UNIVERSITÀ DEGLI STUDI DI ROMA "SAPIENZA"

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

XX CICLO – 2008

LINKÖPINGS UNIVERSITET

LINKÖPING STUDIES IN SCIENCE AND TECHNOLOGY

A Computational Musco-Skeletal Model for Animating Virtual Faces

Marco Fratarcangeli

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

UNIVERSITÀ DEGLI STUDI DI ROMA "SAPIENZA"

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

XX CICLO - 2008

LINKÖPINGS UNIVERSITET

LINKÖPING STUDIES IN SCIENCE AND TECHNOLOGY

Marco Fratarcangeli

A Computational Musco-Skeletal Model for Animating Virtual Faces

Thesis Committee

Prof. Marco Schaerf (Advisor, Italy)
Prof. Robert Forchheimer (Advisor, Sweden)

Reviewers

Prof. Miguel Otaduy
Prof. Zhigang Deng

Copyright © 2008
by Marco Fratarcangeli

ISBN: 1234567890

AUTHOR'S ADDRESS:
Marco Fratarcangeli
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "Sapienza"
Via Ariosto 25, I-00185 Roma, Italy.
E-MAIL: frat@dis.uniroma1.it
WWW: http://www.dis.uniroma1.it/∼frat/

a Roberta
con amore

Acknowledgments

I am grateful to my advisors and mentors, Marco Schaerf and Robert Forchheimer. They always supported me and gave me the possibility to let grow my passion for Science and in particular, Computer Graphics. Through their example of honesty and fairness, they taught me far more than academic skills, and I will not forget it.

I want to thank Igor Pandzic, which always has supported and trusted in me.

Marco Tarini and Fabio Ganovelli for the passionate, creative and resourceful brainstormings and for the fun that we had together in various regions of the world.

Beside the scientific skills that I gathered, the most precious value that I earned during these fascinating years of study has been the people, the friends, that I met. I will always keep them in my hearth, wherever they are. These years have been an impressive and challenging journey, in which sacrifices, efforts and stress did never miss, but it has been a small price compared with the human and scientific experiences that I obtained back. I feel a lucky person because I had the chance to live it.

Lastly, I would like also to express my gratitude to the open source community, which share so much useful software and knowledge; without these, the development of this work would have been much more painful than it has been.

Abstract

Automatic synthesis of facial animation in Computer Graphics is a challenging task and although the problem is three decades old by now, there is still not a unified method to solve it. This is mainly due to the complex mathematical model required to reproduce the visual meanings of facial expressions coupled with the computational speed needed to run interactive applications.

In this thesis, there are two different proposed methods to address the problem of the animation of 3D realistic faces at interactive rate.

The first method is an integrated physically-based method which mimics the facial movements by reproducing the anatomical structure of a human head and the interaction among the bony structure, the facial muscles and the skin. Differently from previously proposed approaches in the literature, the muscles are organized in a layered, interweaving structure laying on the skull; their shape can be affected both by the simulation of active contraction and by the motion of the underlying anatomical parts. A design tool has been developed in order to assist the user in defining the muscles in a natural manner by sketching their shape directly on the already existing bones and other muscles. The dynamics of the face motion is computed through a position-based schema ensuring real-time performance, control and robustness. Experiments demonstrate that through this model it can be effectively synthesized realistic expressive facial animation on different input face models in real-time on consumer class platforms.

The second method for automatically achieving animation consists in a novel facial motion cloning technique. It is a purely geometric algorithm and it is able to transfer the motion from an animated source face to a different target face mesh, initially static, allowing to reuse facial motion from already animated virtual heads. Its robustness and flexibility are assessed over several input data sets.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Animation of virtual faces has been an active research field in Computer Graphics from more than 30 years. The applications of facial animation include such diverse fields as character animation for films and advertising [BL03; im209], computer games, video teleconferencing [Ost98; CPO00], user-interface agents and avatars [CSPE00; Pan02; PHIM04], and facial surgery planning [BNS96; TGG00]. In character animation, it is of critical importance to reproduce accurately the face motion because it is one of the prime source of emotional information.

The difficulty in reproducing a believable facial animation lies mainly in the complex and sophisticated structure of the human head. This makes hard to formulate a mathematical model able to represent the bio-mechanical inner workings of the face. An high accuracy and precision is required because, as humans, we are used to observe and decode facial expressions from the moment we born, and we are expert in easily detecting the smallest artifacts in a virtual facial animation. In the field of entertainment industry, believable animation is achieved by intensive manual labor of skilled artists and technicians. Beside the relevant costs involved, this solution is feasible only for specific geometric head models performing predefined motion.

This thesis presents two methods which addresses the problem of how to achieve believable facial simulation in automatic way and not hard-coded for a specific face. Interactivity is an additional crucial key requirement as it permits for manipulation of the face structure in real-time and faster validation and delivery of the results.

The first proposed approach is to create and visualize a computational model of the human face, in which the elements (particularly bone, skin and muscles), represent their anatomical counterpart and behave nearly like the real organs. This involves motion, deformation and contact between bio-tissues that are viscoelastic, non-linear, anisotropic and structurally heterogeneous, and which the mechanical properties vary according to its composition. This model is not devised to simulate the realistic behavior of the anatomical structure in a medical and bio-mechanical sense; as a matter of fact, the proposed facial model is useful for obtaining believable motion in a short time and at interactive rate, without reproducing the inner

mechanic characteristics of the human tissue, as it happens, for example, in non-interactive approaches [SNF05; SSRMF06] or computer aided surgery [GZDH04]. The main purpose of this work is to obtain a relatively simple model of the human face, governed by few parameters and thus easy to use, but still able to produce convincing and believable results.

The second technique presented here is a facial motion cloning which, given an already animated virtual face, transfers its motion to a static mesh, animating it.

1.2 Problem Description

In this thesis, I describe two methods to help in producing synthetic facial animation: a virtual anatomical model and a facial motion cloning technique.

1.2.1 Anatomical Model Input

The inputs of the virtual anatomical model are a cranial structure of a human head and the superficial skin; the shape of both of them is expressed as:

- an orientated, 2-manifold, triangulated mesh $\in \mathbb{R}^3$;
- a set of landmarks which specifies some meaningful facial features directly on the geometry; it is a subset of the MPEG-4 Facial Definition Points (FDPs) (see Sec. 3.5);

Both the skull and the skin geometries are defined as triangulated meshes $\in \mathbb{R}^3$. In general, a *mesh* is defined as any of the open spaces or interstices between the strands of a net that is formed by connecting nodes in a predefined manner [Liu02]. A mesh provides a certain relationship between the nodes, which is the basics of the formulation of the dynamics and rendering equations. In this case, these connectivity relationships are expressed by triangles. The mesh shall be *orientable*, in the sense that it is possible to set a coherent normal for each point of the surface, and *2-manifold*, meaning that an open interval around each vertex is homomorphic to a disk or semidisk [Ede01], or, in other words, each edge in the mesh is shared among two triangles at the most.

The addressed problems are:

- to produce a sequential process that allows the layering and attachment of muscles and fatty tissue to the underlying bony geometry and to the superficial skin, and
- to animate the whole anatomical structure in order to deform the skin and produce facial motion.

In particular the animation is obtained through the physical simulation of the rigid bony structure coupled with the deformable bodies which lays on the skull in a layered fashion. The uppermost layer is the skin mesh, which is deformed by the underlying tissues producing facial motion. The active contraction of a muscle shall be driven through a control input variable which affects its shape and the shape of the deformable tissues which lay, directly or indirectly, on the muscle. The shape of the deformable tissues shall change due to the muscular active contraction, surface tension and volume preservation characteristics. Optionally, if further meshes, representing the eyes, teeth and tongue are provided as input, they shall be animated through rigid transformation, i.e. rotations.

1.2.2 Facial Motion Cloning Input

The inputs of the facial motion cloning are an animated skin mesh (the source) and a different static skin mesh (the target). As for the anatomical model, both the source and the target mesh have associated a set of landmarks which specifies some particular features on the face. The motion of the source face is expressed as set of blend shapes. The addressed problem is to obtain the same set of blend shapes for the target mesh in order to animate it.

1.3 Methodology

1.3.1 Anatomical Model

In a real head, the facial muscles produce the forces influencing the motion of the muscle itself, of the other muscles, of the underlying bones (like the jaw), and eventually of the skin. This virtual anatomical model, instead, is devised to work in a different way; it is organized in *layers*, where each layer represents a particular part of the face and influences the layers placed above of it. The bottom layer is the bone layer, composed by the upper skull and the jaw; then, there are several layers of facial muscles and fatty tissue, and eventually there is the skin layer. The bone layer moves independently from the above layers (muscles and skin), through rigid transformations, and influences them. For example, if the jaw depresses and opens, all the muscles connected to it will deform accordingly. On top of the skull, there are several muscle layers; each muscle influences the muscles of the layers placed above of it and finally the skin.

The physical simulation is computed according to the Position-based Dynamics (PBD) introduced by Müller [MHHR06]. It is based on the popular numerical integration due to Verlet [Ver67] and widely used in the Computer Graphics context, see [Jak03; Por04] among the others. PBD allows to impose constraints of geometric nature on a deformable surface, like volume preservation of the whole surface or maintaining the distance among two nodes of the mesh during deformation. This permits for modeling the virtual anatomical structures without the use of internal and external forces, which simplifies the deformable model and produces unconditionally stable simulations.

The user is provided with an interactive tool which allows to sketch in a natural way the muscle shapes directly on the bony structure and to the already existing muscles. Once the musco-skeletal has been designed, it is fitted into the skin which is bound to it and animated.

The output of this method is a physically based model which is able to generate facial expressions on the input skin mesh.

1.3.2 Facial Motion Cloning

Facial Motion Cloning (FMC) is a purely geometric algorithm which can be divided in two steps:

1. a Radial Basis Function $G(P) \in \mathbb{R}^3$ is found such that it fits the neutral pose of the source target mesh into the target skin shape and a mapping is set among the target vertices and the source triangular faces;

2. the same function $G(P)$ is applied to all the blend shapes of the source; the triangular faces are deformed and the target vertices are displaced accordingly obtaining the corresponding blend shape for the target mesh.

1.4 Contributions

This work proposes solutions to the stated problems in the design and implementation of anatomically based modeling system for the human head at interactive rate and a novel motion cloning technique. Previous works in interactive facial animation focused mainly on defining an anatomical model using a template skin mesh on top of which the muscle shapes are defined. The anatomical model is then deformed to represent heads with a shape different from the template one [KHS01; ZPS03]. Thus, the motion of the skin meshes is rather similar to the template one. The reason of using a template mesh is probably due to the fact that the dynamics of these models are based on a network of springs and masses, whose parameters depends on the geometric dimensions of the skin. Choosing a fixed template skin allows to fix the parameters for the physical structure, like for example the spring's stiffness.

In the virtual anatomical model presented here, the muscles and the fatty tissues are defined over a skull geometry; the whole structure is then fitted into the target facial skin mesh. The physical simulation is carried out through a position based dynamics schema [MHHR06], which is based on geometrical constraints. This means that the overshooting problems arising when using spring models are overcame, physical parameters like stiffness assume normalized values $\in [0, ..., 1]$, the simulation is unconditionally stable, and the model can adapt easily to skin meshes with different dimensions, producing facial motion which depends from the features of the specific skin mesh.

The major contributions of this work can be listed as follows.

Layered Anatomical Structure: an anatomical physically simulated model, which is not restricted to a specific input skull or skin mesh. Facial muscles and fatty tissues are deployed in layers, which interact and slides on each other, forming an interwoven structure. Thus, a muscle deforms according to its own active contraction, which is controlled by a single variable, or due to the motion of the underlying anatomical structures. Any number of muscles is allowed. The skin is deformed by the muscle motion and rigid motion of the lower mandible. If teeth meshes are present in the input skin mesh, collision among lips and teeth is accounted for. If present, eyes are considered as well and animated though shifting of the texture mapping coordinates.

Position-based formulation: the deformable bodies in the model, namely, the muscles, the passive tissues (fat) and the skin, are physically modeled and simulated through the position based dynamics schema presented in [MHHR06]. This allows the simulation for being more robust and stabler than spring-mass networks, while conserving computational efficiency. A new triangular area preservation constraint is introduced as well.

Muscle Model: facial motion, unlike other body parts, is primarily dependent on muscles as opposed to skeletal structures; thus a particular emphasis has been devoted to the muscle model, both in the shape and deformation characteristics. The facial muscle models are represented as deformable surfaces. There are two kinds of muscles: linear

and circular. Both of these models do not represent the microscopic structure of real muscle, like the internal fiber arrangements, but instead mimic the macroscopic behavior of their real counterpart, like volume preservation which produces bulging when the muscle contracts or thinning when the muscle elongates.

Facial Motion Cloning: it is introduced a Facial Motion Cloning technique employed to transfer the motion of a virtual face (namely the source) to a mesh representing another face (the target), generally having a different geometry and connectivity.

1.5 Overview

The remainder of this dissertation is broken into the following chapters. In Chap. 2, it is presented an overview of previous work in the area of facial animation, physically based deformations, morphing and motion cloning. Chap. 3 provides knowledge about concepts used throughout this thesis. It provides a primer on the anatomical background related to facial animation and the physically-based simulation of deformable bodies in real time, in particular position-based dynamics. Further material is provided regarding morphing with Radial Basis Functions (RBF) and the MPEG-4 FBA standard. Chap. 4 is concerned with the geometrical modeling of the facial muscles and how initial shapes can be defined through the *ad-hoc* developed tool. Chap. 5 details the modeling and the animation of facial model, from the skull to the superficial skin, including accessory parts like teeth and eye,s and the process to assemble and animate the complete model. In Chap. 6, it is described an approach for Facial Motion Cloning and, finally, Chap. 7 concludes by providing an outlook of future research.

1.6 Publications and Collaborations

My earliest research has focused on a previous version of a physically-based anatomical model for virtual faces [FS04; Fra05]. This model is based on a mass-spring network, similar to [LTW95; ZPS02], with the muscle model introduced by Waters [Wat87]. Using this model, several morph targets (also known as blend shapes), can be automatically produced, each of them corresponds to a MPEG-4 Facial Animation Parameter (FAP). By interpolating these morph targets, it is possible to perform speech embedded in an MPEG-4 Face and Body Animation (FBA) data stream. The work described in [FF07] is in the model-based coding context, where the face expressions of a real human head are tracked and used to drive the speech of different virtual faces.

Physically based animation has been investigated in the context of different fields. In virtual robots simulation, a complete environment for a RoboCup soccer match is reproduced in [ZFI07], including four- and two-legged robots, with sensors, game field and ball. This framework has been useful for prototyping and testing artificial intelligence strategies. In the Computer Aided Surgery context, a method to simulate real time knot tying has been presented in [KPFG07], which can be used in an haptic environment to reproduce the suturing process.

The facial motion cloning technique in Chap. 6 has been presented in progressive iterations in [FS05a; FS05b; FSF07], while the muscle model and the anatomical face model in

Chap. 4 and 5 is unpublished material at the moment.

The research activity has been carried out within a joint Ph.D. programme among the University of Rome “La Sapienza”, in Italy and the Linköping Institute of Technology, in Sweden. I spent approximately two years in each institution.

Chapter 2

State of the Art

2.1 Facial Animation Techniques

Computer facial animation is defined as "*an area of computer graphics that encapsulates models and techniques for generating and animating images of the human head and face*" [wik09]. During the years, hundreds of researchers [PW96; NN98; HTMTB03; RP06; NDP*07] have devised many different techniques and methods to achieve this task. The first one producing 3D animations has been invented by Parke [Par72]. Basic expressions in 3D are defined at different moments in the animated sequence and intermediate frames are simply interpolated between two successive basic expressions. He digitized by hand several configurations of the same face model, each one representing a different expression or facial movement, and then linearly interpolated the 3D position of the vertexes in order to achieve motion. There are many synonyms to refer to these static fixed configurations of the face model: key poses, key frames, morph targets or blend shapes. Interpolation of key frames is still an extensively used technique in various fields of computer animation, not only facial animation, because it is simple to implement and computationally inexpensive on most of the hardware platforms. However, defining by hand all the key poses is a tedious task and requires skilled artists, in particular if the face model consists of thousands of vertexes. For lengthy animations, many key poses have to be defined and stored. Moreover, the key poses are valid only for the particular face model used. If other face models are employed, key poses have to be defined for each one of them.

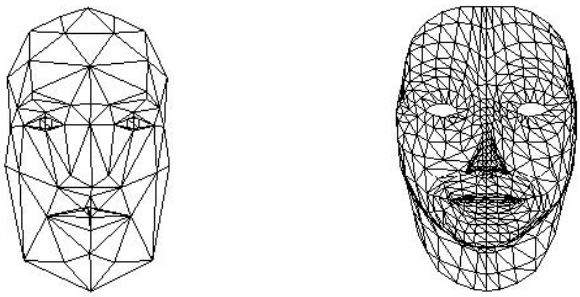
The main effort of research in this field has been to simplify and automatize the control and production of the animation for individual face models.

2.1.1 Famous Models

Numerous facial models have been created over the nearly three decades of research on facial animation. This subsection list models that have had the most impact in the area. The impact is judged on the contribution made by the model, how often the model is referenced and how often it is incorporated into other systems.

Candide

Rydfalk [Ryd87] describes a system called *Candide* (Fig. 2.1a), designed at the University of Linköping in Sweden to display a human face quickly. Candide was designed when graphics hardware was slow and the number of polygons was very important. When designing the system, Rydfalk had four constraints: use of triangles, less than 100 elements, static realism and dynamic realism. He defines static realism as when the motionless face looks good and dynamic realism as when the animated motion looks good. FACS AUs are used to define the animation. The final geometry has less than 100 triangles and 75 vertexes resulting in a very rough model that lacks adequate complexity in the cheeks and lips. With so few triangles, it is very difficult to make the model geometrically look like a particular person. This model is popular among vision researchers (e.g., [LRF93; LLF94; AD03]), for applications such as tracking, model based communication, and compression where a low polygon count model is sufficient and desired. The model has been then updated to be used with the standard MPEG-4 [Ahl01].



(a) (b)

Fig. 2.1: Face models: *(a)*. Candide, *(b)* Waters.

Waters

Waters [Wat87] presented a parameterized muscle model for creating facial animation. The muscle model can be adapted to any physical model, in particular he used a polygonal mesh. The parameterized muscles are given zones of influence and the nodes of the facial model are displaced within these zones using a cosine fallout function. Work using the muscle system by defining ten muscles based on FACS is presented. This model has been the basis for the first physically-based methods for facial animation, in particular [TW90; LTW93; LTW95]. Note how the edges of the mesh are mostly aligned with the Langer's lines (Sec. 3.1.4); this is intentionally done to produce more realistic deformations of the skin and, nowadays, almost every face mesh used in animation is modeled according to this principle (for instance, see Fig. 2.2, from [nl209]).

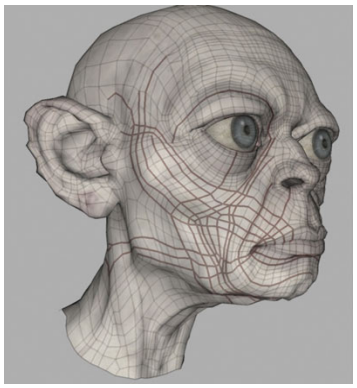


Fig. 2.2: The edges direction of facial meshes are placed along Langer's lines.

2.1.2 Image-based Techniques

Image-based methods are the preferred approach to achieve facial animation in the movie industry (e.g., [BL03]), since their first use [BL85]. In a strictly 2D approach, images are used as key frames and they are morphed between to create animation. The other basic method is to blend the texture on a 3D model using morphing, while also interpolating the 3D vertex configuration between key frames (or blend shapes) [DM96]. This gives motion that appears more realistic because the small deformations that occur in the skin, such as wrinkles and bulges, will appear since they are represented in the new texture. The major drawbacks are that not only must textures for all possible articulations, and all possible combinations of articulations be acquired (and this may not be possible), but they must also be stored. Pighin et al. [PHL*98] describe a system that uses image processing techniques to extract the geometry and texture information. To achieve animation, they create a face mesh and texture for each key frame and interpolate them. The geometry is linearly interpolated while the textures are blended and warped properly.

One of the drawbacks when using blend shapes is the so called *interference*, that is, individual blend shapes often have overlapping (competing or reinforcing) effects, e.g., a blend shape controlling the eyebrow and another one controlling the eyelid. In [LMDN05] the problem is addressed from a linear algebra point of view, improving the orthogonality among blend shapes under the supervision of the animator. Such a method has been used in the movie *Lords of the Rings*, where facial animation was achieved through the use of 946 blend shapes for the "Gollum" character. This rather big dataset has been reduced to 46 in the approach of Deng [DCFN06], which describe a semi-automatic method of cross-mapping of facial data, acquired by motion capture, to pre-designed blend shapes, while maintaining low the weight of the interference.

Face transfer techniques [NN01; Pan03; PKC*06; FSF07] reused existing facial animation by transferring source facial animation to target face models with little manual intervention, but these techniques require a high-quality animated source face. Briefly, the main concept is to transfer the motion from a *source* face to a static *target* face, making this latter animatable. Learning-based methods rely on a dataset of 3D scans of different facial expressions and mouth shapes to build a morphable model [BV99; BBPV03], that is, a vector space of 3D expressions. Difference vectors, such as a smile-vector, can be added to new

individual faces. Unlike physical models, it is treated the appearance of expressions, rather than simulating the muscle forces and tissue properties that cause the surface deformations.

Escher *et al.* [EMT97; EPT98], developed a cloning method based on Dirichlet Free Form Deformation (FFD) and applied it to the synthesis of a virtual face in order to obtain a virtual avatar of a real human head. In FFD algorithms, the deformation is controlled by a a few external points. To achieve volume morphing between the source and the target face meshes, the control points are usually difficult to define and not very intuitive for manipulation. In Expression Cloning developed by Noh [NN01], the movements of a source face are expressed as motion vectors applied to the mesh vertexes. The source mesh is morphed, through the use of RBF volume morphing and neural networks, to match the shape of the target mesh. The motion vectors are transferred from source model vertexes to the corresponding target model vertexes. The magnitude and direction of the transferred motion vectors are properly adjusted to account for the local shape of the model. The Facial Motion Cloning approach developed by Pandzic [Pan03] relies on the fact that the movements of the source face are coded as morph targets. In Pandzic's approach, each morph target is described as the relative movement of each vertex with respect to its position in the neutral face. Facial cloning is obtained by computing the difference of 3D vertex positions between the source morph targets and the neutral source face. The facial motion is then added to the vertex positions of the target face, resulting into the animated target face. The key positions represented by morph targets are expressed by the MPEG-4 Facial Animation Parameters (FAPs) (see Sec. 3.5). Each morph target corresponds to a particular value of one FAP. By interpolating the different morph targets frame-by-frame, animation of the source face is achieved.

2.1.3 Performance-driven Methods

Performance animation is capturing the motion of some performance, and applying it to a facial model to create animation with many different possible methods.

One of the most diffuse techniques is the combined exploitation of marker-based motion capture data with face geometry [Wil90]. Another approach is the use of structured light-systems, which are less precise than the marker based ones, however does not use any marker, thus they are less invasive, and are able to capture dynamic motion like depicted in [ZSCS04; WHL*04; ZH*06].

The combined use of image processing and vision techniques can extract motion data from video. Motion from a sequence of images is extracted by tracking feature points on the face across frames. Sometimes additional markers or makeup is used. Terzopoulos and Waters [TW90] use snakes along with make-up to track facial features, which they tie to muscle actuation. The estimation of muscle activation for physics-based approaches is also used in the most recent and advanced models [SNF05].

In [BBA*07], the facial geometry is acquired as a static scan including reflectance data at the highest possible quality. Then, the expression wrinkles are tracked during movements through the use of a traditional marker-based facial motion-capture system composed by two synchronized video cameras. These data are used to synthesize motion deforming the high-resolution geometry using a linear shell-based mesh-deformation method.

The main difficulties of motion capture are the quality of the data which may include vibration as well as the retargeting of the geometry of the points. Research objectives in this area are to improve the way the motion data is captured while reducing the manual effort of the final user, through as less as possible use of invasive methods and expensive tracking

hardware.

2.1.4 Physically-based Methods

Physically-based approaches animate faces by simulating the influence of muscle contraction onto the skin surface, adn deforming it accordingly. The general approach is to build a model respecting the anatomical structure of the human head. By doing this, the produced motion is very realistic. The drawback is that these methods involve a massive use of computational resources to advance the simulation. Thus, they are not suitable for interactive applications on modestly equipped platforms. Physically-based methods have been used in the simulation of the whole human body and animals as well [WVG97; NTH01; NTHF02; TBHF03].

The first work using this approach is due to Platt and Badler [PB81]. Waters [Wat87] presented a parametric muscle model which simulate the behavior of linear and sphincter facial muscles. In [LTW93; LTW95] a three-dimensional model is automatically built of a general human face adapting a predetermined triangle mesh using the data obtained through a 3D laser scanner. The resulting face model consists of three layers representing the muscle layer, dermis and epidermis. The elastic properties of the skin are simulated using a mass-spring system. The simulation is driven by a second-order Runge-Kutta scheme as a compromise between stability, accuracy, and speed requirements. Additional volume preservation constraints model the incompressibility of the ground substance. To this end, local restoration forces are computed to minimize any deviation in volume of the elements. An alternative integration scheme for the stiff mass-spring system is proposed by Baraff and Witkin [BW98]. They provide the theory for a stable implicit integration using very large time steps. Bro-Nielsen and Cotin [BNS96] use linearized finite elements for surgery simulation. They achieve significant speedup by simulating only the visible surface nodes, the so-called condensation.

An off-line and realistic solution is proposed by Teran *et al.* [TBHF03; ITF04; SNF05; TSIF05; TSB*05]. They build a Finite Element model of the flesh from the visible human dataset. Beside the bony part, the deformable part of the face (facial tissue, muscles and cartilages), are modeled in the form of a tetrahedral mesh with about 850 thousand tetrahedra out of which 370 thousand, in the front part of the face, are simulated. Since the muscles are so thin that they are not captured by the tetrahedral representation, their action is simulated directly like a field of forces acting on the internal structure of the facial mesh. This leads to impressively realistic results at the cost of huge computational resources (8 minutes per frame on a single Xeon 3.06Ghz CPU), beside the effort to build the whole model (it is reported the employment of 5 graduate students for several months).

An anatomically based face model running at interactive rate is provided by Kähler *et al.* [Käh03; KHS01; KHYS02; KHS03; KÖ7]. He devised a computational representation of the anatomical structure of a real head, with skull, muscle and skin in order to model a general template virtual face. The inputs for the model are a pre-defined geometrical mesh built *ad-hoc* for this purpose which represents the superficial skin, together with the underlying skull. Then, the muscle map can be interactively designed by the user through an editing tool. Finally, the different anatomical parts are connected together resulting in the final template face model. To represent different humans, the template model must adapt its shape and appearance. The shape is obtained by fitting the template model into a set of scattered data points obtained by a laser scan of a real person. The appearance, that is the texture of the skin, the eyes and teeth are obtained with the methods in Tarini *et al.* [TYHS02].

Chapter 3

Background Knowledge

3.1 Anatomy of the Human Head

3.1.1 Skull

This section outlines and identifies the cranial substructures necessary to build the ideal human head. The components are identified in Fig. 3.1, image from [PP09]. A detailed explanation follows.

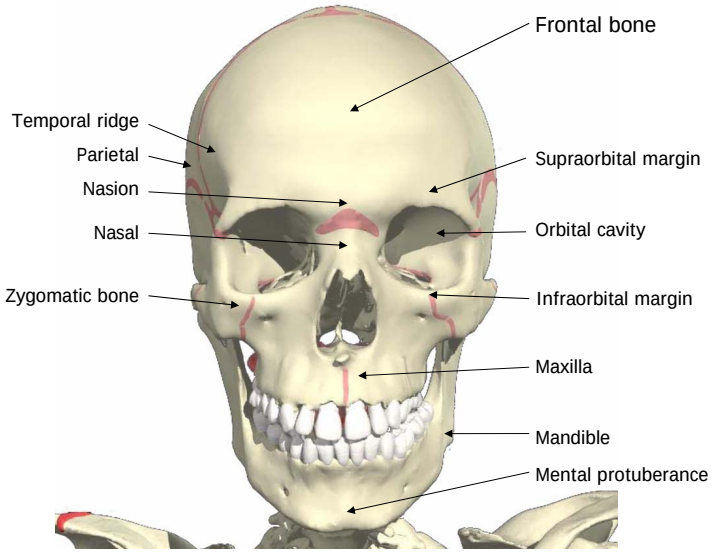


Fig. 3.1: Major features of the human skull.

The *frontal bone* forms the structure of the forehead, and is slightly curved toward the back of the head and the sides. The frontal bone is rather thick and terminates at the brow,

just above the nose and at the temporal ridge on the sides. The *temporal ridge* runs along the side of the upper skull. Its subtle on the skull and nearly unperceivable on the finished head, but is responsible for creating the square-shaped appearance of the upper skull.

Derived from the Latin *parietalis* meaning "belonging to the wall", the parietal bone makes up the side of the head. Its a smooth curved bone that extends outward until it lines up with the back of the jawbone. Along the side of the head, the parietal bone is between the frontal bone and the occipital bone on the back of the head.

The *supraorbital margin* defines one of the most distinctive facial features as it creates the ridge above the eyes. The supraorbital margin is the bone directly under the eyebrows creating the upper portion of the eye sockets. When animating facial expressions, the skin moves over the supraorbital margin. In particular, when the eyebrows raise, the most of them across the middle is moved, but the sides stay locked. That is because they are resting on the supraorbital margin. The tissue just above the upper eyelid is pulled upward. Clearly, it is not the supraorbital margin moving, but rather the sagging skin tissue that surrounds it. When the eyebrows are raised, this tissue is pulled over the supraorbital margin.

The *nasion* is the area where the frontal bone meets the *nasal bone*. Basically, its the little dip at the top of the nose, just before the brow ridge. The nasal bone is comprised of two small oblong bones, side by side, starting at the nasion, and continuing down the face essentially forming the bridge of the nose. The point where the nasal bone terminates usually creates a small bump in the nose. The cartilage that forms the tip of the nose is connected to the nasal bone. A common mistake in facial animation is to move the tip of the nose during facial expression. While subtle movement in the nose does occur, its due to the skin covering being stretched. For the most part, the tip nose is fixed and stable.

The *orbital cavity* is the large hole where the eye is located. It is much larger than the actual eye, which sits rather high in the orbital cavity. The *infraorbital margin* is the lower portion of the orbital cavity and the upper portion of the cheekbone. It creates the ridge under the eye and is directly responsible for creating bags under the eyes. It supports the excess fluids and tissue to create the bags. When the cheeks are raised, the tissue rides up and over the infraorbital margin, collecting under the lower eyelid, forcing it to puff up. Since the muscle tissue ca not move over the infraorbital margin, it collects under it and creates the puffy cheeks. This is particularly noticeable during the smile expression or when winking.

The *zygomatic bone* is the cheekbone that lies directly under the infraorbital margin. The zygomatic bone is obscured by the infraorbital margin from the front view, but is visible on the outer edge where it protrudes from the face, creating the common cheekbone. While smiling, the tissue collects in front of the zygomatic bone, which pushes it outward to create puffy cheeks.

The *maxilla* is the upper jawbone, directly under the nose. The maxilla is stationary and holds the gums and upper row of teeth. The *mandible* consists of the complete lower jawbone and defines the contour of the face. It is the largest facial bone and is the only movable bone on the skull.

During opening of the mouth, the lower jaw rotates around a horizontal axis passing through the mandibular condyles, which are located at the rear extreme of the jawbone and are free to slide a short distance along the temporal bone of the cranium, forming the so-called *temporomandibular joint*. There are a variety of movements permitted by this articulation. The range of actions the mandible is capable of is:

1. *depression*: opening of the mouth from rest; this hinge action occurs up to 15-18 degrees away from the rest position;

2. *elevation*: closing of the mouth to rest;

3. *protrusion*: carrying the mandible forwards from rest;

4. *retraction*: carrying the mandible back to rest;

5. *small amount of lateral movement*: side-to-side movement from the rest position.

3.1.2 Anatomy and Physiology of Muscles

Skeletal muscles are voluntary muscles which contract in order to move the bones they connect. Located throughout the body, these muscles form a layer between the bones of the skeleton and subcutaneous fatty tissue.

Skeletal muscles consist of elongated muscle fibers and fibrous connective tissue which anchors the muscles to the underlying skeleton. The composition of muscle fibers in a muscle determines the potential strength of muscle contraction, its direction and the possible range of motion due to contraction. This fiber arrangement is known as the muscle pennation.

Muscle fibers are anchored to bone or to other muscles through tendons or tough, flat fascial sheets called aponeuroses. Muscle fibers generally attach to aponeurosis or tendon in parallel arrays. Tendon and aponeurosis have elastic and dissipative properties, but unlike muscle, tendon has no active elements so its elasticity is purely passive. Muscle fibers do not attach directly to bone, but apply forces to the skeleton via aponeurosis and tendon. Tendon must be sufficiently stiff to transmit muscle forces to bone without undergoing significant deformation itself. As both muscle and tendon work closely together to create a functional unit of force generation and transmission, they are often referred as a collective bio-mechanical structure, a *musculotendon* unit. The tendon portion of the musculotendon unit attaches to bone is called the *origin* and those connected to soft tissue the *insertion*.

The force vector generated by a pennate array of muscle fibers has a component that lies parallel along the line of action of the tendon that contributes to force and motion at the origin or insertion sites. There is also a perpendicular component which causes muscle fibers to push against each other, against other soft tissue or against bone and it leads to changes in the shape of the muscle during contraction, for example belly bulges. These changes in shape may lead to a change in the direction of the line of action with respect to the site of origin or insertion during contraction.

Anatomists distinguish between two types of muscle contraction:

- *isotonic* contraction, where the length of the muscle changes while the volume remains constant and the muscle produces movement;

- *isometric* contraction, where the muscle contracts or tenses without producing movement or undergoing a change in length;

Often a muscle or the muscle-tendon unit spans more than one joint. Contraction of such a muscle will produce rotation of all of the spanned joints.

Facial muscles differ from most other skeletal muscles in several significant ways. In particular, some mimic muscles are attached to soft tissue or blend with other muscles. Then, most of the facial muscles have tendons which are considerably shorter than the length of the muscle fibers.

The pennation patterns of facial muscles can be reduced to three main types already mentioned above: linear, sphincter and sheet. The fibers may have a complex geometrical arrangement with sheets of fibers oriented in different planes. During contraction local rotations and deformations of sets of fibers will occur in each plane, leading to complex changes in shape. With such complex arrangements of aponeuroses, muscle fiber lengths may vary from one region of the muscle to another in addition to being oriented in different planes.

3.1.3 Facial Muscles

There are many muscles located in a human face, the main part of them have supporting functions, while eleven are instigating muscles and are responsible for facial animation. The facial muscles are divided into four main muscle masses: jaw muscles, mouth muscles, eye muscles and brow muscles. These muscles are illustrated in Fig. 3.2. Since the Corrugator supercilii is mostly hidden by the Frontalis, it is shown in the dedicated Fig. 3.3. Both of the images are from [PP09].

Jaw and Mouth Muscles

The lower cranial muscles can be categorized into jaw muscles and mouth muscles. The jaw muscles control the jawbone while the mouth muscles control the lip and the chin. The jaw muscles include one major muscle and several supporting muscles. The main muscle in the jaw group is the *masseter*, which is used in all the actions involving the elevation and the depression of the mandible, like clench the teeth, biting and chewing. The masseter arises from the anterior two thirds of the lower border of the zygomatic arch and pass downward and backward into the lateral part of the mandible.

The mouth muscle mass contains the largest number of muscles and it is used extensively during lip synching animation. The *levator labii superioris* arises from the maxilla at the inferior margin of the orbit, above the infraorbital margin, and inserts into the skin overlying the lateral side of the upper lip. The primary function of the levator labii superioris muscle is to elevate the upper lip, like in disgust or disdain expressions.

The *zygomaticus major* takes origin from the lateral surface of the zygomatic bone and it passes obliquely downwards to the corner of the mouth where it mingles with the orbicularis oris muscle. Its main action is to pull the corner of the upper lip upwards and outwards, as in smiling and laughing.

The *depressor anguli oris* arises from an extensive area around the external oblique line of the mandible and pass upwards to the corner of the upper lip. This muscle depresses the corner of the mouth and it is crucial for creating expressions like sadness or frowning.

The *depressor labii inferioris* muscle depresses the lower lip and draws it laterally. It arises from he mandible just in front of the mental protuberance and pass upwards and medially to converge with the orbicularis oris muscle in the lower lip. It is associated with the expressions like doubt or fear.

The *risorius* muscle is usually poorly developed. It does not originates from a bone but arises from a connective tissue in correspondence with the masseter muscle. It runs horizontally across the face and inserts into the corner of the mouth. The risorius pulls the corner of the mouth laterlaly as in grinning.

The *orbicularis oris* is the last of the major muscles in the mouth muscle mass. It is a sphincter around the lips like a ring encompassing them. The orbicularis oris muscle is very

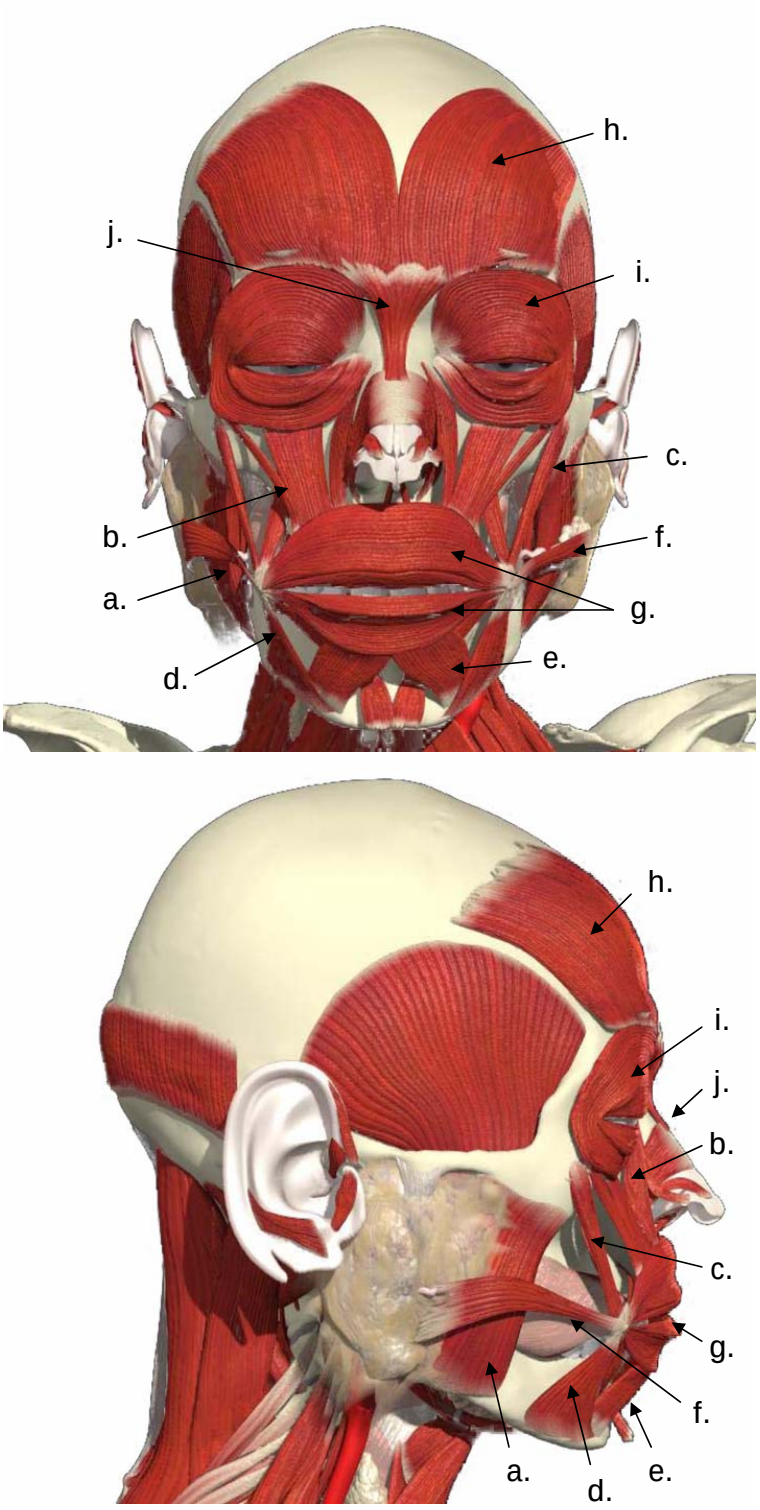


Fig. 3.2: Front and side view of the main facial muscles involved in facial animation. (a). Masseter; (b). Levator labii superioris; (c). Zygomaticus major; (d). Depressor anguli oris; (e) Depressor labii inferioris; (f). Risorius; (g). Orbicularis oris; (h). Frontalis; (i). Orbicularis oculi; (j). Procerus.

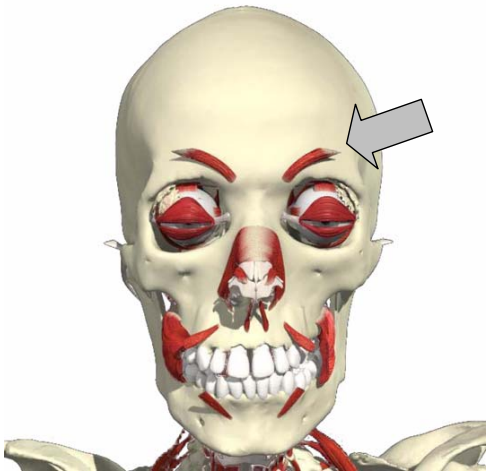


Fig. 3.3: The Corrugator supercilii muscle.

complex muscles and it is capable of various movements, including closure, protrusion and pursuing of the lips.

Brow and Eye Muscles

The *corrugator supercilii* muscle originates from the medial end of the supraorbital margin on the frontal bone and insert into the skin of the middle of the eyebrow. This muscle is used to compress the skin between the eyebrows, which are drawn downwards and inwards, and is used to create expressions such as anger, intense concentration and disgust.

The *orbicularis oculi* muscle may be regarded as a sphincter of the eyelids. The palpebral part is involved in closing the eyelids without effort, i.e. involuntary closure during blinking, and also voluntary movements like wink or squint.

The *procerus* arises from the nasal bone and the lateral nasal cartilage. Its fibers pass upwards to inser into the skin overlying the bridge of the nose. It produces transverse wrinkles over the pbridge of the nose.

Frontalis bellies covers the frontal part of the scalp and have no bony attachments but they blend with the surrounding muscles, in particular with the corrugator supercilii and the orbicularis oculi. The frontal bellies raise the eyebrows and the skin of over the root of the nose, used in movements such as glancing upwards and expressions of surprise and fright. Acting from below the frontal parts also draw the scalp forwards to produce wrinkles on the forehead.

3.1.4 Anatomy and Biomechanics of the Facial Tissue

In *anatomy*, soft tissue is a collective term for almost all structures, which can be named soft in comparison to bones. A basic structural element of facial and other soft tissues is collagen, which amounts up to 75% of dry weight [Fun93]. The remaining weight is shared between

elastin, actin, reticulin and other polymeric proteins. These biopolymers are organized in hierarchical bundles of fibers arranged in a more or less parallel fashion.

The direction of the collageneous bundles correspond closely to the creases on the skin surface and, under tension, define the shape and the amount of the wrinkles. These are the Langer's lines (Fig. 3.4, from [BFMM07]), or cleavage lines, named after the Austrian anatomist Karl Langer (1819-1887), who discovered them in 1861. He showed that the orientation of these lines coincide with the dominant axis of mechanical tension in the skin [Gra08; BFMM07].

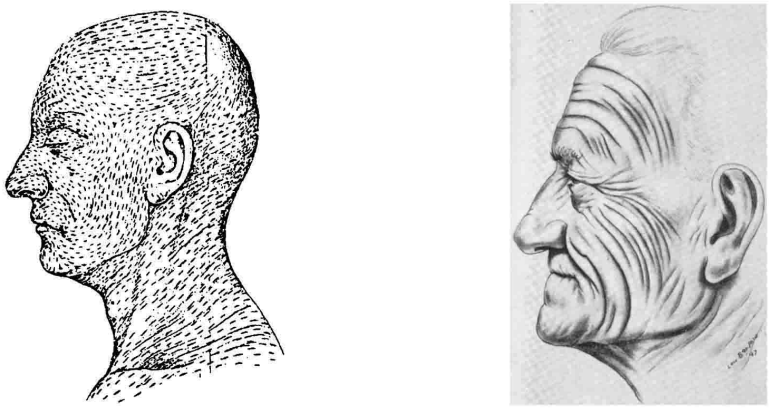


Fig. 3.4: *Left*. Langer's lines for the face and neck area, placed along the collageneous bundles in the skin. *Right*. A composite drawing of the normal wrinkle pattern of the face. Wrinkles appear in the normal direction of the collageneous bundles in the skin.

The facial tissue consists of the several anatomically distinct layers: the skin, subcutis (also named hypodermis), fascia and muscles. Fig. 3.5 shows a schematic cross-section of facial tissue. Skin is subdivided in two main layers : the thin epidermis and the thicker dermis. The dermis layer contains disordered collagen and elastin fibers embedded in the gelatinous ground substance. The thickness of the skin varies between $1,5$mm and 4mm. The dermis layer of the skin is continuously connected by collagen fibers to a subcutaneous fatty tissue, called the hypodermis. In turn, the hypodermis is connected to the fibrous fascia layer, which surrounds the muscle bundles. The contact between the lower subcutaneous tissue layer and the muscle fascia is flexible, which appears as a kind of sliding between the skin and other internal soft tissues.

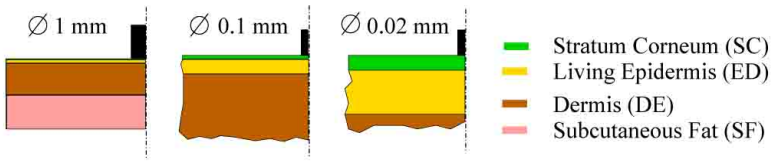


Fig. 3.5: Schematic view of cross-section of human skin, showing 4 layers at various scales.

Biomechanics combines the field of engineering mechanics with the fields of biology and physiology and is concerned with the analysis of mechanical principles of the human body. While studying the living tissue biomechanics, the common practice has always been to utilize the engineering methods and models known from "classic" material science. However, the living tissues have properties that make them very different from normal engineering materials. Numerous experimental and theoretical studies in the field of tissue biomechanics have been carried out in recent years [Fun93; ON99; Hen01; MWMTT03]. Summarizing the facts observed in different experiments with different tissue types, soft tissues generally exhibit *non-homogeneous*, *anisotropic*, *quasi-incompressible*, *non-linear* material properties.

Non-homogeneity, anisotropy. Soft tissues are multi-composite materials containing cells, intracellular matrix, fibrous and other microscopical structures. This means that the mechanical properties of living tissues vary from point to point within the tissue. The dependence on coordinates along the same spatial direction is called *non-homogeneity*. If a material property depends on the direction, such material is called *anisotropic*. Facial tissue is both non-homogeneous and anisotropic. However, there are practically no quantitative data about these properties and thus their importance for modeling of relatively thin facial tissue is uncertain.

Quasi-incompressible material. A material is called incompressible if its volume remains unchanged by the deformation. Soft tissue is a composite material that consists of both incompressible and compressible ingredients. Tissues with high proportion of water, for instance the brain or water-rich parenchymal organs are usually modeled as incompressible materials, while tissues with low water proportion are assumed quasi-incompressible.

Non-linearity. Although the elastin and collagen fibers are considered linear elastic, the stress-strain curve of skin for uniaxial tension is nonlinear due to the non-uniformity of its structure, as can be seen in Fig. 3.6.

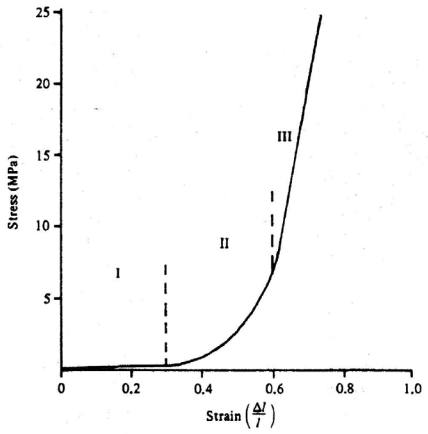


Fig. 3.6: Stress-strain diagram for skin showing the different stages.

The curve can be divided in four stages. In the first stage the contribution of response of the undulated collagen fibers can be neglected; elastin is responsible for the skin stretching, and the stress-strain relation is approximately linear. In the second phase, a gradual straightening of an increasing fraction of the collagen fibers causes an increasing stiffness. In the

third phase all collagen fibers are straight and the stress-strain relation becomes linear again. Beyond the third phase, yielding and rupture of the fibers occur.

3.2 Position-based Dynamics

The simulation techniques to produce physically based animation are often based on the Newton's second law of motion $\mathbf{a}(t) = \mathbf{f}(t)/m$. A set of ordinary or partial differential equations define the force fields applied to the different elements which represents the system. Then, a numerical integration schema is employed to integrate the acceleration to obtain the velocity and the velocity to obtain the position of the element in a given time step. For example, the classic system of differential equations

$$\mathbf{v}(t+\Delta t) = \mathbf{v}_0(t) + \frac{\mathbf{f}(t)}{m}dt \tag{3.1}$$

$$\mathbf{x}(t+\Delta t) = \mathbf{x}_0(t) + \mathbf{v}(t)\,dt \tag{3.2}$$

is resolved through the simple explicit Euler numerical integration schema:

$$\mathbf{v}_n = \mathbf{v}_{n-1} + \frac{\mathbf{f}_n}{m}\Delta t \tag{3.3}$$

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{v}_n \Delta t \tag{3.4}$$

Together, the differential equations and the numerical integration schema constitutes the *physical model*. A physical model is assessed according to its *generality*, *accuracy* and *efficiency*. Generality expresses the validity of the model for different physical phenomena and different conditions; accuracy is how much the simulated quantities are equal to the real ones; efficiency is computation time scaled to time requirements (hard real-time, interactive, off-line). Usually, making a model general and accurate reduces efficiency and vice versa.

Thus, the efficiency with which this system provides the results depends on two main factors: the complexity of the mathematical model of differential equations and the integration schema employed to solve it. The mathematical model can be simplified and relaxed through assumptions which depends on the particular simulated system and the requirements of the animation. For example in a video game, the physics designer will prefer efficiency, thus simplifying as much as possible the mathematical model and losing in accuracy and generality. Actually in entertainment applications, it is important that the results are believable and plausible rather than realistic. The opposite case occurs when an high degree of realism is required, in computer aided surgery or in mechanical engineering applications, where accuracy is preferred to efficiency. However, the system should remain general enough to remain stable and controllable during all the simulation. For example, the Euler explicit schema presented above is known as a method fast to compute and easy to implement, however it is rather prone to instability and the error grows with the time. Instability and error can be reduced if the time step Δt is made smaller [BW93; Ebe04a], in this case there is the need of more iterations to compute the evolution of the system in a given time frame and thus the efficiency is reduced.

Traditionally, numerical integration schema are categorized as explicit or implicit. The former ones deliver the fastest results while the latter is the more accurate, sacrificing computational speed. An overview of the methods used in Computer Graphics to simulate deformable bodies, like mass-spring systems, the finite element method or the finite differences approaches, can be found in the surveys [GM97; NMBC05].

In this work, it is used an approach recently proposed by Müller et al. [MHHR06], called Position Based Dynamics (PBD). In PBD, the physical system is still modeled through equations governing external and internal forces to the deformable bodies, however it is possible to set constraints which represent geometric relationships among particles with a mass. These constraints are expressed by mathematical equations and inequalities and establish rules over geometric quantities (like the distance from one particle to another), which the particles must respect throughout the simulation. This basically means that in PBD it is possible to *directly handle the position* of the particles without introducing any discontinuity in the solution of the equations governing the system. This is possible because the integration schema is based on the one proposed by Verlet in [Ver67]. It is an explicit schema which is based on the Taylor expansion of the Eq. 3.2:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \dot{\mathbf{x}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{x}}(t)\Delta t^2 + \frac{1}{6}\dddot{\mathbf{x}}(t)\Delta t^3 + O\left(n^4\right) \tag{3.5}$$

$$\mathbf{x}(t-\Delta t) = \mathbf{x}(t) - \dot{\mathbf{x}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{x}}(t)\Delta t^2 - \frac{1}{6}\dddot{\mathbf{x}}(t)\Delta t^3 + O\left(n^4\right) \tag{3.6}$$

$$\tag{3.7}$$

By summing up Eq. 3.5 and Eq. 3.6, it leads to:

$$\mathbf{x}(t+\Delta t) + \mathbf{x}(t-\Delta t) = 2\mathbf{x}(t) + \ddot{\mathbf{x}}(t)\Delta t^2 + O\left(n^4\right) \tag{3.8}$$

which, rearranging, becomes:

$$\mathbf{x}(t+\Delta t) = 2\mathbf{x}(t) - \mathbf{x}(t-\Delta t) + \ddot{\mathbf{x}}(t)\Delta t^2 + O\left(n^4\right) \tag{3.9}$$

In this formulation, the velocity term disappeared and the position at the next time step $\mathbf{x}(t+\Delta t)$ depends only from the current forces applied to the particle, the current position and the position at the previous time step. Actually, the velocity term is implicitly expressed in Eq. 3.9 as

$$\mathbf{v}(t+\Delta t) = \frac{\mathbf{x}(t) - \mathbf{x}(t-\Delta t)}{\Delta t} + O(n) \tag{3.10}$$

The Eq. 3.9 has several nice characteristics: it is reversible in time (if a negative time step is used, the system rolls back exactly to the starting point), it is symplectic [Ear06], that is, it conserves the energy of the system, and thus it is stabler than the Euler method. Furthermore, the approximation of the position is $O\left(n^4\right)$ which is two orders of magnitude greater than the Euler one, thus the Verlet method is much more precise than the Euler one.

Since in Eq. 3.9, the velocity is implicitly defined by the current and past position of the particle, the Verlet integration schema allows to directly *project* (that is, displace) the position of the particles in the so-called *legal* positions. If the particle has penetrated a wall, for instance, the position can be shifted right in point where collision happened and the velocity will implicitly compensate to reach the projected position.

In the field of Computer Graphics, Verlet numerical integration has been used firstly by Jacobsen [Jak03] in his *Fysix* engine to simulate rag dolls, cloth and plants in the game "Hitman: Codename 47". Fědor [FŏŠ] uses a similar approach to simulate characters in games and Porcino used it the movie industry [Por04]. In [MHHR06] this approach is improved by providing the possibility to use non linear geometrical constraints into the system and a Gauss-Siedel iterative solver to efficiently solve them. A hierarchical method able to further speed up the computation in the simulation of flat 3D cloth has been proposed in [MÖ8], while another geometrical approach is reported in [MHTG05] where constraints are formulated for entire sets of particles.

Since in this work, the physical model relies on the Position Based Dynamics approach [MHHR06], in order to keep the thesis as self contained as possible, here the PBD is briefly summarized. In [MHHR06], the objects to be simulated are represented by a set of N particles and a set of M constraints. Each particle i has three attributes, namely

- m_i: mass
- $\mathbf{x}_i$: position
- $\mathbf{v}_i$: velocity

A constraint j is defined by the five attributes:

- n_j: cardinality
- $C_j : \mathbb{R}^{3n_j} \rightarrow \mathbb{R}$: scalar constraint function
- $k_j \in [0, ..., 1]$: stiffness parameter
- *unilateral* or *bilateral*: type

With type *bilateral* is satisfied if $C_j(x_{i_1}, ..., x_{i_n}) = 0$. If its type is *unilateral* then it is satisfied if $C_j(x_{i_1}, ..., x_{i_n}) \geq 0$. The stiffness parameter k_j defines the strength of the constraint in a range from zero to one.

Given this data and a time step Δt, the simulation proceeds as follows (from [MHHR06]):

(1) **forall** particles i
(2) initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_0^i$
(3) **endfor**
(4) **loop**
(5) **forall** particles i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\mathbf{f}_ext(\mathbf{x}_i)}{m_i}\Delta t$
(6) **forall** particles i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i\Delta t$
(7) **forall** particles i **do** generateCollisionConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
(8) **loop** solverIterations **times**
(9) projectConstraints$(C_1, ..., C_{M \cup M_{coll}}, \mathbf{p}_1, ..., \mathbf{p}_N)$
(10) **endloop**
(11) **forall** particles i
(12) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
(13) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
(14) **endfor**
(15) **endloop**

Since the algorithm simulates a system which is second order in time, both, the positions and the velocities of the particles need to be specified in (1)-(3) before the simulation loop starts. Lines (5)-(6) perform a simple explicit forward Euler integration step on the velocities and the positions. The new locations $\mathbf{p}_i$ are not assigned to the positions directly but are only used as predictions. Non-permanent external constraints such as collision constraints are generated at the beginning of each time step from scratch in line (7). Here the original and the predicted positions are used in order to perform continuous collision detection. The solver (8)-(10) then iteratively corrects the predicted positions such that they satisfy the M_{coll} external as well as the M internal constraints. Finally the corrected positions $\mathbf{p}_i$ are used to update the positions and the velocities. It is essential here to update the velocities along with the positions. If this is not done, the simulation does not produce the correct behavior of a second order system. the integration scheme used here is very similar to the Verlet method described in Eq. 3.9 and Eq. 3.10.

3.2.1 Gauss-Seidel Solver

The goal of the solver step (8)-(10) is to correct the predicted positions $\mathbf{p}_i$ of the particles such that they satisfy all constraints. The problem that needs to be solved comprises of a set of M equations for the $3N$ unknown position components, where M is now the total number of constraints. This system does not need to be symmetric. If $M > 3N (M < 3N)$ the system is over-determined (under-determined). In addition to the asymmetry, the equations are in general non-linear.

What complicates things even further is the fact that collisions produce inequalities rather than equalities. In the position based dynamics appraoch, non-linear Gauss-Seidel is used. It solves each constraint equation separately.

Again, given $\mathbf{p}$ we want to find a correction $\mathbf{\Delta p}$ such that $C(\mathbf{p}+\mathbf{\Delta p}) = 0$. It is important to notice that PBD also linearizes the constraint function but individually for each constraint. The constraint equation is approximated by

$$C(\mathbf{p} + \mathbf{\Delta p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \mathbf{\Delta p} = 0 \tag{3.11}$$

The problem of the system being under-determined is solved by restricting Δp to be in the direction of $\nabla_p C(\mathbf{p})$ which conserves the linear and angular momenta. This means that only one scalar λ, a Lagrange multiplier, has to be found such that the correction

$$\mathbf{\Delta p} = \lambda \nabla_{\mathbf{p}} C(\mathbf{p}) \tag{3.12}$$

solves Eq. 3.11. This yields the following formula for the correction vector of a single particle i

$$\mathbf{\Delta p_i} = \lambda w_i \nabla_{\mathbf{p_i}} C(\mathbf{p}) \tag{3.13}$$

where

$$s = \frac{C(\mathbf{p})}{\sum_j w_j \left| \nabla_{\mathbf{p_j}} C(\mathbf{p} \right|^2} \tag{3.14}$$

and $w_i = 1/m_i$. As mentioned above, this solver linearizes the constraint functions. However, in contrast to the Newton-Raphson method, the linearization happens individually

per constraint. Solving the linearized constraint function of a single distance constraint for instance yields the correct result in a single step. Because the positions are immediately updated after a constraint is processed, these updates will influence the linearization of the next constraint because the linearization depends on the actual positions. Asymmetry poses no problem because each constraint produces one scalar equation for one unknown Lagrange multiplier λ. Inequalities are handled trivially by first checking whether $C(p) \geq 0$. If this is the case, the constraint is simply skipped. We have not considered the stiffness k of the constraint so far. There are several ways of incorporating the it. The simplest variant is to multiply the corrections $\mathbf{\Delta p}$ by $k \in [0, ..., 1]$.

3.2.2 Stretching constraints

Distance constraints drive two particles $\mathbf{p_1}$ and $\mathbf{p_2}$ to stay at a given distance d from each other. The constraint can thus be expressed by the equation:

$$C(\mathbf{p_1}, \mathbf{p_2}) = |\mathbf{p_1} - \mathbf{p_2}| - d = 0 \tag{3.15}$$

From Eq. 3.13 and 3.14, the correction vectors to use during the constraint projection (step (9) in the algorithm in Sec. 3.2), are expressed as

$$\mathbf{\Delta p_1} = -\frac{w_1}{w_1 + w_2} (|\mathbf{p_1} - \mathbf{p_2} - d|) \frac{\mathbf{p_1} - \mathbf{p_2}}{|\mathbf{p_1} - \mathbf{p_2}|} \tag{3.16}$$

$$\mathbf{\Delta p_2} = +\frac{w_2}{w_1 + w_2} (|\mathbf{p_1} - \mathbf{p_2} - d|) \frac{\mathbf{p_1} - \mathbf{p_2}}{|\mathbf{p_1} - \mathbf{p_2}|} \tag{3.17}$$

By weighting the correction vector with the stiffness parameter $k_{stretch} \in [0, ..., 1]$, the distance constraint become less rigid and its dynamics behave similarly to a Newtonian spring:

$$\mathbf{p_1} = \mathbf{p_1} + k_{stretch} \mathbf{\Delta p_1} \tag{3.18}$$

$$\mathbf{p_2} = \mathbf{p_2} + k_{stretch} \mathbf{\Delta p_2} \tag{3.19}$$

3.2.3 Other constraints

Beside the distance constraint, several other kinds of constraints can be defined on a set of particles. In this work, I used

bending which force two triangular faces sharing one edge to maintain a given dihedral angle. A bending constraint involves four particles and it can be written as:

$$C(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}) =$$

$$acos\left(\frac{(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_3} - \mathbf{p_1})}{|(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_3} - \mathbf{p_1})|} \cdot \frac{(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_4} - \mathbf{p_1})}{|(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_4} - \mathbf{p_1})|}\right) - \varphi = 0 \tag{3.20}$$

where φ is the initial dihedral angle between the two faces.

area which force a triangular area to conserve its area during deformation:

$$C(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = \left(\frac{1}{2} |(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_3} - \mathbf{p_1})| - A_0 = 0 \right) \tag{3.21}$$

where $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ are the vertices of the triangle and A_0 is its initial area.

volume which force the particles to conserve the volume enclosed by the surface to which they belongs. In mathematical terms:

$$C(\mathbf{p}_1, ..., \mathbf{p}_N) = \left(\sum_{i=1}^{N_{triangles}} \left(\mathbf{p}_{t_1^i} \times \mathbf{p}_{t_2^i} \right) \cdot \mathbf{p}_{t_3^i} - V_0 = 0 \right) \tag{3.22}$$

where t_1^i, t_2^i, t_3^i are the three indexes of the vertices belonging to triangle i. The sum computes the actual volume of the closed mesh which is compared against the original volume V_0.

position is a simple constraint which anchors a particle to a given position $\mathbf{p}_{i_0}$:

$$C(\mathbf{p}_i) = |\mathbf{p}_i - \mathbf{p}_{i_0}| = 0 \tag{3.23}$$

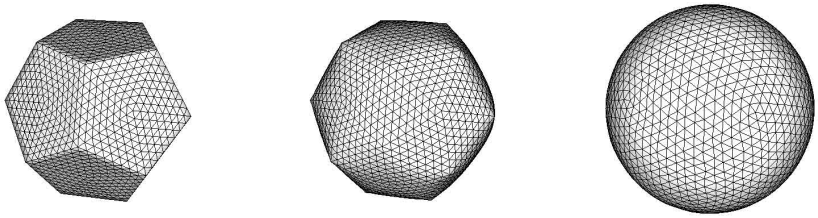


Fig. 3.7: Volume preservation constraint in action: a dodecahedron is inflated by imposing a growing volume value.

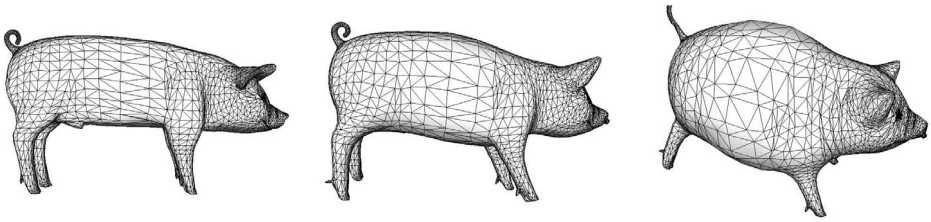


Fig. 3.8: Volume preservation constraint in action: a pig mesh is inflated by imposing a growing volume value.

3.3 Spatial Hashing Data Structure

In the literature there are several different methods devoted to collision detection for deformable objects [ODGB01; TKZ*04]. In this thesis, the Spatial Hashing data structure [THM*03] have been used for collision detection and ray tracing purposes. Spatial hashing is a particularly advantageous approach if the simplexes (points, triangles or tetrahedra), have approximately the same size and are evenly distributed into the space. The concept is to discretize the space through a uniform grid. The size of one side of each cell is the average dimension of the triangle edges. Then, each simplex is stored in all the grid cells which overlap. For fast access to the cell, first the 3D position of a point $\mathbf{p} = (p_x, p_y, p_z)$ belonging to the simplex is approximated with an integer triple:

$$(i_x, i_y, i_z) = \left(\left\lfloor \frac{p_x}{h} \right\rfloor, \left\lfloor \frac{p_y}{h} \right\rfloor, \left\lfloor \frac{p_z}{h} \right\rfloor \right) \tag{3.24}$$

Cells can be stored into an hash map with N buckets, N can be smaller or greater than the total number of grid cells. Greater is N and lesser the number of hash collisions in the buckets, that is each simplex is stored in a different bucket. The correspondence among grid cells and hash map buckets is computed through the hashing function:

$$i = ((i_x \cdot 92837111) \oplus (i_y \cdot 689287499) \oplus (i_z \cdot 283923481)) \, mod N \tag{3.25}$$

The single integer i is the index in an hash table which stores pointers to the simplexes. During the simulation, at the beginning of each time step, all the primitives involved into the collision detection are loaded into the hash table. Note that it is not necessary to clear all the structure; it is sufficient to mark each bucket in the hash table with a time stamp which is incremented at the beginning of the time step. The elements in the hash table which have a time stamp lower than the current one are simply not considered.

This mechanism allows for spatial queries in constant time of primitives belonging to objects under deformation. For example, to perform a ray-triangle operation, it is sufficient to traverse all the cell grid intersected by the ray, starting from its origin, and test for intersection only the simplexes, in this case triangles, stored in the bucket corresponding to the currently traversed cell. As a collision is detected, the process is stopped.

3.4 Radial Basis Function Interpolation

In this project, there is the need of a simple and efficient fitting algorithm for two purposes: to fit the musco-skeletal structure into the skin mesh, as explained in Sec. 5.4, and in the motion cloning technique presented in Chap. 6. Deformation based on scattered data interpolation, like Shepard-based methods and radial basis function methods [FRR96], are easier to manipulate for fitting purposes. In particular, radial basis functions are more effective when landmarks are scarce and they are also computationally efficient. Radial basis functions are used to find a smooth interpolation over a scattered data set.

In mathematical terms, for given data pairs (P_i, Q_i), $(i = 1...n, P_i \in \mathbb{R}^3, Q_i \in \mathbb{R}^3)$, find a function $G : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$, $G \in \mathcal{C}^1$ so that

$$G(P_i) = Q_i \qquad i = 1...n \tag{3.26}$$

Within this project, the points $\{P_i\}$ are located on a surface mesh M_1 and the points $\{Q_i\}$ are the correspondent points on another surface mesh M_2. The morphing function $G(P)$ is defined from these point sets. By applying $G(P)$ to all the vertexes of M_1, the shape of M_1 is fitted into the shape of M_2.

A radial basis function is defined as a basis function whose value depends only on the distance from the data point. A radial basis function method is simply the linear combination of such basis functions:

$$G(P) = \sum_{i=1}^{n} c_i R_i(P) \tag{3.27}$$

where $\{c_i\}$ are 3D coefficients and $\{R_i\}$ are the radial basis functions. We use as basis function the Hardy multiquadrics, defined as:

$$R_i(P) = (d_i^2(P) + r_i^2)^{\frac{1}{2}} \tag{3.28}$$

where d_i is the euclidean distance from P_i, r_i is the stiffness radius controlling the stiffness of the deformation around P_i. The Hardy method is the combination of the sum of the Hardy multiquadrics and a linear term $\mathcal{L}$ used to absorb the linear part of the morphing transformation:

$$G(P) = \sum_{i=1}^{n} h_i \cdot (d_i^2(P) + r_i^2)^{\frac{1}{2}} + \mathcal{L}(P) \tag{3.29}$$

where $\mathcal{L}(P) = h_{n+1} \cdot x + h_{n+2} \cdot y + h_{n+3} \cdot z + h_{n+4}$ is the affine transformation, $\{h_i\}(h_i \in \mathbb{R}^3, i = 1...n+4)$ are called Hardy coefficients. Introducing the $\mathcal{L}(P)$ term, the source mesh will be translated, rotated and scaled according to the position of the target interpolation point set $\{Q_i\}$. For smooth results, the parameters $\{r_i\}$ are set to:

$$r_i = \begin{cases} \min d_i(P_j) & \text{if } i \neq j, \\ 0 & \text{if } i = j. \end{cases} \tag{3.30}$$

In order to find the volume morphing function $G(P)$, we have to define the interpolation point sets $\{P_i\}$ and $\{Q_i\}$ and then solve a linear equation system to compute the Hardy coefficients. The linear equation system to find the $\{h_i\}$ values is obtained by imposing the n conditions $G(P_j) = Q_j$:

$$Q_j = \sum_{i=1}^{n} h_i \cdot (d_i^2(P_j) + r_i^2)^{\frac{1}{2}} + \mathcal{L}(P) \qquad (j = 1...n) \tag{3.31}$$

Furthermore, for linear precision, we impose:

$$\sum_{i=1}^{n} h_i \cdot x_i = 0 \quad (3.32)$$

$$\sum_{i=1}^{n} h_i \cdot y_i = 0 \quad (3.33)$$

$$\sum_{i=1}^{n} h_i \cdot z_i = 0 \quad (3.34)$$

$$\sum_{i=1}^{n} h_i = 0 \quad (3.35)$$

This results in a linear system of $n+4$ equations with $n+4$ unknowns $\{h_i\}(i = 1...n+4)$. The solution exists and it is unique [Mic86].

3.5 MPEG-4 Face and Body Animation Standard

In this section, it is provided a short introduction to MPEG-4 FBA specification, describing the main parameters sets cited in this thesis. Readers familiar with MPEG-4 FBA may wish to skip the section, or use it only as a quick reference.

MPEG-4 is an international standard dealing with the efficient transmission of animation. A significant part of the MPEG-4 is the Face and Body Animation (FBA), a specification for efficient coding of shape and animation of human faces and bodies [PF02; ISO]. Instead of regarding animation as a sequence of images with fixed shape and size, MPEG-4 FBA gives the possibility to express, for each animation frame, the facial movements through a set of weighted parameters. Thus, the animation is synthesized by a proper deformation of the face mesh according to such parameters.

MPEG-4 FBA specifies a face model in its neutral state together with two main data sets: 84 Facial Definition Parameters (FDPs), also called feature points, and 68 Facial Animation Parameters (FAPs). According to the standard, a generic face model is in neutral state when the gaze is in direction of the z-axis, all face muscles are relaxed, eyelids are tangent to the iris, lips are in contact and the mouth is closed in such a way that the upper teeth touch the lower ones [PF02].

FDPs (Fig. 3.9) are control points used to define the shape of a proprietary face model and to provide a spatial reference for defining FAPs. The location of FDPs has to be known for any MPEG-4 compliant face model. The FAP set is subdivided in high- and low- level parameters. Low-level FAPs are used to express basic action that the face can perform, like closing an eyelid or stretching a corner lip. All low-level FAPs are expressed in terms of the Facial Animation Parameter Units (FAPU). They correspond to distances between key facial features and are defined in terms of distances between the FDPs. High-level FAPs are useful to express in a compact way more complex movements like expressions and visemes (that is, the visual counterpart of a phoneme). FAPs allow for representation of the whole set of natural facial movements.

The FBA parameters, both FDPs either FAPs, can be extracted automatically from visual, audio and motion capture systems [AD03; Ahl02; GEZMT99; LESMT99], therefore they

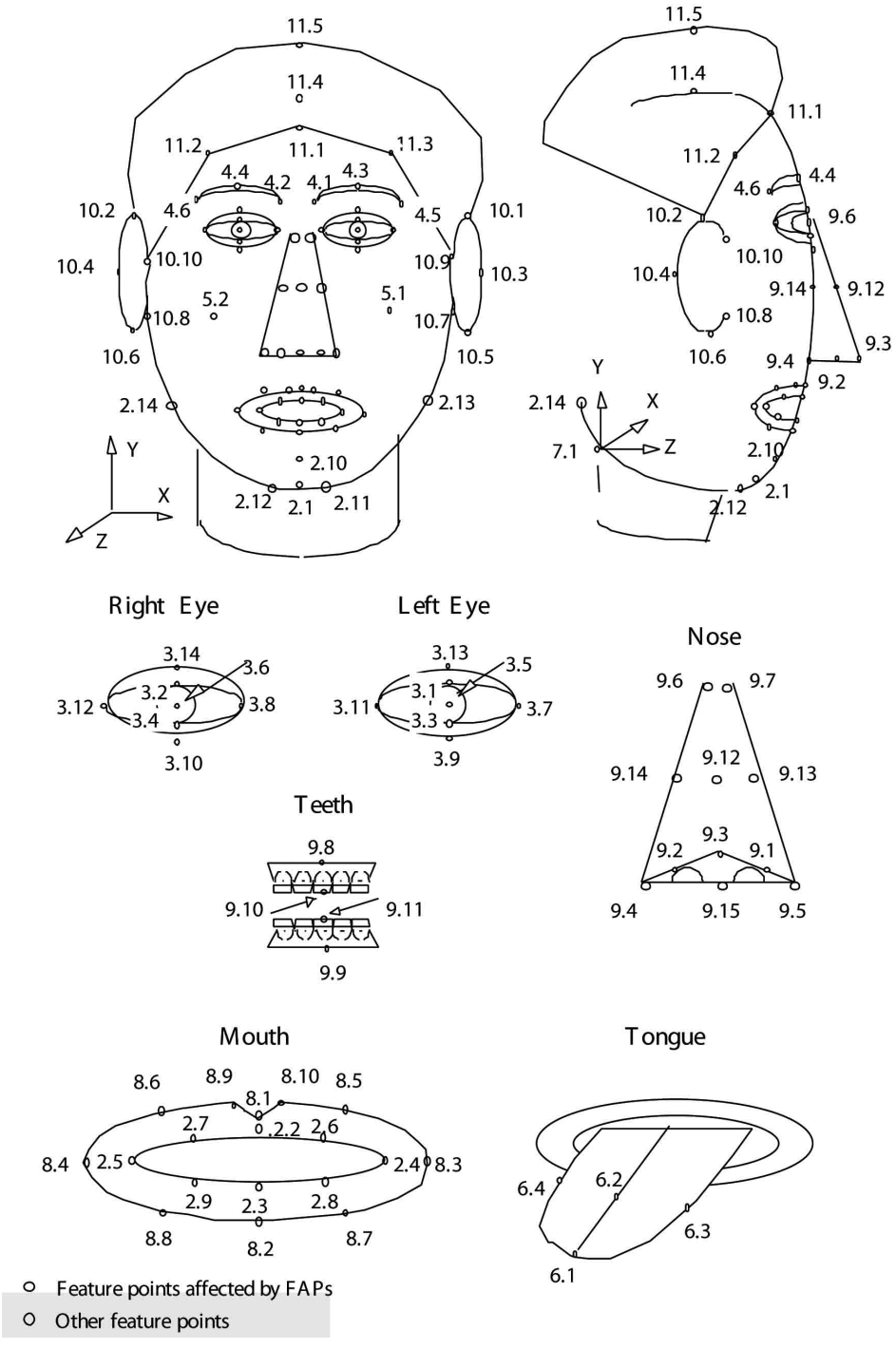


Fig. 3.9: MPEG-4 Facial Definition Points (FDPs).

can be properly compressed and used to achieve, for example, teleconferencing with a very low bandwidth (< 3 Kbps) [Ost98; CPO00].

Chapter 4

The Geometric Muscle Model

4.1 Key Requirements

The major challenge in attempting to accurately reproduce the *microscopic* behavior of the muscles lies in their inherent structural complexity. As most of the living tissues in the human body, they exhibits characteristics like *non-homogeneity, anisotropy, quasi-incompressibility, and non-linear plastic-viscoelasticity*. This basically means that their dynamical behavior changes from point to point and that deformation is reversible only for small stresses. Modeling such a behavior involves usually sophisticated physical models, which in turn introduces a computational overhead, not feasible for allowing interactive simulation on current standard hardware. Furthermore, to define a complex material, like a facial living tissue, there is the need of many parameters, which values is not always known because the difficulty in measuring them.

For these reasons, to synthesize convincing facial expressions in the context of real-time animation, I define a muscle model that does not consider the microscopic structures, like fibers, and relative parameters but instead focuses on mimic the *macroscopic* aspects of the muscle behavior described in Sec. 3.1.2.

In particular,

- only isotonic contraction is considered, since it produces the most visible effects; the muscle can also passively contract and stretch under the influence of the underlying structures, like bones and other muscles;
- the muscle model conserves its volume while under stress; so it becomes thicker and its belly bulges out when it contracts, it becomes thinner as it elongates;
- fibers are represented by action lines which lay directly on the underlying structures (bones or other muscles);
- it allows for interactive definition of both origin and insertion sites as well the shape of the parallel array of fibers.

4.2 Geometric Construction

4.2.1 Action Lines

An *action line* is a piecewise linear curve $A \in \mathbb{R}^3$ lying on a set of triangulated surface meshes. The purpose of the action lines is twofold: (1) they define the bottom contour of the muscle geometry during the simulation, and (2) they provide a mechanism to control the active contraction of the muscle itself.

A *surface point* is a point **sp** $\in S$, where S is a triangulated surface in $\mathbb{R}^3$. A surface point **sp** is uniquely described by the homogeneous barycentric coordinates (t_1, t_2, t_3) w.r.t. the vertexes $A_1A_2A_3$ of the triangular facet to which it belongs, as shown in Fig. 4.1.a.

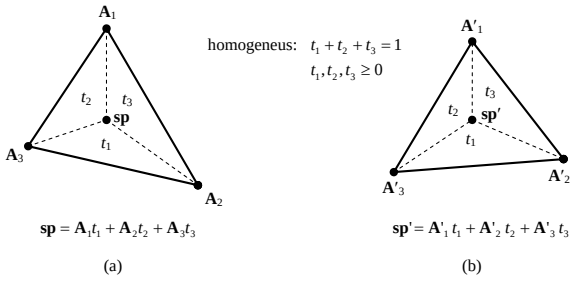


Fig. 4.1: (a). A *surface point* **sp** $\in \Delta A_1A_2A_3$ is defined by the homogeneous barycentric coordinates (t_1, t_2, t_3) w.r.t. the triangle vertexes. (b). As the triangle deforms, the triple (t_1, t_2, t_3) remains constant and **sp** is updated accordingly to **sp′**.

The relevant attributes of a surface point are *position* and *normal*; both of them are obtained through the linear combination of the barycentric coordinates with the corresponding attributes of the triangle vertexes. When these latter displace due a deformation of the triangle, the new position and normal of **sp** are updated as well using the new attributes of the vertexes (Fig. 4.1.b).

Each linear segment of the action line is defined by two surface points; thus, an action line is completely described by the ordered list of its surface points. Note that each single surface point may belong to a different surface S. So, for example, an action line may start on a surface, continue on another surface and finish on a third surface. When the underlying surfaces deform, the surface points displaces and the action line deforms accordingly. A graphical example of this mechanism is shown in Fig. 4.2.

Next section explains how the geometrical model of the muscle is built considering a closed contour of action lines; Sec. 4.3 describes the muscle dynamics and the active contraction mechanism, based on a proper sampling of the action lines.

4.2.2 Specification of Muscle Geometry

The muscle geometry in the facial model, attach, lay and insert directly on the 3D surfaces representing other anatomical structures, like bones and other muscles.

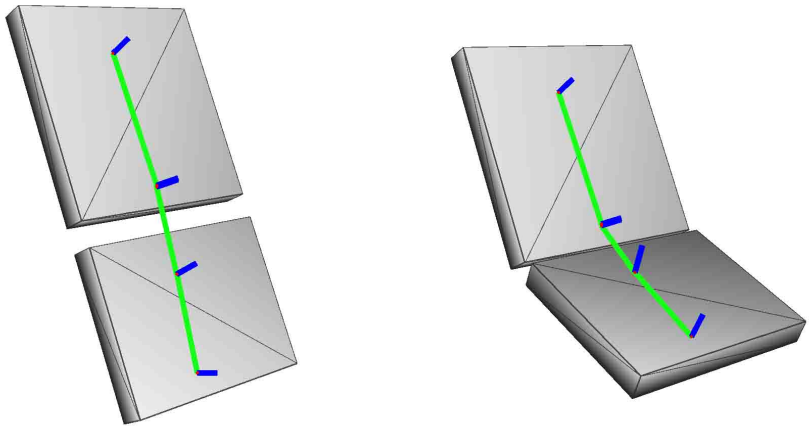


Fig. 4.2: *Left*. A simple action line (green) made of four surface points lying on two aligned triangulated icosahedrons. Surface points are displayed as red dots and normals as blue segments. *Right*. After a simple rotation of the bottom icosahedron, surface points, and thus the action line, update accordingly.

It is thus important to precisely specify the contour of the muscle which is in direct contact with the aforementioned structures. Once this contour is known, it is used as basis to build the geometry of the muscle.

The process needs the following input parameters:

- the polygonal resolution of the geometry, expressed as the number of longitudinal h and latitudinal w sections;
- the bottom contour C of the muscle, defined directly on the geometrical surfaces representing the anatomical structures placed underneath the muscle;
- the thickness t of the muscle.

I illustrate an intuitive and easy user-interface to provide the simulation system with these input parameters in Sec. 4.4.

A 3D triangulated, orientated, 2manifold hexahedral mesh M is built with exactly h longitudinal section and w latitudinal sections (Fig. 4.3).

The contour of one of the larger sides of the mesh is arbitrarily chosen as the *bottom* contour. In the following, it is explained how M is morphed, through the RBF-based fitting algorithm devised in Sec. 3.4, in order to make its bottom contour match with the input contour C. This will produce a mesh M' laying exactly on the surfaces where C is defined. Finally, the thickness of the muscle geometry is adjusted according to the scalar input parameter t.

Note that the size and orientation of the initial mesh M are not important, and thus are not specified, because they do not influence the final shape M'. This is due to one of the feature of the RBF-based algorithm: its invariability w.r.t. affine transformations, like translation and rotation.

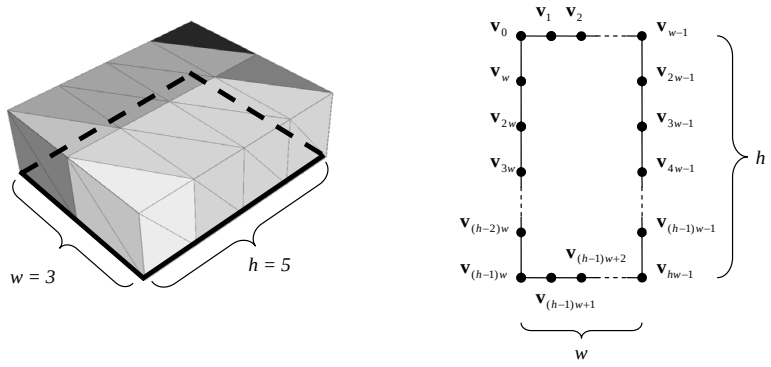


Fig. 4.3: *Left.* Hexahedral mesh procedurally built on the fly as a basis for the muscle geometry; this example has $h = 5$ latitudinal sections and $w = 3$ longitudinal sections. The mesh will be morphed to properly fit in the anatomical model. The bottom contour, partially hidden, is shown bold. *Right.* Schema of the bottom contour with the vertexes forming it. The values of the vertex indexes are obtained for construction.

By using action lines, and surface points, it is rather straightforward to define arbitrary piecewise curves which form a closed contour C lying on existing superficial geometries $\in \mathbb{R}^3$. C is formed by four action lines $A_i, i = 0, ...3$, two latitudinal ones and two longitudinal ones. The extremes of the latitudinal lines are connected through the extremes of the longitudinal ones, as shown in Fig. 4.4.

C is sampled by sampling separately each action line at regular intervals; the sample step for each action line is defined accordingly to the Euclidean norm measured on the segments:

$$step = \frac{\sum_{i=0}^{N} \left\| \mathbf{sp}_i - \mathbf{sp}_{i-1} \right\|}{N}, \qquad N \in \{w, h\} \tag{4.1}$$

where N assumes the value w or h, according to the type of the sampled action line, respectively longitudinal or latitudinal.

Each sample is characterized by its 3D position $\mathbf{s}$ and normal $\mathbf{n_s}$, in a similar way of surface points. However, unlike surface points, samples are not bound to any triangular facet of any surface. The position $\mathbf{s}_i$ and the normal vector $\mathbf{n}_{\mathbf{s}_i}$ of each sample on an action line are found through a simple linear combination of the corresponding surface point attributes:

$$\mathbf{s}_i = \mathbf{sp}_j + \frac{(\mathbf{sp}_j - \mathbf{sp}_{j-1})}{\left\| \mathbf{sp}_j - \mathbf{sp}_{j-1} \right\|} d_i, \qquad i = 0, ..., N, \qquad N \in \{w, h\} \tag{4.2}$$

$$\mathbf{n}_{\mathbf{s}_i} = \frac{\mathbf{n}_{\mathbf{sp}_{j-1}} \left(\left\| \mathbf{s}_i - \mathbf{sp}_j \right\| \right) + \mathbf{n}_{\mathbf{sp}_j} \left(\left\| \mathbf{s}_i - \mathbf{sp}_{j-1} \right\| \right)}{\left\| \mathbf{sp}_j - \mathbf{sp}_{j-1} \right\|} \tag{4.3}$$

where

$$d_i = step * i - \sum_{k=1}^{j-1} \left\| \mathbf{sp}_k - \mathbf{sp}_{k-1} \right\| \tag{4.4}$$

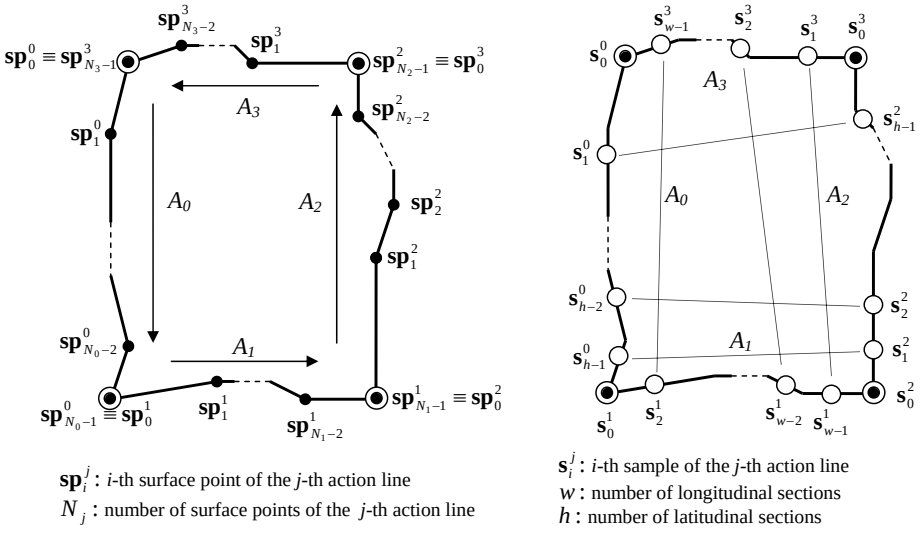


Fig. 4.4: *Left*. The contour C is defined by four action lines $A_i, i = 0, ..., 3$. Each action line is defined by an ordered list of surface points $\{\mathbf{sp}_j\}$. The extremes of the action lines coincide in order to form a closed contour. *Right*. C is sampled at regular intervals and the samples are used to build on the fly the muscle geometry.

and

$$j = max \left\{ l \in \mathbb{N} : \sum_{k=1}^{l} \left\| \mathbf{sp}_k - \mathbf{sp}_{k-1} \right\| < step * i \right\} \tag{4.5}$$

Since the number of samples is w for each latitudinal action line and h for each longitudinal line, the cardinality of the set of samples is $2(w + h)$, that is the same number of vertexes which belong to the bottom contour of the hexahedral mesh M (Fig. 4.3.right). Thus, a bijective mapping among these two sets can be set according to the following rules:

$$\mathbf{s}^0_i \rightarrow \mathbf{v}_j, \qquad i = 0, ..., h-1, \qquad j = i * w \tag{4.6}$$

$$\mathbf{s}^1_i \rightarrow \mathbf{v}_j, \qquad i = 0, ..., w-1, \qquad j = (h-1) * w + i \tag{4.7}$$

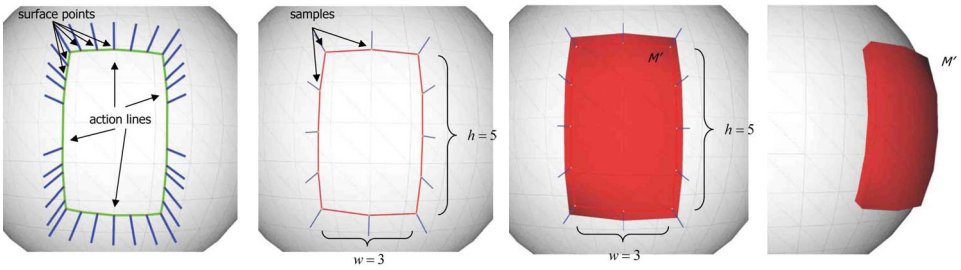
$$\mathbf{s}^2_i \rightarrow \mathbf{v}_j, \qquad i = 0, ..., h-1, \qquad j = (h-i) * w + 1 \tag{4.8}$$

$$\mathbf{s}^3_i \rightarrow \mathbf{v}_j, \qquad i = 0, ..., w-1, \qquad j = w - 1 - i \tag{4.9}$$

where the symbol "$\rightarrow$" means "*in correspondence with*". According to the theory explained in Sec. 3.4, a RBF-based fitting function $G(P)$ is thus computed and used to morph M. After the morphing, the bottom contour of the morphed mesh M' matches with C. An example of the outcome of this process is depicted in Fig. 4.5.

The normals $\mathbf{n}_{\mathbf{s}_i}$ of the samples are useful to adjust the thickness of the morphed muscle geometry $M^{'}$. Given a vertex $\mathbf{v}^{low}_j \in M^{'}$ on the bottom contour, the corresponding vertex $\mathbf{v}^{up}_j$ on the upper layer is given by:

$$\mathbf{v}^{up}_j = \mathbf{v}^{low}_j + \mathbf{n}_{\mathbf{s}_i} * t \tag{4.10}$$



(a) (b) (c) (d)

Fig. 4.5: (a). A contour C formed by four action lines (green) is defined on a spherical 3D surface. Surface Points are in red and normals in blue. (b) C is sampled with $w = 3$ and $h = 5$. Samples are in yellow and normals in blue. (c). The morphed hexahedral mesh M', representing the muscle geometry. (d). A prospective view of M'.

where i and j are respectively the indexes of the samples and the mapped vertexes according to 4.6,...,4.9, and t is the input thickness value.

4.3 Muscle Dynamics and Control

The muscle geometry is the basis to define the geometric constraints which govern the dynamics of the muscle itself. The muscles which are devoted to produce facial expressions belongs basically to two types: sheet and sphincter (Sec. 3.1.3). The action produced by these muscles is always along the tangent of their fibers. To represent them, thus, I use a general template muscle model in which the fibers are represented by a subset of the action lines forming the contour C. The constraints on the template model, and its geometry, are adapted according to the type of muscle that it has to be simulated. In Sec. 4.3.1 and Sec. 4.3.2 it is explained how each action line, and thus the muscle, is able to contract by properly modulating a sampling step.

In the following, the general muscle template is described and, then, how it is adapted to simulate sheet and sphincter muscles.

Initially, let us consider the geometry of the template muscle model as the morphed hexahedral mesh M' defined in the previous section, just after applying the morphing $G(P)$. Each vertex is considered as a particle having mass $m = 1.0$. First, a network of *distance* constraints is built as shown in Fig. 4.6.

Each distance constraint has a stiffness $k \leq 1$, so they behave in similar way of a spring, but without the typical overshooting problems (as stated in Sec. **??**). In particular, there are:

- *Longitudinal* (and *latitudinal*) constraints, linking a particle with its neighbors along longitudinal (and latitudinal) sections on the same side, upper or bottom; their action causes the strongest resistance to in-plane compression or tension.

- *Connection* constraints, linking a particle of one side of the muscle mesh with its corresponding particle on the other side; they resist the traction stresses or compression

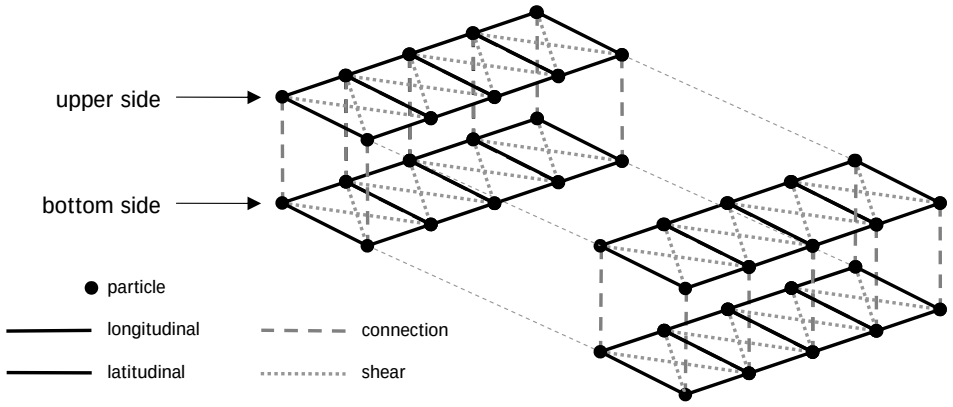


Fig. 4.6: Distance constraints involved in the muscle model dynamics.

between layers. Note that connection constraints are not accommodated into the volume bounded by the muscle surface, they are placed only on the surface itself.

- *Shearing* constraints, that resist to shearing and twisting stresses. As for connection constraints, they are not placed inside the muscle volume.

This network alone is not sufficient to completely describe the dynamics of the muscle. When the muscle contracts, the upper side flexes leading to a bulge in the shape of the muscle. The above soft network does not capture this behavior because there are not any constraints which prevents the particles from moving towards the internal volume of the body; in some extreme cases, the displacement of particles causes the inversion the faces' normal. Placing internal connection or shearing constraints into the muscle volume would not solve this issue: the particles would not collapse into the muscle volume, but on the other hand, the muscle would not be allowed to bulge out under compression. For these reasons, for each couple of triangulated faces f_i, f_j sharing the same edge e_{ij}, a *bending* constraints is defined (Sec. **??**). A bending constraint conserves superficial tension and reacts whenever there is a compression, or an elongation, perpendicular to the edges of the muscle triangulated mesh M.

A further *volume preserving* constraint is defined over all the particles of the muscle geometry, making the muscle thicker under compression and thinner when it elongates.

The softness of the muscular deformable body depends from the stiffness of each single constraint, regardless its kind, and from the number of Gauss-Siedel iterations performed within the numerical integration. Since this latter parameter is fixed and set to the lowest value possible to ensure interactivity (as specified in Sec. **??**), each type of muscle defines its own set of stiffness constraints as described in the following two sections.

4.3.1 Sheet Muscle

A sheet muscle is a flat and thin muscle, whose fibers are arranged in parallel threads. When the muscle contracts, the fibers shortens by equal amounts and pull the skin surface from the

insertion site towards the origin site, which can be considered fixed. Examples of this kind of muscle is the *frontalis* muscle placed in the forehead, used in movements like glancing upwards and expressions of surprise and fright (Sec. 3.1.3).

In order to model this mechanism, given the contour C, as defined in Sec. 4.2.2, and the action lines $A_i, i \in \{0, ...3\}$, we consider the latitudinal action lines A_3 and A_1 as, respectively, the *origin* site and the *insertion* site of the muscle; the longitudinal action lines A_0 and A_2 represent, instead, the fibers of the muscle and define the direction of contraction.

Each particle $\mathbf{v}_i$ of the muscle model which correspond to a sample $\mathbf{s}_j$ in the longitudinal action lines A_1 and A_3 (Eq. 4.6 and 4.8), is bound to its corresponding sample through a *position* constraint (Sec. **??**).

Thus, if a sample $\mathbf{s}_j$ is displaced, the corresponding particle $\mathbf{v}_i$ will be displaced as well. The new position of the samples is computed by considering the Eq. 4.2 with the sampling step:

$$step' = step(1 - c) \tag{4.11}$$

where $step$ is defined in Eq. 4.1, and $c \in [0, 1]$ is the *contraction factor*. If $c = 0$ then the muscle is in rest position, if $c = 1$ then there is the maximal contraction.

If the muscle is very flat and large, additional longitudinal action lines, called *internal*, are embedded into the model. The geometrical construction of the internal action lines is specified in Sec. 4.4, by now it is important to note that they span from the origin to the insertion site parallel to the longitudinal action lines.

An example of contracting sheet muscle is shown in Fig. 4.7.

The constraints, with the corresponding stiffness, involved into the dynamics of the sheet muscle are:

$$\begin{aligned} k^{up}_{latitudinal} &= 1.0 \\ k^{up}_{longitudinal} &= 0.1 \\ k_{connection} &= 1.0 \end{aligned}$$

$$\begin{aligned} k_{bend} &= 0.4 \\ k_{volume} &= 1.0 \\ k_{position} &= 0.2 \end{aligned}$$

The stiffness of the remaining constraints is set to 0; these constraints are thus disabled and not considered further into the simulation. Note the low stiffness of the position constraints: while the difference among expected position and actual position remains acceptable, this low value allows the muscle to slide over the underlying surfaces.

An example of the effect of the volume constraint in the sheet muscle model is shown in Fig. 4.8.

4.3.2 Sphincter Muscle

The sphincter muscle model is used to simulate the *orbicularis oris*, that is the muscle surrounding the oral orifice, capable of various movements, including closure, protrusion and pursuing of the lips (Sec. 3.1.3).

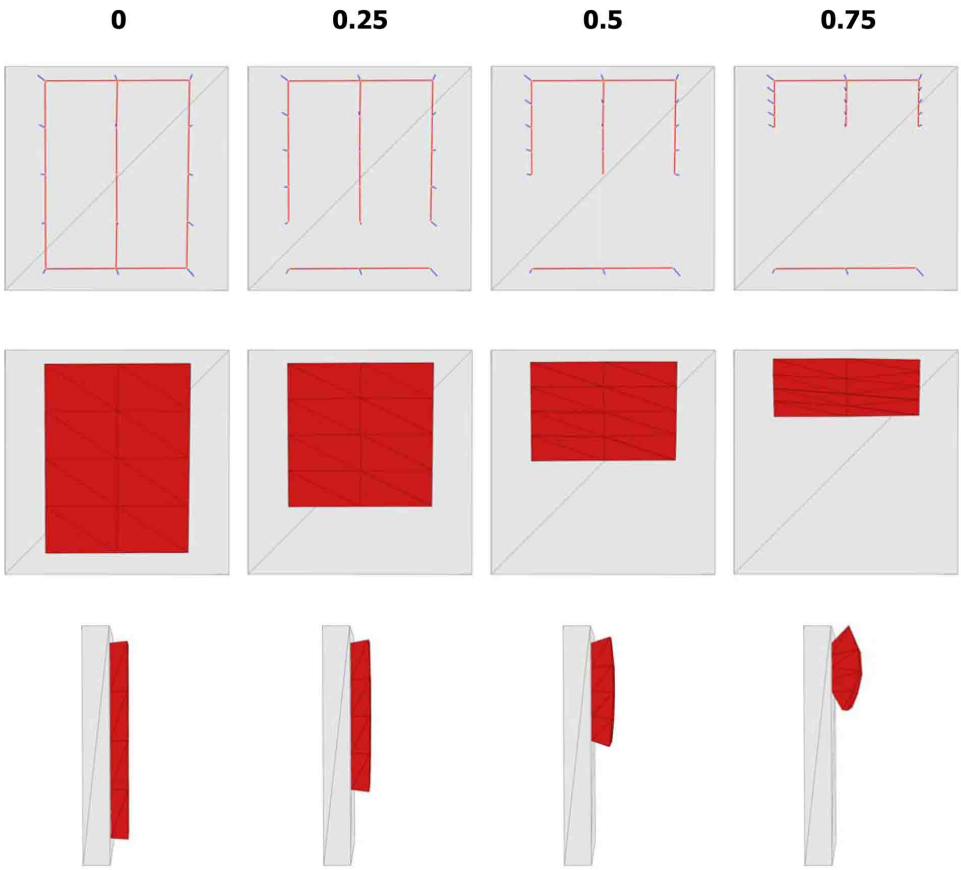


Fig. 4.7: Example of a sheet muscle defined on a flat surface. Contraction level c is 0, 0.25, 0.50 and 0.75. *Upper row*: samples of the bottom contour C, connected by red links for clarity. *Middle* and *bottom rows*: front and side view of the muscle geometry while contracting. Note how the muscle bulges out as the contraction c increases.

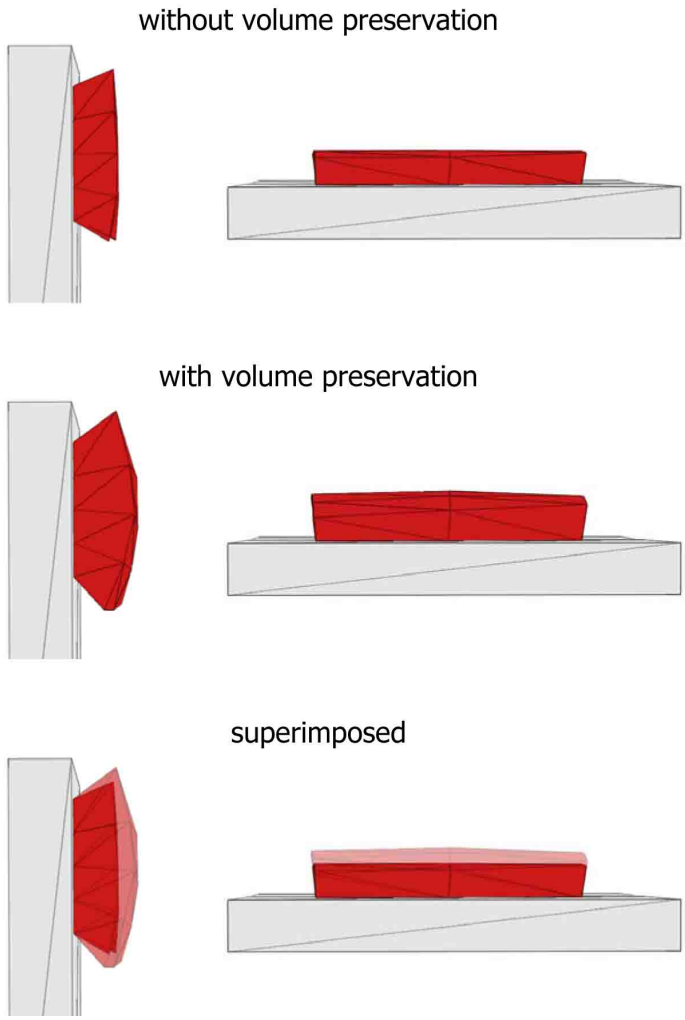


Fig. 4.8: Side (*left*) and top (*right*) view of a sheet muscle while contracting with $c = 0.3$. *Upper row*: with volume preserving constraint; *middle row*: without volume preserving constraint; *bottom row*: the two configurations are superimposed for comparison.

In the sphincter muscle model, the two longitudinal action lines, A_1 and A_3, are both closed curves (their first surface point coincides with their last one), and define respectively the inner and the outer contour of the muscle. The remaining latitudinal action lines, A_0 and A_2, are coincident and both of them connects the first surface point of A_1 with the first surface point of A_3 (Fig. 4.9).

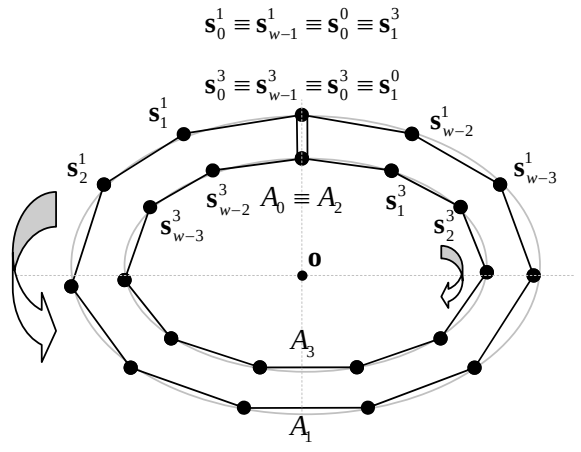


Fig. 4.9: Schematic view of the action lines in the sphincter muscle model.

In the sphincter muscle model there is not a fixed origin site. In fact, while contracting around a virtual center, the body of this kind of muscle slides on the underlying anatomical structures. Like the sheet muscle, the motion of the sphincter muscle model is driven by the samples of the action lines, anchored through position constraints to the corresponding particles on the muscle geometry.

Similarly to other approaches in the context of interactive physically based facial animation [Wat87; TW90; ZPS02], the contraction of a sphincter muscle can be modeled through the use of a parametric ellipsoidal function. The construction of the ellipsoid could have be done through a best fit in the least square sense, however to have more control on the shape of the resulting ellipsis, I preferred to build it in the following manner.

Once the action lines are defined, they are sampled according to Eq. 4.6. A contraction plane CP is computed by fitting, in least square sense [Nea04], the whole set of samples of all the action lines. A first guess of the position of the contraction center $\mathbf{o}_guess$ is estimated as the average of the samples of A_1 projected on CP:

$$\mathbf{o}_{guess} = \sum_{i=0}^{w-1} \mathbf{s'}_i^1 \tag{4.12}$$

where $\mathbf{s'}_i^1$ is the sample projected on CP. A line l_i is defined passing through each pair $\left\langle s_i^1, s_j^3 \right\rangle, i \in 0, ..., w-1 \subset \mathbb{N}, j = w-1-i$. The contraction center position $\mathbf{o}$ is then recomputed as the average of the set of points $\mathbf{c}_i$, where $\mathbf{c}_i$ is the point of l_i closest to $\mathbf{o}_guess$.

The normal of the contraction center $\mathbf{n}_o$ is taken as the normal of the contraction plane $\mathbf{n}_{CP}$. The direction $\mathbf{dir}_a$ of the semi major axis a of the ellipses is the direction of the axis parallel to the x-axis passing through $\mathbf{o}$, projected on the contraction plane CP and normalized. A similar computation is carried out for the direction $\mathbf{dir}_b$ of the semi-minor axis b, considering the y-axis. The amplitude of the two axises is

$$\begin{aligned} |a| &= max_i \{(\mathbf{s}_i - \mathbf{o}) \cdot \mathbf{dir}_a\} \\ |b| &= max_i \{(\mathbf{s}_i - \mathbf{o}) \cdot \mathbf{dir}_b\} \end{aligned}$$

The just defined ellipsis, together with the contract factor c, is used to compute the sample displacements and thus the muscle motion. For each pair $\langle s_i^1, s_j^3 \rangle$ of action line samples,

$$\begin{aligned} x &= (\mathbf{s'}_j^3 - \mathbf{o'}) \cdot \mathbf{dir}_a \\ y &= (\mathbf{s'}_j^3 - \mathbf{o'}) \cdot \mathbf{dir}_b \end{aligned}$$

then

$$\delta = (\mathbf{o} - \mathbf{s}_i^1)(1-c)cos(\frac{\pi}{2}(1 - \frac{\sqrt{y^2b^2 + x^2a^2}}{ab}))$$

$$\mathbf{s}_i^1 = \mathbf{s}_i^1 + \delta \mathbf{s}_j^3 = \mathbf{s}_i^3 + \delta$$

The constraints, with the corresponding stiffness, involved into the dynamics of the sphincter muscle are:

$$\begin{aligned} k_{latitudinal}^{up} &= 0.2 \\ k_{connection} &= 1.0 \\ k_{shear}^{up} &= 0.3 \\ k_{shear}^{low} &= 0.3 \\ k_{bend} &= 1.0 \\ k_{volume} &= 1.0 \\ k_{position} &= 0.2 \end{aligned}$$

As for the sheet muscle, the stiffness of the remaining constraints is set to 0 and thus they are disabled. An example of contraction of a sphincter muscle is shown in Fig. 4.10.

4.3.3 Passive Deformation

In the previous two sections, I described the *active* contraction of sheet and sphincter muscles, that is a contraction depending from an external scalar parameter c. However, in my model the muscle can also *passively* contract and stretch influenced by the deformation of the underlying anatomical structures. As stated in Sec. 4.2.1, action lines are basically an ordered list of surface points, and a surface point deforms according to the triangular facet to which it belongs, because while the triangle vertexes move, its barycentric coordinates remains constant. Since both the shape and the dynamics of a muscle are driven by the sampled

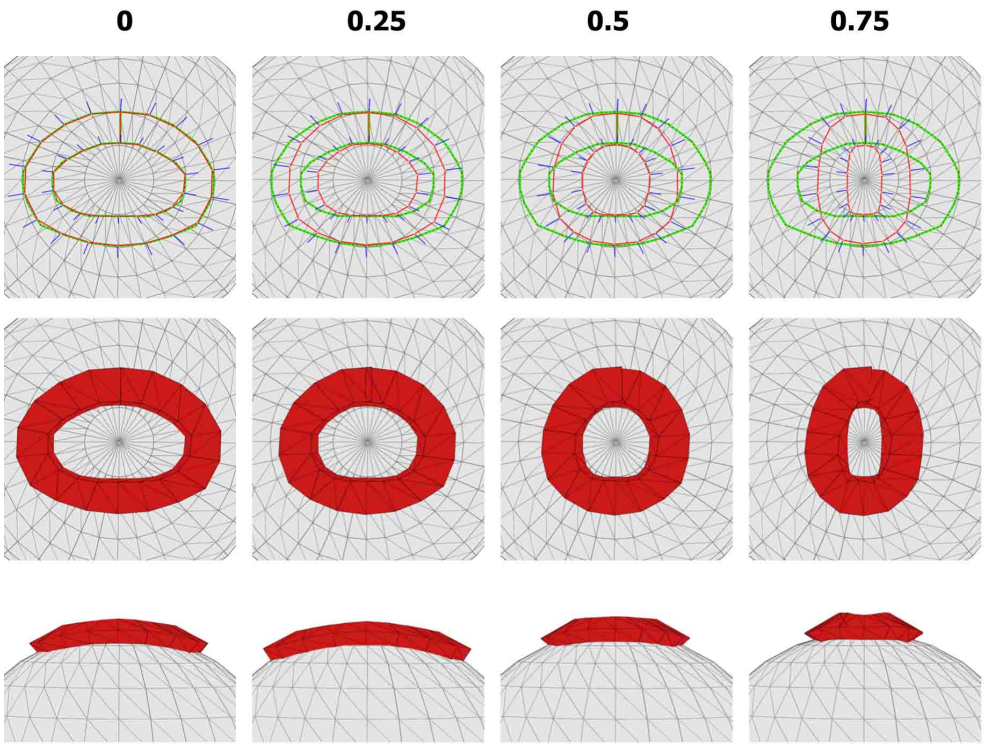


Fig. 4.10: Example of a sphincter muscle defined on a spherical surface, $h = 2$, $w = 19$. Contraction level c is 0, 0.25, 0.50 and 0.75. *Upper row*: action lines (green) and samples of the bottom contour C, connected by red links for clarity. *Middle* and *bottom rows*: front and top view of the muscle geometry while contracting.

action lines, if these latter are defined on a dynamic surface, like another muscle or a movable bone, then the muscle itself will deform according to the surface where it lays.

Fig. 4.11 and 4.12 show examples of this mechanism involving two sheet muscles and a flat surface representing the bone. In Fig. 4.11, the green muscle lays partly on the red muscle and partly on the gray bone. When the red muscle contracts, the green muscle stretches. In Fig. 4.12, the brown muscle lays on the red muscle; as this latter contracts, the brown muscle contracts as well. A further example is depicted in Fig. 4.13, where a yellow sheet muscle lays on two sheet muscles which contracts in opposite directions.

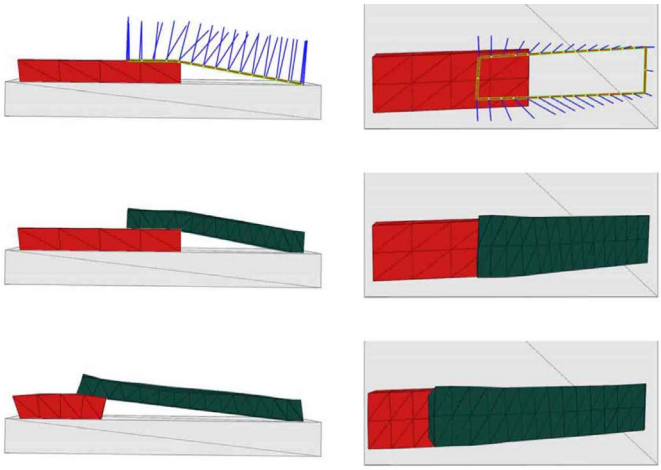


Fig. 4.11: Passive stretching of a sheet muscle (green) laying on another sheet muscle (red) and a flat bone (gray). *Left*. Side view. *Right*. Top view.

4.4 Interactive Editing

One of the input parameters mentioned in Sec. 4.2.2, necessary to build the muscle geometry, is the bottom contour C, defined on the 3D surfaces representing the anatomical structures placed underneath the muscle. Here I describe an interactive method which provides the final user with the ability to specify C directly and in a natural way on the already existing anatomical structures.

The idea is to allow the user to sketch out the action lines, and thus the contour C, directly on the visible anatomical geometries. The surface points forming the action lines are found through a fast ray-tracing algorithm. Using an input 2D pointing device, like a mouse or a pen on a tablet PC, the current 2D position is projected on the near clipping plane and on the far clipping plane of the current view frustum in the 3D space. These two points define a unique ray passing through them, and a surface point is obtained intersecting the ray with the visible geometry (Fig. 4.14).

The ray-tracing algorithm must be fast enough to allow interactive update rates. For this purpose, the 3D space is subdivided in a regular grid of cells; each cell, and the triangular

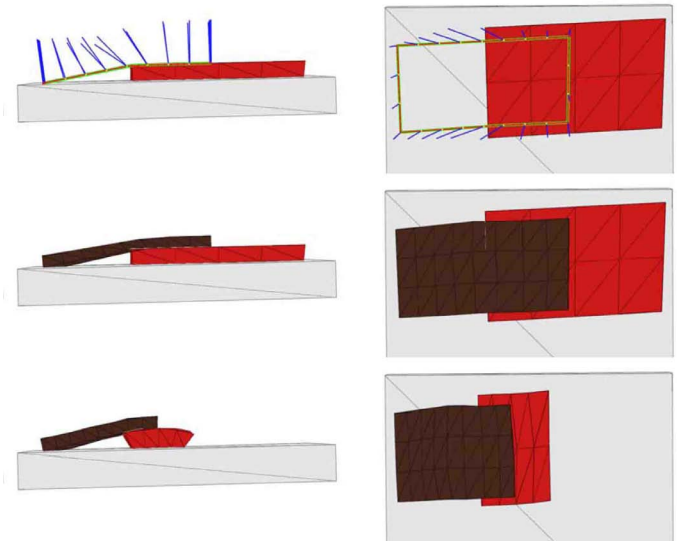


Fig. 4.12: Passive contraction of a sheet muscle (brown) laying on another sheet muscle (red) and a flat bone (gray). *Left*. Side view. *Right*. Top view.

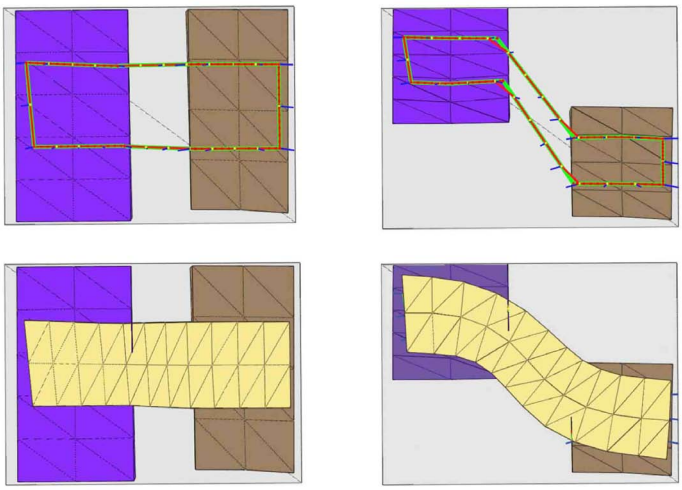


Fig. 4.13: Passive deformation of a yellow sheet muscle laying on two sheet muscles which contracts in opposite directions. *Left*. Relaxed state. *Right*. Both of the underlying muscles contracts.

Fig. 4.14: The 2D position of the pointing device is projected on the near and far clipping plane of the view frustum. A ray is casted (solid line) and traverses the 3D space. The intersection with the musco-skeletal structure define a surface point.

facets which are contained in it, are stored in a spatial hashing data structure 3.3. When a ray traverses the space, from the near to the far plane, for each cell hit by the ray, the contained facets are tested for collision. If an intersection is found, the algorithm stops otherwise continues on the next cell.

To define a single action line, the user manually picks a set of surface points. In the general case, the segment connecting two consecutively picked surface points $\mathbf{sp}_i$ and $\mathbf{sp}_j$ does not lay on the anatomical surface, unless it is flat. Thus, beside this latter, there are two cases: the segment crosses a concave portion of the mesh or penetrates inside a convex portion. In this case, further surface points are obtained automatically by splitting the segment in such a way that the piecewise linear curve running from $\mathbf{sp}_i$ and $\mathbf{sp}_j$ lays mostly on the visible surface according to the mechanism depicted in Fig. 4.15.

The segment-splitting mechanism is also shown in Fig. 4.15.

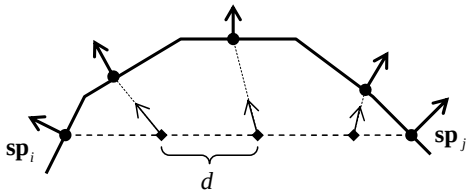
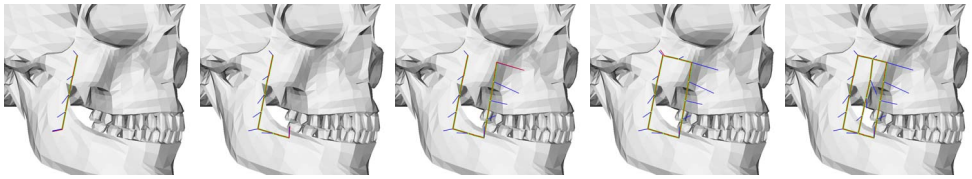


Fig. 4.15: If two picked surface points crosses the mesh, the linking segment is automatically split and further surface points are found on the mesh surface.

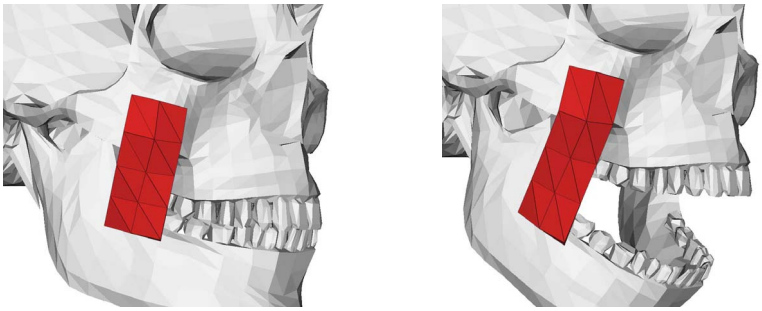
To define the closed contour C for a sheet muscle, the first surface point of an action line A_i must be the last one of A_{i-1}, $i > 0$, and the last surface point of A_3 is the first on of A_0. In the case of the sphincter muscle instead, only the outer and inner action lines A_1 and A_3 have to be specified, because the surface points of the remaining action lines are obtained like shown in Fig. 4.9.

By allowing the user to interactively define the contour C, together with w, h and the thickness t, the muscle geometry which lays on C can be completely specified (Sec 4.2.2). This technique allows the larger number of samples on C to be specified by a fewer set of surface points, that make up the action lines.



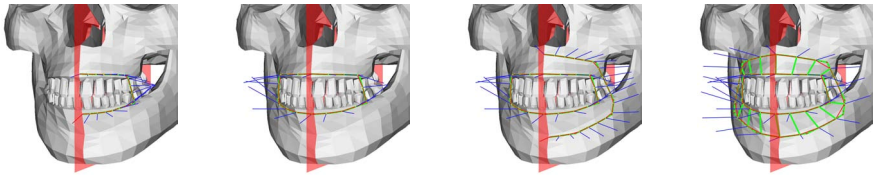
(a) (b) (c) (d) (e)

Fig. 4.16: Interactive definition of a linear muscle on a skull mesh. Note that the longitudinal action lines crosses the empty space between the zygomatic bone and the lower jaw.



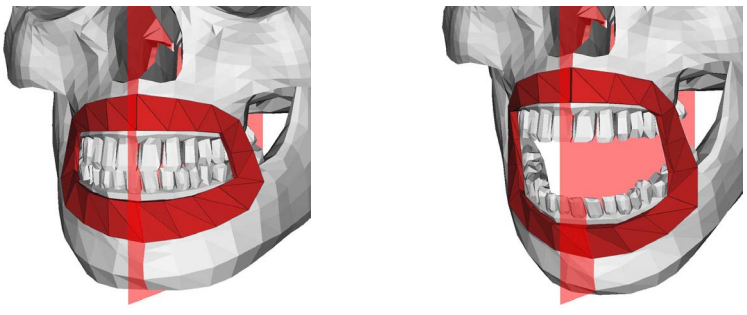
(a) (b)

Fig. 4.17: By modifing the underlying surface (in this case the bony jaw), the linear muscle is properly deformed.



(a) (b) (c) (d)

Fig. 4.18: Interactive definition of a sphincter muscle on a skull mesh. Note that each action line is automatically symmetrized w.r.t. the sagittal plane of the head.



(a) (b)

Fig. 4.19: Modifying the underling structure causes the muscle deformation.

Chapter 5

Facial Model

5.1 Facial Model Overview

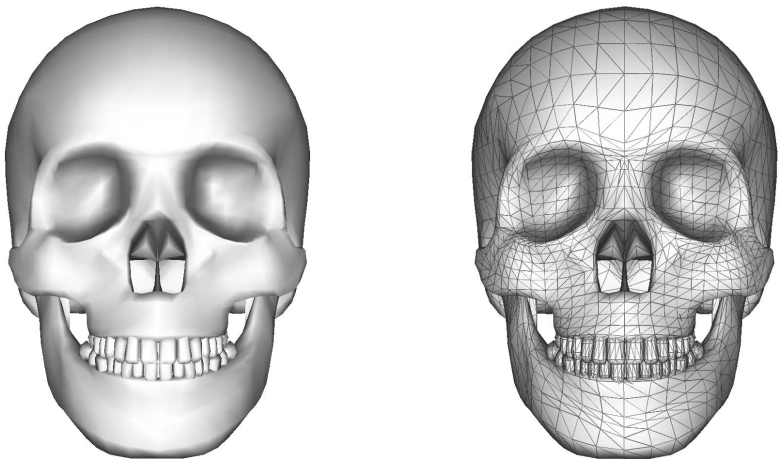
In the human face, muscles forms an interweaving structure which is rather complex to simulate as a whole, for a number of reasons. It is difficult to identify precisely the shape and the contours of a real facial muscle since it is a thin shell which most likely originates from the bony structure, or from another muscle, and inserts and blends into either another muscle or on the superficial facial tissue. The bio-mechanic dynamics of the muscolotendon system is difficult to model into a set of mathematical equations given its non-linear, anisotropic nature. Furthermore, the quantitative parameters which characterize these equations are hard to measure in the living tissues. This is a research area on its own and many works have been published in the literature. In the Computer Graphics field, see for instance [Fun93; WVG97; SPCM97; AT00; AT01; NTHF02]. Each muscle interacts with all the surrounding anatomical structures making the mathematical model even more sophisticated.

The facial model presented in this work is organized in layers; the deepest layer is the bony structure, that is the skull. Then, there are successive layers of muscles and cartilages which lay one on top of the other. The last and most superficial layer is the skin, which is represented by the face mesh to animate. The deepest layers influences the most superficial ones, propagating the deformation until the skin moves producing animation. Before discussing the musculature and its effects on surface form, we briefly mention the influence of the articulated rigid skull.

5.2 Skull

The skull is the basis on which the whole computational model of the face is built. It is represented by a triangulated surface mesh $\in \mathbb{R}^3$ which may have any shape or connectivity. In Fig. 5.1, it is shown the mesh used for the experiments presented throughout this thesis. The face model is not constrained to use this particular skull mesh; for instance, it may use a version with lower polygonal resolution.

The skull is divided in two components: the upper skull and the mandible. The mandible



(a) (b)

Fig. 5.1: Shaded *(a)* and wireframe *(b)* views of the skull mesh used for the experiments. It is formed by 6882 vertexes and 13380 faces.

consists of the complete lower jawbone. The motions of the mandible are modeled through the use of rigid transformations applied to the jaw mesh vertexes. Depression and elevation is represented through a rotation around a *pitch* axis, the small amount of lateral movement as a rotation around a *yaw* axis and, finally, protrusion and retraction as a translation along the *protrusion* axis; the total number of degrees of freedom is thus three.

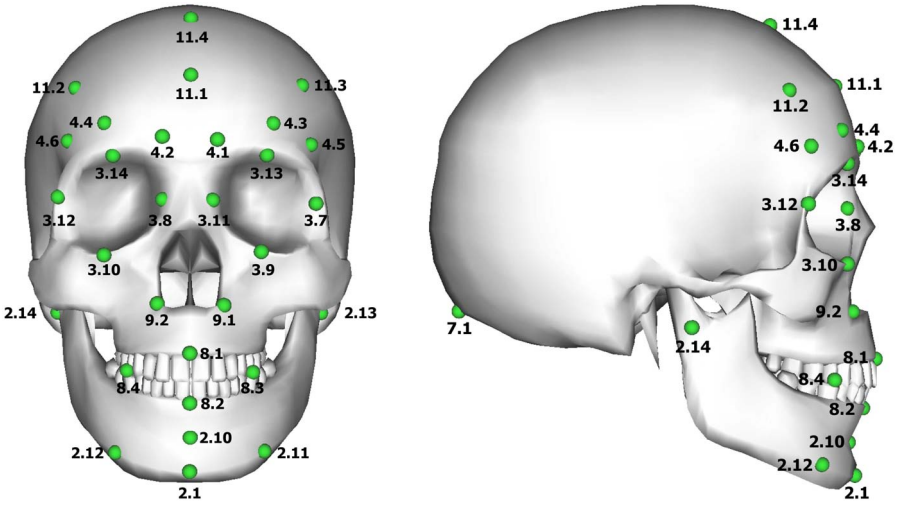
A set of 31 landmarks is defined on the skull surface (Fig. 5.2). The landmarks belongs to a subset of the MPEG-4 FDPs (see Sec. 3.5), and their purpose is twofold: (1) they are useful to define the pitch and yaw rotation axises and the protrusion direction of the mandible, and (2) they provide the spatial references for computing the morphing function to fit the skull and the muscle map into the facial skin mesh (see Sec. 5.4). Note that the landmarks are implemented as surface points (Sec. 4.2.1); this means that their position can be chosen on any point of the skull mesh surface, not necessarily a vertex. To pick each one of the landmarks, it is employed the same ray-tracing algorithm explained in Sec. 4.4.

The pitch rotation axis of the mandible is located at the tip of the condyle, just behind the earlobe; that is, the pitch axis passes through the landmarks 2.13 and 2.14. The maximal pitch rotation allowed is of 18 degrees, while the minimal is -4 degrees (see Fig. 5.3).

In addition to pitch, the mandible may yaw. In this case the rotation axis passes through the landmark 11.4 and the middle point among the landmarks 2.13 and 2.14. The minimal and maximal yaw rotation is, respectively, -5 and $+5$ degrees (see Fig. 5.4).

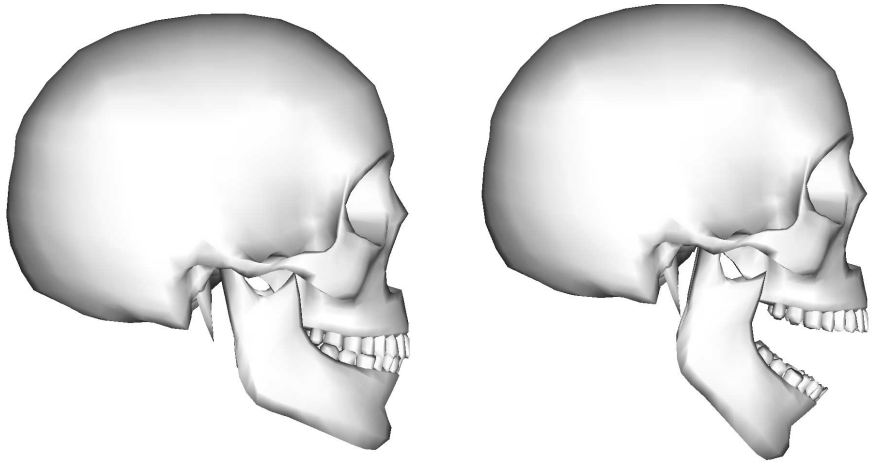
Lastly, the protrusion direction is computed as the line passing through the landmarks 7.1 and 2.10. When the mandible protrudes, all the vertexes belonging to it are simply translated in the protrusion direction (see Fig. 5.5). Protrusion and yaw rotation are necessary to achieve certain facial expressions, such as when chewing or looking dumbfounded and confused.

Given the pitch or yaw normalized rotation axis $\hat{\mathbf{u}}_{\mathbf{rot}} = (x, y, z)$ and a rotation angle θ,



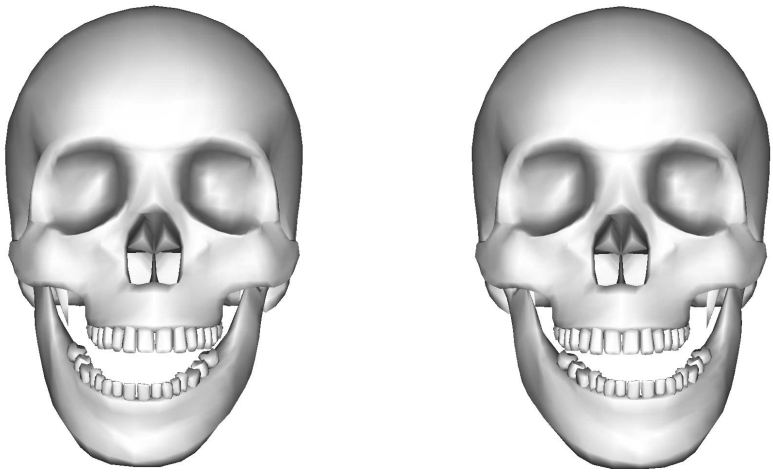
(a) (b)

Fig. 5.2: Front and side views of the 31 landmarks placed on the skull surface. They are a subset of the MPEG-4 FDPs (Sec. 3.5), and thus the naming convention.



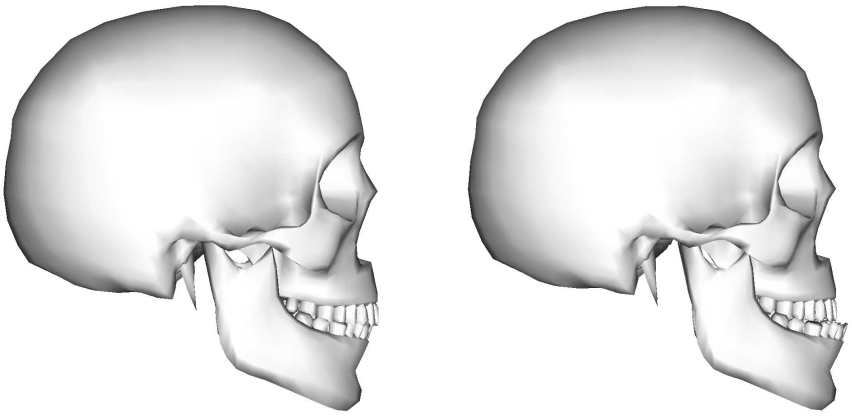
(a) (b)

Fig. 5.3: The skull mesh performing pitch rotation of the mandible.



(a) (b)

Fig. 5.4: The skull mesh performing yaw rotation of the mandible.



(a) (b)

Fig. 5.5: The skull mesh protruding the mandible.

the rotation transformation [SE03; Ebe04b] is

$$rot\left(\hat{\mathbf{u}}_{\mathbf{rot}},\theta\right)=\hat{\mathbf{u}}_{\mathbf{rot}}\hat{\mathbf{u}}_{\mathbf{rot}}^{T}+\cos\left(\theta\right)\left(\hat{\mathbf{u}}_{\mathbf{rot}}\times\mathbf{v_i}\right)+\sin\left(\theta\right)\left(\hat{\mathbf{u}}_{\mathbf{rot}}\times\left(\hat{\mathbf{u}}_{\mathbf{rot}}\times\mathbf{v_i}\right)\right) \tag{5.1}$$

which in a matrix form becomes:

$$rot\left(\hat{\mathbf{u}}_{\mathbf{rot}},\theta\right)=\begin{pmatrix}\left(1-x^{2}\right)c_{\theta}+x^{2} & -zs_{\theta}-xyc_{\theta}+xy & ys_{\theta}-xzc_{\theta}+xz & 0\\ zs_{\theta}-xyc_{\theta}+xy & \left(1-y^{2}\right)c_{\theta}+y^{2} & -xs_{\theta}-yzc_{\theta}+yz & 0\\ ys_{\theta}-xzc_{\theta}+xz & xs_{\theta}-yzc_{\theta}+yz & \left(1-z^{2}\right)c_{\theta}+z^{2} & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{5.2}$$

where $c_\theta=\cos\left(\theta\right)$ and $s_\theta=\sin\left(\theta\right)$. Thus, to correctly rotate any vertex $\mathbf{v_i}=\left(\mathbf{v_x},\mathbf{v_y},\mathbf{v_z},\mathbf{1}\right)$ belonging to the mandible mesh to its transformed configuration $\mathbf{v_i'}$, first $\mathbf{v_i}$ is brought in the local coordinates of the rotation axis, it is rotated and then it is brought back to the world coordinates. This is expressed by the following matrix product:

$$\mathbf{v_i'}=tra\left(\mathbf{p}_{\hat{\mathbf{u}}_{\mathbf{rot}}}\right)rot\left(\hat{\mathbf{u}}_{\mathbf{rot}},\theta\right)tra\left(-\mathbf{p}_{\hat{\mathbf{u}}_{\mathbf{rot}}}\right)\mathbf{v_i} \tag{5.3}$$

where $\mathbf{p}_{\hat{\mathbf{u}}_{\mathbf{rot}}}=\left(p_x,p_y,p_z,1\right)^1$ is any point belonging to the axis $\hat{\mathbf{u}}_{\mathbf{rot}}$ and $tra\left(\mathbf{p}_{\hat{\mathbf{u}}_{\mathbf{rot}}}\right)$ is the translation matrix

$$tra\left(\mathbf{p}_{\hat{\mathbf{u}}_{\mathbf{rot}}}\right)=\begin{pmatrix}1 & 0 & 0 & p_x\\ 0 & 1 & 0 & p_y\\ 0 & 0 & 1 & p_z\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{5.4}$$

For the protrusion and retraction movements, the jaw vertex $\mathbf{v_i}$ is simply translated along the protrusion axis $\hat{\mathbf{u}}_{\mathbf{tra}}$ direction according to:

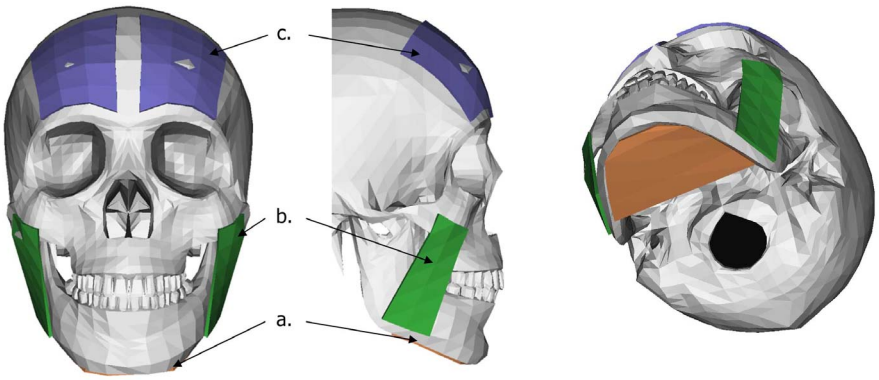
$$\mathbf{v_i'}=tra\left(\hat{\mathbf{u}}_{\mathbf{tra}}\cdot\mathbf{d}\right)\mathbf{v_i} \tag{5.5}$$

where d is the desired amount of displacement.

5.3 Muscle Map

On top of the skull, there is an interweaving structure of muscles, cartilages and facial tissue, mostly fat. In the model presented here, the muscle model presented in Chapter 4 is exploited to build the facial muscles responsible for expressions, the structural muscles which move the jawbone and under the neck, and the fatty tissue under the cheeks. As seen in the previous section, the jaw moves independently through rigid transformations; thus, the structural muscles, like the masseter, does not rotate the jawbone as in a real head but are used as a support for other muscles and the skin.

The muscles are organized in layers. Each layer influences the deformation of the layers on top of it but not the layers underlying it. In Fig. 5.6,5.7,5.8,5.9, there are the different layers forming the muscle map used in the experiments.



(a) (b)

Fig. 5.6: *(a) a. Platysma; b. Risorius; c. Frontalis. (b).* Bottom view of the Platysma muscle, placed under the jaw.

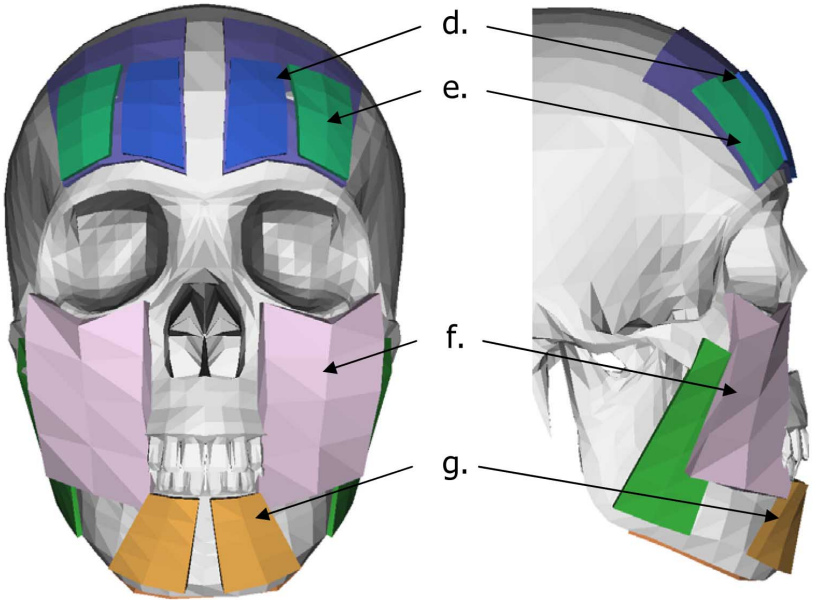


Fig. 5.7: d. Frontalis Inner; e. Frontalis Outer; f. Fatty tissue; g. Depressor Anguli.

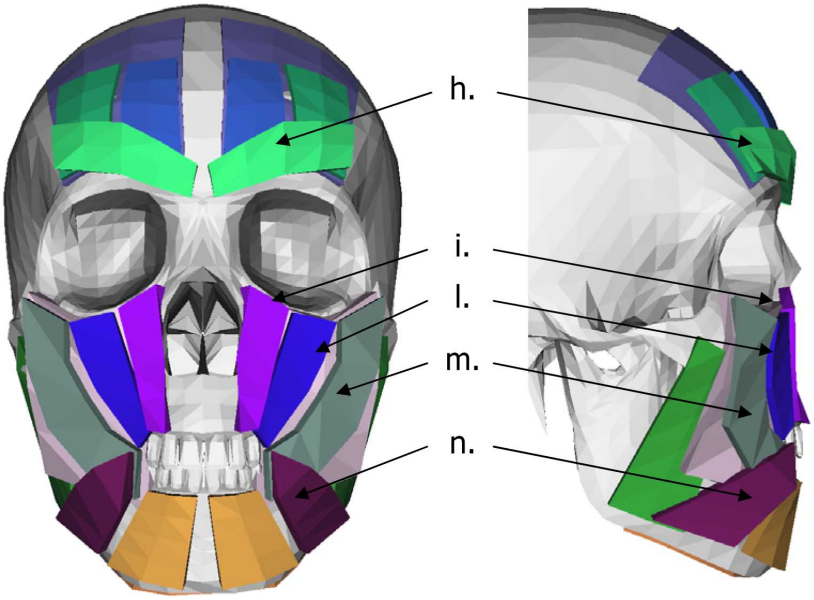


Fig. 5.8: h. Corrugator Supercilii; i. Levator Labii Inner; l. Levator Labii Outer; m. Zygomaticus Major; n. Depressor Anguli.

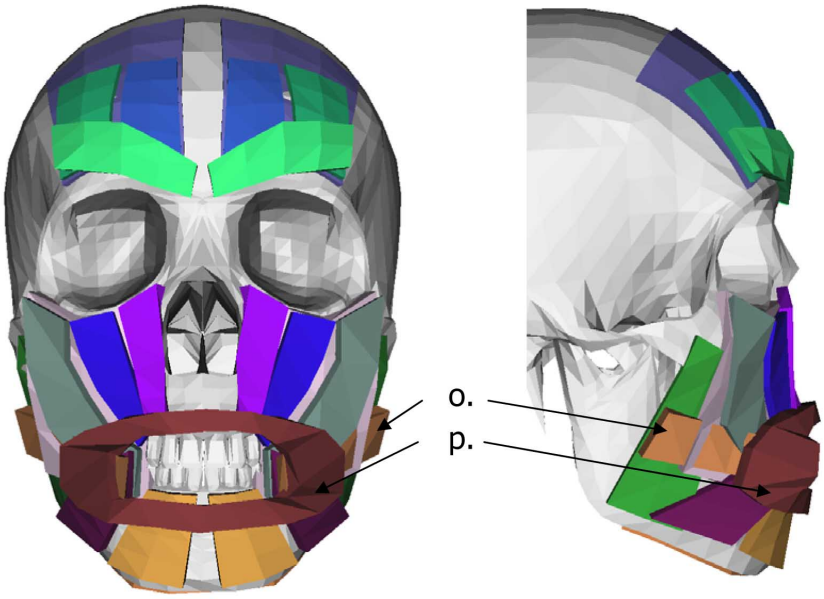


Fig. 5.9: o. Risorius; p. Orbicularis Oris.

The muscle map comprises 25 linear muscles and one circular muscle representing the orbicularis oris. The orbicularis oculi muscles around the eyes and the eyelids are modeled separately as explained in Sec. 5.6.1. Clearly, this map does not exactly represent the real muscular structure of the human head; this is due to the simulated muscle model which has a simplified dynamics compared with the real musco-skeletal system. However, even though there may be not a mapping one-to-one with the real muscle map shown in Fig. 3.2, the virtual muscle map has been devised to mimic all the main expressive functionalities of the real one.

For instance, on the forehead area of a real head, there is a single, large and flat sheet muscle, that is the frontalis bellies, which causes almost all the motion of the eyebrows. In the virtual model, this has been represented by two separate groups of muscles, each one on a separate side of the forehead. Each group is formed by a flat linear muscle (the frontalis) and on top of it two additional muscle (named, for convenience, frontalis inner and frontalis outer). On top of them there is the corrugator, which ends on the nasal region of the skull. Combining together these muscles, the dynamics of the real frontalis bellies are reproduced with a satisfying degree of visual realism, see Sec. 3.1.3, even though the single linear muscle models have a simple dynamics compared with the correspondent real ones.

Some of the linear muscles does not actively move but are just passively deformed by the underlying structures. In particular, the masseter from which originates the risorius muscles; the fatty tissue under the cheeks, on top of which there are the zygomaticus major muscle, the levator labii and part of the orbicularis oris. The last supporting muscle is the platysma, under the chin bone; it is useful as a support for the skin when the jaw rotates, otherwise the skin vertexes would enter in the lower part of the face producing an unrealistic, and unpleasant, effect.

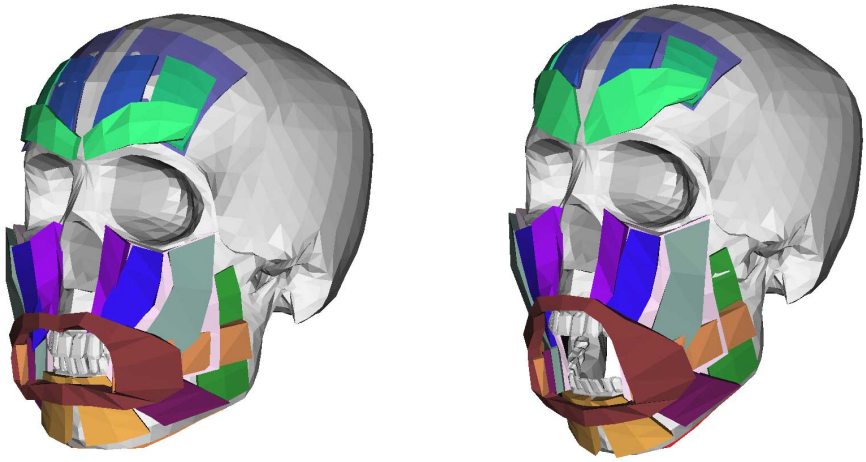
As already stated in Sec. 4.3, each simulated muscle is linked to the underlying structures through position constraints following the position of surface points. Thus, when an anatomical structure deforms, all the surface points lying on it move as well, which in turn influence the motion of the above linked structures. For instance, when the jaw, which is part of the deepest layer, rotates, all the deformable tissues which, totally or partially, lay on it will be deformed as well and, so on, in a sort of chain reaction which eventually arrive at the skin (Sec. 5.5). In Fig. 5.10, the example muscle map is deformed by rotating the jaw and contracting the frontalis bellies. Note how all the above muscles are properly deformed.

5.4 Construction Process

Once the musco-skeletal system has been defined as specified in the previous sections, it can be used to animate the input skin mesh. The skull and the muscle map must be first deformed in order to fit the skin shape. In a successive step (Sec. 5.5), the skin is anchored to the bones and to the muscles in such a way to follow their deformation and animate as well.

The fitting algorithm uses Radial Basis Functions (see Sec. 3.4), to morph the skull mesh. To define the fitting function $G(\mathbf{p})$ it is necessary to provide a set of pairs of spatial positions (P_i, Q_i), $(i = 1...n, P_i \in \mathbb{R}^3, Q_i \in \mathbb{R}^3)$. In this case the initial set of 31 landmarks P_i which

[1] note that the vertex positions are expressed in homogeneous coordinates in order to make the translation matrix a linear operator and to combine it with the rotation matrix. Since in this particular case these are all affine transformation, to obtain the 3D coordinates it is sufficient to remove the last 1.



(a) (b)

Fig. 5.10: *Left*. Skull and muscle map in rest position. *Right*. The jaw depresses while the frontalis bellies contract; the whole muscle map is deformed accordingly.

is located on the skull mesh (Fig. 5.2) is provided as input together with the mesh, as stated in Sec. 1.2. The position of these points identifies a subset of the MPEG-4 Facial Definition Points and specifies the position of facial features which characterizes the face shape. A similar set of landmarks Q_i is defined for the skin mesh and it is provided in input as well.

To improve the quality of the morphing, further 16 landmarks are automatically computed. Given two defined landmarks, $\mathbf{p_1}$ and $\mathbf{p_2}$, the median point $\mathbf{p} = \mathbf{p_1} + (1/2)\,(\mathbf{p_2} - \mathbf{p_1})$ is computed. A cylindrical projection axis is computed as a vertical line through the barycenter of the mesh. A ray perpendicular to the projection axis is passed through the median point p. The origin of the ray is defined outside the mesh and the direction is such that the ray passes through p and intersects the projection axis. The first intersection is appended in the list of correspondences for the mesh.

In Fig. 5.11, it is shown the result of this process for a skull and a skin mesh as well the pairs of initial landmarks used for the enrichment, linked by a black line. These pairs have been defined empirically, by running the morphing algorithm on several skin meshes and assessing visually where the skull mesh was not properly embedded into the skin. Green dots represents the initial set of landmarks provided in input. Red dots are the landmarks obtained from the enrichment process.

The landmarks on the skull surface can be overlapped by the muscles. In this case, the landmarks are brought directly on the muscle surface through a cylindrical projection similar to the one just described: a ray is casted from outside the skull towards the inner cylindrical projection axis and, if intersects a muscle, it is assigned to it. Before the ray tracing, the thickness of the muscles is recomputed according to the skull dimension *after* the morphing:

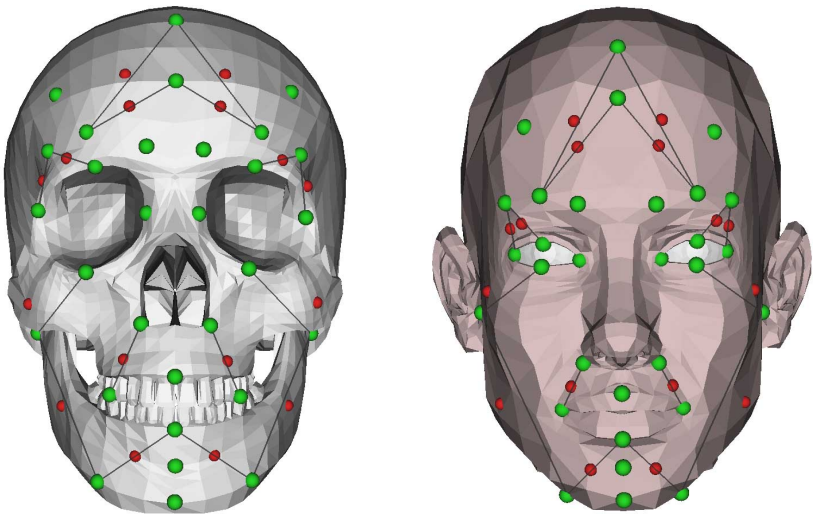


Fig. 5.11: Set of landmarks to compute the morphing function used to fit the skull mesh into the skin. Green dots represent landmarks provided in input, red ones are computed by ray tracing.

$$t' = \frac{d_{head}}{d_{skull}} t$$

where d_{head} and d_{skull} are the diagonal of the Axis Aligned Bounding Box which embed respectively the skull and the skin mesh. By projecting the skull landmarks on the muscles with updated thickness, after the morphing, the muscle map is accommodated just beneath the skin surface.

Given the whole set of landmark pairs (P_i, Q_i) for the skull and the skin mesh, the fitting function $G(\mathbf{p})$ is found by solving the system in Eq. 3.31 - 3.35, through a LU decomposition. $G(\mathbf{p})$ is applied to all the vertexes of the skull mesh which is thus morphed into the skin. The surface points which defines the action lines on the bottom contour of the muscles are updated according to the new positions of the skull faces. The geometry of each muscle is thus rebuilt according to the algorithm in Sec. 4.2.2. Fig. 5.12 shows the result of this process for a sample skin mesh.

Note that the rear part of the skull is not fitted into the skin. This is not important because, as explained in the next sections, the deformation of the skin is not propagated in the back of the head but it happens mostly where the jaw and the muscle map are.

5.5 Skin

Skin is modeled as a deformable body; its properties are defined by geometrical constraints, in a similar way to the muscles. The skin is built starting from the 2-manifold face mesh provided in input (Sec. 1.2). Each node in the mesh is handled as a particle with a mass, which is set to 1.0.

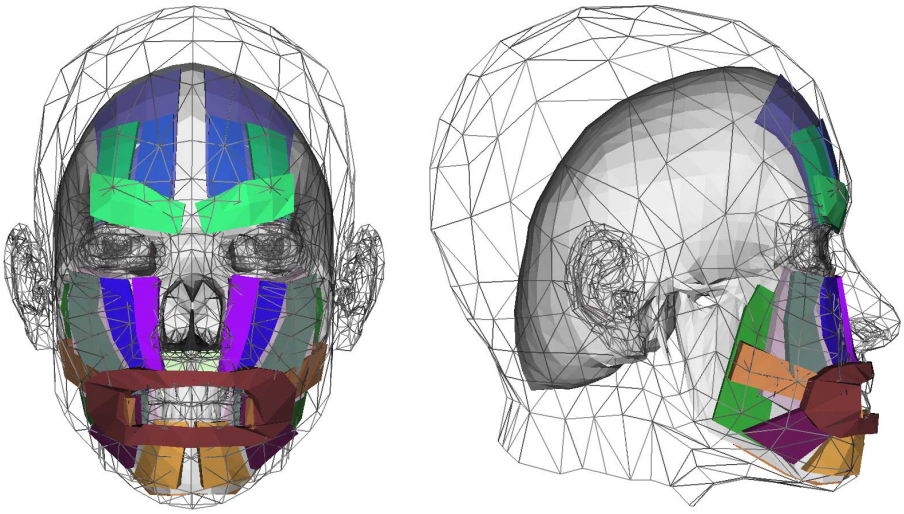


Fig. 5.12: Result of the morphing process. The front part skull and the muscle map, that is the movable part of the musco-skeletal system, are fitted inside the skin mesh.

A stretching constraint is placed along each edge of the triangular faces. For each pair of adjacent triangles $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ and $(p1, p_4, p_2)$, a further stretching constraint is placed along the opposing, not connected, particles p_3 and p_4, as shown in Fig. 5.13. This latter constraint penalizes to bending and twisting stresses. The target length is the euclidean distance among the particles in rest state. A bending constraint is defined on the two faces and the target angle is the dihedral angle among the two faces in rest state. A triangle area preservation is also imposed on each triangular face. Finally, a volume preservation constraint is defined over all the particles belonging to the skin. The target volume is the initial volume of the mesh in rest state. The value of the stiffness for each kind of constraint is found empirically by assessing the visual quality of the animation.

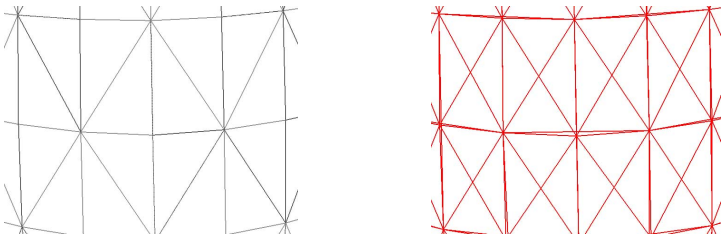


Fig. 5.13: Left. A close up of the input mesh. Right. The corresponding stretching constraints. Further bending constraints are placed along each edge shared by two triangles and an area preservation constraint is defined for each triangle.

The empirical values which produce the most plausible results are:

$$\begin{aligned} k^{stretching} &= 0.15 \\ k^{bending} &= 0.95 \\ k_{area} &= 0.95 \\ k_{anchor} &= 0.2 \\ k_{volume} &= 1.0 \\ damping &= 0.2 \end{aligned}$$

After the skull and the muscle map are fitted onto the skin mesh, further constraints are defined to bind the skin to the underlying musco-skeletal structure. For each particle p in the skin mesh, a ray is casted along the normal, that is towards the outer direction. In fact, after the fitting, portions of some muscles may stay outside the skin. By projecting in the outer direction, the skin vertexes are first bound to these muscles. If no intersection is found, then another ray is casted in the opposite direction of the normal, towards the inner part of the head. The ray in tested against the muscles from the most superficial to the deepest one. If the ray does not intersects any muscle then the skull is tested. The normal of the skin particle is considered as the average normal among the normals of the star of faces to which the particle belongs.

If an intersection is found, then it is defined a surface point sp on the intersected triangular face in the position where the ray intersects the face. A particle q is added to the system and it is bound through a position constraint to sp. A stretching constraint is placed among the particle p and q.

When the bones and the muscles move, the position of the surface points will change accordingly. The set of added particle q is updated as well because it is bound to the surface points through the corresponding position constraint and will displace the skin particles, whose final motion will depend also from the other involved constraints.

Since the main part of facial animation happens in the frontal part of the face, where the main part of the muscles and the jaw are, the deformation of the skin is not propagated to the back of the head. This is achieved by computing the normalized geodesic distance from all the skin particles which are anchored to a muscle or to the jaw to the remaining skin particles. The scalar values field produced by this process is called *influence map* and it is used as a weight for the displacement of the skin particles, as explained in Sec. 5.7. Fig. 5.14 shows the influence map as a color map on input skin meshes used in the experiments.

5.5.1 Eyelids

In a real head, the movements of eyelids are driven by the *orbicularis oculi* muscle. This muscle is composed by three parts: orbital, palpebral and lacrimal. Together they form a sphincter muscle of the eyelids, which causes forced closure motion, i.e. “screwing up” the eye, and closure without effort, like when blinking. The dynamics of the eyelids is very fast compared with the other facial muscles and the muscle model proposed in Chap. 4, is not appropriate to simulate its movements.

To simulate the eyelids, first the skin particles influenced by the orbilularis oris have to be found. This is done by detecting the border of the eyes. The border is formed by

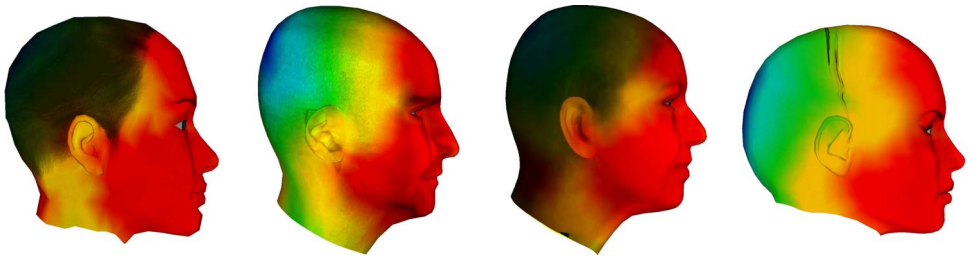


Fig. 5.14: Influence of the musco-skeletal structure on the skin, represented as a color map on different input skin meshes. Red are zones under direct control, blue with no control at all.

triangles having at least one edge which is part of only one triangle. the border is formed by all the vertices which belongs to these edges. Then, all the skin particles which have a *normalized* geodesic distance from the eye border lesser than 0.04 are considered belonging to the eyelids. In this step, it is used the classical Dijkstra's Shortest Path Tree algorithm [Dij59]. The geodesic distance is approximated by allowing to walk only along edges of the mesh.

For each eyelid particle p, a further particle q with an infinite mass is defined and placed in the same position of p. A position constraint is placed to bound the movement of p to the movement of q. During simulation, to animate the eyelids, the particles q are rotated and the particles p following them rotates as well. The rotation axis is defined by the landmarks $(3.7, 3.11)$ for the left eye and $(3.8, 3.12)$ for the right eye. Particles belonging to the upper and lower lids are chosen depending if the position of the particles lie on top or below the rotation axis.

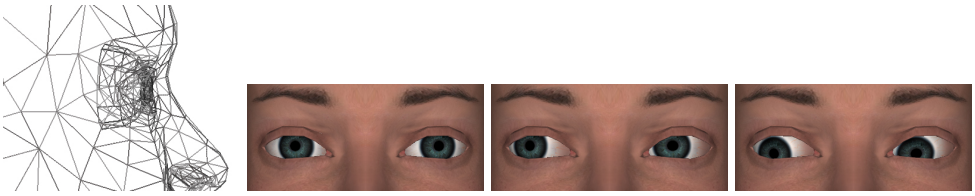
This gives a rather precise control on the movement of the eyelids and permits blinking and other subtle movements without inserting external parts in the skin mesh, like in [Käh03]. However, the geodesic threshold 0.04 is an empirical value. while it worked nicely for the test skin mesh that we employed in the experiments (Fig. 5.19), it is not guaranteed to work for any kind of skin mesh. Thus, the threshold can be manually tuned by the user in case of artifacts. In Fig. 5.15, it is shown an example of the movements of the upper and lower eyelids.

Fig. 5.15: Contraction of the upper eyelid (top row) and of the lower eyelids (bottom row).

5.6 Other facial parts

5.6.1 Eyes

In this model, eyes are an accessory part. Eye meshes are provided in input together with the skin mesh. In this face model, eyes are not animated through rotation of the relative geometries, instead the texture mapping coordinates are shifted accordingly to the desired motion. This choice has be done to allow a wide range of input eye meshes. For example, Fig. 5.16 show an example in which the eyes are modeled as half ellipsoid. If this geometry would rotate, empty space would be visible on the corner of the eyes, and probably it would penetrate with the eyelid. By shifting texture mapping coordinates, instead, the geometry remains fixed and the eyes move naturally.



(a) (b) (c) (d)

Fig. 5.16: (a). Eyes can be modeled as half ellipsoid. (b), (c), (d) Shifting texture mapping coordinates provides the illusion of eye rotation.

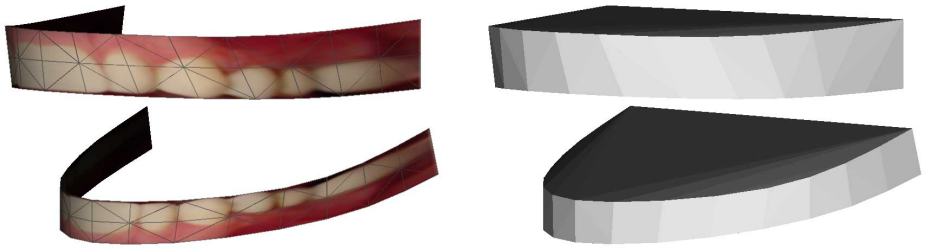
5.6.2 Teeth

As the eyes, teeth are optional part in this model. If the upper teeth mesh or the lower teeth meshes are present in the input skin mesh, then they are embedded into the model. The upper and lower teeth mesh are animated through the same rigid transformations applied to the bony structures. Lower teeth are tied to the jaw motion; when this later rotates or protrudes, the same transformation influences them too.

Beside aesthetic reasons, teeth are employed for collision detection and response against the lips of the face. Lips collide with the teeth when they stretch, for instance when the jaw depresses or the risorius muscles contract. In the preprocessing phase, the QHull [BDH96; qhu09] algorithm is run on the upper and lower teeth meshes to find the respective convex hull, which will be employed during the collision handling. Example of the output of this process is shown in Fig. 5.17. During animation, convex hulls are influenced by the same transformations of the respective teeth meshes.

During animation, immediately after the skin has been updated, the collision handling algorithm against the teeth is run. The algorithm is divided in two phases, broad and narrow. In the broad phase, each skin particle $\mathbf{p}$, with current position (p_x, p_y, p_z) is tested if it is enclosed in the axis-aligned minimum bounding box[2] of the teeth convex hull or not. This test is computationally cheap, as it is sufficient that $\mathbf{p}$ satisfies the inequalities:

[2]The minimum or smallest bounding or enclosing box for a point set in N dimensions is the box with the smallest measure (in this case volume)) which all the points lie within. The axis-aligned minimum bounding box for a given



(a) (b)

Fig. 5.17: (a). Example of upper and lower teeth meshes. (b). the corresponding convex hulls.

$$\begin{cases} (p_x \geq B_x^{min}) \wedge (p_y \geq B_y^{min}) \wedge (p_z \geq B_z^{min}) \\ (p_x \leq B_x^{max}) \wedge (p_y \leq B_y^{max}) \wedge (p_z \leq B_z^{max}) \end{cases}$$

where $\mathbf{B^{min}}$ and $\mathbf{B^{max}}$ are, respectively, the minimal and the maximal point which uniquely define the bounding box.

If $\mathbf{p}$ is not in the bounding box, then it is surely not colliding the teeth and discarded, otherwise the narrow phase is run. A ray is casted from $\mathbf{p}$. The direction of the ray is the intersection among sagittal and coronal (?) plane in case of the upper teeth, or the jaw protrusion direction (see Sec. 5.2) in case of the lower teeth. The intersection among the ray and the convex hull is tested with the ray-tracing algorithm reported in Sec. 4.4. If $\mathbf{p}$ is inside the convex hull, the intersection is not empty and collision is detected. In this case, the intersection point $\mathbf{q}$ is computed and a position constrint which bounds $\mathbf{p}$ to $\mathbf{q}$ is inserted in the set of collision constraints.

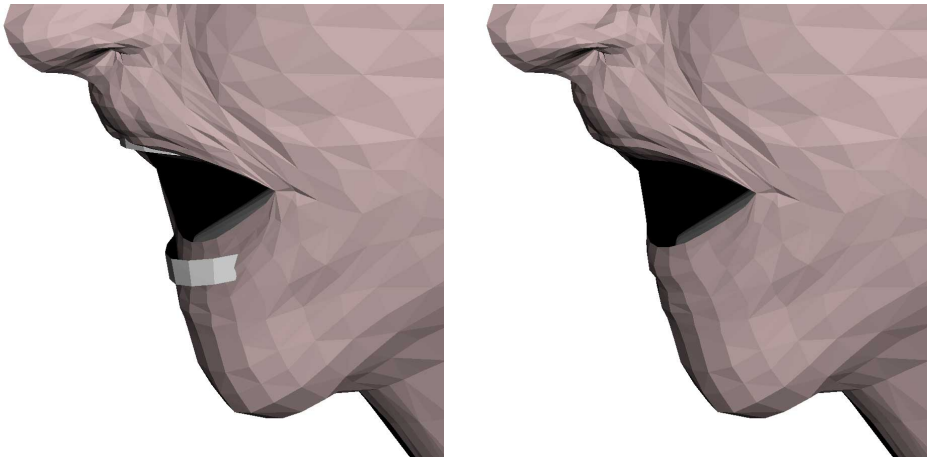
Note that in this collision handling algorithm, only spatial displacements are employed, without inserting any impulsive force in the computation. The physical simulation remains stable because it is of position based type. In Fig. 5.18, it is shown the outcome of this algorithm in case of jaw depression and contraction of the zygomaticus major muscles.

5.7 Animation Algorithm

The animation is carried out through the simulation of each anatomical part in a strict sequential order, which can be summarized by the following algorithm.

(1) **loop**
(2) **update** state of the bony jaw pitch, yaw and protrusion values)
(3) **for each** muscle **do**
(4) **update** contraction value

point set is its minimum bounding box subject to the constraint that the edges of the box are parallel to the (Cartesian) coordinate axes.



(a) (b)

Fig. 5.18: (a). Collision handling disabled, lower teeth mesh comes out from the chin. (b). Collision handling enabled, chin and lips deforms in a natural way.

(5) **for each** eyelid particle
(6) **update** rotation angle

(7) apply rigid transformations to the bony jaw and teeth
(8) **for each** muscle (from the deepest to the more superficial) **do**
(9) **for each** surface points belonging to the lower contour **do**
(10) refresh position according to the underlying triangular face
(11) run one PBD iteration on all the constraints

(12) **for each** skin particle **do**
(13) refresh the corresponding anchor position
(14) detect collision with teeth
(15) run one or more PBD iterations over the skin, including collision constraints
(16) **for each** skin particle **do**
(17) apply influence weight: $\mathbf{p}_i \leftarrow \mathbf{x}_i + weight_i \mathbf{\Delta p}_i$
(18) **for each** eye **do**
(19) shift texture mapping coordinates
(20) **for each** mesh **do**
(21) update normals and axis-aligned bounding box
(22) **end loop**

Lines (2) - (6) update the animation state of the musco-skeletal structure; variables like the pitch of the jaw or the contraction value for each muscle are set. The core simulation steps are in lines (7) - (15). Starting from the skull, which is the deepest layer, all the anatomical structures are updated in sequence. The position of surface points which lay on a layer are

Mesh name	Vertices	Faces	Constraints	Render Time (ms)	Phys Sim Time (ms)	fps
Masha	1466	2822	16461	1	31	31
Marco	2502	4908	28785	1	28	34
Girl	4604	9160	53834	1	45	22
Reana	4904	9541	55687	2	50	19
Skull + Muscles	4297	8352	4919	5	6	90

Table 5.1: Meaningful numbers for each input skin meshes and the musco-skeletal structure.

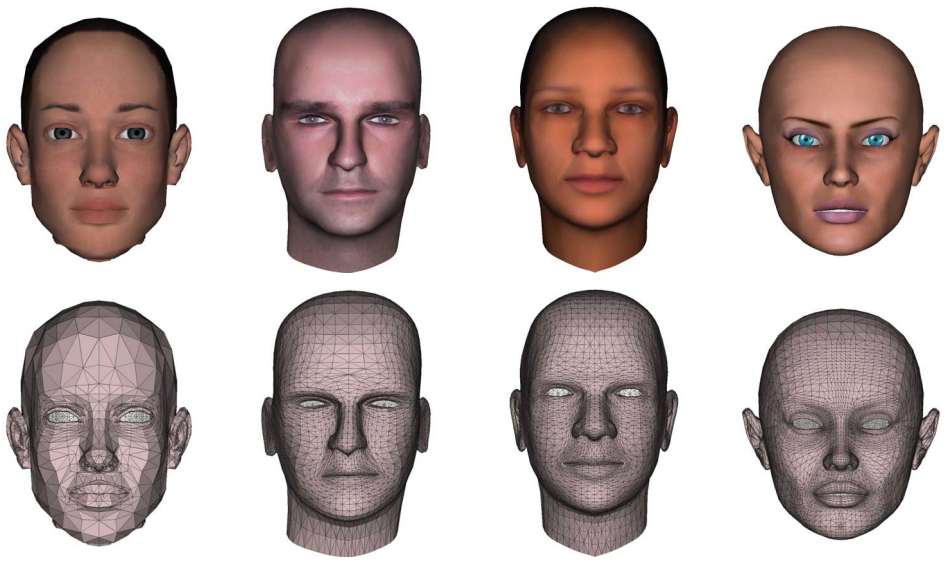
updated as well and influences the deformation in the next layer. After surface points on a layer are updated, PBD iterations are run on the next layer which displace surface points for the upper layer. It is used a fixed time step of 16.6 ms. This sort of chain reaction traverses all the muscle map and finally reaches the skin which is deformed according the underlying musco-skeletal structure. Note that in lines (16) - (17), the displacement Δp_i, which is caused by the PBD simulation, is weighted by the influence map, in order to not propagate the deformation on the back of the head. As a last step, the eye are animated as well in lines (18) - (19) and the loop iterates.

5.8 Results

The whole facial model has been implemented in a real-time software framework, using mostly C++ and OpenGL, beside other open source technologies (see Appendix 7.2.3 for further details). The presented results have been recorded using a consumer class platform, an Intel Core2 Duo CPU at 2.40 GHz, with 2 GB of main memory and a NVIDIA Quadro FX 360M video card. The prototype software run on a single thread, thus only one core is used. The visualization is done through OpenGL vertex arrays because the position of the skin vertices may change in any frame.

The system is visually evaluated by producing the various facial expressions with different target skin meshes, in particular the basic expressions [EFE72]: anger, disgust, fear, joy, sadness and surprise. The time performance is measured as well. The skin meshes used in the experiments are shown in Fig. 5.19. In Table 5.1, are gathered the main numbers which characterize them (number of vertices, triangles, constraints) together with the time performance of the simulation.

Note the highest render time of the skull and muscles compared with skin meshes, even though they have together an equivalent number of drawing primitives than the skin meshes. This is due to the fact that each muscle is rendered with a single drawing call, while the skin meshes need only one. On modern graphics cards, in fact, it is convenient to batch primitives in as less as possible drawing calls [Wlo03] for having fast renderings. In the preprocessing phase, most demanding step is the definition of the muscle map which has been designed in less than one day of work and then have been used for all the input skin meshes. Once the muscle map is ready, time for fitting the musco-skeletal structure into the skin is far less than 1 second, while time for binding to the skin, which involves a ray tracing for each skin vertex,



(a) (b) (c) (d)

Fig. 5.19: Smooth and flat shaded renderings of the skin meshes used in the experiments. (a). Masha, (b). Marco, (c). Girl, (d) Reana.

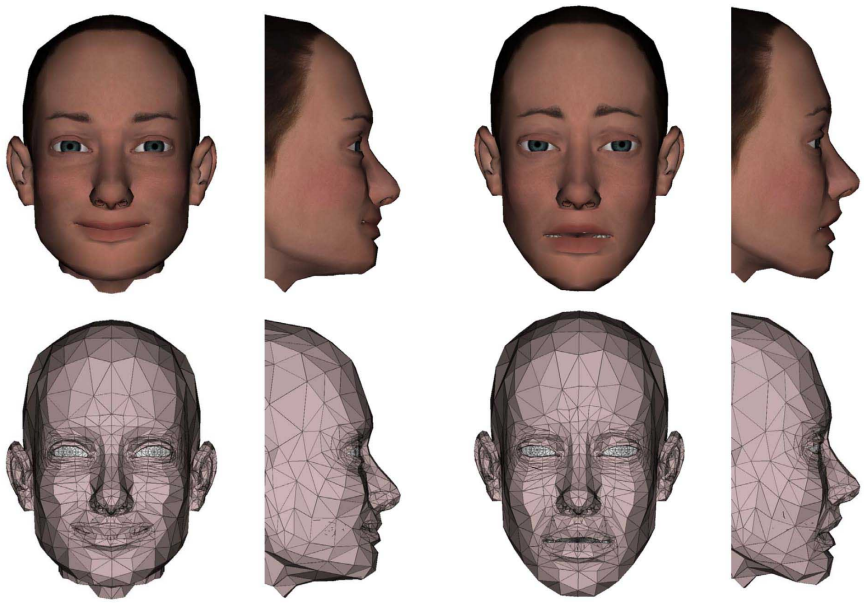
is less than 5 seconds for each input skin mesh.

From Fig. 5.20 to 5.31, there are shown several face models built from the input skin mesh while performing the basic facial expressions.

5.9 Discussion

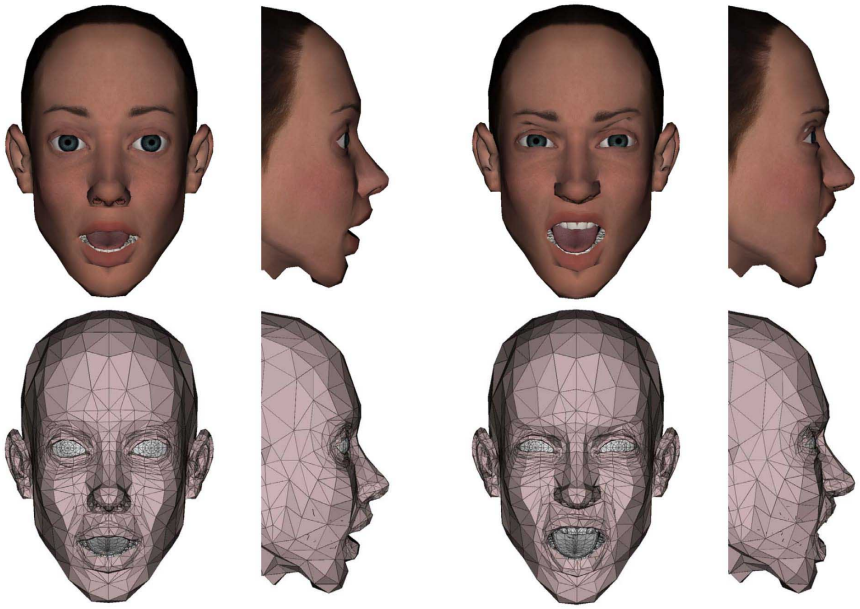
The presented computational face model is flexible enough to adapt to different target skin meshes and animate them. This is due to the fact that the simulation is carried out through Position Based Dynamics. There are not external or internal newtonian forces involved, the facial motion is caused by geometrical constraints and spatial displacements applied to specific parts of the model. To contract a muscle, it is sufficient to re-sample its action lines according to a normalized control variable (Sec. 4.3), and the deformation is propagated to the upper layers. Actually, this method can be easily generalized to simulate the facial dynamics of heads different from humans, for example cyclops or animals.

The stiffness values of the constraints, as well the mass values of the particles, are found empirically by assessing the visual quality of the animation. Since these values are normalized among zero and one, the parameters are suitable for virtual heads of different scale and shape. However, they can be adjusted interactively by the user through the design tool (App. 7.2.3). There is not a correspondence among these values and the bio-mechanic parameters which define the real skin material, like for example the Young modulus. The geometric constraints used in PBD does not have a correspondence with the differential equations which govern the dynamics of the human head, thus their configuration must be chosen in such a



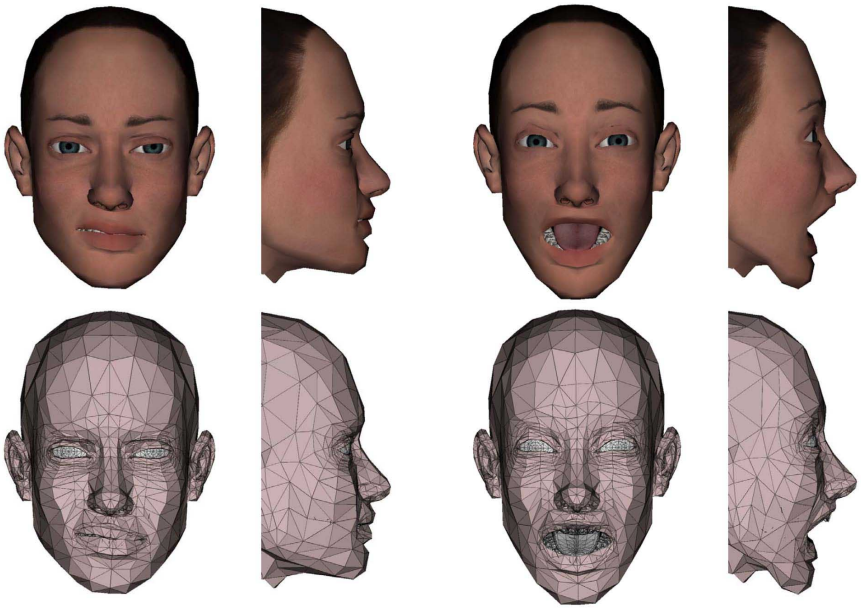
(a) (b)

Fig. 5.20: Masha, (a). Joy. (b). Sadness.



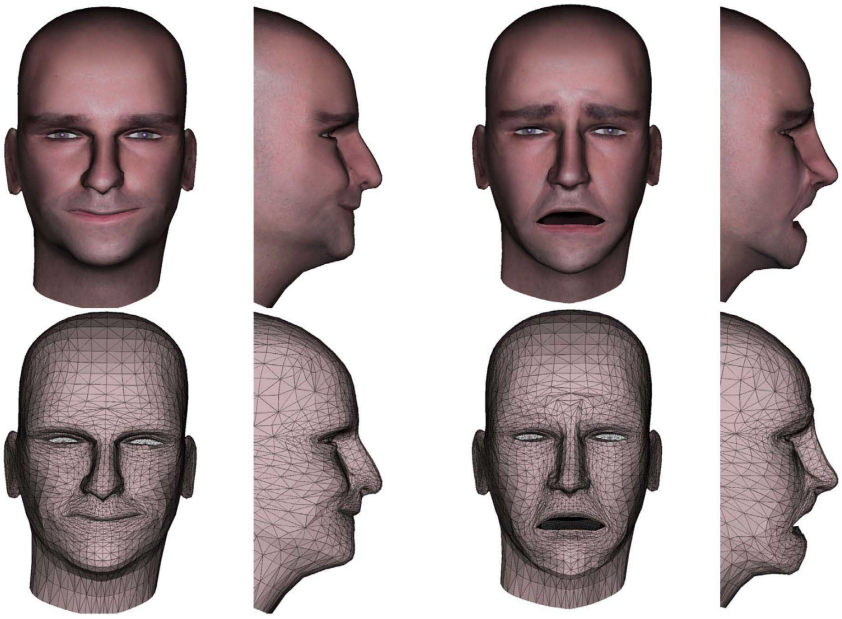
(a) (b)

Fig. 5.21: Masha, (a). Surprise. (b). Anger.



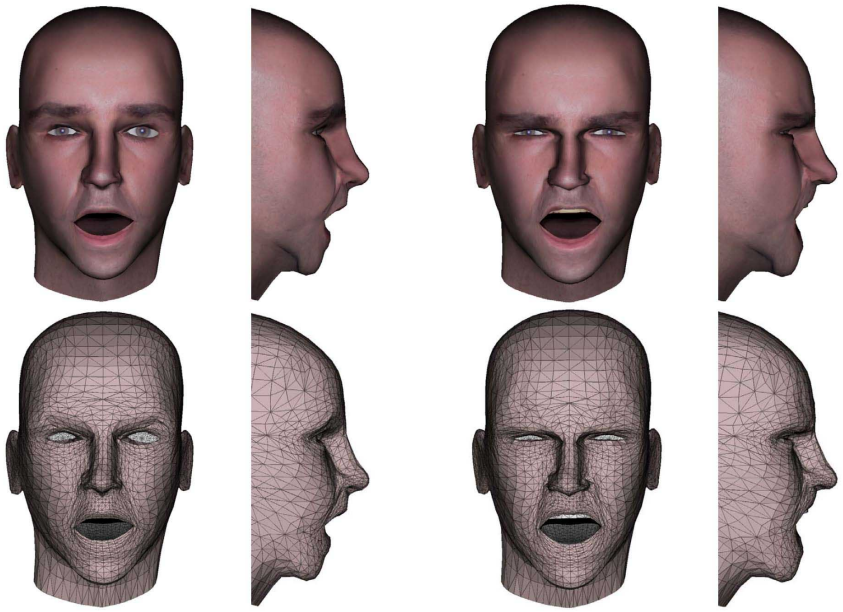
(a) (b)

Fig. 5.22: Masha, (a). Disgust. (b). Fear.



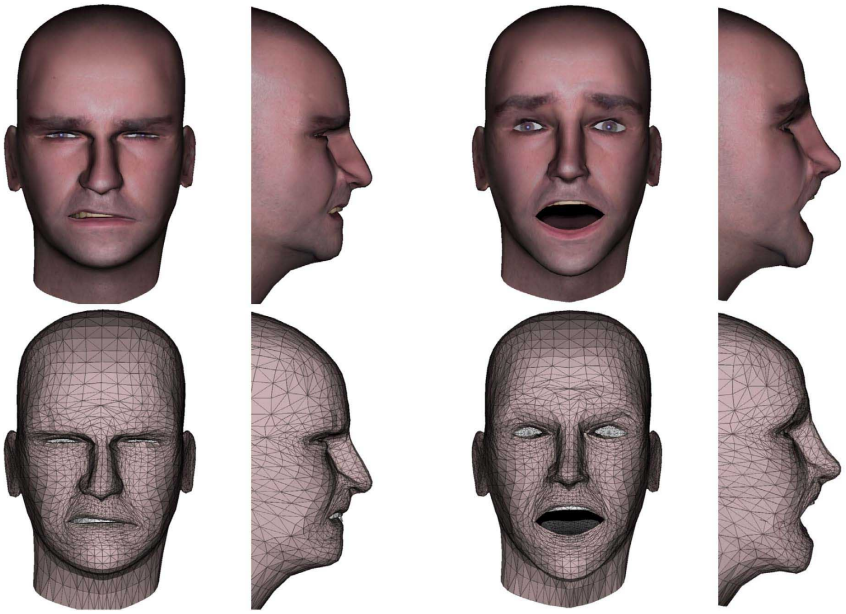
(a) (b)

Fig. 5.23: Marco, (a). Joy. (b). Fear.



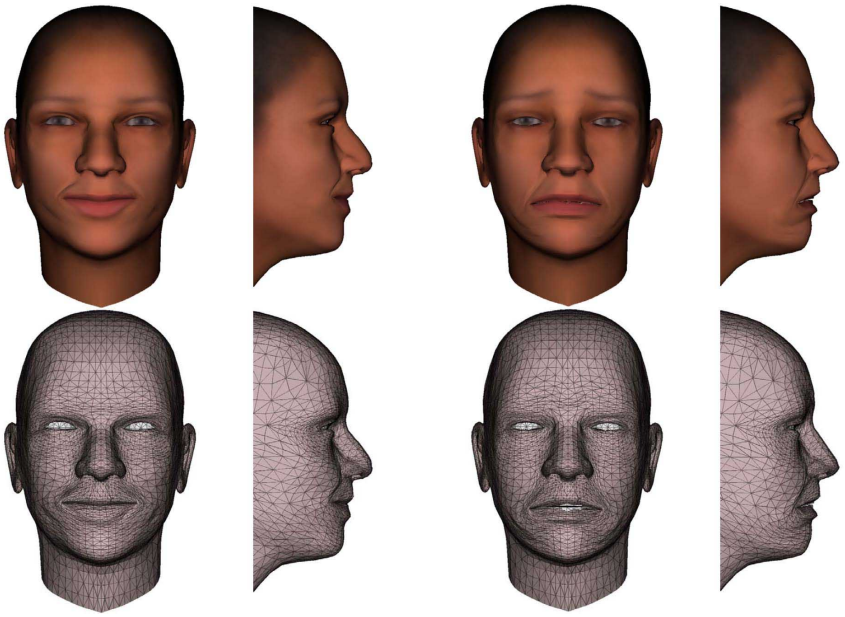
(a) (b)

Fig. 5.24: Marco, (a). Surprise. (b). Anger.



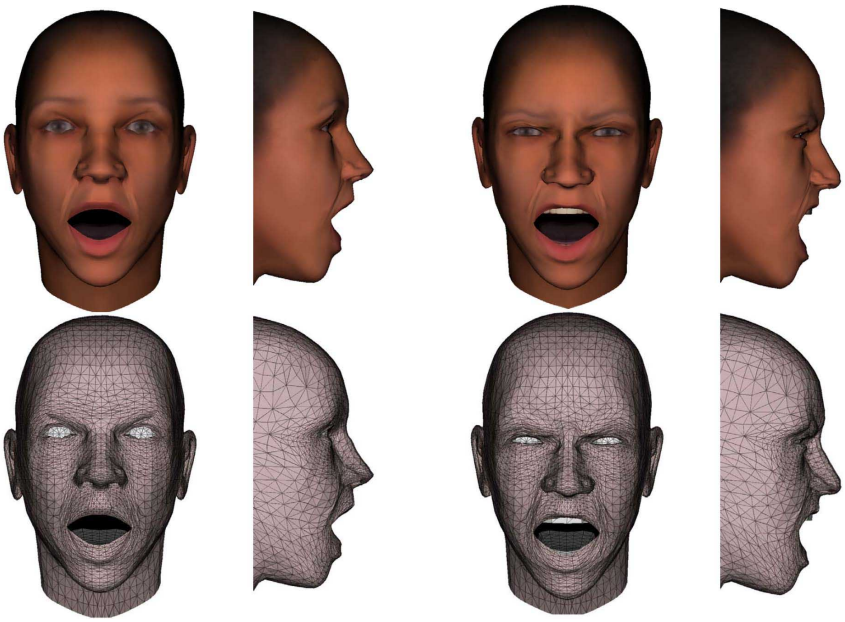
(a) (b)

Fig. 5.25: Marco, (a). Disgust. (b). Fear.



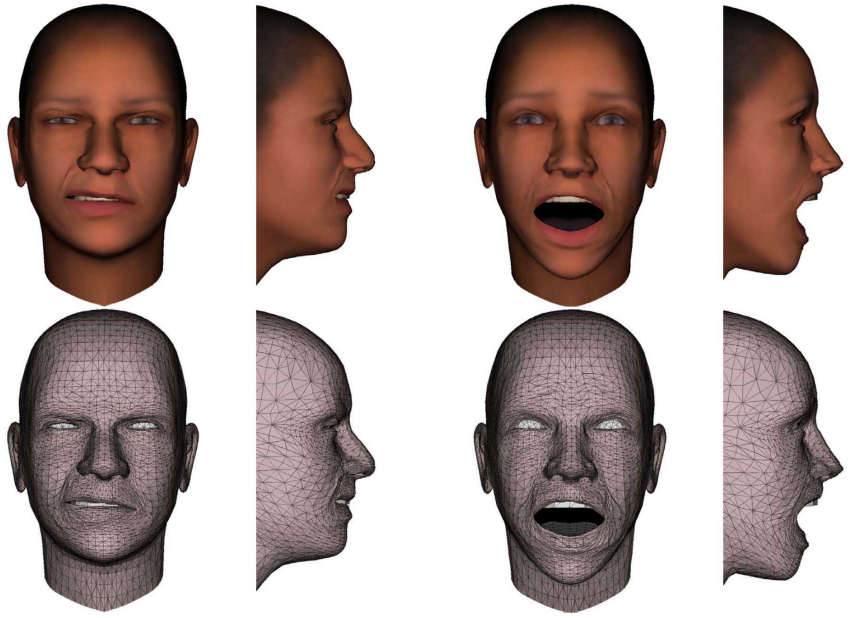
(a) (b)

Fig. 5.26: Girl, (a). Joy. (b). Sadness.



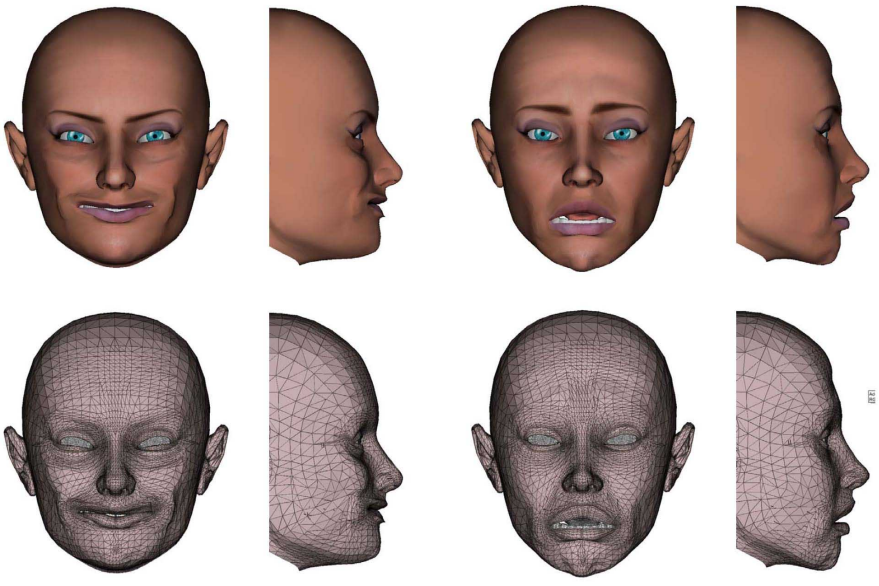
(a) (b)

Fig. 5.27: Girl, (a). Surprise. (b). Anger.



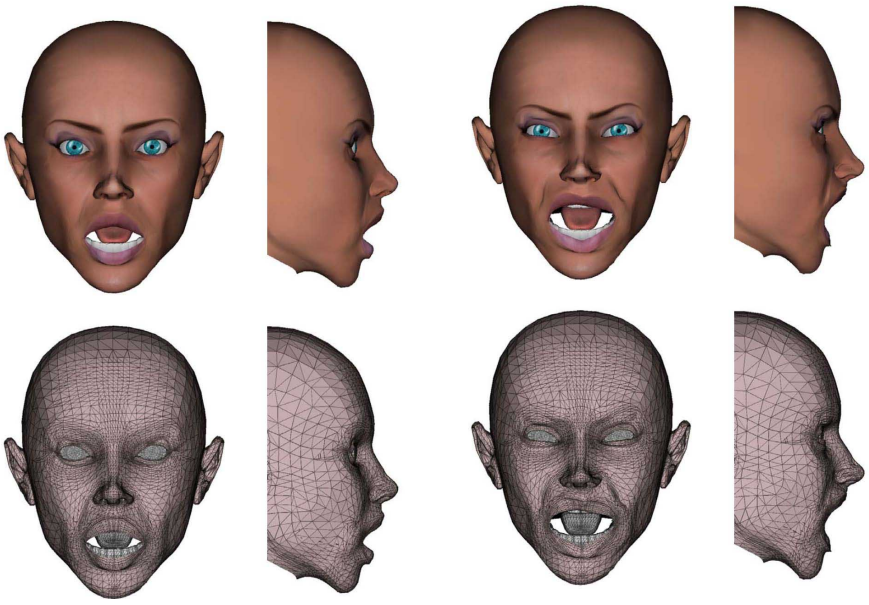
(a) (b)

Fig. 5.28: Girl, (a). Disgust. (b). Fear.



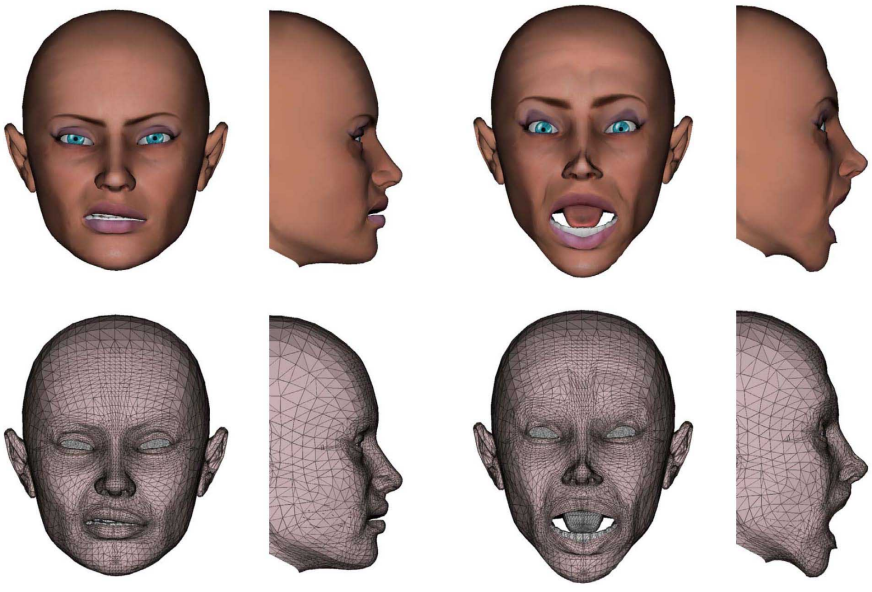
(a) (b)

Fig. 5.29: Reana, (a). Joy. (b). Sadness.



(a) (b)

Fig. 5.30: Reana, (a). Surprise. (b). Anger.



(a) (b)

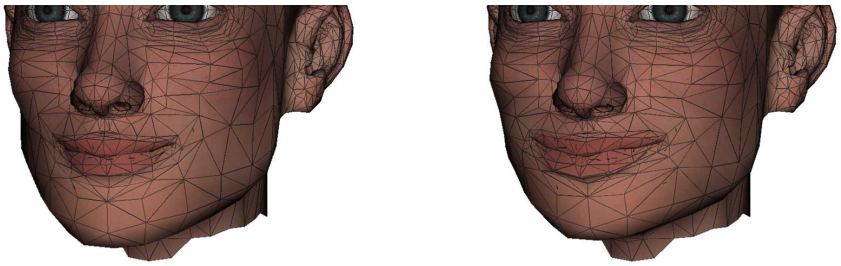
Fig. 5.31: reana, (a). Disgust. (b). Fear.

way to reproduce the macro behavior of the living tissues. For example, the skin resists to bending and twisting stresses. This is modeled with high values in the bending constraints and further stretching constraints placed among opposing particles in adjacent faces (Fig. 5.13). The skin is mostly composed by water and collagen and thus it conserves its volume during deformation; this is modeled through the constraints defined on all the triangular facets and with the volume preservation constraint defined on all the skin particles. In this way, the inversion of the normal faces is strongly penalized and the surface of the skin remains smooth while deforming; the animated skin meshes are thus suitable for accurate off-line renderings.

The polygonal resolution of the skin meshes influences the speed with which the simulation reach a steady configuration. The finer the resolution, higher the number of constraints and slower the convergence of the Gauss-Seidel solver. This can be compensated by using an higher number of PBD iterations in the projection of the constraints of the skin; in this case the frame rate drops, however interactivity is maintained. I experimented an alternative approach by using an improved version of the simulation algorithm, called Hierarchical Position Based Dynamics, recently proposed by Müller in [MÖ8]. This approach works well for flat pieces for clothes; however for curved meshes, like the skin, does not produce believable motion and it needs further investigations.

In the face model, there is not a clear support for the lips. This may generate artifacts when lips are stretched, like in Fig. 5.29a). One line of future work to fix this issue, is the improvement of the skin model by transforming the triangulated skin mesh into a tetrahedral mesh and then defining a volume preservation constraints for each tetrahedra.

The muscles and the cartilages are sketched directly on the already existing musco-skeletal structures (bones and other muscles), and then it is fitted into the target skin mesh. This allows for designing a complex facial structure in which there are different layers of muscles and deformable tissues, like the fatty tissue under the cheeks or supporting muscles like the masseter muscle which spans between the zygomatic bone and the jaw. The deformation of the cheeks is thus more plausible, for instance while smiling or in the anger expression (Fig. 5.32).



(a) (b)

Fig. 5.32: The Masha skin mesh smiles with a thin (a) and a thick (b) fatty tissue under the cheeks.

Among the work already existing in literature, the research of Kähler [Käh03; KHS01; KHYS02; KHS03; KÖ7] is the most similar to the face model presented here, however there are important differences.

In Kähler, the muscles are defined on a template face surface, built for this purpose, and

then they are bound to the underlying skull. This limits the applicability of such approach because a different muscle model shall be designed for each different virtual head. This involves a rather big amount of manual work, considering that a mass-spring network is used to solve the dynamics, and thus the physical parameters (e.g. stiffness) should be redefined as well according to the spatial dimensions of the virtual face. To solve such a problem, Kähler deforms the shape of the template head to match different humans; however, he still animates always the same template mesh.

In my model, instead, facial muscles are drawn directly on the skull as explained in Chap. 4. This allows to define any number of muscular layers (the total number of layers is a choice of the muscle designer); the most superficial muscles are thus passively deformed when the underlying ones contract leading to a more realistic animation. Furthermore, it allows to apply the musculoskeletal model (skull, muscles and passive tissues), to different virtual heads which become ready to be animated.

Such a flexibility is also due to the use of Position-based Dynamics (PBD), instead of mass-spring networks. PBD and mass-spring networks are both techniques to physically simulate particle systems at interactive rate but are rather different from each other. With mass-spring networks, it is possible to define only local relationships between the particles (e.g. a spring among two particles). In PBD, instead, it is possible to define both local and global relationships (e.g. area and volume preservation), and, unlike in springs, the stiffness of constraints is normalized between 0 and 1. These factors make PBD more feasible than mass-spring networks to model the sophisticated biomechanical behavior of the facial muscoskeletal system of faces with very different size and shape.

Differently from Kähler, the fat is explicitly modeled as passive deformable tissue. For instance, see the cheeks in the muscle map in Fig. 5.7 and in Fig. 5.32. Actually, it would be possible to model also other types of passive tissues, like the cartilages of the nose. However, this requires further investigation.

To solve particles' positions, the Gauss-Seidel iterative method is used. Even though this numerical technique is simple, easy to implement and with a low computation cost, in some cases it may lead to small instabilities. However, it is important to note that the main purpose of the facial model is to automatically synthesize blend shapes, so the small instabilities that may happen during the simulation are not so critical.

The purpose of the facial model is not in real time animation system for facial animation; this would have not a practical application in the real world because it would involve a big computational cost for a real time application, like any other existing technique in literature apart linear interpolation. The main purpose of this facial model is to produce facial blend shapes by editing muscle configurations, which then can be animated through interpolation, making easier the work of artists and strongly reducing the production time.

The whole mechanism makes the model robust and computationally cheap, and suitable for directly producing facial animation, or to support the artist in producing the blend shapes for key frame interpolation.

Chapter 6

Facial Motion Cloning

6.1 Description

In this chapter, it is described a Facial Motion Cloning (FMC) method that is used to copy the facial motion from one face to another. The facial movements are represented by a set of morph targets encoded by the MPEG-4 Facial and Body Animation (FBA) standard. A morph target is a variation of the face that has the same mesh topology, but different vertex positions. Essentially, it is the source face performing a particular key position. Each morph target corresponds to a basic facial action encoded by a MPEG-4 FBA parameter. The linear interpolation of the morph targets is able to represent a wide range of facial movements. Thus, an artist could produce one detailed face including all morph targets, then use this FMC implementation to quickly produce the corresponding full set of morph targets for a new and completely different face. The morph targets of the source face can be produced manually by artists, by motion capture techniques or in automatic way, for instance by applying the anatomical model presented in the previous chapters.

There are two main advantages in using morph targets encoded by MPEG-4 FBA. The first one is in the low requirements of CPU-time usage, since the animation is achieved by linear interpolation of the morph targets. The second one lies in the capability of the MPEG-4 FBA to express and precisely control the whole range of facial expressions [PF02]. Thus, by computing the morph targets for the target face, it is possible to use them to perform generic face animation encoded in a MPEG-4 FBA stream in a computationally cheap manner.

6.2 Algorithm Overview

The FMC method is schematically represented by Figures 6.1 and 6.2 and can be summarized as follows.

Fig. 6.1: Facial Motion Cloning mechanism. A RBF volume morphing $G(P)$ is performed between the source and the target face.

Fig. 6.2: Facial Motion Cloning mechanism. Using the same deformation function $G(P)$, all the source morph targets are cloned to the target face.

Given a manually picked set of 48 feature points on the input face meshes, it is computed a scattered data interpolation function $G(P)$ employed to precisely fit the shape of the source into the shape of the target mesh through a volume morphing. Then, each vertex of the target face is mapped to the corresponding triangular facet of the deformed source mesh in its neutral state through a proper projection. The target vertex position is expressed as a function of the vertex positions of the source triangular facet through barycentric coordinates. At this point, all the source MTs are deformed by applying to each one of them the same morphing function $G(P)$ used to fit the neutral source into the neutral target mesh. Hence, the new position of each target vertex is computed considering the location of the corresponding deformed source facet, obtaining the target MT. The whole set of morph targets is called Animatable Face Model (AFM) and it can be directly animated by commercial MPEG-4 FBA players [VT:09].

Inputs to the method are the source and target face meshes. The source face is available in neutral state as defined in the MPEG-4 FBA specification. The morph targets (MTs), corresponding to the MPEG-4 FAPs, of the source face are available as well. The target face exists only in the neutral state. For each neutral face mesh, the corresponding MPEG-4 Feature Definition Point (FDP) set must be defined, that is, each FDP is manually mapped onto a vertex of the input face meshes. We need 48 out of 84 FDPs because the FDPs corresponding to group 10 (ears), 6 (tongue), FDPs from 9.8 to 9.11 (teeth) and 11.5 (top of the head) are not considered. The process to manually map the 48 FDPs on the input faces requires 10-30 minutes according to the user skills. The goal is to obtain the target face with the motion copied from the source face. No assumption is made about the number of vertices or their connectivity in the input models beside the fact that the triangulated meshes are 2-manifold and orientable.

The task of the shape fitting process is to adapt the generic source face mesh to fit the target face mesh. The input are the source and target face meshes. The source and the target faces are both available in neutral position as defined in the MPEG-4 FBA specification (Section 3.5 and [PF02]). For each neutral face mesh, the corresponding subset of 48 MPEG-4 Facial Definition Points (FDPs) must be defined as specified in Section 6.2. Taking these feature points as a reference, some face features are extracted, that is, the eye contour and the inner lip contour (Section 6.2.1).The FDPs are used together with the vertices belonging to the eye and lip contour features to find a first guess of the scattered data interpolation function $G(P)$ that roughly fits the source into the target mesh (Section 6.2.2). The precise fitting of the moving zone of the deformed source model is obtained by iteratively refining $G(P)$ (Section 6.2.3).

6.2.1 Eye and Lip Feature Extraction

Starting from the manually picked MPEG-4 FDPs, the eye and lip features are extracted from the input meshes through automatic algorithms.
Driven by some of the MPEG-4 FDPs belonging to the eye group, a proper path is found in the mesh graph going from a start point to an end point, where the start and the end point are defined according to the round robin UP $\Rightarrow$ LEFT $\Rightarrow$ DOWN $\Rightarrow$ RIGHT $\Rightarrow$ UP. To find the path between the start and the end point, a greedy strategy is employed:

```
    Right Eye FDPs: 3.2 ⇒ 3.8 ⇒ 3.4 ⇒ 3.12 ⇒ 3.2
Left Eye FDPs:  3.1 ⇒ 3.7 ⇒ 3.3 ⇒ 3.11 ⇒ 3.1

v = start_point
while (v ≠ end_point)
  d_min = very big value
  for each neighbor n of v do
    n_n = normalize (n - v)
    d = distance ((v + n_n), end_point)
    if (d_min > d)
      d_min = d
      v_min = n
    end if
  end for
  insert v_min in the eye contour set
  v = v_min
end while
```

The inner lip contours are found by applying the following algorithm, once for the upper lip and once for the lower one:

```
start_point = FP 2.5 (inner right corner lip)
end_point = FP 2.4 (inner left corner lip)
direction = end_point - start_point
direction_xy = (direction_x, direction_y, 0)
v = start_point
insert v in the lip contour set
while (v ≠ end_point)
  for each neighbor n of v
    a = angle (direction_xy, (n - v)_xy)
  end for
  v = n having smaller [greatest] a, a∈ (-Π/2, Π/2)
  insert v in the upper [lower] lip contour set
end while
```

Note that a MPEG-4 FBA compliant synthetic face has the gaze and the nose tip towards the positive z-axis and that the lips are in contact but they are not connected [PF02]. The correctness of these extraction algorithms is not assessed, however they work in a satisfactory way with all our test face models.

6.2.2 Scattered Data Interpolation

Having computed the face feature points on both the source and the target face mesh in their neutral state, it is built the smooth interpolation function $G(P)$ that fits precisely the source model into the target face model according to the process depicted in Sec. 3.4.

$G(P)$ can be computed once the correspondence set $\{(P_i, Q_i)\}$ has been defined. The denser the correspondence set is the closer the resulting fit. The 48 FDPs specified in Section 6.2 are considered as a first guess of the interpolation point set $\{(P_i, Q_i)\}$. $G(P)$ is computed and applied to the source mesh obtaining a rough fitting. Then, the correspondence set is enriched by inserting the vertices belonging to the eye and lip contours of the source and the point lying on the nearest edge of the correspondent target contours. $G(P)$ is recomputed with the enriched set of correspondences and applied again on the source mesh in its neutral state obtaining again a rough fitting but this time with a correct alignment of the eye and lip features.

6.2.3 Correspondence Set Refinement

After morphing the source face mesh to roughly fit the target, the fitting is improved by specifying additional correspondences. A vertex P of the source face is called a *movable* vertex, if its position in one of the morph targets is not equal to its position in the neutral face. Thus, such a vertex will potentially move during the animation. The movable vertices are projected from the deformed source mesh on the target face surface by casting rays with a fixed length along the vertex normals[1] and compute the intersection of the ray with the target mesh. A fixed length of $ENS0 * 0.3125$ is used for the casted rays, where ENS0 is the MPEG-4 FAPU defining the distance between the eye and nose.

By doing this, for each movable vertex of the source face P_i, the corresponding intersection point Q_i on the target surface mesh is found. Having chosen a fixed length, only a part of the movable vertices will have a correspondent point on the target surface. This is because, after the initial rough fit, only the nearest vertices will be close enough to the target mesh surface to permit a ray-facet intersection. I also experimented outward rays with unlimited length. Better results have been achieved with the fixed ray length, probably because in this way the interpolated surface slowly stick to the target surface without inserting high-frequency elements.

The first 125 moving vertices having greatest error $e_i = \|Q_i - P_i\|$ are considered the pair (P_i, Q_i) is inserted in the correspondence set. If a point P_i is already in the set, then only the position of the correspondent Q_i is updated. The linear system (3.31)-(3.35) is solved again to find $G(P)$ for the enriched correspondence set and the scattered data interpolation algorithm is re-run to update the whole source face model from its neutral state. This process is iterated until no more movable vertices are inserted in the correspondence set. This may happen for two different reasons:

[1] The normal of a vertex is considered as the average of the normals of the faces the vertex is part of.

- all the movable vertices have been inserted in the correspondence set;
- the actual set has enough correspondence pairs to fit carefully the source to the target mesh.

By applying the refined $G(P)$ to all the vertices of the neutral source face mesh, this latter fits precisely the target mesh in the area where there are vertices that potentially move when the animation is performed.Only the moving zone is considered because the task is to copy the source face motion and there is no interest in the static zones. Figure 6.3 shows an example of this iterative fitting process for some test models. In the central column, the source face is solid while the target mesh is wireframe rendered. Step 0 is the initial rough fit. In Step 1, the eye and the lip features are aligned. After some iterative steps, the source fits to the target face in the moving zone.

Fig. 6.3: Source shape fitting iterative process.

6.3 Cloning Process

The input of the motion cloning process is formed by the scattered data interpolation function $G(P)$ just refined and the morph targets, corresponding to the MPEG-4 FAPs, of the source face. To be precise, the algorithm need the 64 MTs corresponding to low-level FAPs and the 20 MTs corresponding to high-level FAPs (14 visemes and 6 emotions), for a total of 84 MTs. The target vertices are mapped on the deformed source mesh through the same projection used in Section 6.2.2, this time casting the fixed-length rays from the target vertices towards the deformed source mesh. At this stage of the process, the deformed source and the target meshes are very similar to each other and each target vertex can be considered as located where the casted ray intersects the proper triangular face of the source mesh. Thus, the barycentric coordinates are computed for each target vertex considering the vertices of the correspondent source triangular facet. The position of the target vertices can be expressed as a linear combination of the positions of the three corresponding source vertices.

Then, for each particular source MT S_i, the corresponding target MT T_i is obtained by applying the following algorithm:

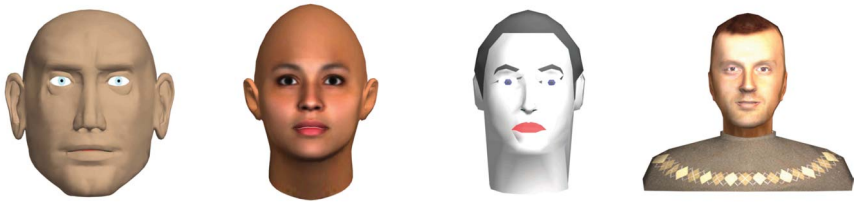
```
T_i = neutral target face (stored in T_0);
apply G(P) to S_i only in each vertex P ∈ D_i;
for each vertex Q ∈ T_i
  if (P_a ∈ D_i) ∨ (P_b ∈ D_i) ∨ (P_c ∈ D_i)
    Q = P_a · b + P_b · c + P_c · a;
end for;
```

where D_i is the set of moving vertices of S_i, P_a, P_b and P_c are the vertices of the source triangular facet corresponding to the vertex Q and a, b and c are the proper barycentric coordinates. Note that the scattered data interpolation function $G(P)$ is applied *only to the movable vertices* D_i of each particular source MT S_i in order to speed up the cloning process. That is, the same global deformation function $G(P)$ used to fit the source into the target, is applied to the local part of the source MTs S_i that will move when S_i is employed during the animation. Then, the position of the corresponding target vertices is computed by linear combination of the barycentric coordinates with the new positions of the source triangular facet obtaining, finally, the resulting target MT T_i.

As a final step, the global motion of the head is copied as well as the movements of the tongue and teeth (if present) through affine transformations like in Pandzic [Pan03].

6.4 Results

Experiments have been performed on an Intel Pentium M 1,73 GHz processor with 512 MB RAM. Some of the used face models for the experiments are shown in Figure 6.4. The test models have different polygonal resolutions, shapes and connectivity, both symmetric as well as asymmetric. A full set of high- and low-level FAP motions (i.e. morph targets), was



(a) (b) (c) (d)

Fig. 6.4: Test models used in our experiments. (a) `joakim`, 909 vertices and 1752 faces; (b) `beta`, 2197 vertices and 4118 faces); (c) `data`, 367 vertices and 677 faces; (d) `kevin`, 498 vertices and 956 faces.

available for each model. All motion was cloned from model `joakim` and `beta` to each other model, producing the grids of cloned animated models for each motion in Fig. 6.5-6.8. Looking at each row shows how an expression is cloned from one face to all the other faces; looking at each column shows how the expression is cloned onto the same face from different sources.

The comparison in terms of computation time between Pandzic's method [Pan03] and this algorithm is straightforward, since it is used the same input and produced the same kind of output. In Table 6.4, it is presented the computation time for some cloning processes performed during the tests along with other interesting data. For comparison, the last column presents the computation time for Pandzic's cloning approach.

To assess the quality of this approach, the same source animatable faces have been cloned to themselves and the error between the source and the output model is computed as the

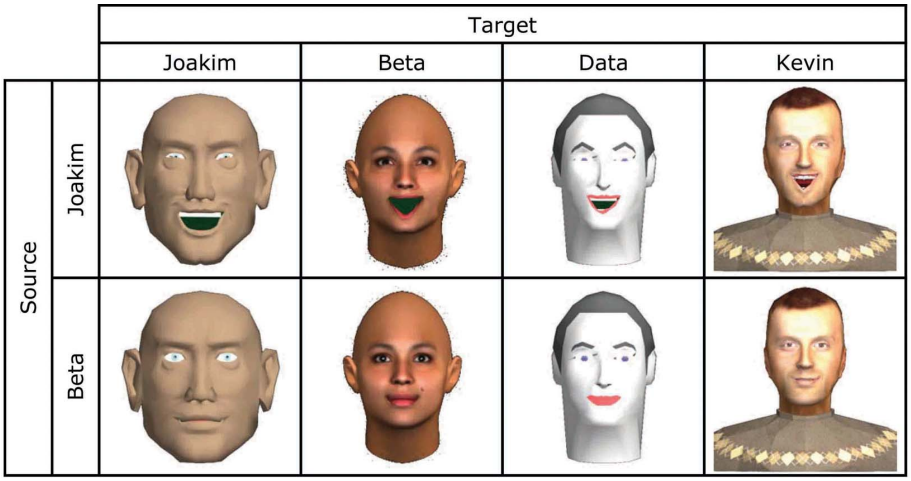


Fig. 6.5: Cloning grids for expression joy.

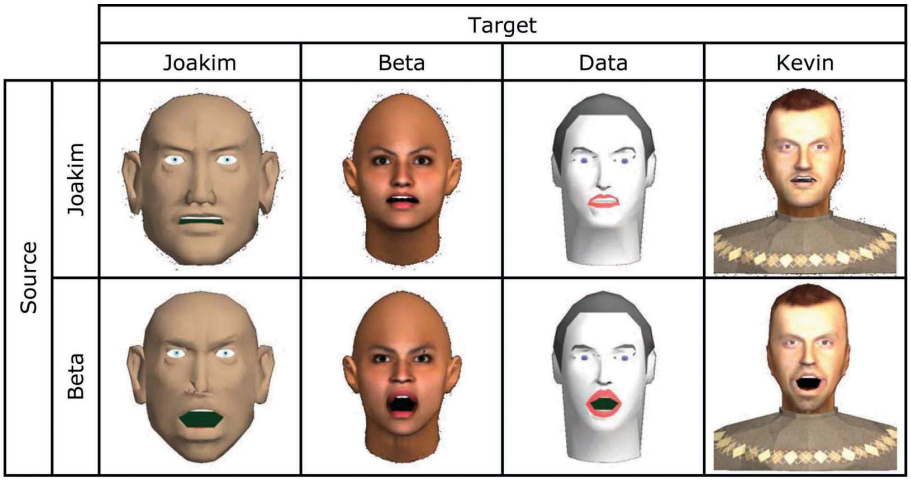


Fig. 6.6: Cloning grids for expression anger.

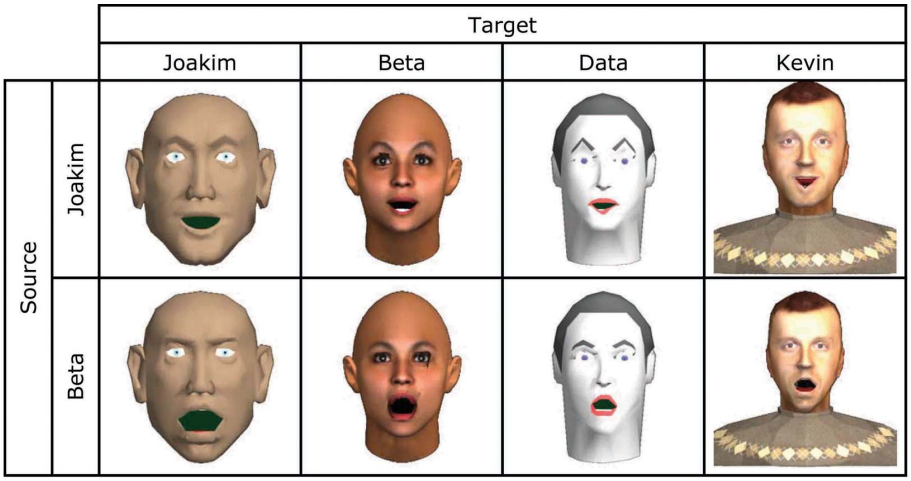


Fig. 6.7: Cloning grids for expression surprise.

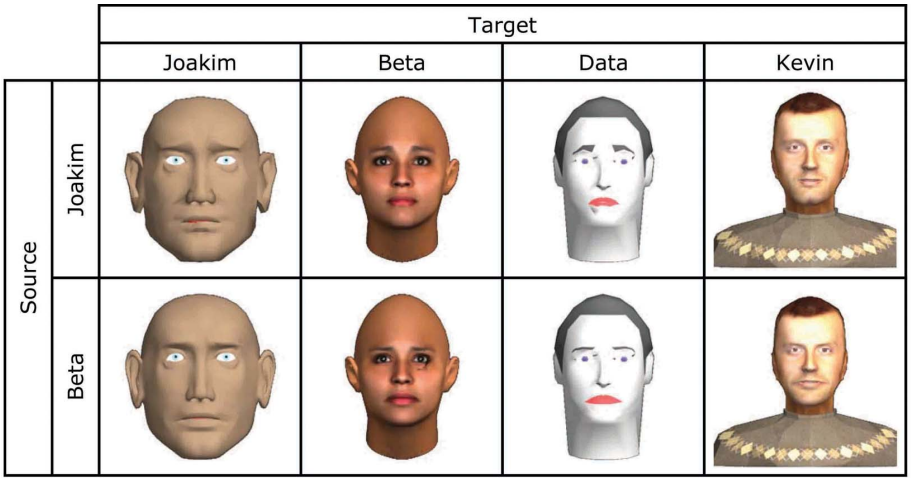


Fig. 6.8: Cloning grids for expression sadness.

Table 6.1: MVs: Moving Vertices. CPs: Correpondence points. Is: Iterations to refine. RT: $G(P)$ Refinement Time. CT: Cloning Time. TT: Total Time.

	MVs	**CPs**	**Is**	**RT**	**CT**	**TT**	**TT_Pzc [Pan03]**
`beta ⇒ data`	396	377	8	4.0 s	1.0 s	5.0 s	5.3 s
`data ⇒ beta`	280	280	5	1.3 s	3.6 s	5.0 s	12.4 s
`joakim ⇒ data`	479	461	8	4.4 s	1.0 s	5.4 s	4.5 s
`data ⇒ joakim`	280	275	4	1.0 s	1.2 s	2.3 s	14.1 s
`beta ⇒ kevin`	396	382	9	4.6 s	0.8 s	5.4 s	9.1 s
`kevin ⇒ beta`	294	290	5	10.6 s	0.6 s	11.3 s	13.6

average of the error of each of the 84 morph targets:

$$e = \left(\frac{1}{84} \sum_{j=1}^{84} \frac{\sum_{i=1}^{N_v} \left\| v_i^S - v_i^T \right\|}{\sum_{i=1}^{N_v} \left\| v_i^S \right\|} \right) \times 100 \tag{6.1}$$

where N_v is the number of the vertices of the face mesh, v_i^S and v_i^T are, respectively, the i-th vertex of the source and the corresponding one on the cloned output model. Table 6.4 presents the obtained values. Fig. 6.9 shows the error distribution over the test models.

Table 6.2: Average error on cloning source faces with themselves.

	source ⇒ source
`beta`	0.028 %
`data`	0.280 %
`joakim`	0.084 %
`kevin`	0.141 %

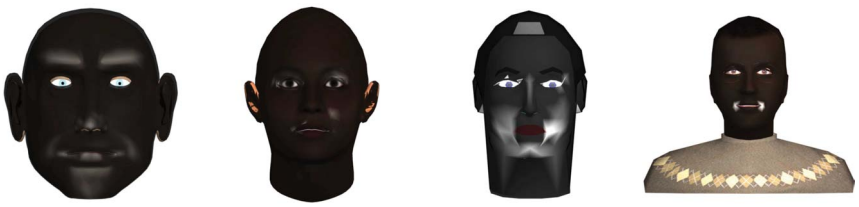


Fig. 6.9: Visual distribution of the error. Error magnitude is proportional to the color brightness.

While differences between these visual results and [Pan03; NN01] are subjective, there are still differences in implementation:

- Noh uses RBF, together with neural networks, to align the source to the target face and then the motion vectors for each target vertex are computed *locally* using proper affine transformations. In my approach instead, the RBF $G(P)$ is needed to align the source and the target with an iterative enrichment of the correspondence set. Then, $G(P)$ is reused to deform the source MTs and copy the motion on the target face. The fact that the cloning is carried out only where needed make this process computationally light and probably faster than Noh's approach. However, Noh's approach is almost fully automatic, while in my approach there is still need to manually map 48 FDPs;
- this algorithm provides MPEG-4 FBA compliant talking face able to perform generic animation, while Noh's approach is able to clone an animation given *a priori*.

6.5 Discussion

The proposed method deals with the problem of reusing existing facial motion to produce in a short amount of time a ready-to-be-animated MPEG-4 FBA compliant talking head. Apart from an initial manual picking of 48 correspondence points all the techniques presented here are fully automatic. In terms of visual results shown, even though only a small subset of them could be presented here, most of facial movements for expression and low-level FAPs are copied correctly to the target face models.

One limitation of this method is that the target mesh cannot be of higher resolution than the source otherwise the barycentric coordinates will just pull any vertices in the interior of that control triangle onto the plane of that triangle. However, using a high-resolution source should not cause major problems. In Fig. 6.5-6.8, higher-resolution models `joakim` and `beta` are cloned to lower resolution models `data` and `kevin`.

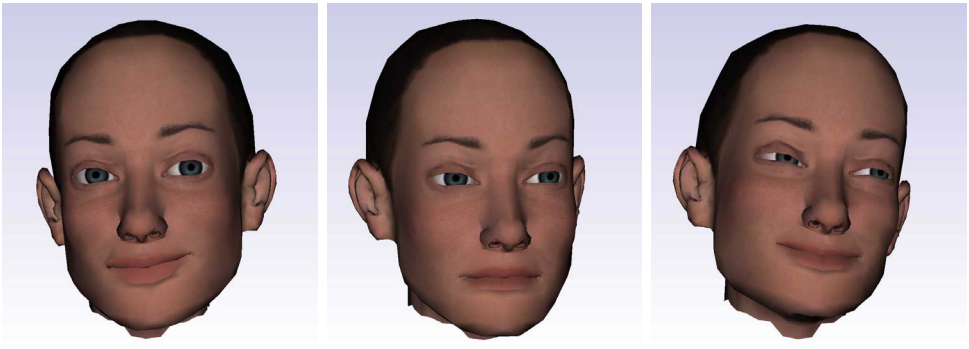
A further problem is in the lack of anatomical knowledge of the algorithm. If, for example, a slim source face is cloned to a fat target face, the produced motion will reflect the slim nature of the source and it will result unnatural. It is due to the user of the method to choose proper source and target meshes.

The main computational effort during the cloning process lies in the refinement process of the $G(P)$ interpolation function. This is because each pair of correspondence points corresponds to a linear equation in the system to resolve in order to obtain $G(P)$. The asymptotic behavior of the linear equation-solving algorithm (LU decomposition) is $O(n^3)$, where n is the number of moving vertices of the source face. Since the correspondences can be very close each other, we think that not all of them are necessary and, as a future work, it would be useful identify and not consider the less significant ones. Furthermore, face feature tracking could be applied to the polygonal meshes in order to retrieve the FDPs making the whole process fully automatic.

After the cloning process is finished, the target face is ready to perform *generic* facial animation encoded into a MPEG-4 FBA stream. The computational cost of the animation depends from the player employed. In this case, I used the lightweight MPEG-4 FBA player provided by [VT:09] in which the animation is achieved, for each frame, through linear interpolation between the proper morph targets and rendered in OpenGL. Thus, the computational cost is rather low. Combined with facial feature tracking, MPEG-4 FBA talking heads can potentially be used for a very low bitrate visual communication in a model-based coding scenario [CPO00; Pan02] (teleconferencing, games, web interfaces).

Chapter 7

Conclusions



(a) (b) (c)

Fig. 7.1: Masha, random expressions. (a). "*Hey, there*", (b). "*You don't convince me*", (c). "*This is going to be checkmate*".

The sophisticated structure of the real human head and the sensible human perception to facial motion artifacts, requires a huge effort from artists and technicians to produce believable facial animation of virtual characters. In this work, I presented two approaches which assist in this delicate task, which automatize the main part of the process and allows to save production resources.

In the virtual anatomical model, the inner structures of the head are simulated as rigid and deformable bodies. A musco-skeletal structure is designed through an *ad-hoc* tool and then it can be fitted to different target skin meshes, whose animation can be controlled by the jaw movements and the muscle contractions.

In the facial motion cloning (FMC) approach, the movements of a source are copied to a target skin mesh, initially static. Since it depends essentially from the position of the vertices of the input meshes, and not from their connectivity, it is particularly suitable for target skin meshes which have an asymmetric topology w.r.t. to the sagittal plane of the head. These

kind of meshes, which are usually obtained by automatic level-of-detail techniques, are not very well handled by the anatomic model and FMC is a substitute method to obtain animated virtual faces.

7.1 Key Contributions

7.1.1 Muscle Shape Definition

Muscles are defined through an interactive designing tool which allows the user to sketch action lines directly on the skull and already existing muscles. The action lines are sampled and a geometric hexahedral mesh is fitted into the sample points to reach the desired shape. The process is natural and intuitive and allows for defining within hours a complete, multi-layered, musco-skeletal structure ready to animate different input skin meshes.

7.1.2 Computationally Cheap Biomechanical Model

The muscle model is a key part of the system; in particular the linear muscle model is responsible for representing sheet and linear muscles as well the fatty tissue under the skin surface. The modeling through geometric constraints allows for reproducing the macro behavior of the muscles, like bulging due to volume preservation, while keeping the computational cost low and the simulation unconditionally stable. The stiffness parameters which characterizes the model dynamics are normalized in a range between zero and one; it is shown that the same musco-skeletal structure can be adapted to different skin meshes. Muscular contraction is controlled by only one scalar parameter, which makes easy to control a whole muscle map. A further advantage of this technique is the easiness in handling collisions, which happens between the skin and teeth structures: penetrations can be resolved completely by projecting points to valid locations without introducing any impulsive force.

7.1.3 Anatomical Face Model

The musco-skeletal model, including fatty tissue, is assembled together with the skin and the other anatomic elements, like teeth and eyes, to represent a complete facial structure. Each element is separately controlled providing the possibility to synthesize a wide range of facial expressions. The whole modeling has been carried out with the final purpose to animate the widest range of input skin meshes. For example, eyes are not animated through geometric rotations, which can produce artifacts, the illusion of movement is provided through the shifting of the texture mapping coordinates.

7.1.4 Facial Motion Cloning

In the anatomical model, the skin is built starting from the input skin mesh. Hence, if an input mesh has an asymmetric topology, the skin model may behave not correctly. For these cases, a purely geometric algorithm is provided which copy the motion information from an already animated skin mesh to a different static target. The method has been tested and have successfully transfered animation among different face meshes. However, having no

knowledge of the anatomical characteristics of the meshes involved, it is possible to transfer the motion from a fat face to a slim face and this can produce not believable motion. The output animated face mesh is compliant with the MPEG-4 FBA standard which make it usable in commercial products, in special way in model-based coding applications.

7.2 Future Research

The anatomical model can be a valid tool to perform further research.

7.2.1 Potential Extensions of the Anatomical Model

The skull mesh is not hard-coded into the method. Thus, if a monstrous skull mesh, like the one belonging to an ogre or a cyclops, is used, it would be possible to design the muscles and animate the corresponding skins. This could be valid for the main part of mammal animals as well. The interactive editor to design muscles may be also employed in more complex systems for off-line facial animation. Furthermore, given the physically-based nature of the anatomical approach, it would also possible to model external interactions as a result of contact. Thus, the anatomical model may be suitable for facial animation under external influences.

7.2.2 Improvement of Position Based Dynamics

As most of the methods for interactive physically-based animation, in PBD there is not a clear correspondence among the parameters governing the simulated model and the real system. Usually, estimated or empirically obtained values are employed to achieve plausible motion. A line of future research is to employ methods from artificial intelligence, like genetic algorithms or neural networks, to automatically learn these physical parameters. In the case of facial animation, the ground truth data could be video sequences of talking people whose motion is tracked, parametrized and applied to the virtual face. The objective of the AI algorithm shall be to define the physical parameters such that the virtual face motion matches the real one.

Another improvement to PBD is the use of the multi grid Gauss-Seidel solver for sophisticated meshes like those representing the face skin. This would allow for better performance and faster convergence of the constraints projection.

7.2.3 Face Parametrization

By using the anatomical model, it is easy to automatically synthesize a wide range of facial expressions. If a Principal Component Analysis is run on the whole set of facial configurations achievable by a given skin meshes, it would be possible to generate a small and orthogonal set of blend shapes, which interpolated each other would reproduce the main part of the input facial configurations. This would allow for an extremely efficient encoding of the animation of the face, usable for instance in the context of video conference.

Bibliography

[AD03] AHLBERG J., DORNAIKA F.: "Efficient Active Appearance Model for Real-Time Head and Facial Feature Tracking". In *IEEE International Workshop on Analysis and Modelling of Faces and Gestures (AMFG)* (Nice, October 2003).

[Ahl01] AHLBERG J.: *Candide-3 - an updated parameterised face*. Tech. Rep. LiTH-ISY-R-2326, Department of Electrical Engineering, Linköping University, Sweden, 2001.

[Ahl02] AHLBERG J.: *Model-based Coding – Extraction, Coding and Evaluation of Face Model Parameters*. Phd thesis no. 761, Department of Electrical Engineering, Linköping University, Linköping, Sweden, September 2002.

[AT00] AUBEL A., THALMANN D.: Efficient muscle shape deformation. In *in Deformable Avatars , IFIP TC5/WG5.10 DEFORM2000 Workshop* (2000), Kluwer.

[AT01] AUBEL A., THALMANN D.: Interactive modeling of the human musculature. In *In Proceedings of Computer Animation* (2001), pp. 7–8.

[BBA*07] BICKEL B., BOTSCH M., ANGST R., MATUSIK W., OTADUY M., PFISTER H., GROSS M.: Multi-scale capture of facial geometry and motion. *ACM Trans. Graph. 26*, 3 (2007), 33.

[BBPV03] BLANZ V., BASSO C., POGGIO T., VETTER T.: Reanimating faces in images and video. In *Proceedings of EUROGRAPHICS* (Granada, Spain, 2003), Brunet P., Fellner D., (Eds.).

[BDH96] BARBER C. B., DOBKIN D. P., HUHDANPAA H.: The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4 (1996), 469–483.

[BFMM07] BUSH J., FERGUSON M. W., MASON T., MCGROUTHER G.: "The dynamic rotation of Langer's lines on facial expression.". *Journal of Plastic, Reconstructive & Aesthetic Surgery 60*, 4 (April 2007), 393–399.

[BL85] BERGERON P., LACHAPELLE P.: Controlling facial expressions and body movements in the computer-generated animated short "tony de peltrie". In *SIGGRAPH 85 Advanced Computer Animation seminar notes* (July 1985).

[BL03] BORSHUKOV G., LEWIS J. P.: Realistic human face rendering for "the matrix reloaded". In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications* (New York, NY, USA, 2003), ACM, pp. 1–1.

[BNS96] BRO-NIELSEN M., S. C.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer Graphics Forum* (1996), vol. 15, pp. 57–66.

[BV99] BLANZ V., VETTER T.: A morphable model for the synthesis of 3d faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

[BW93] BARAFF D., WITKIN A.: "An Introduction to Physically Based Modeling". In *SIGGRAPH Course Notes* (1993).

[BW98] BARAFF D., WITKIN A.: "Large Steps in Cloth Simulation". In *Computer Graphics Proceedings* (Orlando, July 1998), Annual Conference Series, SIGGRAPH 98, ACM SIGGRAPH.

[CPO00] CAPIN T., PETAJAN E., OSTERMANN J.: Very low bitrate coding of virtual human animation in mpeg-4. In *IEEE International Conference on Multimedia and Expo (II)* (July 2000), vol. 2, pp. 1107–1110.

[CSPE00] CASSELL J., SULLIVAN J., PREVOST S., ELIZABETH C. (Eds.): *Embodied Conversational Agents*. The MIT Press, April 2000.

[DCFN06] DENG Z., CHIANG P.-Y., FOX P., NEUMANN U.: Animating blendshape faces by cross-mapping motion capture data. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 43–48.

[Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[DM96] DECARLO D., METAXAS D.: The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)* (Washington, DC, USA, 1996), IEEE Computer Society, p. 189.

[Ear06] EARN D.: *Ergodic Concepts in Stellar Dynamics*, vol. 430/1994 of *Lecture Notes in Physics*. Springer Berlin / Heidelberg, 2006, ch. Symplectic integration without roundoff error., pp. 122–130.

[Ebe04a] EBERLY D.: *Game physics*. Morgan Kaufmann, 2004.

[Ebe04b] EBERLY D. H.: *3D Game Engine Design*. Morgan Kaufmann, 2004.

[Ede01] EDELSBRUNNER H.: *Geometry and topology for mesh generation*. Cambridge University Press, New York, NY, USA, 2001.

[EFE72] EKMAN P., FRIESEN W. V., ELLSWORTH P.: *Emotion in the Human Face*. Oxford University Press, 1972.

[EMT97] ESCHER M., MAGNENAT-THALMANN N.: "Automatic 3D Cloning and Real-Time Animation of a Human Face". In *Computer Animation* (Geneva, Switzerland, June 1997).

[EPT98] ESCHER M., PANDZIC I., THALMANN N. M.: Facial deformations for mpeg-4. In *CA '98: Proceedings of the Computer Animation* (Washington, DC, USA, 1998), IEEE Computer Society, p. 56.

[FF07] FANELLI G., FRATARCANGELI M.: "A Non-Invasive Approach For Driving Virtual Talking Heads From Real Facial Movements". In *3DTV Conference* (Kos Island, Greece, May 2007), IEEE Xplore.

[Fra05] FRATARCANGELI M.: "Physically Based Synthesis of Animatable Face Models". In *Proceedings of the 2nd International Workshop on Virtual Reality and Physical Simulation (VRIPHYS05)* (Pisa, Italy, November 2005), ISTI-CNR, The Eurographics Association, pp. 32–39.

[FRR96] FANG S., RAGHAVAN R., RICHTSMEIER J.: "Volume Morphing Methods for Landmark Based 3D Image Deformation". In *SPIE International Symposium on Medical Imaging* (Newport Beach, CA, February 1996), vol. 2710, pp. 404–415.

[FS04] FRATARCANGELI M., SCHAERF M.: "Realistic Modeling of Animatable Faces in MPEG-4". In *Computer Animation and Social Agents* (Geneva, Switzerland, July 2004), MIRALAB, Computer Graphics Society (CGS), pp. 285–297.

[FS05a] FRATARCANGELI M., SCHAERF M.: "Facial Motion Cloning Using Global Shape Deformation". In *Eurographics Short Presentations* (Dublin, Ireland, August 2005), The Eurographics Association and The Image Synthesis Group, pp. 89–92. ISSN 1017, 4656.

[FS05b] FRATARCANGELI M., SCHAERF M.: "Fast Facial Motion Cloning in MPEG-4". In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005. (ISPA 2005)* (Zagreb, Croatia, September 2005), IEEE - Signal Processing Society, pp. 310–315. issn 1845-5921.

[FSF07] FRATARCANGELI M., SCHAERF M., FORCHHEIMER R.: Facial motion cloning with radial basis functions in mpeg-4 fba. *Graph. Models 69*, 2 (2007), 106–118.

[Fun93] FUNG Y.: *Biomechanics: Mechanical Properties of Living Tissues*. Springer, New York, 1993.

[FŏŠ05] FĚDOR M.: Fast character animation using particle dynamics. In *Proceedings of GVIP 2005* (2005).

[GEZMT99] GOTO T., ESCHER M., ZANARDI C., MAGNENAT-THALMANN N.: "MPEG-4 Based Animation with Face Feature Tracking". In *Computer Animation and Simulation '99* (1999), pp. 89–98.

[GM97] GIBSON S., MIRTICH B.: *A survey of deformable models in computer graphics.* Tech. Rep. TR-97-19, MERL, Cambridge, MA, 1997.

[Gra08] GRAY H.: *Gray's Anatomy: The anatomical basis of Medicine and Surgery*, 40th ed. Churchill-Livingstone, 2008. ISBN: 978-0-443-06684-9.

[GZDH04] GLADILIN E., ZACHOW S., DEUFLHARD P., HEGE H.: Anatomy-and physics-based facial animation for craniofacial surgery simulations. *Medical and Biological Engineering and Computing 42*, 2 (2004), 167–170.

[Hen01] HENDRIKS F.: *Mechanical behaviour of human skin in vivo - a literature review*. Unclassified Report 820, Koninklijke Philips Electronics, 2001.

[HTMTB03] HABER J., TERZOPOULOS D., MAGNENAT-THALMANN N., BLANZ V.: *Facial Modeling and Animation - Eurographics 2003 Tutorial Notes*. Eurographics, -, September 2003.

[im209] Image Metrics. `http://www.image-metrics.com`, February 2009.

[ISO] Moving Pictures Expert Group. MPEG-4 International standard. ISO/IEC 14496. http://www.cselt.it/mpeg.

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of deformation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 131–140.

[Jak03] JAKOBSEN T.: Advanced character physics. `http://www.gamasutra.com/view/feature/2904/advanced_character_physics.php`, 2003.

[KÖ7] KÄHLER K.: *3D Facial Animation- Recreating Human Heads With Virtual Skin, Bones, and Muscles.* Verlag, October 2007.

[Käh03] KÄHLER K.: *A Head Model with Anatomical Structure for Facial Modeling and Animation*. Phd thesis, Universität des Saarlandes, Saarbrücken, Germany, December 2003.

[KHS01] KÄHLER K., HABER J., SEIDEL H.-P.: Geometry-based muscle modeling for facial animation. In *GRIN'01: Graphics interface 2001* (Toronto, Ont., Canada, Canada, 2001), Canadian Information Processing Society, pp. 37–46.

[KHS03] KÄHLER K., HABER J., SEIDEL H.-P.: Reanimating the dead: reconstruction of expressive faces from skull data. *ACM Trans. Graph. 22*, 3 (2003), 554–561.

[KHYS02] KÄHLER K., HABER J., YAMAUCHI H., SEIDEL H.-P.: Head shop: generating animated head models with anatomical structure. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM, pp. 55–63.

[KPFG07] KUBIAK B., PIETRONI N., FRATARCANGELI M., GANOVELLI F.: "A Robust Method for Real-Time Thread Simulation". In *ACM Symposium on Virtual Reality Software and Technology (VRST)* (New Port Beach, CA, USA, November 2007), ACM, (Ed.).

[LESMT99] LEE W.-S., ESCHER M., SANNIER G., MAGNENAT-THALMANN N.: "MPEG-4 Compatible Faces from Orthogonal Photos". In *CA* (1999), pp. 186–194.

[Liu02] LIU G.: *"Mesh Free Methods – Moving beyond the finite element method"*. CRC Press, 2002.

[LLF94] LI H., LUNDMARK A., FORCHHEIMER R.: Image sequence coding at very low bitrates: A review,". *IEEE Trans. Image Processing 3* (1994), 589–609.

[LMDN05] LEWIS J. P., MOOSER J., DENG Z., NEUMANN U.: Reducing blendshape interference by selected motion attenuation. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM, pp. 25–29.

[LRF93] LI H., ROIVAINEN P., FORCHHEIMER R.: 3-D motion estimation in model-based facial image coding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 15*, 6 (1993), 545–555.

[LTW93] LEE Y., TERZOPOULOS D., WATERS K.: Constructing physics-based facial models of individuals. In *Proceedings of the Graphics Interface '93 Conference* (Toronto, ON, Canada, May 1993), pp. 1–8.

[LTW95] LEE Y., TERZOPOULOS D., WATERS K.: Realistic modeling for facial animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM, pp. 55–62.

[MÖ8] MÜLLER M.: Hierarchical position based dynamics. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys2008)* (Grenoble, November 2008).

[MHHR06] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Madrid, November 6-7 2006).

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. In *Proceedings of SIGGRAPH'05* (Los Angeles, USA, July 31 - August 4 2005), pp. 471–478.

[Mic86] MICCHELLI C.: "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions". *Constructive Approximations 2* (1986), 11–22.

[MWMTT03] MAUREL W., WU Y., MAGNENAT THALMANN N., THALMANN D.: *Biomechanical Models for Soft Tissue Simulation*, 1st ed. ESPRIT Basic Research. Springer, February 2003.

[NDP*07] NEUMANN U., DENG Z., PIGHIN F., NOH J., GARCHERY S., MAGNENAT-THALMANN N., LEWIS J.: *Data-Driven 3D Facial Animation*. Computer Science. Springer London, October 2007.

[Nea04] NEALEN A.: An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. `http://www.nealen.com/projects`, 2004.

[nl209] New line films. http://www.newline.com, February 2009.

[NMBC05] NEALEN A., MÜLLER M. KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. In *Eurographics 2005 state of the art report (STAR)* (Dublin, Ireland, August 29 - September 02 2005).

[NN98] NOH J., NEUMANN U.: A survey of facial modeling and animation techniques. *University of Southern California Technical Report* (1998), 99–705.

[NN01] NOH J., NEUMANN U.: "Expression Cloning". In *SIGGRAPH* (2001), ACM SIGGRAPH, pp. 277–288.

[NTH01] NG-THOW-HING V.: *Anatomically-based models for physical and geometric reconstruction of humans and other animals*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ont., Canada, Canada, 2001. Adviser-Fiume,, Eugene.

[NTHF02] NG-THOW-HING V., FIUME E.: Application-Specific Muscle Representations. In *Graphics Interface* (2002), pp. 107–116.

[ODGB01] OSULLIVAN C., DINGLIANA J., GANOVELLI F., BRADSHAW G.: Collision Handling for Virtual Environments. In *Computer Graphics Forum (Proc. of Eurographics)* (2001).

[ON99] ÖZKAYA N., NORDIN M.: *Fundamentals of Biomechanics: Equilibrium, Motion, and Deformation*, 2nd ed. Springer, June 1999.

[Ost98] OSTERMANN J.: "Animation of Synthetic Faces in MPEG-4". In *Computer Animation* (Philadelphia, Pennsylvania, June 1998), pp. 49–51.

[Pan02] PANDZIC I. S.: Facial animation framework for the web and mobile platforms. In *Web3D '02: Proceedings of the seventh international conference on 3D Web technology* (New York, NY, USA, 2002), ACM, pp. 27–34.

[Pan03] PANDZIC I.: "Facial Motion Cloning". *Graphical Models Journal, Elsevier 65*, 6 (2003), 385–404.

[Par72] PARKE F. I.: Computer generated animation of faces. In *ACM'72: Proceedings of the ACM annual conference* (New York, NY, USA, 1972), ACM, pp. 451–457.

[PB81] PLATT S. M., BADLER N. I.: Animating facial expressions. *SIGGRAPH Comput. Graph. 15*, 3 (1981), 245–252.

[PF02] PANDZIC I., FORCHHEIMER R. (Eds.): *"MPEG-4 Facial Animation – The Standard, Implementation and Applications"*, 1st ed. John Wiley & Sons, LTD, Linköping, Sweden, 2002.

[PHIM04] PRENDINGER, HELMUT, ISHIZUKA, MITSURU (Eds.): *Life-Like Characters. Tools, Affective Functions, and Applications*. Cognitive Technologies. Springer, 2004.

[PHL*98] PIGHIN F., HECKER J., LISCHINSKI D., SZELISKI R., SALESIN D. H.: Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 75–84.

[PKC*06] PYUN H., KIM Y., CHAE W., KANG H. W., SHIN S. Y.: An example-based approach for facial expression cloning. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM, p. 23.

[Por04] PORCINO N.: *Game Programming Gems 4*. Charles River Media, 2004, ch. Writing A Verlet Based Physics Engine, pp. 231–240.

[PP09] PRIMAL PICTURES L.: Interactive Head and Neck, CD 5, February 2009. `http://www.primalpictures.com`.

[PW96] PARKE F. I., WATERS K.: *Computer facial animation*. A. K. Peters, Ltd., Natick, MA, USA, 1996.

[qhu09] QHull code for convex hull. `http://www.qhull.org`, February 2009.

[RP06] RADOVAN M., PRETORIUS L.: Facial animation in a nutshell: past, present and future. In *SAICSIT '06: Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries* (Republic of South Africa, 2006), South African Institute for Computer Scientists and Information Technologists, pp. 71–79.

[Ryd87] RYDFALK M.: *CANDIDE, a parameterized face.* Tech. Rep. LiTH-ISY-I-866, Department of Electrical Engineering, Linköping University, Sweden, 1987.

[SE03] SCHNEIDER P. J., EBERLY D. H.: *Geometric Tools for Computer Graphics*, 1st ed. Morgan Kaufmann Publishers, 2003.

[SNF05] SIFAKIS E., NEVEROV I., FEDKIW R.: Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. 24*, 3 (2005), 417–425.

[SPCM97] SCHEEPERS F., PARENT R. E., CARLSON W. E., MAY S. F.: Anatomy-based modeling of the human musculature. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 163–172.

[SSRMF06] SIFAKIS E., SELLE A., ROBINSON-MOSHER A., FEDKIW R.: Simulating speech with a physics-based facial muscle model. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 261–270.

[TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R.: Finite volume methods for the simulation of skeletal muscle. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 68–74.

[TGG00] TESCHNER M., GIROD S., GIROD B.: Direct computation of nonlinear soft-tissue deformation. In *Vision, Modeling and Visualization VMV'00* (Saarbrücken, Germany, November 22-24 2000).

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANERTS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization VMV 2003* (Munich, Germany, November 19-21 2003), pp. 47–54.

[TKZ*04] TESCHNER M., KIMMERLE S., ZACHMANN G., HEIDELBERGER B., RAGHUPATHI L., FUHRMANN A., CANI M. P., FAURE F., THALMANN M. N., STRASSER W.: Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)* (2004), Eurographics Association, pp. 119–139.

[TSB*05] TERAN J., SIFAKIS E., BLEMKER S. S., NG-THOW-HING V., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 317–328.

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM, pp. 181–190.

[TW90] TERZOPOULOS D., WATERS K.: Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer Animation 1*, 2 (1990), 73–80.

[TYHS02] TARINI M., YAMAUCHI H., HABER J., SEIDEL H.-P.: Texturing faces. In *Proceedings of the Graphics Interface 2002 (GI-02)* (Mississauga, Ontario, Canada, may 2002), Canadian Information Processing Society, pp. 89–98.

[Ver67] VERLET L.: Computer experiments on classical fluids. ii. equilibrium correlation functions. *Physical Review 165* (1967), 201–204.

[VT:09] Visage Technologies - the character animation company. `http://www.visagetechnologies.com`, February 2009.

[Wat87] WATERS K.: A muscle model for animation three-dimensional facial expression. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM, pp. 17–24.

[WHL*04] WANG Y., HUANG X., LEE C. S., ZHANG S., LI Z., SAMARAS D., METAXAS D., ELGAMMAL A., HUANG P.: High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum 23* (September 2004).

[wik09] Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Computer_graphics`, February 2009.

[Wil90] WILLIAMS L.: Performance-driven facial animation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM, pp. 235–242.

[Wlo03] WLOKA M.: Batch, batch, batch. what it does really mean? In *Game Developers Conference* (2003).

[WVG97] WILHELMS J., VAN GELDER A.: Anatomically based modeling. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 173–180.

[ZFI07] ZARATTI M., FRATARCANGELI M., IOCCHI L.: *"RoboCup 2006: Robot Soccer World Cup X"*, vol. 4434 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 2007, ch. "A 3D Simulator of Multiple Legged Robots based on USARSim", pp. 13–24. Best Paper Award at RoboCup International Symposium 2006, Bremen, Germany June 2006.

[ZH*06] ZHANG S., HUANG P., ET AL.: High-resolution, real-time three-dimensional shape measurement. *Optical Engineering 45*, 12 (2006).

[ZPS02] ZHANG Y., PRAKASH E., SUNG E.: "Hierarchical Face Modeling and Fast 3D Facial Expression Synthesis". In *Proc. XV Brazilian Symposium on Computer Graphics and Image Processing* (October 2002), IEEE Computer Society Press, pp. 357–364. Best Paper Award.

[ZPS03] ZHANG Y., PRAKASH E., SUNG E.: "Efficient Modeling on An Anatomy-Based Face and Fast 3D Facial Expression Synthesis". In *Computer Graphics Forum* (June 2003), vol. 22, The Eurographics Association, Computer Graphics forum, Blackwell Publishing Ltd., pp. 159–169.

[ZSCS04] ZHANG L., SNAVELY N., CURLESS B., SEITZ S. M.: Spacetime faces: high resolution capture for modeling and animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 548–558.

Muscle Designer Tool

.1 Main Functionalities

The Muscle Designer has been written in order to test and prototype all the components of the anatomical model and animate custom skin meshes. For this purpose, it support the following main functionalities:

1. load a custom skull mesh and allows for defining the landmarks on it;
2. design and edit interactively both circular and linear muscles and allows for tune any of their parameters;
3. load a custom skin mesh, with corresponding eyes, teeth and tongue; allows for editing the landmarks;
4. build the complete model by fitting the skull and the muscle models into the skin and binds the skin to the underlying structure;
5. tune any parameter of the model on the fly;
6. modify muscle contractions in order to achieve facial animation;
7. load and save the model in any time of its specification;
8. load and save a set of muscle contractions representing a set of emotions;
9. export blend shapes in order to animate the face in different rendering engines.

It allows to load a custom skull mesh, interactively design and edit both linear and circular muscles. Furthermore it allows for loading a skin mesh with eyes, teeth and tongue, load a set of expressions.

.2 External Technologies

The tool has been wrote in portable c++ even though has been tested only on the Windows XP platform. A rather wide set of open source libraries have been used to build the tool. They are:

- OpenGL. (Open Graphics Library) It is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. Within this tool, its Windows implementation has been used for visualize 3D data structures.

- VCGLib. Developed by the Visual Computing Lab of the ISTI - an institute of the Italian National Research Council (CNR). It is a portable C++ templated library for the manipulation, processing of triangle and tetrahedral meshes. Main developer and founder of the library is Prof. Paolo Cignoni. Within this tool, it has been used mainly analyze meshes, find the borders, load and save, matrix-vector computation. It has been extended with several features like the RBF morphing.
- IDOLib. Developed by the same research group which made VCGLib, IDOlib is a C++ library which aims to provide a simple and as general as possible tool for developing software for simulating deformable objects. It provides basic definition and tools (particle type, PDE integration) as well as known method presented in literature. Main developer and founder of the library is Prof. Fabio Ganovelli. Within this tool, it has been the basis on which build the PBD engine and the spatial hashing data structure.
- Boost. It is a set of free peer-reviewed portable C++ source libraries. Within this tool, the Serialization library haas been used to load and save the state of both the application and the anatomical model.
- Qhull. It is a library which computes the convex hull, Delaunay triangulation, Voronoi diagram, halfspace intersection about a point, furthest-site Delaunay triangulation, and furthest-site Voronoi diagram. It has been used to compute the convex hull of the teeth for collision handling.
- wxWidgets. A cross-platform GUI and tools library for GTK, MS Windows, and MacOS. It has been used for visualizing and handling main GUI as well for log functions.
- FreeImage. It is a library for developers who would like to support popular graphics image formats like PNG, BMP, JPEG, TIFF. It has been used to load the textures of the face models independently from wxWidgets.

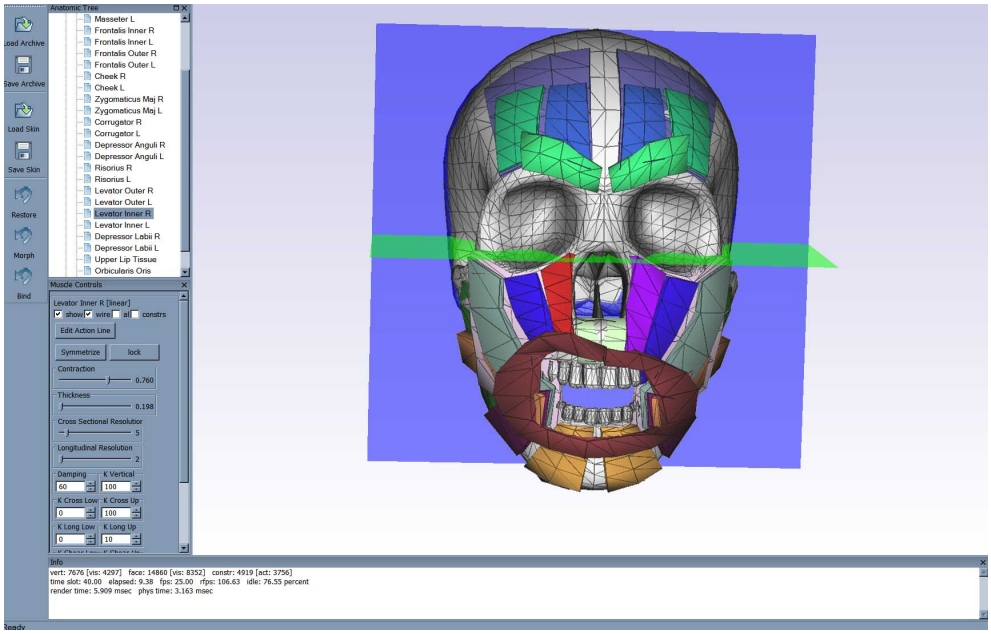

Anatomic Tree
Masseter L
Frontalis Inner R
Frontalis Inner L
Frontalis Outer R
Frontalis Outer L
Cheek R
Cheek L
Zygomaticus Maj R
Zygomaticus Maj L
Corrugator R
Corrugator L
Depressor Anguli R
Depressor Anguli L
Risorius R
Risorius L
Levator Outer R
Levator Outer L
Levator Inner R
Levator Inner L
Depressor Labii R
Depressor Labii L
Upper Lip Tissue
Orbicularis Oris
Muscle Controls
Edit Action Line
Symmetrize
lock
Contraction
Thickness
Cross Sectional Resolution
Longitudinal Resolution
Damping
K Vertical
K Cross Low
K Cross Up
K Long Low
K Long Up
Info
Load Archive
Save Archive
Load Skin
Save Skin
Restore
Morph
Bind

Fig. 2: GUI.

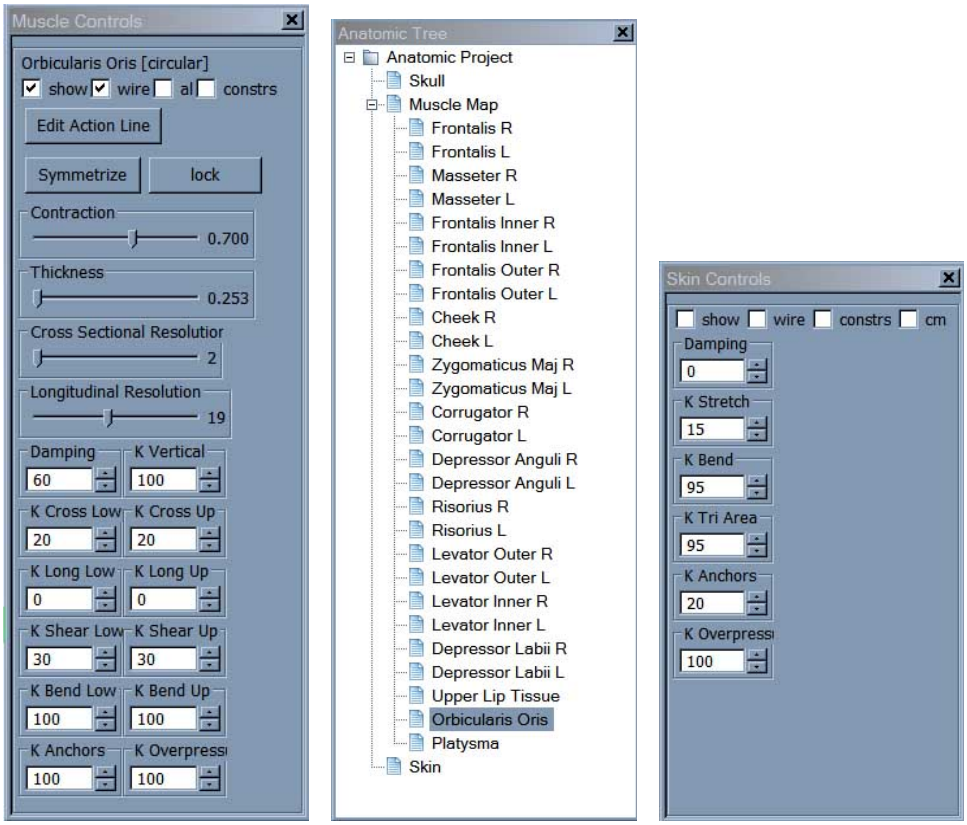


Fig. 3: GUI panels 1

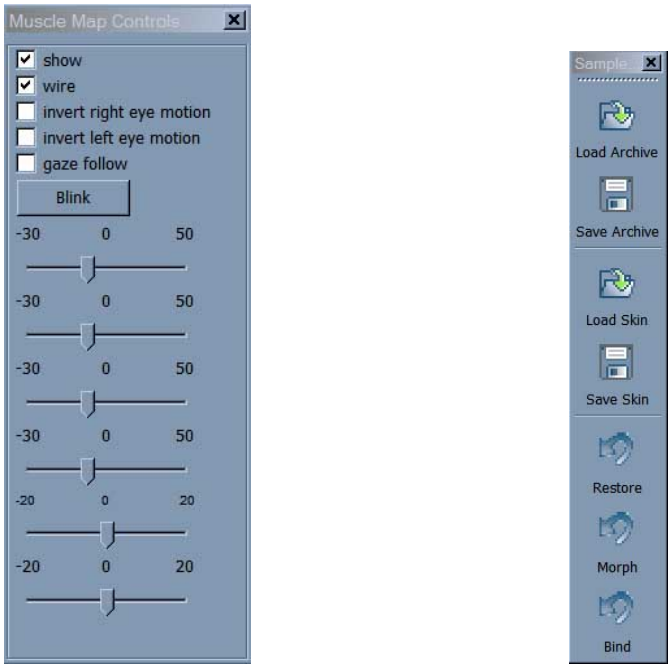


Fig. 4: GUI panels 2