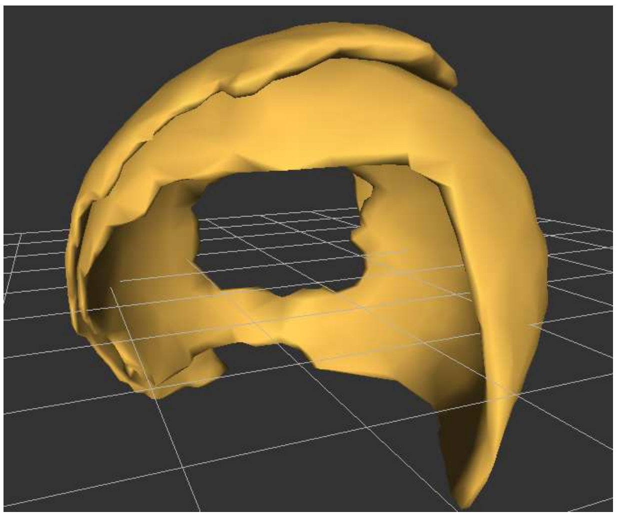


(c) Situation at $t + \Delta s$ after velocity correction.

Figure 4.22: Example of velocity correction seen in 2D for clearness.



(a) The 3D human hip joint.



(b) The hip parts evaluated here.

Figure 4.23: Test setup for evaluation of collision detection and response methods.

slide on each other all along the motion. Figure 4.23 shows the test setup highlighting the parts involved in the collision detection and response procedures.

Next subsections present the experiments done manipulating some input parameters. They are all based on a simulation of 4 seconds in which the hip flexes up to 90° and twists (internal rotation) up to 57°. Each subsection focuses on one particular problem. In every case, we measured the obtained computing time for the three main time consuming processes: kinematical motion; physical deformation; collision detection and avoidance.

4.3.3.1 What is the optimal hash table size?

We have run 6 simulations with different configurations of spherical sliding collision detection and geometrical response with velocity correction. The tests were all performed

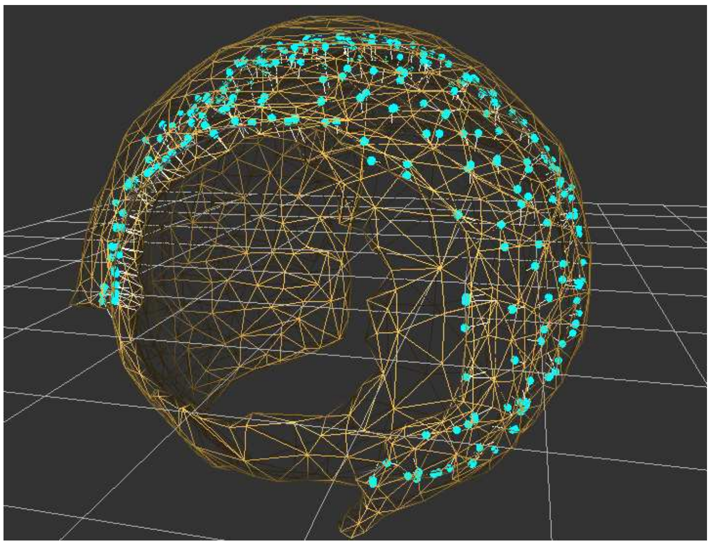


Figure 4.24: Wireframe view showing all pairs face-vertex for the hip cartilage contact handling. The short white lines go from the vertices of the mobile mesh to their closest point (green spheres) on the fixed mesh. Each pair triangle-vertex is tested for distance to determine if a collision exists.

integrating the simulation with a basic Euler formulation at a rate of 500 iterations per second. We varied the size of the hash table from 10x10 to 400x400. For each table resolution, in addition to the computing times mentioned in the introduction, we also measured the initialization time and the number of calls to triangle-vertex distance calculation. Figure 4.24 shows the pairs triangle-vertex in a typical situation for the hip. Figures 4.25 and 4.26 bring the data we obtained on comparative graphs. We do not analyze the deformation and kinematics computation times here because the hash table size does not influence them.

Analyzing the graph of figure 4.25, we see that the computation time for collision treatment goes down as the hash size is increased, tending to remain constant at around 45 seconds for very large tables. The initialization time, in turn, increases fast due to the $O(n^2)$ complexity of the algorithm. Then, we compared the computation time for collision treatment with the number of distance tests (or non-null hits on the table lookup) in figure 4.26. They are proportional, which allowed us to conclude that after a certain hash resolution the relation triangle hash-cell becomes one-to-one, avoiding false hits, what is positive. However, increasing even more the resolution does not improve performance during simulation, is more memory consuming, and makes the initialization process extremely long.

4.3.3.2 Numerical integration method vs. timestep size

For this test we computed 16 simulations with spherical sliding collision detection and geometrical response with velocity correction. This time, we used the same hash size (40x40) for all of them. We have basically chosen 8 different timesteps, varying from 100 to 600 iterations per second. For each timestep value we computed the same

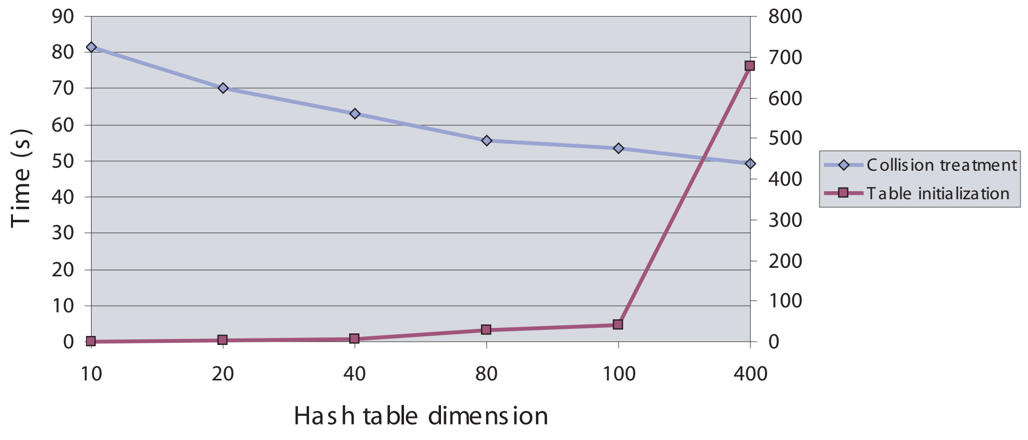


Figure 4.25: Calculation time for initialization and lookup for different hash table sizes.

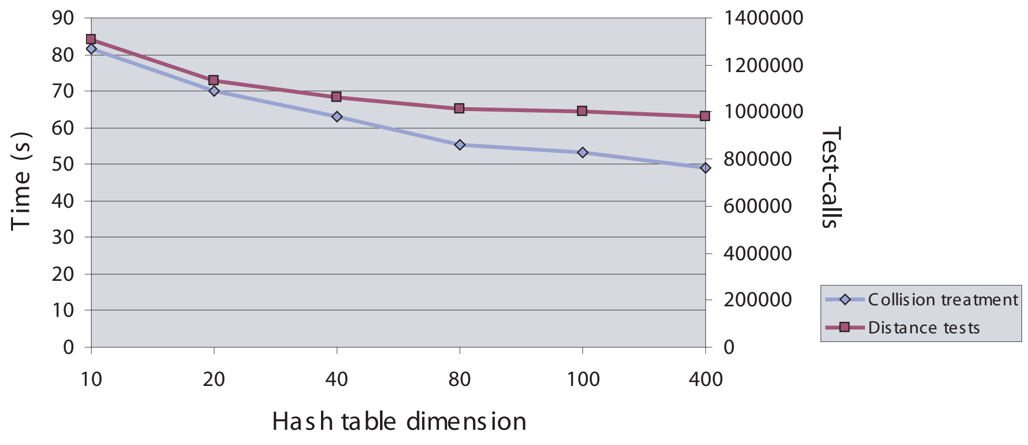


Figure 4.26: Calculation time for lookup and number of distance tests for different hash table sizes.

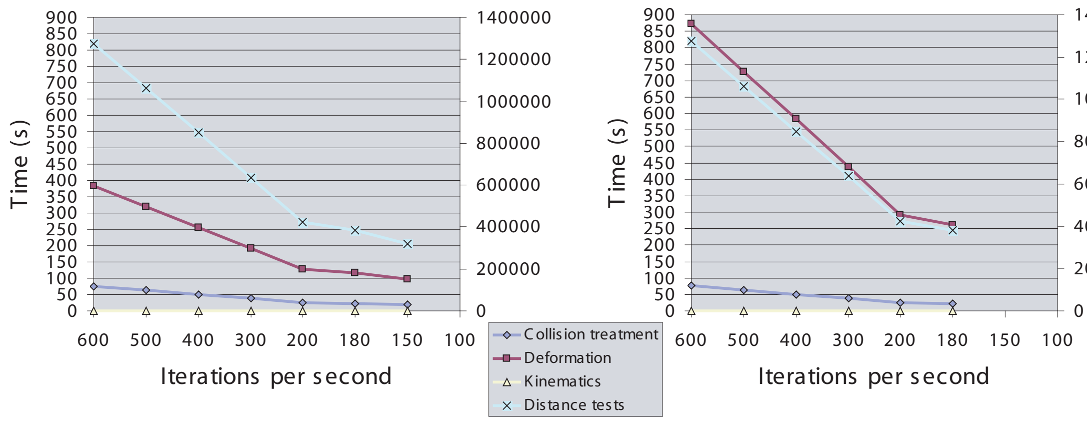


Figure 4.27: Calculation time and number of distance tests for different timesteps. At the left using Euler formulation, and at the right using Runge-Kutta 4.

simulation twice, one using Euler formulation and the other using Runge-Kutta of order 4 for numerical integration. The goal is to expose the two different numerical integration methods to an instability generator situation that is the permanent contact management with collision detection and response methods. It should allow to find how large a timestep can be in such situations. Remember that larger timesteps mean faster simulation. The output data is displayed on figures 4.27 and 4.28.

Analyzing the graphs we see that the kinematical part is not time consuming, and that the deformation is the process that takes more time. The time to compute deformation and collisions, and the number of distance tests go linearly down as the timestep is increased, what was already expected. That Runge-Kutta 4 needs more time to compute the same number of steps was equally expected. What can be said to be surprising is that we can use a larger timestep, 6.6 ms, with Euler than with Runge-Kutta 4. This latter have shown to be instable already at 5.5 ms, being acceptable only at 5 ms, while Euler diverged only close to 10 ms. Note that the diverging values have not been plotted in the graphs. We discuss the numerical integration effects with more details in section 3.4.3.

4.3.3.3 Spherical sliding vs. proximity structure

Here we compare the same simulation changing only the collision detection method. Euler is used for both cases and the collision response is also the same.

In figure 4.29 we see that the deformation time is always the same with both methods, and that the simulation converges with a timestep up to 6.6 ms and diverge at 10 ms. We also see that the collision detection with spherical sliding is almost one order of magnitude faster than with proximities. In figure 4.30 we compare the number of distance tests, and we observe that with proximities this number is significatively inferior than with spherical sliding. The explanation is that with proximities the closest triangle must be updated to each vertex of one of the meshes at every iteration. This

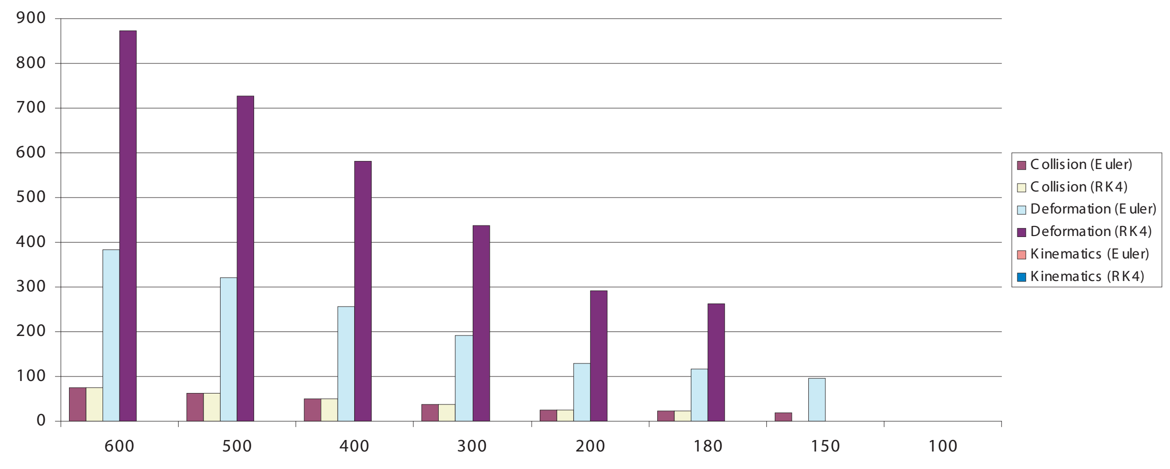


Figure 4.28: The contribution of each process (collision, deformation, kinematics) on time consumption along the different timesteps with the two integration methods.

update takes time, making the spherical sliding relatively more efficient.

4.4 Conclusion

After a review of the state of the art in collision detection and response techniques, we implemented and tested some of them in the specific case of continuous contact along large areas between deformable objects. We also introduced two novel approaches for collision detection: proximities structure and spherical sliding.

We cross evaluated them in the context of a virtual hip joint simulation. We observed that the spherical sliding method has the best performances, while the proximities structure has a wider range of suitable cases.

We also observed that the collision response method and the deformation model can be very sensitive to numerical integration errors, especially with stiffer viscoelastic objects. We demonstrated that for physical realism the collision response not necessarily need to be based on forces. With our deformation model, geometric changes on the surface become physically realistic due to the internal reactions of the soft tissue.

We did not include friction for energy dissipation in the contact model. We evaluated a friction coefficient with colliding spheres and concluded that it is negligible with the dynamics our target applications. This should be considered in future works if they aim at simulating motion with very different dynamic behavior.

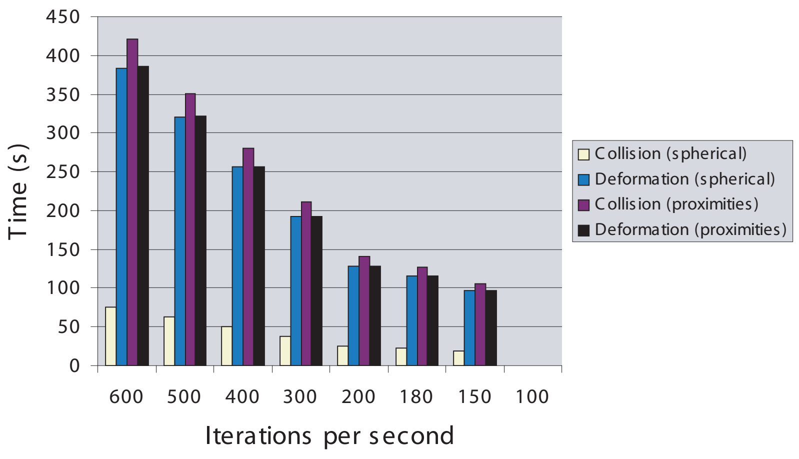


Figure 4.29: Comparing computing time for collision treatment and deformation between spherical sliding and proximities methods.

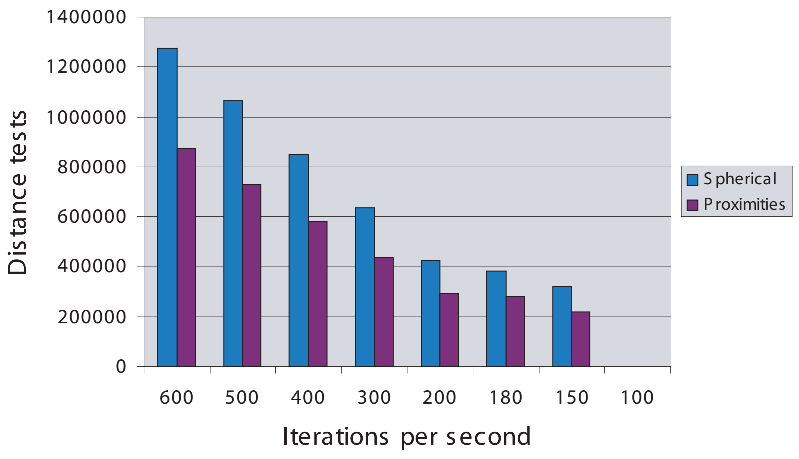


Figure 4.30: Comparing number of distance tests for collision response between spherical sliding and proximities methods.

CHAPTER 5

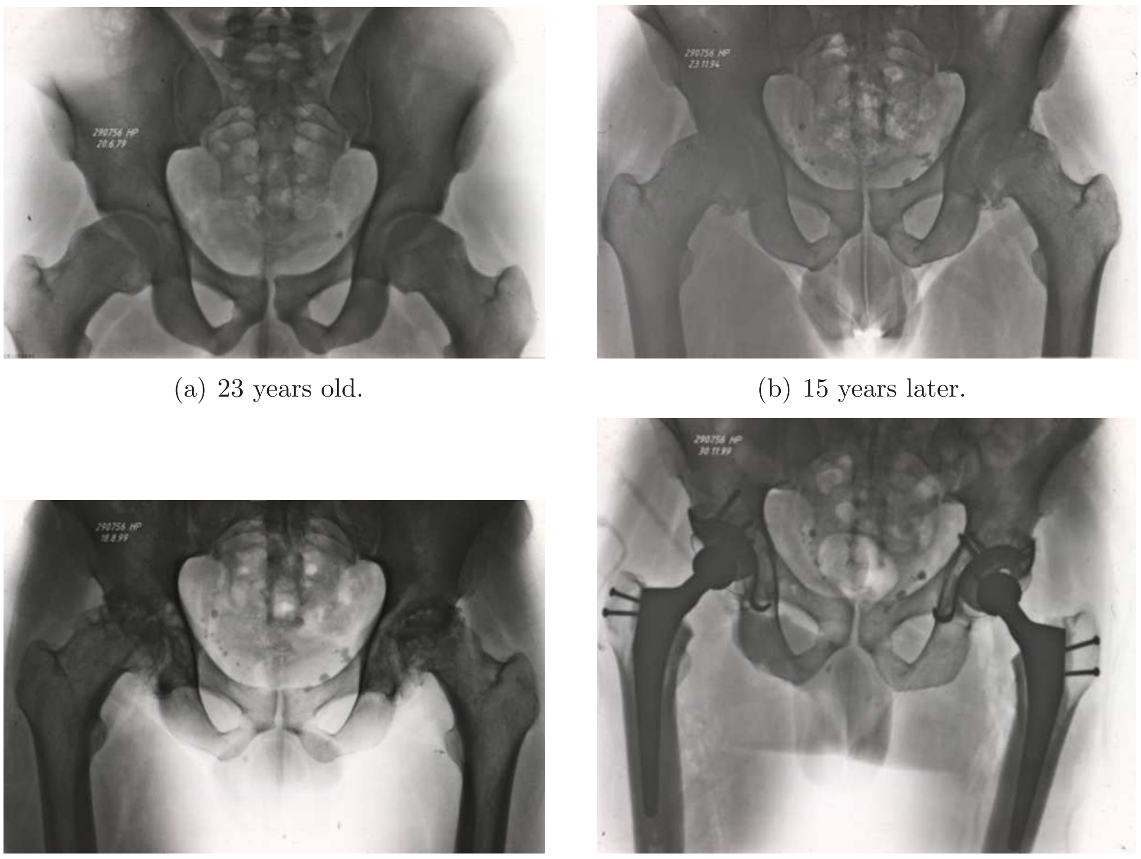
The hybrid hip case study

5.1 Introduction

The human hip joint (articulatio coxae/iliofemoralis) is a classic ball and socket joint that allows movement in all three planes: coronal (flexion-extension), sagittal (abduction-adduction), transverse (internal-external rotation). The external appearance in the hip joint area and its motion are due to the presence of several inner structures amongst we can highlight bones, cartilages, ligaments, muscles, tendons, fat and skin. All these structures are deformable, but bones are sharply stiffer and are often considered rigid. Bones, cartilages and ligaments are much involved in the contact existent within the joint. Thus, they are directly responsible for most of the hip range of motion constraints. See figure 5.2 for a coronal cross- section view of the hip.

Pain on the hip area is one of the main causes of surgical treatment in Orthopedics. One of the pathologies often requiring hip surgery is the hip arthritis (see figure 5.1). This inflammation affects almost 100% of the elder population, and according to physical activity may touch young patients. The usual treatment to severe hip arthritis is the replacement of the hip by a prosthesis. Hence, arthritis in young patients is a serious problem without convenient treatment to date, because the prothesis lifetime is short and it must be replaced after a few years. Preventive surgery and/or posture correction can avoid a young patient to develop hip arthritis. We believe that a 3D functional model of the joint is useful as a tool to analyze, diagnose, plan and assess preventive treatment for people exposed to risk of hip pain, which are potential patients of hip arthritis.

The remaining of this chapter describes the human hip with its structures and functions, how we combine the models of chapters 2, 3 and 4 to build a virtual model of the hip for our case study, and some prototype applications we setup to assess medical usability of our models.



(a) 23 years old.

(b) 15 years later.

(c) 20 years later.

(d) After hip surgery.

Figure 5.1: Hip arthritis, one of the main causes of hip pain (courtesy HUG).

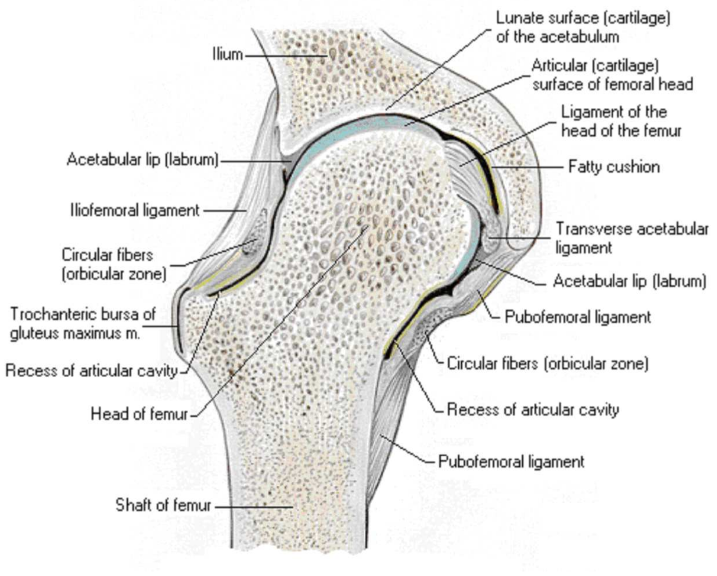

Ilium
Lunate surface (cartilage) of the acetabulum
Articular (cartilage) surface of femoral head
Ligament of the head of the femur
Acetabular lip (labrum)
Fatty cushion
Iliofemoral ligament
Circular fibers (orbicular zone)
Transverse acetabular ligament
Acetabular lip (labrum)
Trochanteric bursa of gluteus maximus m.
Pubofemoral ligament
Recess of articular cavity
Circular fibers (orbicular zone)
Head of femur
Recess of articular cavity
Pubofemoral ligament
Shaft of femur

Figure 5.2: Coronal cross-section of the hip joint. Bones, ligaments and cartilages are shown (adapted from diverse online sources).

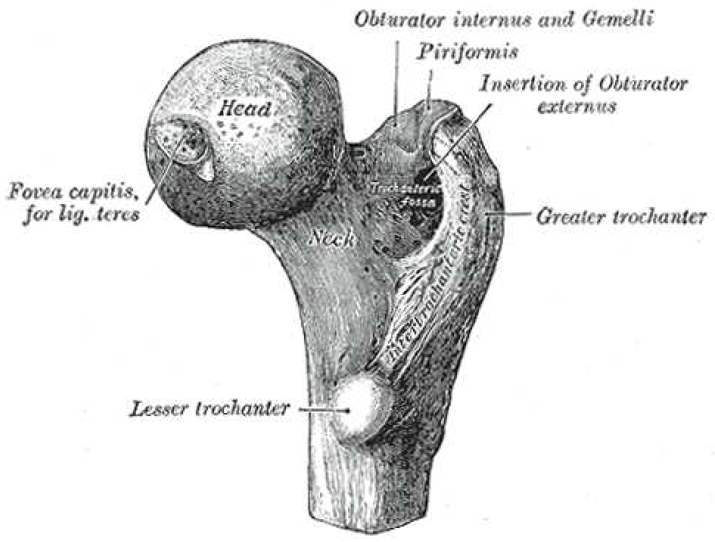


Figure 5.3: Upper extremity of the human right femur with anatomical landmarks indicated [Gray, 2000].

5.2 The human hip and its structures

5.2.1 The bones of the hip

The **femur** is the longest and strongest bone in the human skeleton; it is cylindrical in the greater part of its extent. In the erect posture it is not vertical, being non-parallel with its fellow. It inclines gradually medialward, so as to approach its fellow near the knee-joint binging it closer to the line of gravity of the body. The femur, like other long bones, is divisible into a body and two extremities.

The upper extremity of the femur (figure 5.3, which is the most important in this work, presents a head, a neck, and a greater and a lesser trochanter. The head is globular and forms rather more than a hemisphere, is directed upward, medialward, and a little forward. Its surface is smooth, coated with cartilage in the fresh state, except over an ovoid depression, the *fovea capitis femoris*, which is situated a little below and behind the center of the head, and gives attachment to the ligament teres.

The **pelvis**, also called hip-bone is a large, flattened, irregularly shaped bone, constricted in the center and expanded above and below. It consists of three parts, the ilium, ischium, and pubis; the union of the three parts takes place around a large cup-shaped articular cavity, the acetabulum, which is situated near the middle of the outer surface of the bone (see figure 5.4. The ilium is the superior broad and expanded portion that extends upward from the acetabulum. The ischium is the lowest and strongest portion of the bone; it forms, with the pubis, a large aperture, the obturator foramen. The most important part of the pelvis in the context of this work is the **acetabulum** (cotyloid cavity). It is a deep, cup-shaped, hemispherical depression, directed downward, lateralward, and forward. It is bounded by a prominent uneven rim, which is thick and strong above, and serves for the attachment of the glenoidal labrum (cotyloid ligament), which contracts its orifice, and deepens the surface for articulation. It presents below a deep notch, the acetabular notch, which serves for the attachment of the ligamentum teres. The rest of the acetabulum is formed by a curved cartilaginous surface, the lunate surface, for articulation with the head of the femur.

5.2.2 The cartilages of the hip

Two cartilaginous surfaces are present in the hip joint. One covers almost completely the head of the femur - except the fovea capitis where the teres ligament attaches - and is often called **femoral head cartilage**. It is thick centrally and thin peripherally. The other, conversely, is thicker peripherally, has a horse-shoe shape and covers the acetabulum, being called **acetabular cartilage** or **lunate surface**. They are opposing surfaces, are regularly and reciprocally curved, and at any given time only two fifths of the femoral head occupies the acetabulum. The labrum mentioned in section 5.2.1 serves to extend the bony acetabulum into a true hemisphere as well as deepening it thus increasing joint stability. See figure 5.5.

5.2.3 The ligaments of the hip

The articular capsule and ligaments work as static stabilizers of the hip joint, especially the iliofemoral ligament, which tightens in extension enabling the hip to assume a stable close packed position; in the other hand, muscles are considered as dynamic stabilizers. Contrarily of what happens in other joints, the role of ligaments on determining the hip joint motion and range of motion is less significant.

The articular capsule and the other 6 ligaments of hip joint are described bellow. It is known, however, that these 7 structures are not separated. Actually, they are blended together to compose the complex of the articular capsule. The differences in names and classification are due to functional variations in thickness and orientation of fibers, as we can observe in the figure 5.6.

The **articular capsule** is strong and dense. Above, it is attached to the margin of the acetabulum 5 to 6 mm beyond the glenoidal labrum behind; but in front, it is attached to the outer margin of the labrum, and, opposite to the notch where the margin of the cavity is deficient, it is connected to the transverse ligament, and by a few fibers to the edge of the obturator foramen (muscle). It surrounds the neck of the femur, with most fibers running in the same direction of the neck axis, and is attached, in front, to the intertrochanteric line; above, to the base of the neck; behind, to the neck, about 1.25 cm above the intertrochanteric crest; below, to the lower part of the neck, close to the lesser trochanter. From its femoral attachment some of the fibers are reflected upward along the neck as longitudinal bands, termed retinacula. The capsule is much thicker at the upper and forepart of the joint, where the greatest amount of resistance is required; behind and below, it is thin and loose. It consists of two sets of fibers, circular and longitudinal. The circular fibers, zona orbicularis, are most abundant at the lower and back part of the capsule, and form a sling or collar around

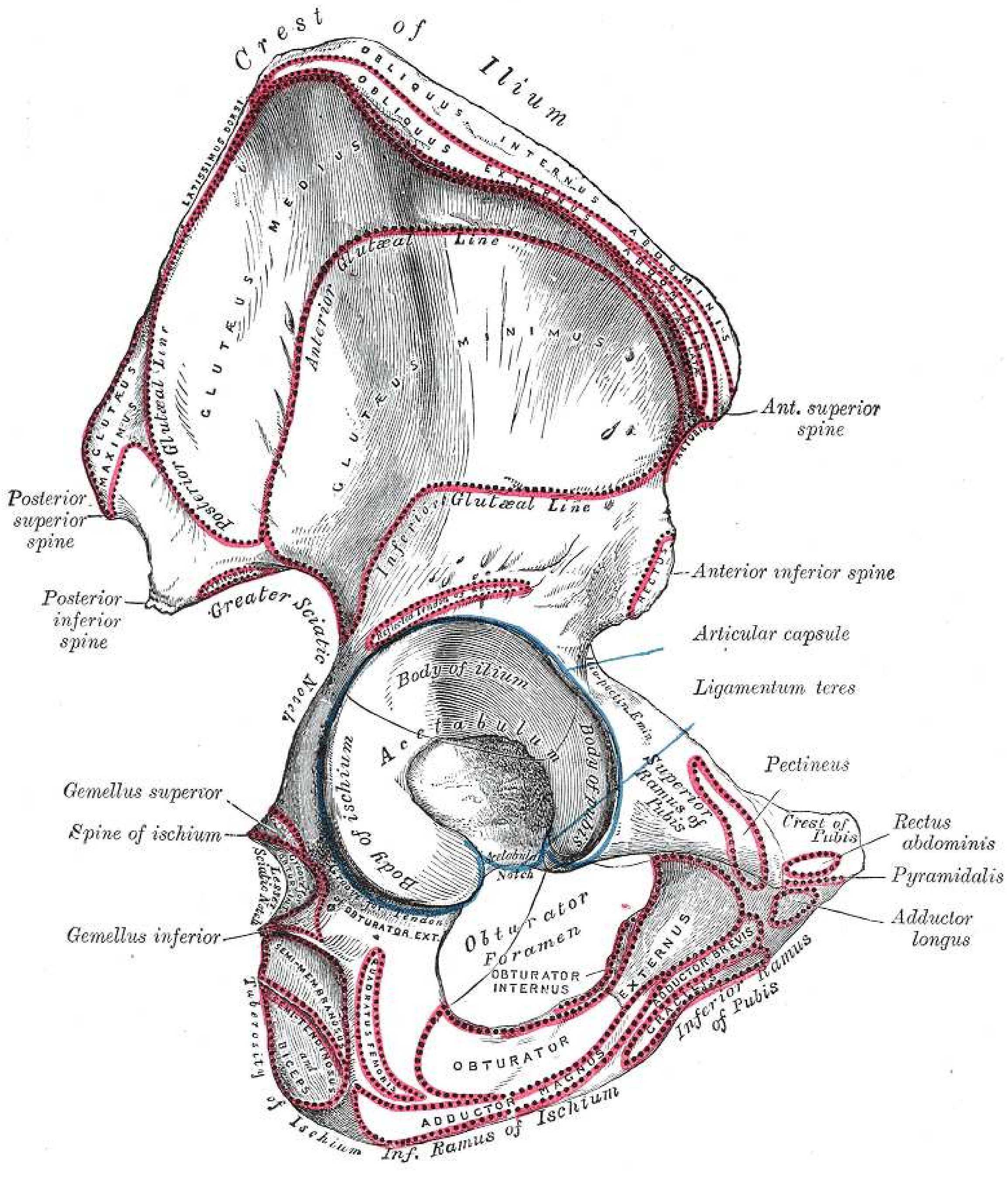
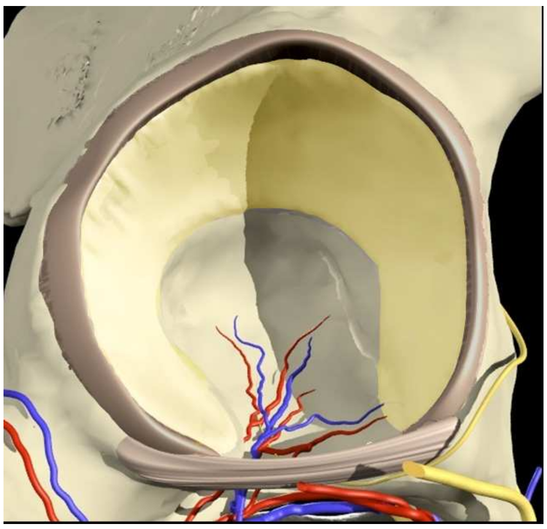


Figure 5.4: Acetabular view of the pelvis indicating ligaments and muscles attachments as well as the anatomical landmarks [Gray, 2000].


(a) Lateral view of the horseshoe shaped cartilage of the hip acetabulum (yellowish part) with surrounding labrum (darker grey rim).

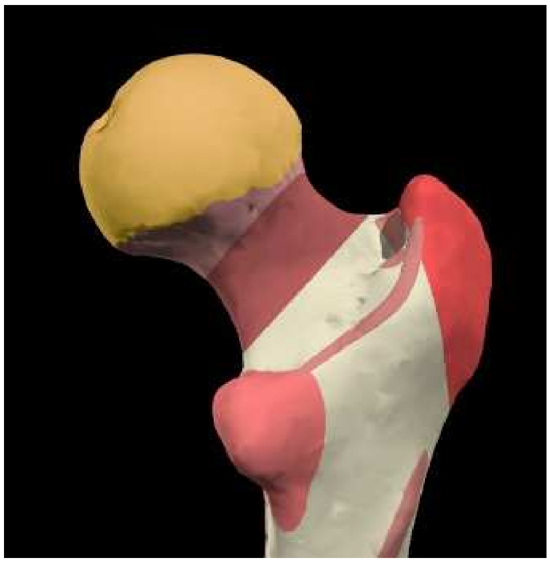
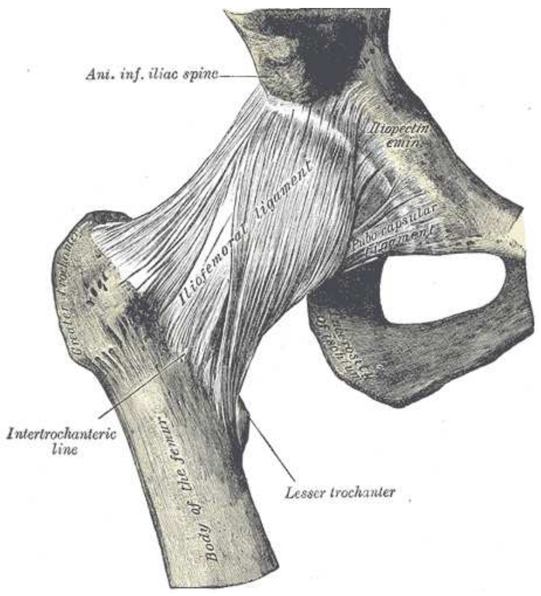

(b) Posterior view of the superior extremity of the human right femur. Yellowish part indicates the cartilage cap and reddish parts show ligaments and muscles attachment areas.

Figure 5.5: The cartilages of the hip joint (adapted from [Haddad et al., 2001]).

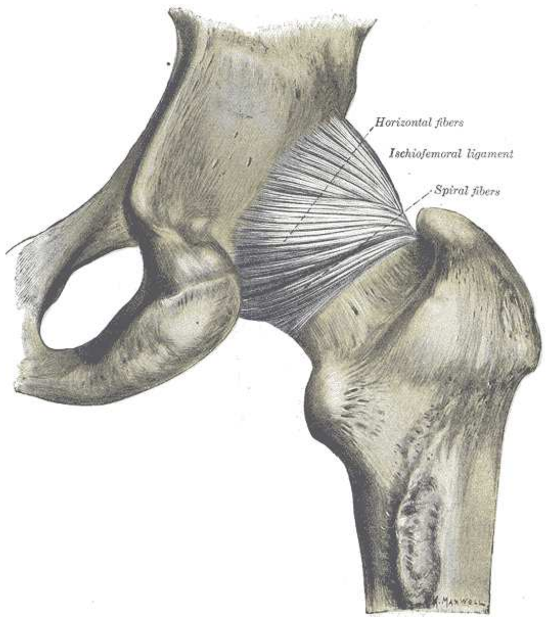
the neck of the femur. Anteriorly, they blend with the deep surface of the iliofemoral ligament, and gain an attachment to the anterior inferior iliac spine. The longitudinal fibers are greatest in amount at the upper and front part of the capsule, where they are reinforced by distinct bands or accessory ligaments, of which the most important is the iliofemoral ligament.

The **pubocapsular** ligament, also called **pubofemoral** is attached above to the obturator crest and the superior ramus of the pubis; below, it blends with the capsule and with the deep surface of the vertical band of the iliofemoral ligament. It reinforces the capsule, checks medial rotation and tightens on abduction.

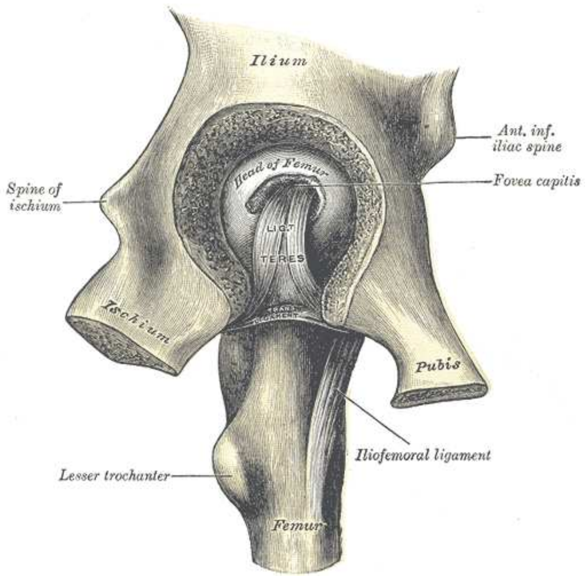
The **iliofemoral** ligament is a band of great strength that lies in front of the joint; it is intimately connected with the capsule, and serves to strengthen it. With its twisted fibers, it is one of the strongest ligaments in the body. It is attached, above, to the lower part of the anterior inferior iliac spine; below, it divides into two bands, one of which passes downward and is fixed to the lower part of the intertrochanteric line; the other is directed downward and laterally and is attached to the upper part of the same line. Between the two bands there is a thinner part of the capsule. In some cases there is no division, and the ligament spreads out into a flat triangular band, which is attached to the whole length of the intertrochanteric line. This ligament is frequently called the Y-shaped ligament. Its functions are: checking medial rotation and extension of the femur; prevention of the closed-packed position of the hip checking adduction somewhat; and what is most important, allows one to stand with minimum muscular activity, because the ligament is tightened when the pelvis rolls backwards during standing.



(a) Right-hip frontal view.



(b) Right-hip posterior view.



(c) Left-hip medial view.

Figure 5.6: The hip joint capsule with ligaments [Gray, 2000].

The ligament **teres femoris** is the ligament of the head of the femur. It is a triangular, somewhat flattened band implanted into the antero-superior part of the fovea capitis femoris (see section 5.2.1 and 5.2.2 for reference). Its base is attached by two bands, one into either side of the acetabular notch, and between these bony attachments it blends with the transverse ligament. This ligament is made tense when the thigh is semiflexed and the limb then adducted or rotated outward; it is, on the other hand, relaxed when the limb is abducted. It has, however, little influence as a ligament, its main function being serving as support to transmit blood vessels.

The **ischiocapsular** (also **ischiofemoral**) consists of a triangular band of strong fibers, which spring from the ischium below and behind the acetabulum, and blend with the circular fibers of the capsule. It is distally attached to the posterosuperior aspect of the neck of the femur(where the neck meet the greater trochanter). Its function is to check internal rotation and extension of the femur.

The **ischiofemoral** ligament is attached proximally to the ischium posterior and posteroinferior to the acetabular rim in the superior and lateral directions.

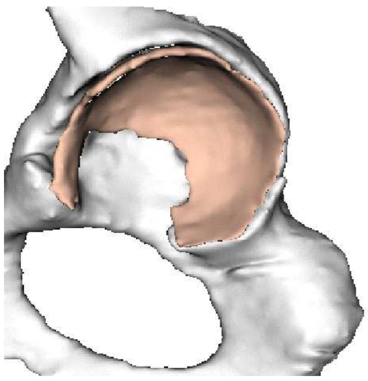
The **glenoidal labrum**, also called **acetabular lip** or **acetabular rim**, is the fibrocartilaginous edge which forms a ring around the circular outer border of the acetabulum and deepens that cavity. At the same time it protects the edge of the bone, and fills up the inequalities of its surface. It bridges over the notch as the transverse ligament, and thus forms a complete circle, which closely surrounds the head of the femur and assists in holding it in its place. It is triangular on section, its base being attached to the margin of the acetabulum, while its opposite edge is free and sharp. When the angular limits of the joint are reached, the femoral neck can hit the labrum. In people whom activities involve repetitive or constant extreme postures of the hip, these hits may cause the labrum to become rigid like bone, and after some time, this rigid structure damage the femoral cartilage, causing a pathology known as **arthritis**. [Ferguson, 2000] is a complete reference on the biomechanics of the labrum.

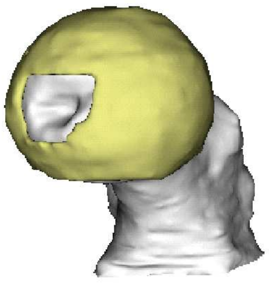
The **transverse acetabular** is in reality a portion of the glenoidal labrum, though differing from it in having no cartilage cells among its fibers. It consists of strong, flattened fibers, which cross the acetabular notch, and convert it into a foramen, forming a bridge over the artery in the ligament of the femoral head through which the nutrient vessels enter in the joint.

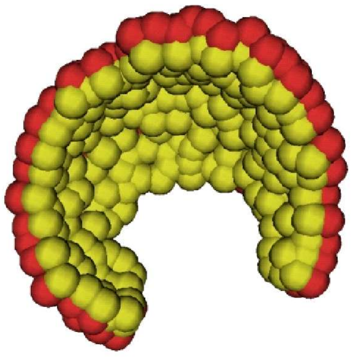
5.3 Building the virtual hip

Based on the anatomical and biomechanical information of section 5.2, as well as on the models of chapters 2 and 3, we have built a generic functional 3D model of the hip joint. This model combines, then, Kinematics and Mechanics to allow for the simulation of the hip such that a set of useful physical parameters can be extracted, which support the development of medical applications.

3D boundary models of organs (meshes) are reconstructed from segmented MRI data [Lydia Yahia-Cherif, 2003]. The 3D meshes representing bones are considered rigid bodies and are used as is. Cartilage meshes have their volume discretized into a


(a) 3D acetabular cartilage


(b) 3D femoral cartilage


(c) Discretized acetabular cartilage

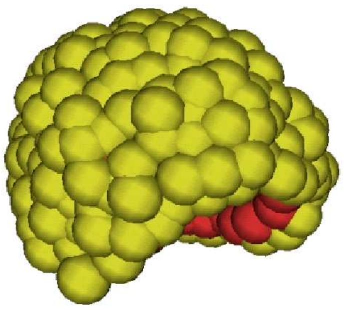

(d) Discretized femoral cartilage

Figure 5.7: The cartilage caps of the femoral head and the acetabular cup discretized into a set of spherical regions.

number of spherical regions representing the molecules of the deformation model (figure 5.7).

The vertices on the surfaces of the cartilage caps are then linked to the neighborhood of underlying molecules with weights according to distances, and follow the model deformations (see skinning in section 3.4.5. Bony motion is generated using the kinematical model (figure 5.8).

Later on we also added ligament models to complete the hip (see figure 5.9). We modelled all hip ligaments, including the acetabular labrum, which is not actually a ligament. However, we did not include them all at the same time in our hip model. Instead, we chose the ligaments interesting for the situations we wanted to test.

5.4 Prototype medical applications

We developed a set of applications exploiting the virtual hip joint of section 5.3. Such applications allow for the biomechanical analysis, the clinical diagnosis and the surgery planning around that joint. Their main goal in the context of this work is showing

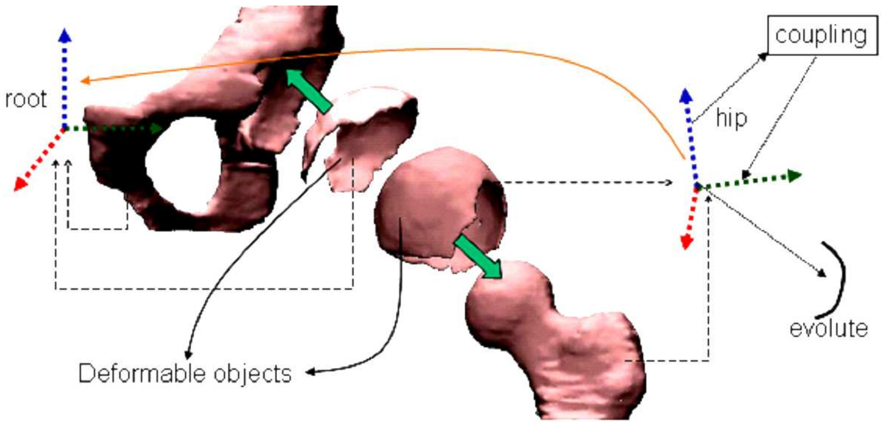
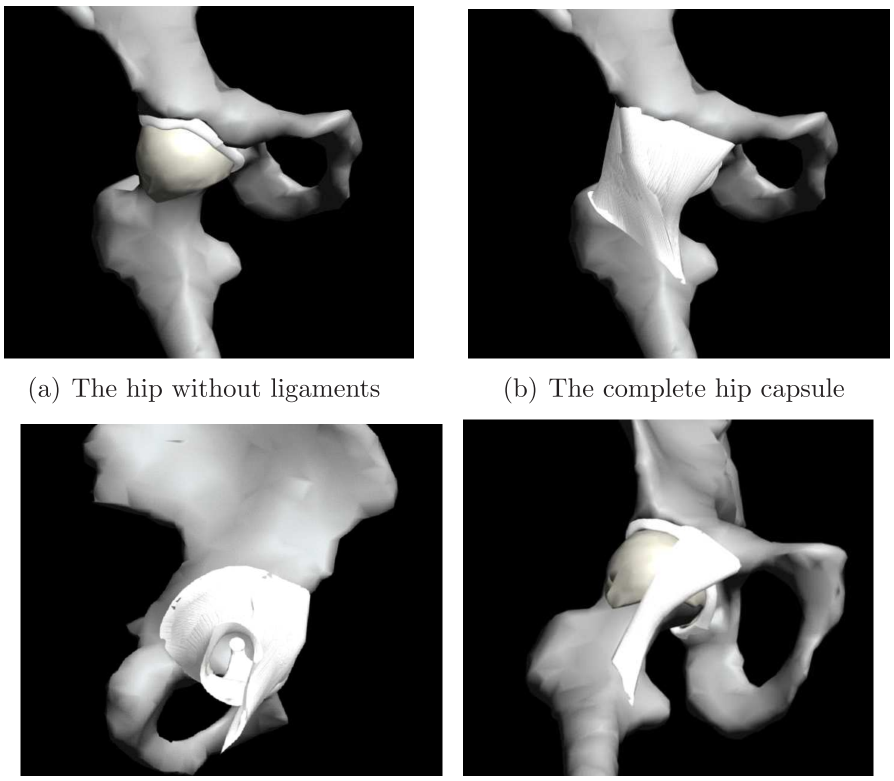


Figure 5.8: Schema of our kinematical hip.



(a) The hip without ligaments (b) The complete hip capsule

(c) Without the femur to inspect inside the capsule (d) Only the pubofemoral ligament

Figure 5.9: Our hip model with the ligaments.

that the joint model of section 5.3, and consequently the models of chapters 2, 3 and 4, fulfill the requirements for medical applications. Sections 5.4.1 to 5.4.4 present details of those applications.

5.4.1 Stress distribution for joint congruity assessment

A uniform and smooth distribution of stress within an articulation denotes the congruity of the articulation, i.e., its capability to transfer motion between the parts it connects with minimum effort and damage. A more congruent hip joint is generally the one which the two cartilage surfaces are more spherical, providing a more uniform contact and consequently a more uniform stress distribution. Assessing stress and strain distribution on the cartilage caps allows analyzing the joint surface, which determines the joint congruency and helps in diagnosis and surgical planning for a number of hip pathologies.

We describe in this section, an interactive test application we have built to compute and analyze the stress distribution and its variation along time during the joint motion [Sarni et al., 2004]. This application allows the user to visualize stress or strain (as seen in chapter 3 stress and strain are correlated) on a 3D model of human hip joint. The user can change the point of view, choose to show or hide specific structures, use different color maps or use transparency to inspect internal parts that are usually masked at different phases of the joint movement.

The virtual hip of section 5.3 is simulated. During simulation, we first calculate the stress around each molecule of the cartilage caps. This is done using equation 3.3, where the force applied is given by the deformation engine (composing all incident forces) while the area is calculated from the molecule radius. Then, for every vertex v_i on the cartilage surface, we combine the stress σ calculated for its anchor molecules a_i ponderated by their respective distances (see equation 5.1).

$$\sigma_{v_i} = \sum_{i=0}^{n} \frac{\sigma_{a_i}}{\|\vec{v_i} - \vec{a_i}\|^2} \tag{5.1}$$

Strain, in turn, is calculated for every molecule relating the nominal and current lengths of its m elastic links, and then mapped for every vertex in the form:

$$\varepsilon_{v_i} = \sum_{i=0}^{n} \frac{\sum_{j=0}^{m} \frac{l_j}{l_{0j}}}{\|\vec{v_i} - \vec{a_i}\|^2} \tag{5.2}$$

Once we have the stresses calculated for all vertices of the mesh, we use color mapping for effective visualization of the stress or strain values (figure 5.10). Color mapping is a common scientific visualization technique to display physical quantities on three-dimensional objects. Being effective and simple, it is widely used in CAD/CAM [Gallagher and Nagtegaal, 1989] [Kuschfeldt et al., 1997] and medical applications [Ik Soo Lim and Thalmann, 2003] for shape analysis. This technique consists in mapping scalar data to colors, and displaying the colors on the computer system. The

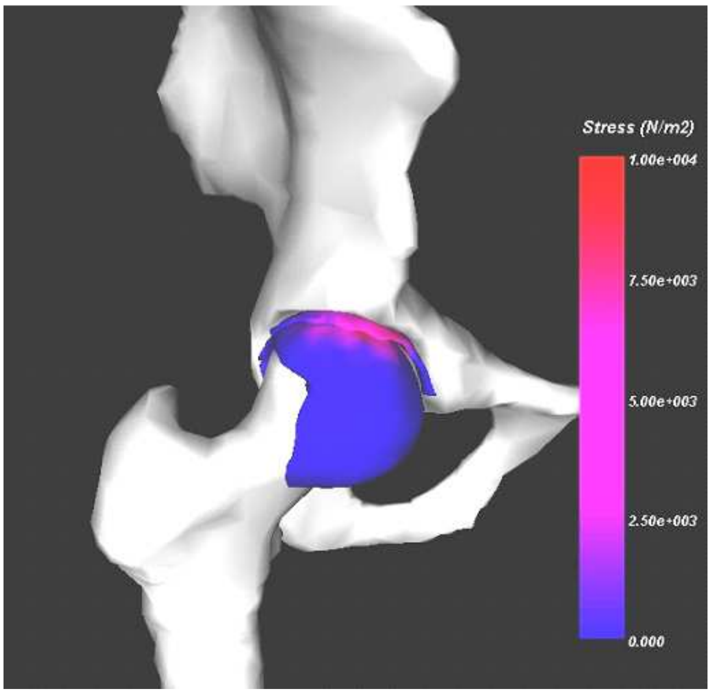
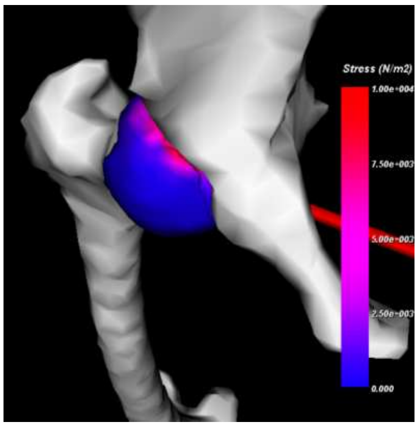


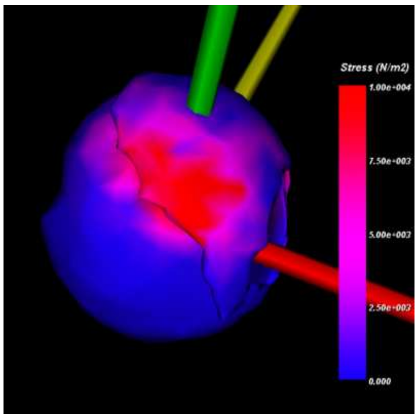
Figure 5.10: Stress distribution mapping on hip joint cartilage using a S-curve, blue-to-red transfer function.

scalar mapping is implemented by indexing into a color lookup table where scalars serve as indices. Color mapping can also be performed by the use of transfer functions, which are expressions that map scalar values into color specifications. A lookup table is a discrete sampling of a transfer function and can be created from transfer functions by sampling the transfer function at discrete points [Schroeder et al., 1997]. Effective color mapping requires careful selection of color lookup tables and transfer functions. This selection requires sensitivity to the qualities of human perception and any special features in the data itself. Levkovitz studied the merits of color scales for medical images and introduced the notion of an optimal color scale [Levkowitz and Herman, 1992]. Lim et al. developed an automatic classification method for color maps [Lim et al., 2003]. Using different transfer functions (linear, S-curve, logarithmic, etc.) and color domains, they generated 168 color tables that were projected onto a 2D layout, which makes exploration easy. This method is applied in this work to select appropriate color tables or transfer functions.

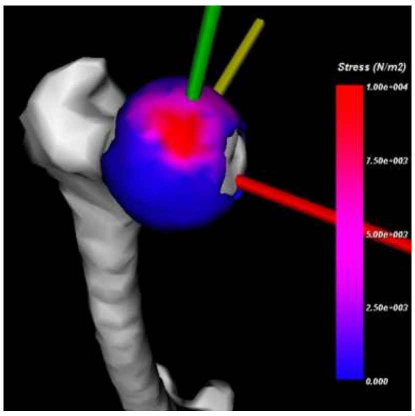
The simulation generates data that can be interactively visualized allowing free exploration of the simulation setup. In figure 5.11, all six images display the same posture but with different options: in the image 5.11(a) all surfaces are visible; in 5.11(c) acetabular parts are transparent to reveal a hidden surface of the femoral cartilage cap; in 5.11(e) the opposite, femoral parts are hidden and a user can inspect stress on the acetabular cartilage (virtual camera is displaced to see the interior); in 5.11(b), both cartilage caps are visible while bones are invisible; in 5.11(d) and 5.11(f) we have respectively the acetabular and femoral cartilage surfaces only. A simulation of evolving stress during motion is also presented on figure 5.12.



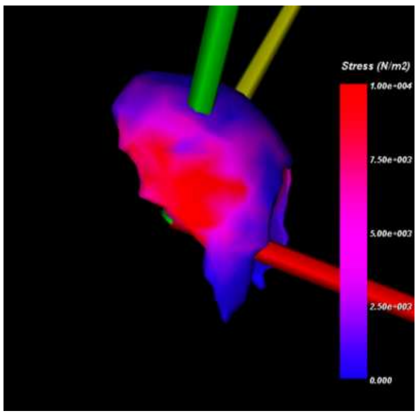
(a)



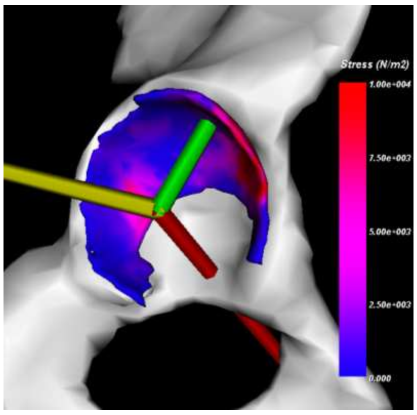
(b)



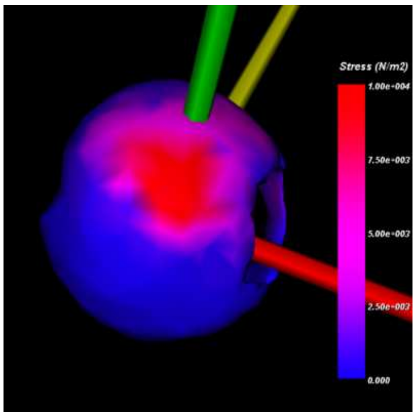
(c)



(d)



(e)



(f)

Figure 5.11: Different views for stress on the hip. Interactive visualization techniques allow free exploration of the simulation setup. All six images display the same posture but with different options: in the image (a) all surfaces are visible; in (c) acetabular parts are transparent to reveal a hidden surface of the femoral cartilage cap; in (e) the opposite, femoral parts are hidden and a user can inspect stress on the acetabular cartilage (virtual camera is displaced to see the interior); in (b), both cartilage caps are visible while bones are invisible; in (d) and (f) we have respectively the acetabular and femoral cartilage surfaces only.

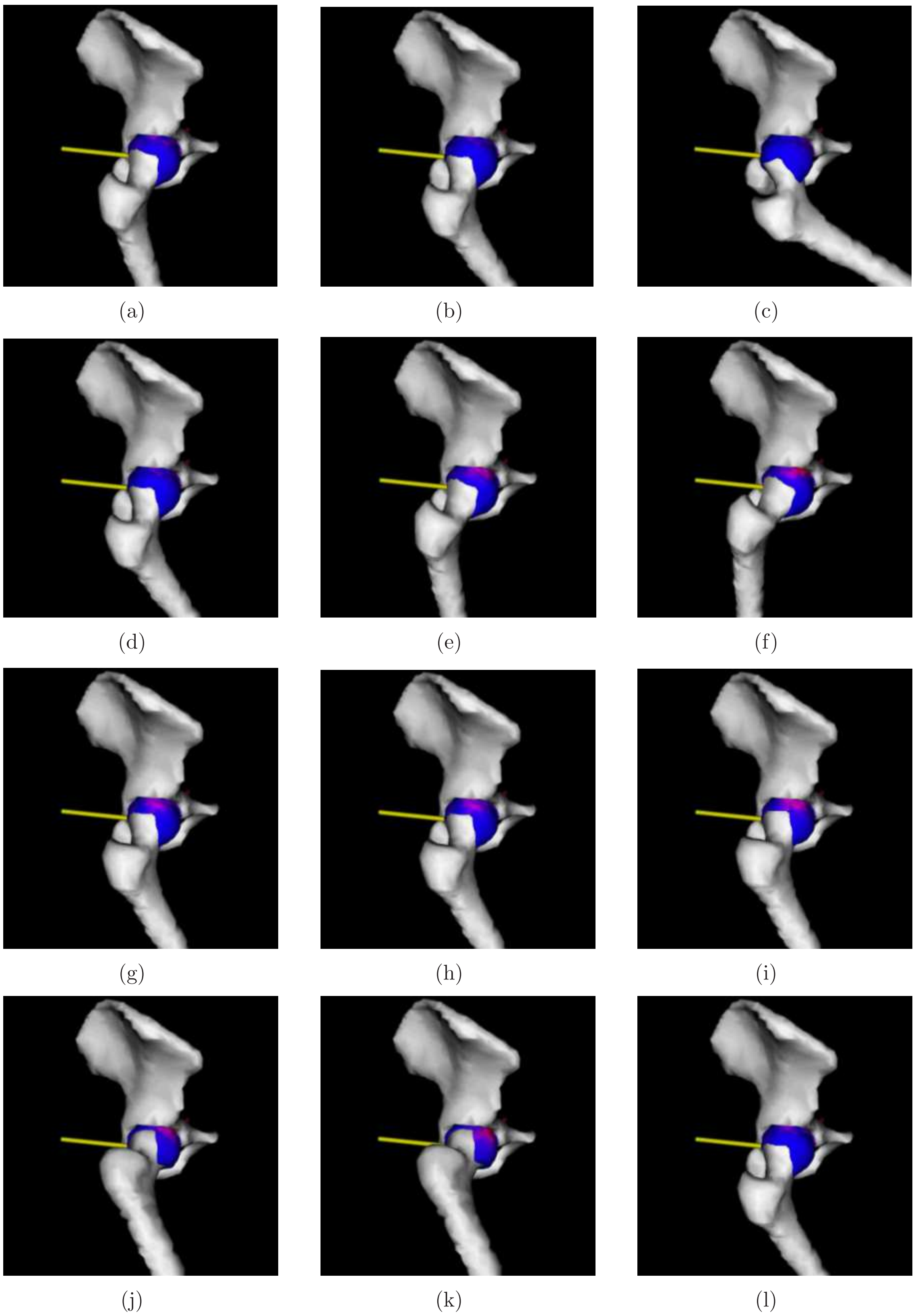
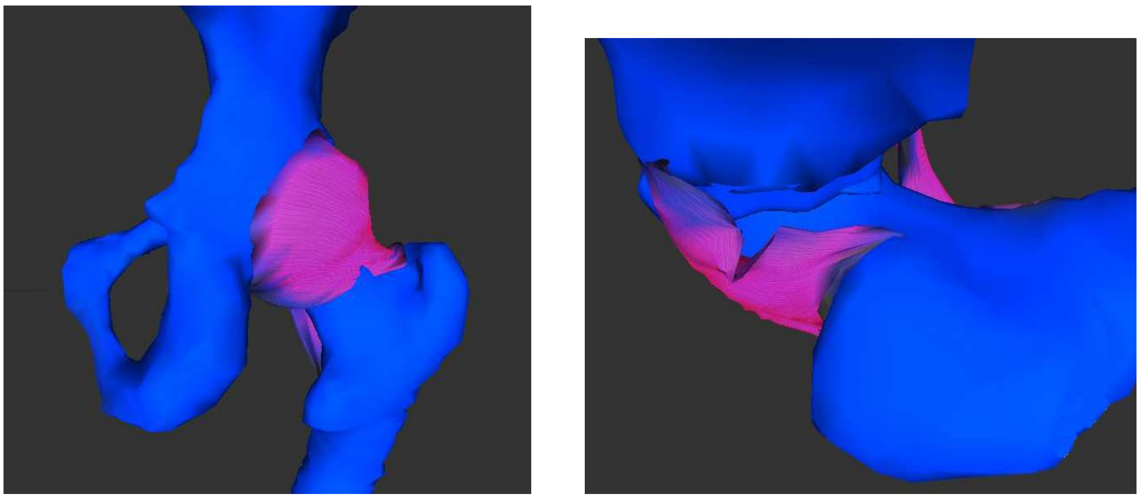


Figure 5.12: Simulated right hip joint evolving along time with stress mapped on cartilage surface. The frames (a) to (f) show a cycle of flexion-extension, and the frames from (g) to (l) show an abduction-adduction cycle. Note that stress is different for equivalent positions taken in forth or back motion because of dynamics.



(a) Total internal rotation at 0° of flexion. (b) Total internal rotation at 90° of flexion.

Figure 5.13: The hip joint in two extreme postures.

5.4.2 Range of motion estimation based on ligament constraint

The range of motion of articulations can be caused by several different constraints. The nature of such constraints can be very complex. One typical example of constraint, which seems simple to understand at first sight but is not, is bony mechanical constraint. It seems obvious that, when the joint moves to an extreme position and the bones touch, you cannot go further and the joint limit has been found. However, even in that case the joint rotation axes can undergo a displacement, provoking a physiological dislocation of the joint and hence pushing the limit of some degrees; this phenomena is particularly visible around the shoulder joint.

Besides bony mechanical constraint, one can mention congruency of sliding surfaces shape, ligaments tensional limit, and even muscular, fatty and skinny volume impingement as factors determining the joint range of motion.

The goal of the application described in this section is determining the range of motion for the hip joint in the case where the ligament plays the role of principal constraint. Our approach consists of moving the joint towards an extreme position while monitoring the stress on the ligament. At the moment in which the stress reach the ligament ultimate strength (or failure stress) the motion stops and we assume the current angle is the limit for that motion.

Specifically, we illustrate the internal rotation motion, which is limited by the ischiocapsular or ischiofemoral ligament. Two situations are analyzed: total internal rotation at zero degree of flexion (figure 5.13(a)); total internal rotation at 90° of flexion (figure 5.13(b)).

The failure stress value for the ischiofemoral ligament can be found in the Biomechanics literature. We have chosen an average value because the values we have found show some variations due to the method used for testing and the quality of the samples. Remember that ligaments are living tissues and their properties are very sensitive to

Source	**Mean stress at failure** (MPa)	**Standard deviation** (MPa)	**Young's modulus** (MPa)
Hewitt	2.0	1.4	N/A
Stewart	2.58	2.16	8.13

Table 5.1: Ischiofemoral ligament stress at failure.

test conditions and donor's characteristics (age, gender, activity, etc.). See table 5.1 for the data we considered.

We built a model of the hip including the pelvis and the femur bones, the acetabular and the femoral head cartilages, and the ischiofemoral ligament. The ligament elasticity (E) was set to $0.8MPa$ in the fibers orientation and 50% of that perpendicularly - remember that ligament is anisotropic. Then we applied total internal rotation at zero degree of flexion and total internal rotation at 90° of flexion to the joint. The maximum angles we obtained were 68° and 53° respectively.

5.4.3 Ligament elasticity estimation from range of motion

Ligaments elasticity have been measured in Biomechanics and several publications exist presenting data from those measurements. However, it is known that the obtained values are very sensitive to the measure technique and to the samples condition. In vivo measurements remaining unpracticable up to now, the elasticity measured varies a lot depending on the conservation of the cadaveric samples. In addition, appropriate cadaveric samples are difficult to obtain, those available being from elder donors which mechanical properties do not correspond to young individuals.

Moreover, if the coefficient of elasticity is to be used for clinical applications where customized data are necessary to diagnose to a specific patient, generic data can be very imprecise and any existent measure technique would be too much invasive.

Given this context, the goal of this application is to estimate the elasticity of a specific ligament according to the range of motion the patient can afford, which is easily measured by a clinician aided by a *goniometer* (figure 5.14). In our application, a virtual model of the patient's joint is kinematically driven to a extreme position known to be constrained by the ligament being analyzed. At that position, a real ligament is close to its rupture stress. Knowing the rupture stress, the equation 5.3 allows to inversely calculate the Young's modulus for the ligament.

$$E = \frac{\sigma \cdot l_0}{\Delta l} \tag{5.3}$$

In our experiment, knowing that the internal rotation of the hip in 90° of flexion is checked by the ischiofemoral ligament, we suppose that rotating internally the hip of a healthy person beyond the angle they can support with no pain will tear that ligament. So, after measuring the maximal internal rotation angle on a volunteer (40°), we simulated their hip driving it to the same limit posture. During this first movement, we considered the ischiofemoral as being very soft in our model (1 MPa). Along this

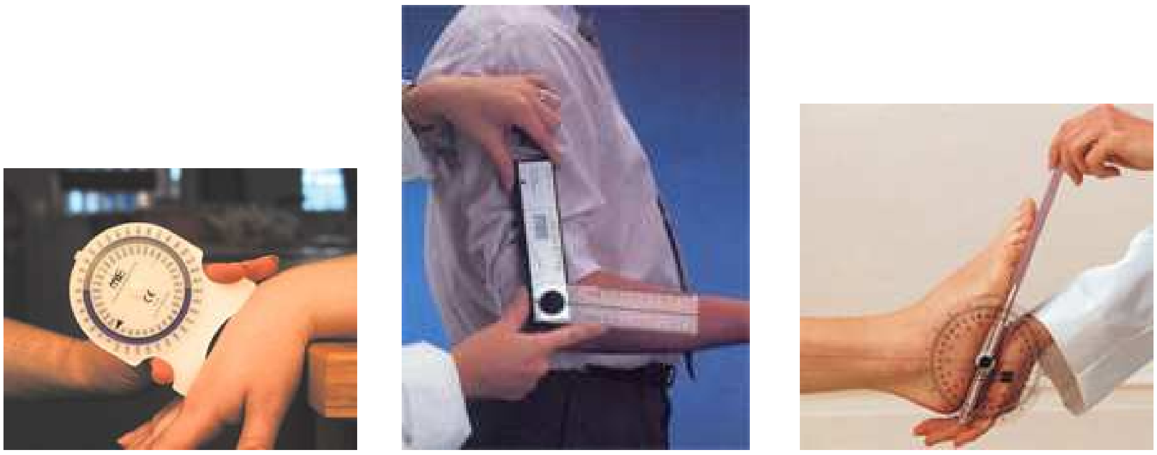


Figure 5.14: Examples of goniometer.

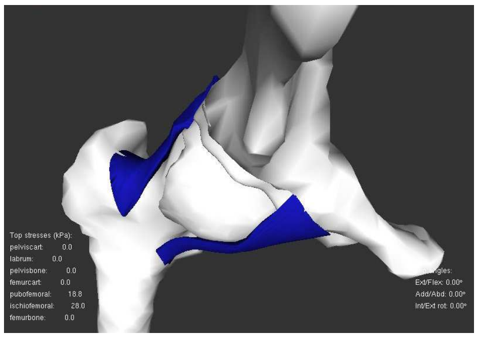
phase, the maximum stress measured on the ligament was 1.5 MPa. We continued the simulation from that point, but now increasing the stiffness of the ligament of 1 MPa per second of simulation. At every iteration we measured the maximum stress on the tissue until it reached the mean failure stress value of 2.5 MPa that we had found on the literature according to table 5.1. At that point we stopped the simulation and concluded that the stiffness valid at that moment corresponds to the elasticity (E) of that ligament in the direction we stressed it, i.e. the longitudinal direction in relation to the fibers, and is of about 8 MPa.

5.4.4 Pre- and post-operative evaluation of stress distribution

With this application we present a case study using a spreadsheet-like interface for exploring biomedical datasets generated by our biomechanical model of the hip [Sarni et al., 2005]. The case we analyze here is an osteotomy corrective surgery that reorients the femoral body in relation to the femoral neck, somewhat like the intertrochanteric osteotomy described by Imhäuser [Imhäuser, 1977]. That is a procedure recommended by most authors as a surgical treatment of severe slipped capital femoral epiphysis. Then we compare estimated contact area of the pre- and post-operative hips from a threshold of the computed stress [Maciel et al., 2005].

For many clinical applications, like the present one, the effort of using realistic computational and graphical models is important but is not sufficient in itself since clinicians need tools that can effectively help them in performing the regular tasks they are used to perform on biomedical images. Resulting biomedical datasets can be multi-dimensional and are usually very large, which makes them difficult to explore and understand [Lim et al., 2003]. Therefore there is a need for intuitive representations for end users to deal with this increasing complexity. One solution is the use of spreadsheet-like interfaces, which are a generalization of conventional spreadsheets where cells can contain graphical objects such as images, volumes, or animations or even widgets to interact with data. In this class of visualization systems, screen space is spent on operands rather than operators, which are usually more interesting to the end user [hsin Chi and Riedl, 1998]. They also benefit from the fundamental properties of spreadsheets where it is easy to organize, compare, analyze and perform operation on data. Our spreadsheet framework consists basically of: cells, operators and dependencies. Cells are the basic data elements. They can contain numbers, images, curves, vectors, matrices or widgets for interactive cells. The cells are organized in a tabular layout, which makes them easy to browse. Operators are applied to cells or ranges of cells and define the dependencies between the cells. A firing algorithm controls dependencies as in conventional spreadsheets and automatically updates the cells to reflect changes.

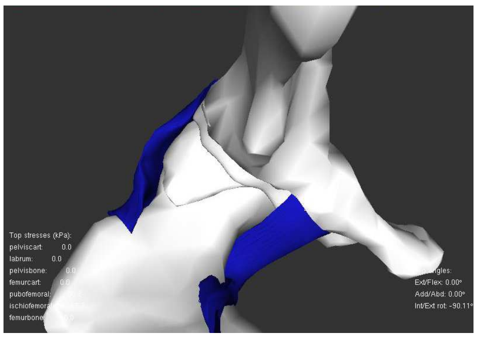
A right hip model has then been built to illustrate a use case where the pre- and post-operative joint congruity is analyzed. The elements present in our hip are: the femur and the pelvis bones; the femoral head and the pelvis cup cartilages; the ischiofemoral ligament; the acetabular rim (labrum). The bones and the labrum are considered rigid, and the elasticity for the cartilages and ligament is defined to be 10 kPa. It is softer than the mean value found on the literature, but it allows our simulation to converge faster, while the stress distribution remains coherent. In addition, fibers orientation is taken into account for the ligament, in a way that it is anisotropic. The motion we performed is 90° of flexion plus total internal rotation, a typical motion for clinical examination in Orthopedics (see figure 5.15). To simulate the osteotomy operation, we deformed the 3D femur moving the distal extremity internally on the frontal plan, such that the hypothetic patient has to abduct his hip to keep the knee at its place. We represent this abduction as a reorientation of the femoral head - and consequently of the whole femur - such that the new anatomical axis of the femur keeps aligned with the original one (see figure 5.16). Then we applied the same motion on the osteotomized joint that we had applied on the original. We could observe that the stress distribution on the cartilage surfaces, and consequently the contact area, change in relation to the pre-operative joint for the same motion (see figure 5.17). In a real clinical case, the surgeon could simulate different reorientations prior to operation and then choose the angles that best improve the congruity of the patient's joint.



```
Top stresses (kPa):
pelviscart:      0.0
labrum:      0.0
pelvisbone:      0.0
femurcart:      0.0
pubofemoral:      18.8
ischiofemoral:      28.0
femurbone:      0.0
```

```
ngles:
Ext/Flex: 0.00°
Add/Abd: 0.00°
Int/Ext rot: 0.00°
```

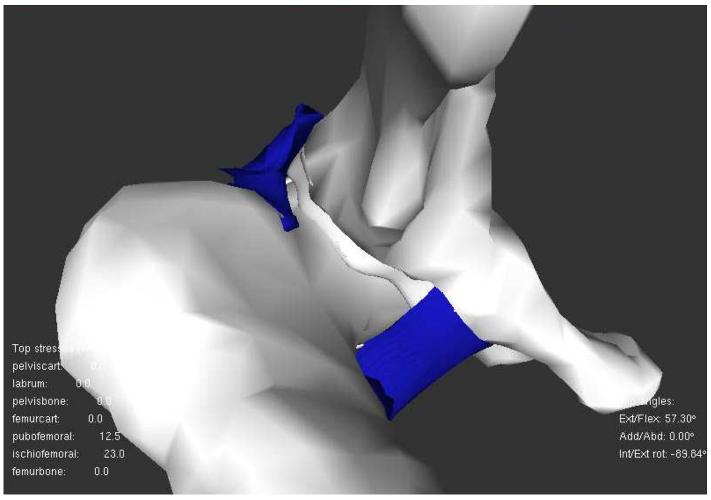
(a) Rest position



```
Top stresses (kPa):
pelviscart:      0.0
labrum:      0.0
pelvisbone:      0.0
femurcart:      0.0
pubofemoral:
ischiofemora
femurbone
```

```
ngles:
Ext/Flex: 0.00°
Add/Abd: 0.00°
Int/Ext rot: -90.11°
```

(b) 90° flexion



```
Top stress
pelviscart:
labrum:      0.0
pelvisbone:      0.0
femurcart:      0.0
pubofemoral:      12.5
ischiofemoral:      23.0
femurbone:      0.0
```

```
ngles:
Ext/Flex: 57.30°
Add/Abd: 0.00°
Int/Ext rot: -89.84°
```

(c) 90° flexion + total internal rotation

Figure 5.15: Typical motion for clinical examination in Orthopedics.

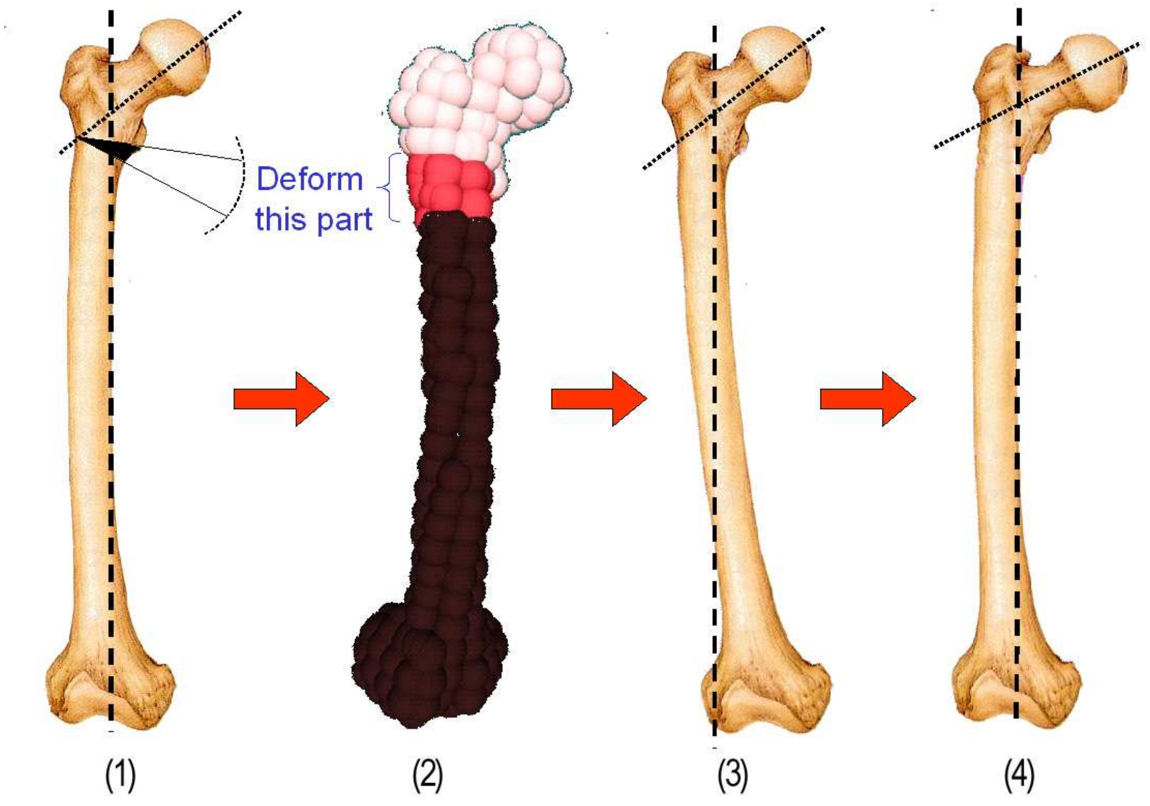


Figure 5.16: Pipeline of the simulated osteotomy. The original femur (1) is discretized into molecules (2), the light and dark grey parts are rigid and the red part is deformable. Kinematical adduction of the thigh in relation to the femoral neck (3) moves the distal epiphysis of the femur out of the anatomical axis. Bringing it back to its position to keep the knee at its place requires abduction of the hip such that the angle between the two dashed lines change in rest posture (4).

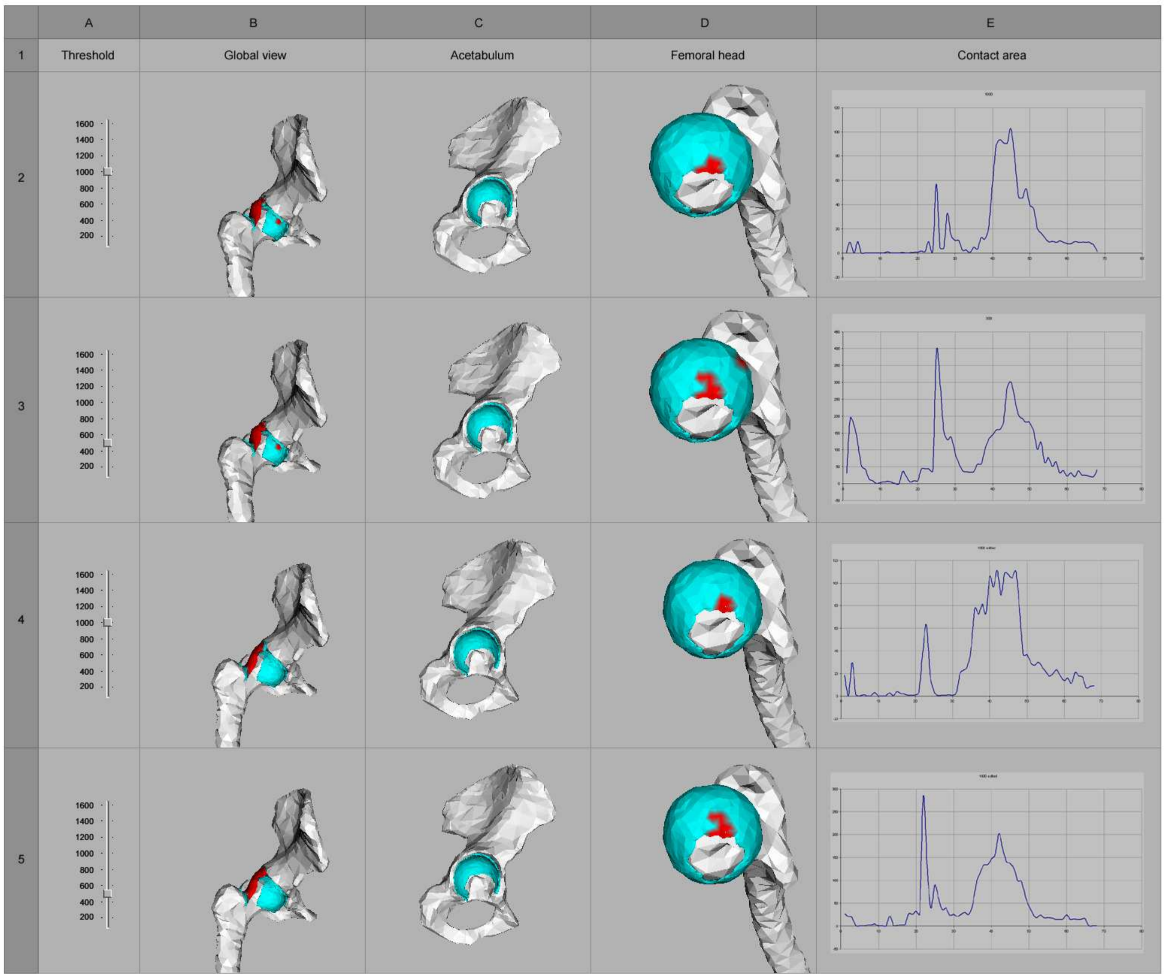


Figure 5.17: A template comparing pre- and post-operative virtual hip. The contact area is estimated based on the stress calculated on the cartilage surfaces. A stress threshold was defined in A2 and used to determine the contact area on the animation in B2. Other views were derived (C2 and D2) and a graph of the contact area history is shown in E2. The row 2 was then copy-pasted into rows 3 to 5. Rows 2 and 3 represent simulated pre-operative situation for different threshold values. Rows 4 and 5 represent simulated post-operative situation.

CHAPTER 6

Conclusion

6.1 Summary

The goal of this thesis was to propose a 3D conceptual articulations model suitable for medical applications. The directives guiding the model conception were:

- to extend the traditional idealized rotational joints to consider anatomical and biomechanical features;
- to be at the same time simple enough to be implemented on computer, and realistic enough to allow for medical applications;
- to be visual in order for applications to be able to explore the joint in a 3D simulation environment.

Our approach to develop such a model was hybrid: both physically based soft tissues deformation and skeleton kinematics were used. An enhanced skeleton model was proposed which takes into account anatomical features lacking in usual computer graphics models. At the same time, a discrete soft tissues model was presented, which can be parameterized in such a way that it characterizes different material properties, and simulates in a 3D physical environment.

The problem of permanent contact inside the joint was also dealt with. Existing collision detection and response techniques were studied and a new one focusing on the joint particularities was designed.

Finally, we implemented a small set of medical applications for which the model suits. The applications uncover information which aid on medical diagnosis and biomechanical analysis of the joint, particularly by calculating strains and tensions along the tissues volume.

6.2 Contributions

6.2.1 Anatomy-based articulation model

Joint models in Computer Graphics being often based on Robotics, they almost always consider only the ideal case of axial rotations. However, the human anatomy is much more complex then the one of robots.

We contributed to change this situation by presenting an articulation model which contemplates some relevant anatomy requirements. Our model includes different joint types depending on their motion degrees of freedom. Such degrees of freedom can be represented both as rotations and translations. Rotational and translational axes are not fixed on the joint center like in robotics. They can be associated to displacement curves in order to glide on the paths determined by those curves as they rotate or translate. This models the joint center displacement, particularly visible in the case of the knee or the shoulder, but present in nearly all articulations.

The model also controls the range of motions, not allowing the joint to move beyond its limits. Such limits are not rigid, though. Two levels of limits are considered: comfort angles and limit angles. This particularity was used in this work to keep track of the limits imposed by non-rigid mechanical constraints such as the influence of ligaments, for example. Nevertheless, we can also use these limits to tune inverse kinematics methods for motion control if our joint model is used with traditional virtual humans animation.

The kinematical control of the articulation is done at a higher level. A unique normalized parameter, a real number between 0 and 1, defines the status of each degree of freedom. From this parameter, the joint model calculates the angular orientation, the position of the axis on the sliding curve, and the influence from other motion axes positions. The combination of all those give the actual position of the joint-coordinates frame.

6.2.2 Biomechanics-based deformation model

A number of models for tissue deformation exists in the literature of Mechanics, Biomechanics and Computer Graphics. They are based on various approaches and give results of different quality. Some are extremely physically accurate as well as very complex and need enormous computational resources to be calculated. Others are focused on visual appearance, leaving physical accuracy to a second plan, and at the same time that they can be computed in realtime. In-between, a number of different approaches try to maximize realism and minimize computing complexity in order to find the best compromise.

Our approach, which we call *molecular model*, was based on a traditional physics-based technique in Computer Graphics: mass-spring systems. Our approach for numerical integration of the system was the use of classical explicit methods, for instance Euler and Runge Kutta or fourth order.

Thus, our contribution is around one of the limitations of mass-spring systems: the inability or extreme difficulty to control stiffness to characterize tissue elasticity due to the large number of parameters inherent to the system.

We tested a number of approaches which allow for a certain control over the objects elasticity, which is given by the material Young's modulus. We finally propose an interactive approach which ensures control of the linear part of the stress-strain relation for a given range of forces for arbitrary mass-spring meshes. Given the fact that the range of forces inside a joint is known and can be found in the literature of Biomechanics, our model guarantees an accurate enough tissue characterization for the intended medical applications.

6.2.3 Quasi-permanent contact model

Collision detection and collision avoidance are important research threads in Computer Graphics because they increase the realism of virtual environments. In a virtual physical environment aimed at medical applications, however, the realism of the collision cannot be merely visual. Contact must provide information which can be used for medical decision making.

Contact management inside an articulation model is a specific situation with two opposed characteristics. On one hand, the permanent contact and pressure makes the problem more complex and difficult to optimize. On the other hand, the articulation is a somewhat closed system in which a number of assumptions can be done, for instance: the topology does not change during physiological activity.

In this context, we bring two contributions. First, we detect collisions efficiently using a method which takes advantage of the known joint topology. Second, we avoid penetration at the collision loci without directly calculating response forces, relying on the propagation of strain provided by our soft tissues deformation model.

Our collision detection method, which we call *sliding sphere*, measures in constant time the signed distance to the closest point for each point of two meshes subject to collide. It makes use of a hash table to determine at any time which element of one mesh is the closest to any given element of the other mesh. The method has shown to be effective for contact between cartilage caps and for the hip ligaments. It may not be effective for the knee cruciate ligaments, for example, because they do not respect a layered topology.

Our penetration avoidance method consists in putting the two colliding elements aside geometrically such as they lie on each other instead of penetrating further. To increase stability for computer simulation, we also look one timestep forward at every simulation iteration and tune the velocities of potentially colliding elements one step further, in order for them to lie on each other in the next frame. These geometrical changes on the surface are then propagated throughout the tissue which physically reacts to the tension they impose on the springs closed to the surface.

The combination of the two methods has shown to be efficient and effective for the tested applications. It does not increase the numerical instability of the system; we demonstrated that the numerical errors we have with the deformation model are not negatively affected by our contact management model.

6.2.4 Biomechanics-based articulation model

The main contribution of this thesis, as foretold by its title, is an articulation model for medical applications. We combined the kinematical skeleton provided by our anatomy-based articulation model with our biomechanics-based model of soft tissues, and with our contact management model to build up such as 3D articulation model.

The bones are rigid and move following kinematical rules controlled by the parameters of the anatomic joint model. Motion can be specified from patients motion capture, thus reproducing real motions, or specified by a specialist, which gives flexibility. On top of the bones, more specifically around the locus where they meet, virtual cartilages and ligaments deform under the constraints imposed by the bones motion. Physical parameters such as stress distribution along the soft tissues can then be evaluated.

The obtained articulation motion has shown to be visually realistic and the physical parameters calculated are meaningful for the intended applications. Nevertheless, the absence of global clinical validation is a main deficiency, even if some validation could be done involving clinically measured ranges of motions. Validating biological systems is a very hard task because in-vivo data are difficult to obtain. One possible approach for further work could be using dynamic MRI to compare deformations.

6.2.5 Force-feedback from deformable objects

Another contribution is the use of a haptic interface with deformable objects. Haptic interfaces are becoming popular in Virtual Reality applications. They can enrich the amount and the quality of the information perceived by a user. They can be valuable to medical applications. Consequently, we tested the compliance of our deformable model with such interfaces.

Haptic applications are realtime, and force feedback devices require high frequency updates (at least 400 Hz but usually 1000 Hz). It is for this reason that most haptic applications involve rigid objects only. To achieve such performances with deformations, we created coarse resolution objects with our deformation model (27 mass elements). We then presented uniformly shaped objects characterized by different elasticities to a group of users to try to identify them, classifying them in order of stiffness.

The perception test was successful, the users being able to recognize different elasticities despite the limitations of the force-feedback used. However, more complex objects, like the ones of the articulation model, cannot be simulated in realtime, which makes the use of haptics still inviable.

6.2.6 Medical applications

We also contributed by setting up a few applications exploiting our models in the fields of computer aided diagnosis and surgery planning. We built a model of the hip joint and developed the following four medical applications:

- **Stress distribution for joint congruity assessment**

  Considering the loads provoked by bones motion, this application computes stress on the cartilage caps. It uses methods from scientific visualization, such as color mapping, to visualize the stress in non-photorealist rendering mode.

  The application aids medical doctors to build a more accurate mental model of the patient case, correlating abnormal stress distribution with pain, for example.

- **Range of motion estimation based on ligament constraint**

  This application aims at estimating the joint range of motion. Unlike most existing approaches, which are based on bone contact to define range of motions, we impose an additional constraint by considering the presence of the ligaments.

  Using information about the maximum stress a ligament can bare before rupture, we move our joints to extreme positions while we measure the current maximum stress on the ligament at each timestep. The positions in which the maximum measured stress is greater than the failure stress for the material are considered unreachable, defining joint limits.

- **Ligament elasticity estimation from range of motion**

  This application uses the inverse approach to the previous one. Here, we use the range of motions measured by a clinician as input, and aim at estimating the ligament elasticity.

  Knowing that for certain postures moving further than the limits of the range of motions causes ligament rupture, and knowing the average ligament rupture stress, which can be found in Biomechanics literature, we estimate the Young's modulus of this tissue. We start simulating with a very soft ligament, and we drive the joint to an extreme position. At that point we start increasing the ligament stiffness at every timestep, always controlling the maximum stress, which will be consequently increasing as well. When the maximum measured stress reaches the rupture stress, we stop simulating, and consider that the current stiffness we have obtained corresponds to the actual ligament elasticity.

- **Pre- and post-operative evaluation of stress distribution**

  Reorienting a malformed bone by osteotomy is a practice in orthopedical surgery. However, planning such a surgery always involve unknown elements which can only be partially controlled, and only by an experienced surgeon. The Imhäuser osteotomy, for instance, aims at reorienting the femoral head in order to improve the hip range of motions. It can be necessary, for example when a patient has part of the femoral head cartilage destroyed and contact with the acetabulum wants to be avoided.

We proposed an application allowing to simulate the effects of such a surgery. Range of motions and stress distribution are first evaluated on the pre-operative model. Then, the bone is edited to provide a new joint configuration, analogously to what happens in a surgical procedure. Finally, the analysis of the post-operative virtual model allows for the assessment of the improvements obtained. In case it is not satisfactory, different femoral head orientations can be tested until the surgeon is satisfied with the result on the 3D model. They can then reproduce the same parameters in the real surgery with a better chance of success.

6.3 Perspectives

The articulation model resulting from this work allows for numerous developments in medical applications in Orthopedics. However, some future work must still be done in order for the model to be used in clinical practice.

From the clinical point of view, two major elements must be worked further before permitting for real patients to be submitted treatment with the help of applications based on this model.

The first is clinical validation, a number of clinical tests are necessary to affirm the model is valid. One manner we can describe to do that is parallel prediction. After a surgeon diagnoses and plan intervention for a real patient, we parallelly simulate the patient joint preoperatively, and submit the model to the same procedure applied by the surgeon on the patient. We must be able to predict surgery outcome, and compare our prediction with the real outcome after surgery and recovery.

Second, the current procedure to obtain individualized patient data is slow, expensive and incomplete. New hardware and techniques for image segmentation and 3D reconstruction would be necessary to make the clinical use of such computer aided approaches viable. Besides, some individualized information, such as accurate mechanical properties of tissues, cannot be obtained so far. New equipment and techniques are in constant evolution in mechanical and biomechanical sciences, but at this day we only have fragile estimations.

From the technical point of view, the deformation model can be improved in terms of material characterization. Although we can already control elasticity, viscosity and anisotropy, one important variable is not under control: shear modulus. It is known that a material is mostly characterized by these two moduli: Young's modulus and shear modulus. Thus, our model lacks in accuracy when shear forces have an important part of the total force in the system. One approach to control shear can be by applying an iterative test as we did for elasticity. In this case, the angle between the force and the elastic connections must be taken into account, and strings stiffness tuning must depend on that angle.

In terms of performance, our model can be improved on the implementation level. Parallel processing, exploiting the new programable graphics cards can reduce algorithm complexity for numerical integration, for example, from $O(n)$ to $O(1)$, because the n springs displacements can potentially be computed in parallel in one GPU cycle. Collision detection can also support GPU implementation.

Furthermore, biomechanical joints should be integrated into computer animation systems. We believe they can have a positive influence on the way virtual characters move, having some advantages over the very simplified joints used nowadays.

Looking farther in the future, we imagine a next generation of virtual joins in which no axis will be present: articulations in which the motion depends only on the mechanical interactions of their different elements, on the constraints imposed by their geometry and behavior as deformable structures. Nevertheless, before this can be done, the available computer hardware must improved. Meanwhile, a possible intermediary solution could be to loose the current kinematical constraints by non-rigidly attaching the joint-coordinate frame. This could be done by simulating a spring attracting the instantaneous center to an ideal center, such that the center position is a cloud of possible positions independent of the joint angles.

List of Figures

Bibliography

Maya User Manual. Alias/Wavefront, 2001.

M. Amrani, F. Jaillet, and B. Shariat. Deformable objects modeling and animation: Application to organs' interactions simulation. *Journal for Geometry and Graphics*, 4(2):181–188, 2000.

Amaury Aubel. *Anatomically-Based Human Body Deformations.* PhD thesis, Ecole Polytechnique Fédérale De Lausanne (EPFL), Lausanne, Switzerland, 2002.

Amaury Aubel and Daniel Thalmann. Realistic deformation of human body shapes. In *Proceedings of Computer Animation and Simulation 2000*, pages 125–135, 2000.

Amaury Aubel and Daniel Thalmann. Interactive modeling of the human musculature. In *Proceedings Computer Animation 2001*, pages 167–255, 2001.

C. Babski. *Virtual Humanoid on the Web.* PhD thesis, Ecole Polytechnique Fédérale De Lausanne (EPFL), Lausanne, Switzerland, 2000.

George Baciu, Wingo Sai-Keung Wong, and Hanqiu Sun. Recode: an image-based collision detection algorithm. *Journal of Visualization and Computer Animation*, 10 (4):181–192, 1999.

Paolo Baerlocher. *Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures.* PhD thesis, EPFL, 2001.

Srikanth Bandi and Daniel Thalmann. An adaptive spatial subdivision of the object space for fast collision detection of animated rigid bodies. *Computer Graphics Forum*, 14(3):259–270, 1995.

David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics*, volume 23, pages 223–232. Proceedings of ACM SIGGRAPH, July 1989.

David Baraff. Non-penetrating rigid body simulation. In *State of the Art Reports of Eurographics*, volume 12, page 24, September 1993.

David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-667-0. doi: http://doi.acm.org/10.1145/192161.192168.

David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM Press. ISBN 0-89791-999-8. doi: http://doi.acm.org/10.1145/280814.280821.

Alan H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1984. ACM Press. ISBN 0-89791-138-5.

Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An introduction to splines for use in computer graphics & geometric modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. ISBN 0-934613-27-3.

Dimitri E. Beskos. *Boundary Element Methods in Mechanics*. Mechanics and Mathematical Methods. North-Holland, 1989.

F. M. Blakeley. Cyberman. Chrysler Corporation, June 1980.

James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1 (3):235–256, 1982. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/357306.357310.

S. Bonner and R. B. Kelley. A representation scheme for rapid 3-d collision detection. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 320–325, August 1988.

Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley Professional, 1st edition, September 1998. ISBN 0201571684.

Ronan Boulic and Olivier Renault. *New Trends in Animation and Visualization*, chapter 3D Hierarchies for Animation, pages 59–78. John Wiley & Sons ltd, UK, 1991.

William J. Bouma and Jr. George Vanecek. Collision detection and analysis in a physical based simulation. In *Proceedings of the Eurographics Workshop on Animation and Simulation*, pages 191–203, 1991.

David Bourguignon and Marie-Paule Cani. Controlling anisotropy in mass-spring systems. In *Proceedings of the 11th Eurographics Workshop on Computer Animation and Simulation*, pages 113–123, 2000.

Gareth Bradshaw and Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.*, 23(1):1–26, 2004. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/966131.966132.

Morten Bro-Nielsen and Stéphane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proceedings of Eurographics '96 - Computer Graphics Forum*, pages 57–66, 1996.

Joel Brown, Stephen Sorkin, Cynthia Bruyns, Jean-Claude Latombe, Kevin Montgomery, and Michael Stephanides. Real-time simulation of deformable objects: Tools and application. In *Computer Animation 2001*, pages 228–236, November 2001.

D. L. Butler, E. S. Grood, F. R. Noyes, and R. F. Zernicke. *Basic Orthopaedic Biomechanics*, chapter Biomechanics of Ligaments and Tendons, pages 125–181. ., 1984.

Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 99–104, New York, NY, USA, 1992. ACM Press. ISBN 0-89791-479-1. doi: http://doi.acm.org/10.1145/133994.134017.

J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1989. ACM Press. ISBN 0-201-50434-0. doi: http://doi.acm.org/10.1145/74333.74358.

Yu-Kuang Chang and Alyn P. Rockwood. A generalized de casteljau approach to 3d free-form deformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 257–260, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-667-0. doi: http://doi.acm.org/10.1145/192161.192220.

David T. Chen and David Zeltzer. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. *SIGGRAPH Comput. Graph.*, 26(2):89–98, 1992. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/142920.134016.

Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. I-collide: an interactive and exact collision detection system for large-scale environments. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 189–ff., New York, NY, USA, 1995. ACM Press. ISBN 0-89791-736-7. doi: http://doi.acm.org/10.1145/199404.199437.

Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1990. ACM Press. ISBN 0-201-50933-4. doi: http://doi.acm.org/10.1145/97879.97900.

Sabine Coquillart and Pierre Jancéne. Animated free-form deformation: an interactive animation technique. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 23–26, New York, NY, USA, 1991. ACM Press. ISBN 0-89791-436-8. doi: http://doi.acm.org/10.1145/122718.122720.

Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/2945.764872.

Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press, Boston, february 1992. ISBN 0121923509.

F. W. da Silva. Um sistema de animação baseado em movimento capturado. Master's thesis, LCG da COPPE/UFRJ, Rio de Janeiro, 1998. in Portuguese.

Fernando Wagner da Silva, Luiz Velho, Paulo Roma Cavancanti, and Jonas Gomes. An architecture for motion capture based animation. In Luiz Henrique de Figueiredo, editor, *Anais do X SIBGRAPI*, Campos do Jordão, Brazil, Outubro 1997.

Gilles Debunne, Marie-Paule Cani, Mathieu Desbrun, and Alan Barr. Adaptive simulation of soft bodies in real-time. In *CA '00: Proceedings of the Computer Animation*, page 15, Washington, DC, USA, 2000. IEEE Computer Society.

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 31–36, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-374-X. doi: http://doi.acm.org/10.1145/383259.383262.

Angel Pasqual del Pobil and Miguel Angel Serna. *Spatial Representation and Motion Planning*, volume 1014 of *LNCS*. Springer-Verlag, 1995.

J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, 22:215–221, 1955.

Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 1–8, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-632-7.

David P. Dobkin and David G. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 400–413, New York, NY, USA, 1990. Springer-Verlag New York, Inc. ISBN 0-387-52826-1.

François Faure. An energy-based approach for contact force computation. *Computer Graphics Forum*, 15(3):357–366, August 1996. ISSN 1067-7055.

Stephen J. Ferguson. *Biomechanics of the Acetabular Labrum.* PhD thesis, Queen's University, Kingston, Ontario, Canada, March 2000. Department of Mechanical Engineering.

William A. Fetter. A progression of human figures simulated by computer graphics. *IEEE Computer Graphics and Applications*, 2(9):9–13, November 1982. ISSN 0272-1716.

Susan Fisher and Ming C. Lin. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation*, pages 99–111, New York, NY, USA, 2001. Springer-Verlag New York, Inc. ISBN 3-211-83711-6.

J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Co., 1990. ISBN 0-201-12110-7.

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5. doi: http://doi.acm.org/10.1145/344779.344899.

A. Fuhrmann, C. Groß, and V. Luckas. Interactive animation of cloth including self collision detection. In *Proceedings of Winter School of Computer Graphics (WSCG 2003)*, pages 203–208, 2003.

Y. C. Fung. *Biomechanics: Its Foundations and Objectives.* Prentice-Hall Inc., Englewood Clifts, NJ, 1972.

Y. C. Fung. *Biomechanics: mechanical properties of living tissues.* Springer-Verlag, USA, second edition edition, 1993.

R. S. Gallagher and J. C. Nagtegaal. An efficient 3-d visualization technique for finite element models and other coarse volumes. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 185–194, New York, NY, USA, 1989. ACM Press. ISBN 0-201-50434-0. doi: http://doi.acm.org/10.1145/74333.74352.

F. Ganovelli, J. Dingliana, and C. O'Sullivan. Buckettree: Improving collision detection between deformable objects. In *Proceedings of SCCG*, 2000.

Stéphane Garchery, Ronan Boulic, Rolga Capin, and Prem Karla. *Handbook of Virtual Humans*, chapter Standards for Virtual Humans, pages 373–391. John Wiley & Sons ltd, UK, 2004.

Alejandro Garcia-Alonso, Nicol&#225;s Serrano, and Juan Flaquer. Solving the collision detection problem. *IEEE Comput. Graph. Appl.*, 14(3):36–43, 1994. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.279041.

Allen Van Gelder. Approximate simulation of elastic membranes by triangulated spring meshes. *J. Graph. Tools*, 3(2):21–42, 1998. ISSN 1086-7651.

Sarah Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratory, 1997.

E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimentional space. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.

Alan Barr Gilles Debunne, Mathieu Desbrun and Marie-Paule Cani. Interactive multiresolution animation of deformable models. In *10th Eurographics Workshop Computer Animation and Simulation*, pages 133–144, Sep 1999.

Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Comput. Graph. Appl.*, 7(5):14–20, 1987. ISSN 0272-1716.

S. Gottschalk. Separating axis theorem. Technical Report TR96-024, UNC Chapel Hill, 1996.

S. Gottschalk, Ming C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-746-4. doi: http://doi.acm.org/10.1145/237170.237244.

J.-P. Gourret, N. M. Thalmann, and D. Thalmann. Simulation of object and human skin formations in a grasping task. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1989. ACM Press. ISBN 0-201-50434-0. doi: http://doi.acm.org/10.1145/74333.74335.

Naga Govindaraju, Stephane Redon, Ming C. Lin, and Dinesh Manocha. CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. In *ACM SIGGRAPH/Eurographics Workshop On Graphics Hardware*, pages 25–32, 2003.

Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph.*, 24(3):991–999, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073301.

Henry Gray. *Anatomy of the human body.* Philadelphia: Lea & Febiger, 1918; Bartleby.com, 2000. ISBN 1-58734-102-6.

Brian Guenter. A system for simulating human facial expression. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *State-of-the-art in Computer Animation*. Springer-Verlag, 1989.

Fares Haddad, Jorge Galante, Edmund Chao, Sarah Muirhead-Allwood, Andrew Chippendale, and Marchi Maheson. *Interactive Hip CD-Rom*. Primal Pictures Inc., cd-rom edition, October 2001. ISBN 1902470370.

James K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988. ISSN 0097-8930.

Susan J. Hall. *Basic Biomechanics*. William C Brown Pub, 2nd edition, January 1995. ISBN 0815140770.

Koichi Hamada and Yoichi Hori. Octree-based approach to real-time collision-free path planning for robot manipulator. In *ACM´96-MIE*, pages 705–710, 1996.

M. Hauth, J. Groß;, and W. Straßer. Interactive physically based solid dynamics. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 17–27, Aire-la-Ville, Switzerland, 2003. Eurographics Association. ISBN 1-58113-659-5.

David Hawkins. *Biomechanics of Musculoskeletal Tissues (online text-book)*. UC-Davis, 2002.

Bruno Heidelberger, Matthias Teschner, and Markus H. Gross. Real-time volumetric intersections of deforming objects. In *VMV*, pages 461–468, 2003.

Bruno Heidelberger, Matthias Teschner, and Markus H. Gross. Detection of collisions and self-collisions using image-space techniques. In *WSCG*, pages 145–152, 2004.

M. Held. Bounding-volume hierarchy. Available online, Nov 2004. URL `http://www.cosy.sbg.ac.at/ held/projects/collision/bvt.html`.

M. Held, J. Klosowski, and J. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 205–210, 1995.

John Hewitt, Farshid Guilak, Richard Glisson, and T. Parker Vail. Regional material properties of the human hip joint capsule ligaments. *Journal of Orthopaedic Research*, 19(3):359–364, May 2001.

John D. Hewitt, Richard R. Glisson, Farshid Guilak, and T. Parker Vail. The mechanical properties of the human hip capsule ligaments. *The Journal of Arthroplasty*, 17 (1):82–89, January 2002.

Gérard Hégron and Bruno Arnaldi. *Computer Animation Motion and Deformation Control*. Number TN2 in Eurographics Technical Reports. Eurographics, September 1992.

Simone E. Hieber, J. H. Walther, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs. *Technology and Health Care*, 12:305–314, 2004.

Gentaro Hirota. An implicit finite element method for elastic solids in contact. In *Proceedings of Computer Animation*, 2001.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-701-4. doi: http://doi.acm.org/10.1145/218380.218414.

Qing hong Zhu, Yan Chen, and Arie Kaufman. Real-time biomechanically-based muscle volume deformation using FEM. *Computer Graphics Forum*, 17(3):??–??, September 1998. ISSN 0167-7055 (print), 1467-8659 (electronic).

Jocelyn Houle and Pierre Poulin. Simplification and real-time smooth transitions of articulated meshes. In *GRIN'01: No description on Graphics interface 2001*, pages 55–60, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society. ISBN 0-9688808-0-0.

Ed Huai hsin Chi and John Riedl. An operator interaction framework for visualization systems. In *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*, pages 63–70, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-9093-3.

P. Hubbard. Real-time collision detection and time-critical computing. In *Proceedings of the 1 st Workshop on Simulation and Interaction in Virtual Environments*, pages 92–96, July 1995.

Thomas C. Hudson, Ming C. Lin, Jonathan Cohen, Stefan Gottschalk, and Dinesh Manocha. V-collide: Accelerated collision detection for vrml. In *Proceedings of Second Symposium on the Virtual Reality Modeling Language (VRML'97)*, pages 119–125, New York, 1997. ACM Press.

M. Hughes, C. DiMattia, M. C. Lin, and D. Manocha. Efficient and accurate interference detection for polynomial deformation. In *CA '96: Proceedings of the Computer Animation*, page 155, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7588-8.

Dave Hutchinson, Martin Preston, and Terry Hewitt. Adaptive refinement for mass/spring simulations. In Ronan Boulic and Gérard Hégron, editors, *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 31–45, New York, NY, USA, 1996. Springer-Verlag New York, Inc. ISBN 3-211-82885-0.

Sofiane Sarni Ik Soo Lim and Daniel Thalmann. Colored visualization of shape differences between bones. In *CBMS 2003: Proceedings of the 16th IEEE Symposium* *on Computer-Based Medical Systems*, pages 273–278, Los Alamitos, CA, USA, 2003. IEEE Computer Society Press.

G. Imhäuser. Late results of imhauser's osteotomy for slipped capital femoral epiphysis. *Z Orthop Ihre Grenzgeb*, 115(5):716–725, 1977.

Instron. Instron corporation web site. Available online. URL `http://www.isntron.com`.

Horace Ho-Shing Ip, Maria S. W. Lam, Ken C. K. Law, and Sam C. S. Chan. Animation of hand motion from target posture images using an anatomy-based hierarchical model. *Computers & Graphics*, 25(1):121–133, 2001.

G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 131–140, New York, NY, USA, 2004. ACM Press. ISBN 3-905673-14-2. doi: http://doi.acm.org/10.1145/1028523.1028541.

Doug L. James and Dinesh K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.*, 22(1): 47–82, 2003. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/588272.588278.

Doug L. James and Dinesh K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.*, 23(3):393–398, 2004. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1015706.1015735.

Doug L. James and Dinesh K. Pai. Artdefo: accurate real time deformable objects. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 65–72, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: http://doi.acm.org/10.1145/311535.311542.

Johan Jansson and Joris S. M. Vergeest. A general mechanics model for systems of deformable solids. In *Proceedings International Symposium on Tools and Methods of Concurrent Engineering*, pages 361–373, 2000.

Johan Jansson and Joris S. M. Vergeest. A discrete mechanics model for deformable bodies. *Computer-Aided Design*, 34(12):913–928, 2002.

P. Jiménez, F. Thomas, and C. Torras. 3D Collision Detection: A Survey. *Computers and Graphics*, 25(2):269–285, Apr 2001.

A. Joukhadar, F. Garat, and C. Laugier. Parameter identification for dynamic simulation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 1928–1933. IEEE Press, 1997.

A. Joukhadar, A. Deguet, and C. Laugier. A collision model for rigid and deformable bodies. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 982–988. IEEE, May 1998.

A.; Laugier C. Joukhadar, A.; Wabbi. Fast contact localisation between deformable polyhedra in motion. In *Proceedings of Computer Animation '96*, pages 126–135, June 1996.

Faouzi Kallel and Jonathan Ophir. Tissue mechanical attributes imaging: Principles and methods. In *CBMS*, pages 147–159, 2000.

M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

III Kenneth E. Hoff, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: http://doi.acm.org/10.1145/311535.311567.

III Kenneth E. Hoff, Andrew Zaferakis, Ming Lin, and Dinesh Manocha. Fast and simple 2d geometric proximity queries using graphics hardware. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 145–148, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-292-1. doi: http://doi.acm.org/10.1145/364338.364383.

Jan Klein and Gabriel Zachmann. Adb-trees: Controlling the error of time-critical collision detection. In *VMV*, pages 37–45, 2003.

James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/2945.675649.

D. Knott and D. Pai. Cinder: Collision and interference detection in real-time using graphics hardware. In *Proceedings of Graphics Interface '03.*, 2003.

James U. Korein and Norman I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, 2(9): 71–74, 76–81, November 1982. ISSN 0272-1716.

R. K. Korhonen, M. S. Laasanen, J. Toyras, J. Rieppo, J. Hirvonen, H. J. Helminen, and J. S. Jurvelin. Comparison of the equilibrium response of articular cartilage in unconfined compression, confined compression and indentation. *Journal of Biomechanics*, 35(7):903–909, July 2002.

Ellen Kreighbaum and Katharine M. Barthels. *Biomechanics: a qualitative approach for studing human movement.* Macmillan Publishing Co., 3rd edition, 1990.

S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum*, 17(3):315–326, 1998.

K. H. E. Kroemer, H. J. Kroemer, and K. E. Kroemer-Elbert. *Engineering Physiology: Bases of Human Factors/Ergonomics.* Van Nostrand Reinhold-Wiley, New York, 2nd edition, 1990.

Paul G. Kry and Dinesh K. Pai. Continuous contact simulation for smooth surfaces. *ACM Trans. Graph.*, 22(1):106–129, 2003. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/588272.588280.

Tsuneya Kurihara and Natsuki Miyata. Modeling deformable human hands from medical images. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 355–363, New York, NY, USA, 2004. ACM Press. ISBN 3-905673-14-2. doi: http://doi.acm.org/10.1145/1028523.1028571.

Sven Kuschfeldt, Thomas Ertl, and Michael Holzner. Efficient visualization of physical and structural properties in crash-worthiness simulations (case study). In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 487–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press. ISBN 1-58113-011-2.

Benoît Lafleur, Nadia Magnenat-Thalmann, and Daniel Thalmann. Cloth animation with self-collision detection. In T. L. Kunii, editor, *Modeling in Computer Graphics*, pages 179–187, Berlin, 1991. springer-Verlag.

J. Lander. Skin them bones. *Game Developer*, pages 11–16, May 1998.

T. Larsson and T. Akenine-Möller. Collision detection for continuously deforming bodies. In *Proceedings of Eurographics*, 2001.

Thomas Larsson and Tomas Akenine-Möller. Efficient collision detection for models deformed by morphing. *The Visual Computer*, 19(2-3):164–174, 2003.

Yuencheng Lee, Demetri Terzopoulos, and Keith Walters. Realistic modeling for facial animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 55–62, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-701-4. doi: http://doi.acm.org/10.1145/218380.218407.

Robson Lemos, M. Epstein, W. Herzog, and B. Wyvill. Realistic skeletal muscle deformation using finite element analysis. In *Proceeding of the 14th Brazilian Symposium on Computer Graphics and Image Processing.*, pages 192–199. IEEE Computer Society, October 2001.

Haim Levkowitz and Gabor T. Herman. Color scales for image data. *IEEE Comput. Graph. Appl.*, 12(1):72–80, 1992. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.135886.

Tsai-Yen Li and Jin-Shin Chen. Incremental 3d collision detection with hierarchical data structures. In *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 139–144, New York, NY, USA, 1998. ACM Press. ISBN 1-58113-019-8. doi: http://doi.acm.org/10.1145/293701.293719.

Ik Soo Lim, Pablo de Heras Ciechomski, Sofiane Sarni, and Daniel Thalmann. Planar arrangement of high-dimensional biomedical data sets by isomap coordinates. In *CBMS*, pages 50–55, 2003.

M. Lin and J. Canny. Efficient collision detection for animation. In *Proceedings of the 3rd Eurographics Worshop on Animation and Simulation*, September 1992.

M. C. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *Proceedings of the 8th IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998.

Ming C. Lin and John F. Canny. A fast algorithm for incremental distance calculation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1008–1014, 1991.

Ming C. Lin and Dinesh Manocha. Fast interference detection between geometric models. *The Visual Computer*, 11(10):542–561, 1995.

Ming Chieh Lin. *Efficient Collision Detection for Animation and Robotics.* PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, December 1993. Chair-John F. Canny.

Laurent Moccozet Nadia Magnenat-Thalmann Lydia Yahia-Cherif, Benjamin Gilles. Individualized bone modelling from mri: Application to the human hip. In *CARS 2003: Proceeding of the 16th International Congress Computer Assisted Radiology and Surgery*, London, UK, 2003. Elsevier.

Anderson Maciel, Luciana Porcher Nedel, and Carla M. Dal Sasso Freitas. Anatomy-based joint models for virtual human skeletons. In *CA '02: Proceedings of the Computer Animation*, page 220, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1594-0.

Anderson Maciel, Ronan Boulic, and Daniel Thalmann. Towards a parameterization method for virtual soft-tissues based on properties of biological tissue. In *5th IFAC 2003 Symposium on Modelling and Control in Biomedical Systems (including Biological Systems)*, Melbourne, Australia, June 2003a. Elsevier.

Anderson Maciel, Ronan Boulic, and Daniel Thalmann. Deformable tissue parameterized by properties of real biological tissue. In XXX, editor, *International Symposium on Surgery Simulation and Soft Tissue Modeling (IS4TM)*, 2003b.

Anderson Maciel, Sofiane Sarni, Olivier Buchwalder, Ronan Boulic, and Daniel Thalmann. Multi-finger haptic rendering of deformable objects. In *Eurographics Symposium on Virtual Environments (In Cooperation with ACM Siggraph)*, Grenoble, 2004.

Anderson Maciel, Sofiane Sarni, Ronan Boulic, and Daniel Thalmann. Stress distribution visualization on pre- and post-operative virtual hip joint. In *Proceedings of the*

5th Annual Meeting of the International Society for Computer Assisted Orthopaedic Surgery (CAOS 2005), Helsinki, Finland, June 2005.

Nadia Magnenat-Thalmann and Daniel Thalmann. *Handbook of Virtual Humans.* John Wiley & Sons ltd, UK, 2004.

Walter Maurel and Daniel Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics*, 24(2):203–218, 2000.

Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics*, 24(4):29–38, August 1990. ISSN 0097-8930.

Stan Melax. Dynamic plane shifting bsp traversal. In *Graphics Interface*, pages 213–220, May 2000.

Ulisses T. Mello and Paulo Roma Cavalcanti. A point creation strategy for mesh generation using crystal lattices as templates. In *IMR*, pages 253–261, 2000.

Johannes Mezger, Stefan Kimmerle, and Olaf Etzmuß. Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG*, 11(2):322–329, 2003. ISSN 1213-6972.

Gavin S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 169–173, New York, NY, USA, 1988. ACM Press. ISBN 0-89791-275-6. doi: http://doi.acm.org/10.1145/54852.378508.

B. Mirtich and J. Canny. Impulse-based dynamic simulation. Technical report, Department of Computer Science, University of California at Berkeley, 1994.

Brian Mirtich. V-clip: fast and robust polyhedral collision detection. *ACM Trans. Graph.*, 17(3):177–208, 1998. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/285857.285860.

Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 181–188, 1995.

Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49–54, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-573-4. doi: http://doi.acm.org/10.1145/545261.545269.

Matthias Müller, Matthias Teschner, and Markus Gross. Physically-based simulation of objects represented by surface meshes. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 26–33, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2171-1. doi: http://dx.doi.org/10.1109/CGI.2004.75.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Trans. Graph. - Proceedings of ACM SIGGRAPH*, 24(3):471–478, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073216.

Tomas Möller. A fast triangle-triangle intersection test. *J. Graph. Tools*, 2(2):25–30, 1997. ISSN 1086-7651.

Laurent Moccozet and Nadia Magnenat Thalmann. Dirichlet free-form deformations and their application to hand simulation. In *CA '97: Proceedings of the Computer Animation*, page 93, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7984-0.

Gary Monheit and Norman I. Badler. A kinematic model of the human spine and torso. *IEEE Comput. Graph. Appl.*, 11(2):29–38, 1991. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.75588.

C. Monserrat, V. Hernández, M. Alcañiz, M. C. Juan, and V. Grau. Evaluation and study of a new deformable model based on boundary element methods. In H.U. Lemke, M.W. Vannier, K. Inamura, and A.G. Farman, editors, *Computer Assisted Radiology And Surgery (CARS'99)*, 1999.

Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988. ISSN 0097-8930.

Van C. Mow and W.C. Hayes. *Basic Orthopaedic Biomechanics*. Lippincott Williams and Wilkins, July 1997. ISBN 0397516843.

Van C. Mow, , and Anthony Ratcliffe. *Basic Orthopaedic Biomechanics*, chapter Structure and Function of Articular Cartilage and Meniscus, pages 113–177. Raven Press, New York, 1991.

MTS. Mts systems corporation web site. Available online. URL `http://www.mts.com`.

Alessandro Nava, Edoardo Mazza, Frederic Kleinermann, Nick J. Avis, and John McClure. Determination of the mechanical properties of soft human tissues through aspiration experiments. In *MICCAI (1)*, pages 222–229, 2003.

Bruce Naylor, John Amanatides, and William Thibault. Merging bsp trees yields polyhedral set operations. *SIGGRAPH Comput. Graph.*, 24(4):115–124, 1990. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/97880.97892.

Luciana Porcher Nedel and Daniel Thalmann. Modeling and deformation of the human body using an anatomically-based approach. In *CA '98: Proceedings of the Computer Animation*, page 34, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8541-7.

Laasanen MS Silvennoinen J Helminen HJ Jurvelin JS Nieminen MT, Toyras J. Prediction of biomechanical properties of articular cartilage with quantitative magnetic resonance imaging. *Journal of Biomechanics*, 37(3):321–328, March 2004.

Ian J. Palmer and Richard L. Grimsdale. Collision detection for animation using sphere-trees. *Comput. Graph. Forum*, 14(2):105–116, 1995.

J. S. Pang and J. C. Trinkle. Complementarity formulations and existence of solutions of dynamic multirigid-body contact problems with coulomb friction. In *Mathematical Programming*, volume 73, pages 199–226, 1996.

Frederic I. Parke and Keith Waters. *Computer facial animation*. A. K. Peters, Ltd., Natick, MA, USA, 1996. ISBN 1-56881-014-8.

Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. Quasi-rigid objects in contact. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–119, New York, NY, USA, 2004. ACM Press. ISBN 3-905673-14-2. doi: http://doi.acm.org/10.1145/1028523.1028539.

Cary B. Phillips and Norman I. Badler. Jack: a toolkit for manipulating articulated figures. In *UIST '88: Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software*, pages 221–229, New York, NY, USA, 1988. ACM Press. ISBN 0-89791-283-7. doi: http://doi.acm.org/10.1145/62402.62436.

Guillaume Picinbono, Hervé; Delingette, and Nicholas Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *Graph. Models*, 65(5):305–321, 2003. ISSN 1524-0703. doi: http://dx.doi.org/10.1016/S1524-0703(03)00045-6.

Les Piegl and Wayne Tiller. *The NURBS book*. Springer-Verlag, London, UK, 1995. ISBN 3-540-55069-0.

John C. Platt and Alan H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, August 1988. ISSN 0097-8930.

Madhav K. Ponamgi, Dinesh Manocha, and Ming C. Lin. Incremental algorithms for collision detection between polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):51–64, 1997. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/2945.582346.

Jovan Popovic', Steven M. Seitz, Michael Erdmann, Zoran Popovic', and Andrew Witkin. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer Graphics and interactive techniques*, pages 209–217. ACM Press/Addison-Wesley Publishing Co., 2000. ISBN 1-58113-208-5. doi: http://doi.acm.org/10.1145/344779.344880.

Emmanuel Promayon, P. Baconnier, and Claude Puech. Physically-based deformations constrained in displacements and volume. *Computer Graphics Forum*, 15(3):155–164, August 1996. ISSN 1067-7055.

X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proceedings of Graphics Interface (GI 1995)*, pages 147–154. Canadian Computer-Human Communications Society, 1995.

S. Quinlan. Efficient distance computation between non-convex objects. In *IEEE Intern. Conf. on Robotics and Automation*, pages 3324–3329. IEEE, 1994.

Z. Huang T. Molet J. Shen P. Kalra L. Moccozet R. Boulic, T. Capin and H. Werner. General purpose hierarchy. Technical report, Lig-EPFL, 1994. ESPRIT Project 6709.

Laks Raghupathi, Laurent Grisoni, Francois Faure, Damien Marchal, Marie-Paule Cani, and Christophe Chaillou. An intestinal surgery simulator: Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):708–718, November/December 2004.

Bernie Roehl. Specification for a standard vrml humanoid. Available online at the H-ANIM WG wep page, 1998. URL `http://h-anim.org/Specifications/H-Anim1.1/`.

S. H. Martin Roth, Markus H. Gross, Silvio Turello, and Friedrich R. Carls. A bernstein-bézier based approach to soft tissue simulation. *Computer Graphics Forum*, 17(3): 285–294, September 1998. ISSN 0167-7055 (print), 1467-8659 (electronic).

Sofiane Sarni, Anderson Maciel, Ronan Boulic, and Daniel Thalmann. Evaluation and visualization of stress and strain on soft biological tissues in contact. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI 2004)*, pages 255–262, 2004.

Sofiane Sarni, Anderson Maciel, Ronan Boulic, and Daniel Thalmann. Spreadsheet framework for visual exploration of biomedical datasets. In *IEEE Symposium on Computer Based Medical Systems*, Dublin, Ireland, 2005.

Alessandro Sarti, Roberto Gori, and Claudio Lamberti. A physically based model to simulate maxillo-facial surgery from 3d ct images. *Future Gener. Comput. Syst.*, 15(2):217–221, 1999. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/S0167-739X(98)00065-X.

Coenraad Frederik Scheepers. *Anatomy-based surface generation for articulated models of human figures.* PhD thesis, 1996. Adviser-Richard E. Parent.

Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 163–172, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: http://doi.acm.org/10.1145/258734.258827.

William Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics.* Prentice Hall, 2nd edition, November 1997. ISBN 0139546944.

Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, volume 20, pages 151–160, New York, NY, USA, 1986. ACM Press. ISBN 0-89791-196-2. doi: http://doi.acm.org/10.1145/15922.15903.

Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, New York, NY, USA, 1998. ACM Press. ISBN 0-89791-999-8. doi: http://doi.acm.org/10.1145/280814.280882.

Kenji Shimada and David C. Gossard. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, pages 409–419, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-672-7. doi: http://doi.acm.org/10.1145/218013.218095.

Mikio Shinya and Marie-Claire Forgue. Interference detection through rasterization. *The Journal of Visualization and Computer Animation*, 2(4):132–134, October–December 1991. ISSN 1049-8907.

Karan Singh and Evangelos Kokkevis. Skinning characters using surface oriented free-form deformations. In *Graphics Interface*, pages 35–42, May 2000.

Kristofer J. Stewart, Rohan H. Edmonds-Wilson, Richard A. Brand, and Thomas D. Brown. Spatial distribution of hip capsule structural and material properties. *Journal of Biomechanics*, 35(11):1491–1498, November 2002.

Nicholas Ayache Stéphane Cotin, Hervé Delingette. Efficient linear elastic models of soft tissues for real-time surgery simulation. Epidaure 3510, INRIA - Sophia Antipolis, October 1998.

Avneesh Sud, Miguel A. Otaduy, and Dinesh Manocha. DiFi: Fast 3D Distance Field Computation Using Graphics Hardware. *Computer Graphics Forum*, 23(3):557–566, 2004. ISSN 0167-7055.

J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 68–74, Aire-la-Ville, Switzerland, 2003. Eurographics Association. ISBN 1-58113-659-5.

D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation*, 1(2):73–80, December 1990. ISSN 1049-8907.

D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics (Proc. SIGGRAPH'87)*, 21(4):205–214, 1987.

Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, volume 22, pages 269–278, New York, NY, USA, 1988. ACM Press. ISBN 0-89791-275-6. doi: http://doi.acm.org/10.1145/54852.378522.

M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. In Cristophe Schlick and Werner Purgathofer, editors, *State of the Art Reports of the 25th Annual Conference of the European Association dor Computer Graphics (Eurographics 2004)*, STAR, pages 119–139. INRIA and the Eurographics Association, August 2004.

Matthias Teschner, Sabine Girod, and Bernd Girod. Direct computation of nonlinear soft-tissue deformation. In *Vision, Modeling, and Visualization (VMV-00)*, pages 383–390, 2000.

Matthias Teschner, Bruno Heidelberger, and Markus Gross Matthias Müller, Danat Pomeranets. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *Proceedings of Vision, Modeling, Visualization VMV'03*, pages 47–54, 2003.

Greg Turk. Interactive collision detection for molecular graphics. Technical report, Chapel Hill, NC, USA, 1990.

Russell Turner. Leman: A system for constructing and animating layered elastic characters. In *Proc. Computer Graphics International*, pages 185–203, 1995.

Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 91–96, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-701-4. doi: http://doi.acm.org/10.1145/218380.218419.

Davide Valtorta and Edoardo Mazza. Dynamic measurements of soft tissue viscoelastic properties with a torsional resonator device. In *MICCAI (2)*, pages 284–292, 2004.

Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *J. Graph. Tools*, 2(4):1–13, 1997. ISSN 1086-7651.

T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. *Computer Graphics Forum*, 20(3):260–267, September 2001.

Pascal Volino. *Mechanical Simulation of Deformable Surfaces for Animation of Synthetic Garments.* PhD thesis, Université de Genève, Geneva, November 1998. Adviser-Nadia Magnenat-Thalmann.

Pascal Volino and Nadia Magnenat-Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum*, 13(3):C/155–C/166, 1994. ISSN 0167-7055 (print), 1467-8659 (electronic).

Pascal Volino and Nadia Magnenat-Thalmann. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In D. Terzopoulos and D. Thalmann, editors, *Eurographics Workshop on Computer Animation and Simulation*, pages 55–65. Springer-Verlag, 1995.

Pascal Volino, Martin Courchesne, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 137–144, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-701-4. doi: http://doi.acm.org/10.1145/218380.218432.

Pascal Volino, Frederic Cordier, and Nadia Magnenat-Thalmann. From early virtual garment simulation to interactive fashion design. *Computer-Aided Design Journal (CAD journal)*, 37:593–608, March 2005.

Jurgen Weineck and Thomas J. Dekornfeld. *Functional Anatomy in Sports*. Year Book Medical Pub, 2nd edition, August 1990.

Jeffrey A. Weiss, John C. Gardiner, and Carlos Bonifasti-Lista. Ligament material behavior is nonlinear, viscoelastic, and rate-independent under shear loading. *Journal of Biomechanics*, 35(7), July .

Jane Wilhelms. Modeling animals with bones, muscles, and skin. Technical report, Santa Cruz, CA, USA, 1994.

Jane Wilhelms. Animals with anatomy. *IEEE Computer Graphics and Applications*, 17(3):22–30, 1997. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.586015.

Andrew Witkin, Kurt Fleischer, and Alan Barr. Energy constraints on parameterized models. In *Proceedings of ACM SIGGRAPH, Annual Conference Series*, pages 225–232, 1987.

Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum*, 20(3), 2001. ISSN 1067-7055.

D. Zeltzer and K. Sims. A figure editor and gait controller for task level animation. *Course Notes, Synthetic Actors: The Impact of Robotics and Artificial Intelligence on Computer Animation*, August 2 1988.

David Zeltzer. Representation of complex animated figures. In *Proceedings of Graphics Interface '82*, pages 205–211, May 1982.

Curriculum Vitae

Name	Anderson Maciel
Date of birth	22 January 1977 in Caxias do Sul, Brazil
Nationality	Brazilian and Italian
Mother tongue	Portuguese
Languages	Good knowledge of English, French and Italian

EDUCATION :

1994-1998

Bachelor (Computer Science)

University of Caxias do Sul (Brazil).

Thesis: Collision Detection between Rigid Polyhedra Pairs Applied to Asimov Project

1999-2001

Master of Science (Computer Science)

Federal University of Rio Grande do Sul, Porto Alegre (Brazil).

Thesis: Anatomy-based Articulation Model for Virtual Humans

2001-2002

Post Graduation (Virtual Reality and Multimodal Interaction)

École Polytechnique Fédérale de Lausanne (Switzerland)

Thesis: Deformable Tissue Parameterized by Properties of Real Biological Tissue

PROFESSIONAL ACTIVITIES :

1994-1998

Junior research assistant (scholarship by CNPq) at UCS (Brazil) Working in the ASIMOV project on 3D modeling of mechanical manipulators.

1998-1999

Software engineer at N&L Informática working on business management software development.

2000

Free-lancer for HSBC working on their home-banking system.

1999-2001

Research assistant (scholarship by CAPES) at UFRGS (Brazil) Working in the Virtual Patients project on anatomical joints modeling.

Systems analyst at Javabr working on web development in Java.

Teaching at Javabr working on Java training courses for third party companies.

PUBLICATIONS :

- A. Maciel, S. Sarni, R. Boulic, D. Thalmann, D. **Stress Distribution Visualization on Pre- and Post-Operative Virtual Hip Joint** In. Computer Aided Orthopedic Surgery 2005, Helsinki, Finland.
- S. Sarni, A. Maciel, R. Boulic, D. Thalmann. **Spreadsheet Framework for Visual Exploration of Biomedical Datasets** In. IEEE Symposium on Computer Based Medical Systems 2005, Dublin, Ireland.
- A. Maciel, S. Sarni, O. Buchwalder, R. Boulic, D. Thalmann. **Multi-Finger Haptic Rendering of Deformable Objects** In. Eurographics Symposium on Virtual Environments (In Cooperation with ACM Siggraph), 2004, Grenoble, France.
- S. Sarni, A. Maciel, R. Boulic, D. Thalmann. **Evaluation and Visualization of Stress and Strain on Soft Biological Tissues in Contact** In. International Conference on Shape Modeling and Applications, 2004, Genova, Italy. IEEE Computer Society Press.
- A. Maciel, R. Boulic, D. Thalmann. **Towards a Parameterization Method for Virtual Soft Tissues Based on Properties of Biological Tissue** In. 5th IFAC Symposium on Modelling and Control in Biomedical Systems (including Biological Systems), Melbourne, Australia. 2003 Elsevier Ltd.
- A. Maciel, R. Boulic, D. Thalmann. **Deformable Tissue Parameterized by Properties of Real Biological Tissue** In. IS4TM, 2003, Juan-les-Pins, France. Lecture Notes in Computer Science: Surgery Simulation and Soft Tissue Modeling. Springer-Verlag.
- A. Maciel, L. P. Nedel, C. M. D. S. Freitas. **Anatomy-Based Joint Models for Virtual Human Skeletons** In. Computer Animation 2002, Geneva, Switzerland. IEEE Computer Society.
- A. Maciel, G. A. Assis, R. V. Dorneles. **ASIMOV - Educational Framework for the Modeling, Programming and Simulation of Mechanical Manipulators** In. European Simulation Multi-conference, 1999, Warsaw, Poland. Erlangen SCS Publishing House, v. I.