MODELING AND ANIMATING PERSONALIZED FACES

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Fatih Erol

January, 2002

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Uğur Güdükbay (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Bülent Özgüç

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

MODELING AND ANIMATING PERSONALIZED FACES

Fatih Erol
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Uğur Güdükbay
January, 2002

A very important and challenging problem in computer graphics is modeling and animation of individualized face models. In this thesis, we describe a facial modeling and animation system attempting to address this problem. The system uses muscle-based generic face model and deforms it using deformation techniques to model individualized faces. Two orthogonal photos of the real faces are used for this purpose. Image processing techniques are employed to extract certain features on the photographs, which are then refined manually by the user through the facilities of the user interface of the system. The feature points located on the frontal and side views of a real face are used to deform the generic model. Then, the muscle vectors in the individualized face model are arranged accordingly. Individualized face models produced in this manner are animated using parametric interpolation techniques.

Keywords: facial animation, rendering, texture mapping, deformation, feature extraction.

ÖZET

KİŞİSELLEŞTİRİLMİŞ YÜZ MODELLEME VE ANİMASYONU

Fatih Erol
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Uğur Güdükbay
Ocak, 2002

Kişisel insan yüzünü gerçekçi olarak modellemek ve canlandırmak bilgisayar grafiği alanında önemli ve zor bir problemdir. Bu tezde, bahsedilen problemi çözmeye yönelik olarak geliştirilen bir yüz modelleme ve animasyon sistemi anlatılmaktadır. Sistemde kullanılan kas tabanlı genel yüz modeli deformasyon teknikleri ile deforme edilerek kişisel yüz modelleri oluşturulabilmektedir. Bunun için insan yüz modelinin birbirine dik olan ön ve yanal fotoğrafları kullanılmaktadır. Resim işleme teknikleri ile fotoğraf üzerinde bazı özellikler bulunduktan sonra, kullanıcı arabiriminin sağladığı fonksiyonlar aracılığıyla yüz üzerindeki tanımlayıcı noktaların yerleri tam olarak belirlenmektedir. Ön ve yanal yüz fotoğraflarındaki bu noktalar kullanılarak genel model deforme edilmektedir. Daha sonra, sistem kişiselleştirilmiş modeldeki kas vektörlerinin yeni pozisyonlarını belirlemektedir. Bu yöntemle oluşturulan kişisel yüz modelleri parametrik interpolasyon teknikleri ile hareket ettirilmektedir.

Anahtar sözcükler: yüz animasyonu, boyama, doku eşleme, deformasyon, özellik çıkarımı.

Acknowledgement

I would like to express my gratitude to Assist. Prof. Dr. Uğur Güdükbay for his great help in preparation of the thesis, as well as for his support and guidance throughout my master's study.

I am also greatly thankful to Prof. Dr. Bülent Özgüç and Assoc. Prof. Dr. Özgür Ulusoy for their interest and acceptance of reviewing the thesis. I sincerely appreciate their supportive remarks, and my thanks are due to all reviewers of the study for their patience with me.

I would like to thank to everybody who permitted their face photographs be taken and used in the study. I appreciate their help and any kind of support they have given.

Finally, I would be happy to express my deepest gratitude and love to my whole family, who have never given up their endless morale support throughout my whole life.

To my family.

Contents

List of Figures

Chapter 1

Introduction

Computer graphics is under intense academic research mainly since four decades ago when Ivan Sutherland started working about drawing figures on cathode ray tube (CRT) in 1962 with his program named Sketchpad. Use of computers for producing visual output has been a great part of life in many domains especially for the last 10 years. Not only with numerous applications as great effects in films, making unbelievable things seem as happening, there are also many other domains the computers are used to come up with solutions for easing or giving color to human life. It is not unusual that some industrial products like machinery pieces or car bodies are designed with a computer; the physical behavior is modeled with some hardware that is controlled and evaluated on a computer; or beautiful artwork is drawn using computer imagery techniques. Even user interface design, which will make life easier with using many electronic devices, is a matter of computer graphics going under research to yield best techniques.

When such tools are thoroughly being used for human good, the modeling and animation of human body also turns out to be an important domain to work on. Modeling behavior of human body in some situations, or the effects on human body within some controlled environment is among the first useful areas that come to mind considering the necessity of computer generated and animated human body models. The film industry is also using related techniques with scenes that would be very dangerous or impossible to film with real actors.

We even have many film productions that are fully/partly computer animated, classified as animations in the market. Many people are great fans for the Disney-Pixar animations like the Toy Story I/II. The characters in the film are either human beings, or toys like cowboy doll, toy-cars, which are animated to produce expressions giving life to their body by reflecting their emotions.

In the whole human body, the most important part turns out to be the face region from the viewpoint of the watcher usually, since it carries the greatest part of duty for reflecting emotions and expressions as well as carrying the mouth where speech is to be incorporated into the animation. The face is the best known place in the human body, and any human will be using the face to recognize people mostly. The human face is a very important and complex communication channel, which is very familiar and sensitive for human perception, causing even very subtle changes in facial expressions to be easily detected. As the face is such important to reflect the body to model and the human brain is very talented to recognize individual faces among a similar face universe, its realistic and detailed animation becomes a research area in computer graphics. We can see the results like human body animations, talking heads, and computer generated art. The animation of the face is mainly producing believable facial expressions on the digital face model.

As most of the application areas of computer facial modeling and animation are demanding, a very important and challenging problem is modeling and animation of individualized face models. In this thesis, we present a facial modeling and animation system attempting to address this problem. The system uses muscle-based generic face model and deforms it using deformation techniques to model individualized faces. Two orthogonal photos of the real faces are used for this purpose. Image processing techniques are employed to extract certain features on the photographs, which are then refined manually by the user through the facilities of the user interface of the system. The feature points located on the frontal and side views of a real face are used to deform the generic model. Then, the muscle vectors in the individualized face model are arranged accordingly. Individualized face models produced in this manner are animated using a muscle-based animation scheme and parametric interpolation techniques.

1.1 Organization of the Thesis

Chapter 2 introduces historical background for facial animation research and main application areas of facial animation. The remaining chapters give detailed explanation of the techniques used in our facial animation system, as well as mentioning briefly some of the other methods that may be used for developments on the current system. The following four chapters are about parts of a facial animation system. Each of the chapters is organized such that first a general information is presented about the subject, then brief information is given on related techniques that may be used for further development, and finally the corresponding part in our facial animation system is explained with references to how it could be improved. Chapter 3 is on face modeling. Brief information on the face anatomy, face modeling techniques, and properties of our face model are presented in this chapter. Chapter 4 is about facial feature extraction, where we make short explanations on facial features and techniques that are suitable to use for extraction of some parameters from face, and then show the method we have used for roughly locating the feature locations. Chapter 5 discusses the face deformation process. In this chapter, we explain the techniques we used to deform the face model according to some parameters after a brief introduction to related work in this area. Chapter 6 is on rendering and animation, explaining a technique for more realistic texture mapping, as well as the part we have built. Then, we explain the techniques for facial animation, and discuss the method we have used in detail. We conclude the thesis with a brief summary of work carried out, reminding and adding on further developments that could be done in order to make the current system better for general or specific uses.

Chapter 2

Previous Work

This chapter briefly explains the research conducted in the facial modeling and animation. Historically, Parke's research is considered as the pioneering work in this area [53]. Parke started to design a very crude polygonal representation of the head, that would be capable for building animations which moved the eyes and mouth. Parke came out with pieces of animations that looked rather satisfying for the time. He produced the facial expression polygon data from real faces using photogrammetric techniques and the animation was built by interpolating between the expressions. In 1974, Parke completed his first parameterized facial model. This model defines the face with a large set of parameters, like distance between eyes, width of mouth, etc. The parameters can be varied to build differently shaped face models. The animation could be done by changing certain segments like size of mouth opening. This parameterized face model unifies the physical description and animation processes. Later, the model was extended to encompass more advanced techniques of generating speech and expression.

Chernoff's work used computer-generated two-dimensional face drawings to represent a k-dimensional space [13]. A detailed encoding scheme was derived using a simple graphical representation of the face. Gillenson of Ohio State University built an interactive system to assemble and edit two-dimensional line-drawn facial images in order to form a computerized photoidenti-kit system [27].

Later, Platt developed a muscle-controlled face model [60]. The model simulated the muscles and facia of the face, naturally producing the bulging and bagging, which occurs when actions are applied to the face. However, this scheme becomes less useful when we have to take care of computational and descriptive complexity. Brennan's work on techniques for computer-produced two-dimensional facial caricatures came on [8], while Weil of MIT reported his work that used a video disk-based system to interactively select and composite facial features at the same time [78]. This study was developed later to come up with computer-based techniques to age facial images especially of children.

Bergeron and Lachapelle produced the short animation film named Tony de Peltrie, which was a landmark for facial animation, by combining three-dimensional facial expressions and speech in order to tell the story [4]. The fictional animated character's face was controlled by parameters which were mapped to a test subject's face. The changes in subject's face when performing certain expressions were digitized and applied to animate the model, resulting in a library of standard expressions. Both lip synchronization and general expression animations were created from this basic library. But the tediousness in this key-frame animation based on control of parameters of a fixed caricature-style model pointed out the weakness of using low-level description of the face for final control.

Pearce et al. encoded a rule-based phoneme compiler that takes a sequence of phonemes specified by the animator and translates them into the appropriate mouth and lip actions [57]. Lewis and Parke used an alternate approach for limited speech analysis using linear predictive coding on spoken phrases to automate the lip-synched animation creation [41]. Hill later described a rule-based phoneme generator using spoken text as input and generating phonemic output [29]. All three methods yield parametric descriptions of human face based on Parke's model, but Hill notes that he would like to adapt the method to a face model with real muscle and bone structure using Facial Action Coding System. The Facial Action Coding System, abbreviated as FACS, is defined by Ekman and Friesen to represent anatomic notation of the expressive abilities of the face [20]. The system describes the actions of face in terms of visible and discernible changes to regions such as eyebrows, cheeks and lips.

Waters formed a new muscle model approach to facial expression animation [77]. In this work, which is also the underlying animation technique for this thesis, the musculature of face was controlled to create a variety of facial expressions. A more detailed explanation for the technique will be presented in Chapter 6. In his work, he algebraically described the function of a muscle as a regular distortion upon a flat grid. The technique is fast and reasonably effective, but may be ignoring some of the peculiar nuances of human face due to its generality. Waite described a similar scheme where he controlled the distortion of entire grid using B-spline surface as opposed to Waters' parametric control of a simple spring-grid distortion of the skin surface [75]. Waite's method allowed more exact modification of basic shape change at the expense of an anatomically less intuitive model.

Magnenat-Thalmann and her colleagues developed an abstract muscle action model [45]. Their scheme formed procedural definitions of the functions of abstract muscles on abstract faces. It was mappable to alternate data sets, and Marilyn Monroe and Humphrey Bogart's faces were emulated as application in the study.

There are also studies, like [16], which use a finite-element model of skin tissue to simulate skin incisions and wound closure, that has been a door to application in surgical planning procedures.

Pieper developed a physically-based model for animating the face, which uses a great amount of detail in the internal structures like facia, bone and muscles[58]. The goal of the system was simulation of reconstructive surgery, so had to accurately represent and preserve the face actions, sacrificing the efficiency due to complexity.

When Pixar produced Tin Toy, receiving an Academy Award, the domain faced the strength of computer animation, and capabilities of computer graphics. In this animated short, a muscle model was used to articulate the facial geometry of a baby into a variety of expressions [55]. Many animations have been built since then, especially coming from Pixar. Toy Story I/II, Dinosaur are among those films where faces of both humans or other objects were animated using

similar techniques.

With use of optical range scanners like Cyberware's optical laser scanner [31], it became very easy to obtain wealth of data for facial animation. Williams used facial image texture maps as a means for three-dimensional facial expression animation [80]. Lee et al. described techniques to map individuals into a canonical representation of the face that has known physically based motion attributes, with the help of new enhanced image processing and scanning technology [40]. In their physically based facial model, they integrated the representation of various layers of facial tissue with dynamic simulation of the muscles' movement. The skin is constructed from a lattice whose points are connected by springs, where the spring stiffness value depends on the biomechanical property of the layer it is a part of. The model offers a point-to-point control and is able to yield more accurately subtle facial deformations such as wrinkles and furrows which are difficult to reproduce with a geometric model.

The recent trade in research is mainly recognizing and tracking faces in video images using model-based [6, 84] or optical-flow based [5, 22] techniques and extracting data from them like facial features which could be used to derive emotional states of the faces. There are some techniques used for locating human faces in images, and locating facial features in face images. The thesis also mentions briefly some of the main studies on parts of a facial animation system like feature extraction and animation in related chapters. With the development of technology, data required to build better models can be acquired more easily which in turn gives rise to new techniques in facial animation. The human brain is very capable of storing and retrieving images, like learning new faces, and recognizing them in very complex environments, which the computers are far away from simulating as well.

Despite so much research on modeling and animating face, synthesizing a realistic face still remains a difficult problem in computer graphics. Each work adds up new techniques to make the produced result look more realistic with better models and texture mapping and animation, however, we can say that the facial animation Turing test has never been passed. We can define the Turing

test for a subject as being unable to differentiate between a computer generated face and real face image. This is due to the fact that "there is no landscape we know as well as the human face" [23], and even a slightest mistake in synthesized face can be perceived by anyone watching. Each work in this domain can be a step to make us understand how the brain works for storing, retrieving and recognizing images, so that we could later be able to synthesize very realistic faces, and simulate brain behavior in similar studies.

2.1 Application Domains

This section includes some example work that shows the application of studied methods in numerous domains, proving usefulness of facial animation. As already mentioned, the most recognizable part of the body that can be perceived by a watching person to differentiate which person it belongs to is the face, therefore, it is sure that facial animation is usually very important as a part of a body animating system, in order to communicate emotions. But still, only the facial region can be in use as application in some domain. Among the very diverse application domains for facial animation are computer games and entertainment, medicine, teleconferencing, social agents and avatars, information assistance systems, and computer-controlled experiment generators. Following is a few examples to such applications.

Film industry: The largest motivator, developer and consumer of three-dimensional facial character animation is the animation industry itself. Advertising and film production are where character animation is extensively used in an ad-hoc manner. Each new project usually requires new models and parameters to be designed and animated, which is generally labor-intensive. Therefore, such needs force new researches to be carried out in order to come up with better models more easily. The development of better equipment is a way that will ease to build more realistic animations, as well as more efficient procedures and techniques.

The systems are traditionally based on key-frame animation, with many parameters that influence the appearance of the face. For example, the models used in Pixar's Toy Story have several thousand control points each [59].

Gaming: The game industry also has a very important dependence on the development of CPUs and graphic processors with better real-time performances. As a result of more capable processors, the gaming industry comes up with games that include better graphics, which means better animations to interact with the player.

The computer games titled Full Throttle and The Curse of Monkey Island use facial animation for their two-dimensional cartoon characters. This trend continues in 3D games like Tomb Raider and Grim Fandango that use facial animation as a key tool to communicate the story to the player [36].

Medicine: The areas of interest with facial animation in medicine are the facial tissue surgical planning and simulation of effects of (dental, plastic facial) surgery [34, 73]. In cases where the facial bones need to be rearranged due to trauma or growth defects, systems that could be used to edit the effects of operation in three dimensions are required. Since such systems will usually require very detailed anatomic modeling, accurate model acquisition and building techniques need to be used as well as caring for efficiency of the system. Example systems usually generate the models using the tomography scans of the head, and the bone surfaces are modeled using iso-surface algorithms like Marching Cubes [42].

For facial tissue simulation, the system needs to emulate the response of skin and muscles after they have been cut and tissue has been removed or rearranged. Plastic surgery is one such application. So each detail in the tissue and bones and muscles may need to be synthesized in the model with an understanding and simulating of the skin tissue dynamics.

Teleconferencing: Video teleconferencing (distant lecturing, visual phones) is about transmitting and receiving facial images, which need to be done in real-time. Despite better communication limits meaning greater bandwidth and larger physical speed, the data to be transmitted is usually very large

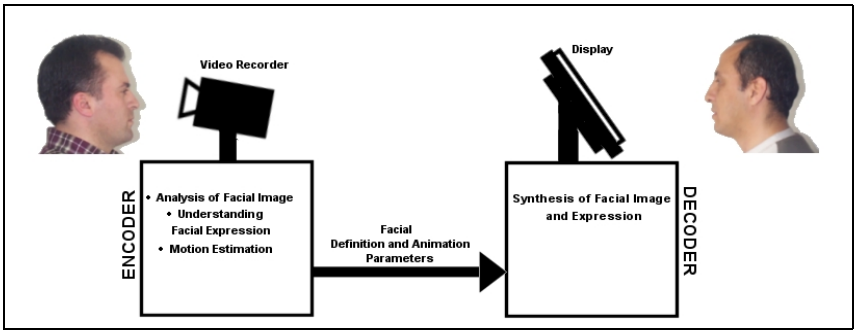


Figure 2.1: The sketch of a one-way teleconferencing system.

bringing a need for better compression algorithms. Research is being carried out in model-based coding schemes [14] and algorithms applied to facial images.

Such systems usually acquire the facial image frame by frame with a video recorder at one end, then these images are analyzed using computer vision and image understanding algorithms. The image analysis techniques gather information about the shape, place and orientation of the head as well as particular features extracted from the face. The data encoded in some way is transmitted to the decoder at the other end, which synthesizes the facial model and the facial expression according to the incoming parameters, that is displayed as an animation of face. Each new frame forms such parameter data at the encoder end, and transmits it to the decoder end to continue facial animation on the display. Figure 2.1 sketches such a one-way video teleconferencing protocol.

Social agents and avatars: Another application domain of facial animation is user interfaces that include characters and agents in order to interact directly with the user [47]. These may be part of many different kinds of applications that have some interface to the user. The animated character or agent may be built to understand commands coming from the user, direct required application to get the wanted results, and then narrate either the result or state of the operation to the user. A simple example can be the quirky characters which navigate the Internet for a result, or search for a file, or search for help on some topic, and display their activity state and results' state through facial expressions.

Educational programs may have such interfaces that communicate with the user using animations accordingly. Talking Tiles is an application of HyperAnimation that aids with the teaching of language skills [26]. There are also applications in order to teach the hearing-impaired how certain words are pronounced through animation of a face model. In order to be more clear on more developed applications, we can give the example of robots designed in order to understand human commands, and act accordingly, which can be thought as animations when the robot is considered as a humanoid model on computer that will narrate results with facial expressions. Virtual reality characters that may be part of e-commerce systems, distance education systems, guidance systems or games will require extensive use of facial animation.

Chapter 3

Face Modeling

In order to realize facial animation, we have to build a face model that will serve our needs. While building the face model, we need to take care of the considerations of points like efficiency of computation for time and space as well as the detail of the model to construct, which will determine how realistic it will look or behave. While a simple 400 vertex polygon mesh could work for a simple facial mask animation, we may need to implement different techniques for representing an accurate face for planning surgical operations that will need to have bones, muscles and a well-synthesized skin tissue with a good model of calculating effects of movements of muscles, bones and skin accordingly.

In this chapter, we first present very brief information on the face anatomy in order to give an idea of what parts we may need to synthesize for a total head. Then, we will talk about how to represent the geometry for the face model. We list some techniques for acquiring data for the face surface, and finally we present information about the face model and corresponding data structure we used in the thesis.

3.1 Face Anatomy

The human body has been studied in great detail for centuries both by medicine in order to understand the inner structure and by artists for producing wonderful drawings and sculptures. We want the face model to look and behave realistic to some extent, therefore we need to understand how the structure of a real face is and how the pieces work together in order to have a good synthesis of it.

A detailed description of the facial anatomy takes place in [56], where both explanations for the face structure and references to get more information are presented. We will just give some important points as a summary and emphasize what may need to be taken care of while modeling the part.

Although the face model we have used in the thesis is just the facial mask showing the front part of the face, we will talk about the head including facial mask, as well as the hair and ears. The head anatomy is composed of mainly the bones which we name as the skull, the muscles which work in order to produce facial expressions and actions like jaw or eyelid opening/closing, and the skin tissue which is what we see when we look at the face.

3.1.1 Bones

The bones of the face add up to a structure called the skull. The skull is both a protective casing for the brain and a foundation for the face. We will not give the nomenclature for each of the bone in this section, but we will talk about how the skull is organized. The skull is formed of the cranium, which is the part that covers and protects the brain, and the skeleton of the face forming the rest. The upper part is where the cranium takes place, and has no parts that can move.

The skeleton can be defined as the bones that take place in the frontal view of the face, excluding the forehead which is supported by the frontal bone as a part of cranium. The upper third of the facial skeleton consists of the orbits and nasal bones, the middle third consists of the nasal cavities and maxillae where

the upper teeth are located, and the lower third consists of the mandibular region carrying the mandible, which is the only freely jointed bone structure of the face. Figures 3.1 and 3.2 display the frontal and lateral views of the skull, respectively.

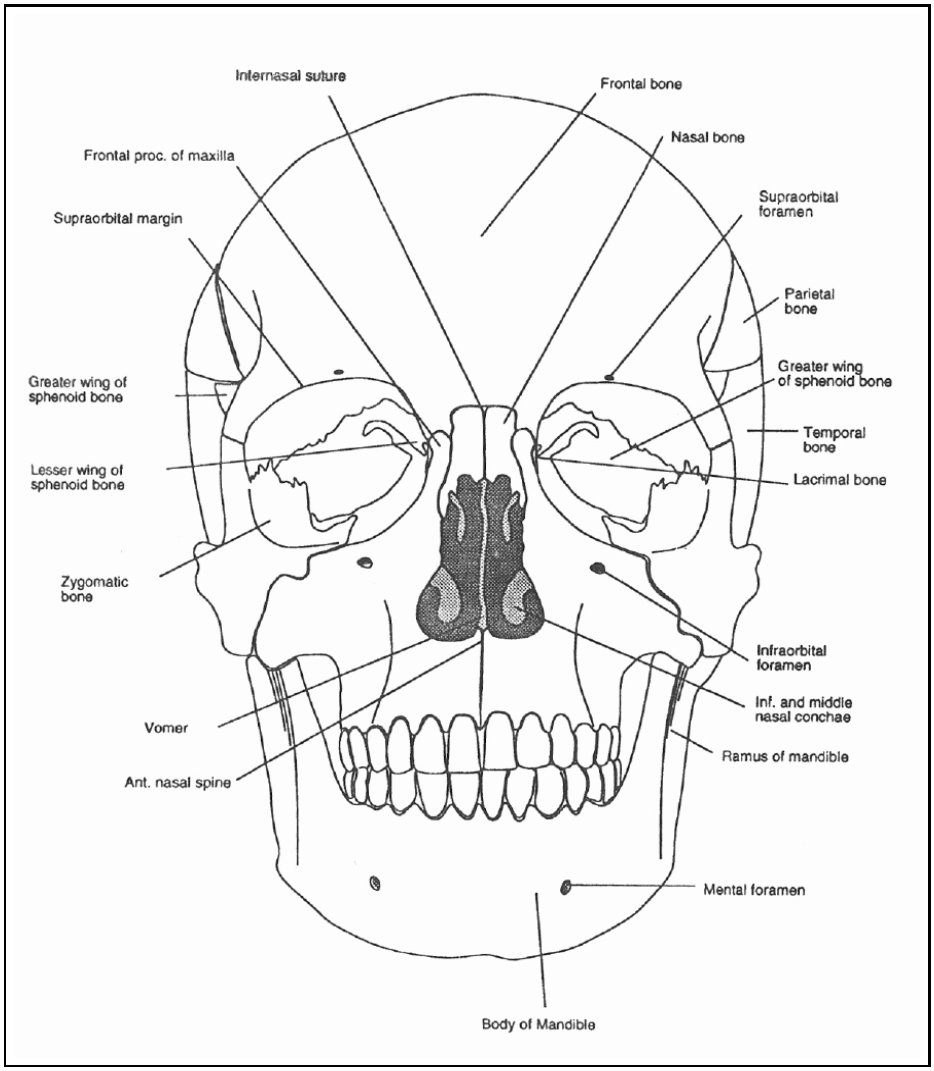


Figure 3.1: The frontal view of the skull [56].

The cranium looks like a deformed sphere covered with the skin having hair on it. It has some roughness on it, but is usually modeled smoothly, and it connects to the neck in the back lower part. The facial skeleton's upper parts shape the facial view, carrying the eyes, nose, cheeks and the upper teeth in it. The lower part, called the mandibular region carries the lower teeth, chin and part of the lower cheeks. The mandible is the only moving bone of the face, which adjusts the mouth opening, so we should be able to simulate the jaw rotation according

to the working of mandible.

The neck part carries a seven-cervical-part of vertebra, and is surrounded by a cylindrical looking flesh, and carries some muscles (and other tissue) that connect head to the body.

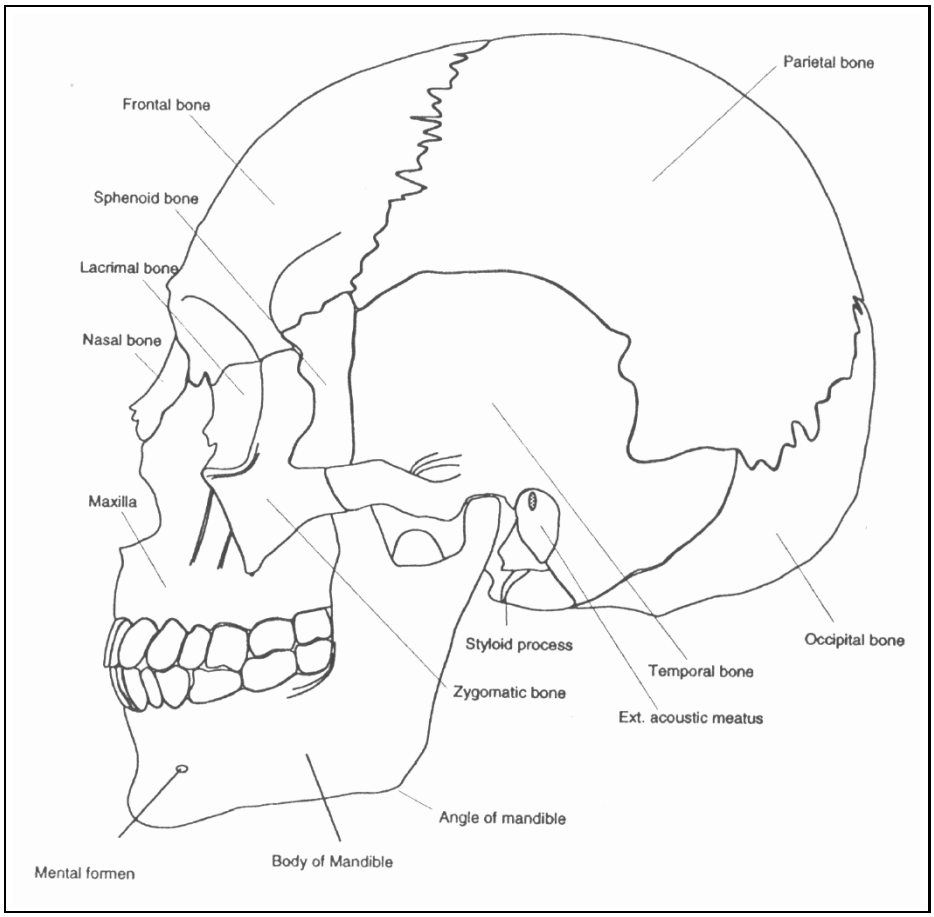


Figure 3.2: The lateral view of the skull [56].

3.1.2 Muscles

Muscles, being the organs of motion in the body, work by contraction and relaxation between the parts their ends are connected to. Muscles connect either two bones, bone and skin, two different areas of skin, or two organs in order to cause motion between them. Figures 3.3 and 3.4 display the frontal and lateral views

of the facial muscle structure, respectively.

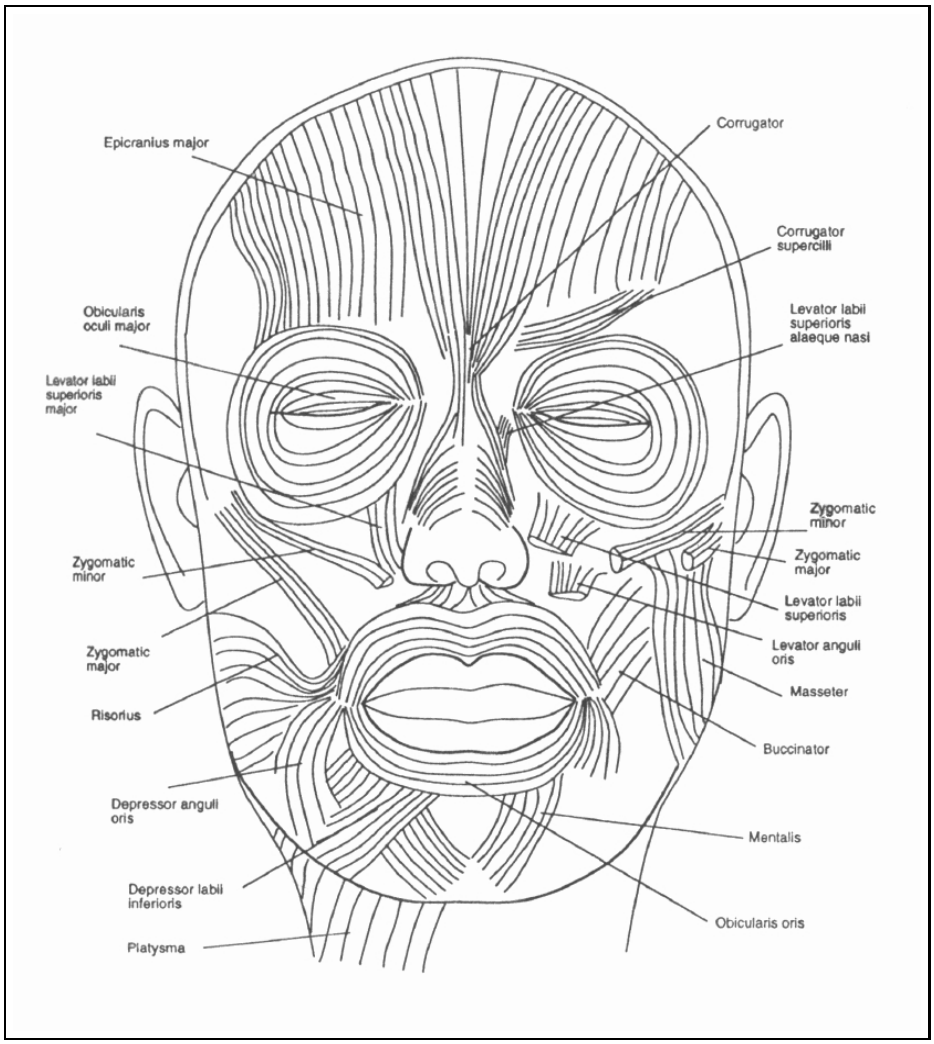


Figure 3.3: The frontal view of the facial muscle structure [56].

The muscles of the face are commonly known as the muscles of facial expression, but some of them carry out other actions like moving cheeks and lips during mastication and speech, or opening/closing of eyelids, which themselves can be part of facial expression. The facial expression muscles are superficial and all attach to a layer of subcutaneous fat and skin at the insertion point which is the end they attach to move. Some of them are totally on the skin with both origin and insertion end points, like the orbicularis oris surrounding the mouth. The muscles on the face function as teams working synergistically and not independently.

The orbicularis oris surrounding the mouth is important in speech animation since it gives its shape to mouth while talking. Most important of the other facial expression muscles that are synthesized in our face model will be explained in that section. The mandible moves with coordinated action of the muscles attached to it, and can protract (move forward), retract (pull backward), elevate (close mouth), depress (open mouth). The movement of the mandible can either be modeled as effect of movement of corresponding muscles, or some other way like rotating the jaw part of the face around the joint region. There are also muscles which take place around the ear, in the neck, and in the tongue.

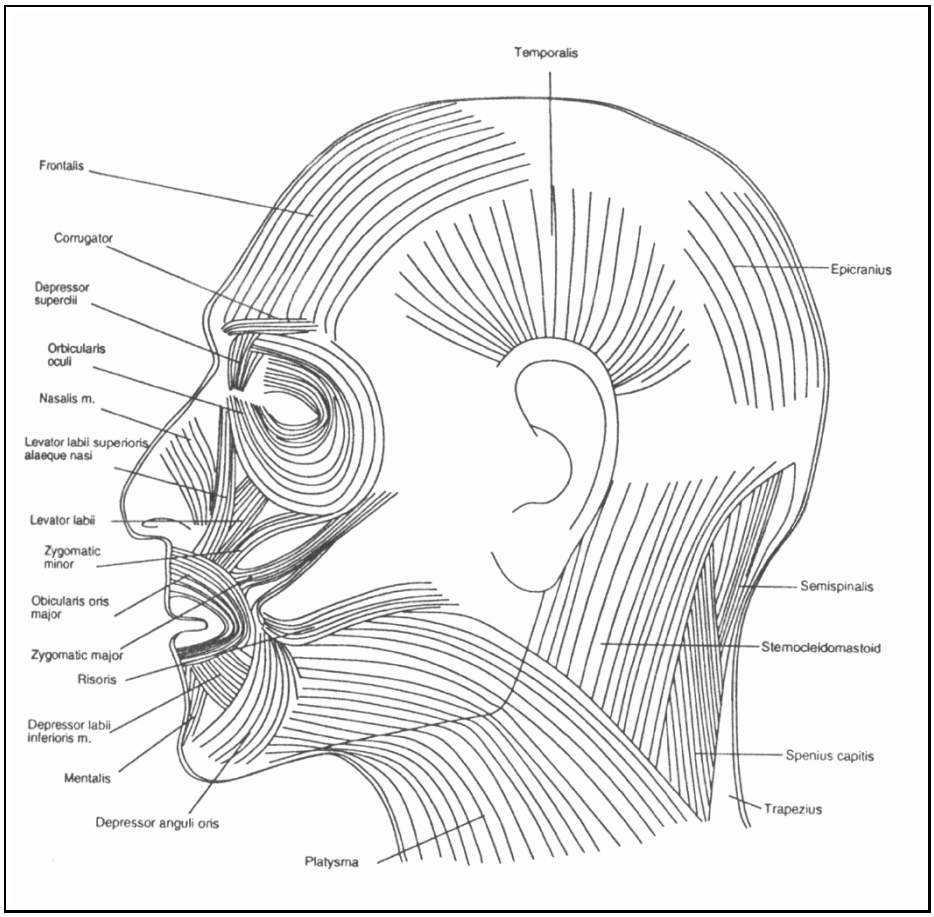


Figure 3.4: The lateral view of the facial muscle structure [56].

3.1.3 Skin

The skin covers the entire external surface of the human form and acts as a highly specialized interface between the body and its surroundings. Its multicomponent microstructure can be defined mainly as a intertwined network of collagen, nerve fibers, small blood vessels and lymphatics covered by a layer of ephitelium and transfixed at intervals by hairs and the ducts of sweat glands. Figure 3.5 gives a basic sketch of the skin layers.

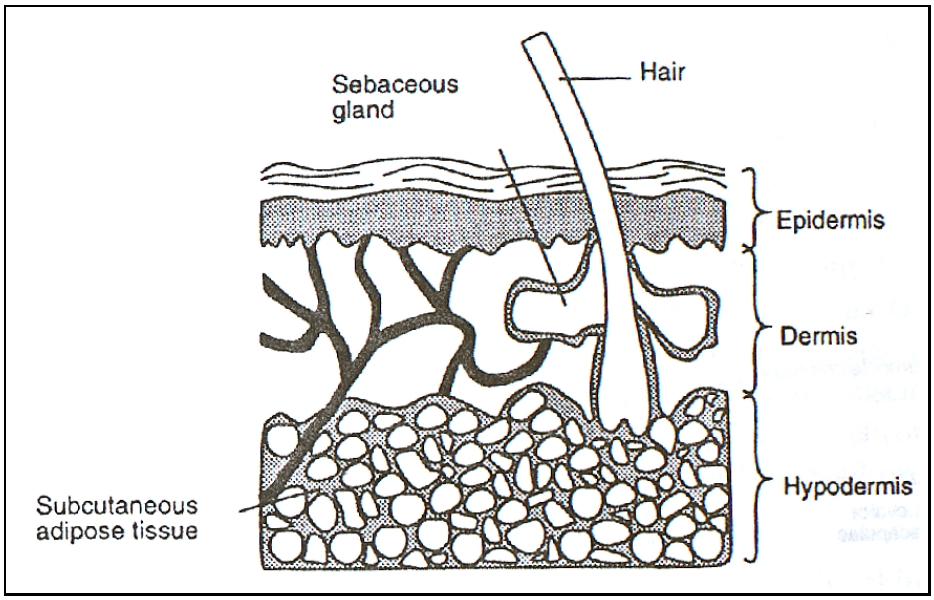


Figure 3.5: Skin layers [56].

The thickness of the skin varies all over the face. Some part of the skin has wrinkles due to loss of underlying fatty tissue and elasticity as the person grows older. In facial animation considered as computer animation, the skin is very important as it is the part displayed to the person watching. Bones and muscles are not visible (if the model is not for surgery purposes) usually, so we have to model the skin and render it realistically using some techniques like texture mapping.

Other parts to be considered while modeling are the eyes, tongue, teeth and the hair. Basically, the eye is usually modeled as a half sphere representing the front part of eyeball that can be seen between the opening of the eyelids with a

colored (or textured) part for the pupil. In a synthetic eye model, the eyes should converge and not stare into the space, the pupil should be able to dilate, and the highlight reflection of the surface of the iris and cornea should be modeled with care.

3.2 Modeling Faces

After describing some properties of the face anatomy, we will present information about commonly used procedures for getting data for a face model, and representing it in digital world. The final model we have built should involve the geometric descriptions and animation capabilities along with representation of additional attributes such as surface colors and textures. Normally, a face model will be designed to approximate the real face, since it cannot exactly represent the real anatomy due to its complexity like wrinkles and creases as well as the great detail in the structure of the skin tissue. Only medical visualization and surgical planning will require much more detail in the model; character animation will use models that approximate the wanted features.

The animation capabilities of the face is determined by the structure of the model itself, so, decisions must be made in the stage of modeling to come with the right model serving its objective. Jaw moving, eyelid opening, facial expressions are among animation capabilities that a face may be designed to perform. Therefore, the necessary details and flexibility should be included in the model.

3.2.1 Facial Geometry

In this section, we list a few of the methods to represent the face model geometrically that would let effective animation and efficient rendering. Having the geometric representation of the model, we can implement procedures to deform its shape for forming expressions and animation and render the result to have it look more realistic.

3.2.1.1 Volume Representations

A way to represent face is to use one of the many volume representation techniques. Among these techniques are *constructive solid geometry (CSG)*, *volume element (voxel) arrays*, and *aggregated volume elements*, such as octrees.

CSG is a method that is commonly employed in computer-aided mechanical design systems. The basic idea is to use Boolean operators on regular mathematical shapes like cube, sphere, cylinder, planes to form the final shape. Since the face is much more complex, such a method is not very useful in face design, but it may be useful for simple cartoon faces.

Voxel representation is used mostly in medical imaging to describe anatomical structures. Usually, two-dimensional slices of three-dimensional structures are obtained first and then assembled to get to the final model. Computer tomography or magnetic resonance techniques are used to acquire the two-dimensional slices of data. Since the whole volume of the model is kept as unit volume elements each having some property, this technique requires large amount of memory. Also animation will be more difficult for such representations. Instead of direct voxel methods, techniques such as Marching Cubes [42] can be used to extract surface geometry models of anatomical structures from the voxel data. Animation can then be done using the extracted surface models.

3.2.1.2 Surface Representations

The preferred geometric basis to represent facial models is the surface primitives and structures. The surface structures should allow different shapes and changes in shapes in order to be able to represent new faces as well as to animate them.

Following is presented briefly some of the possible surface description techniques that can be used for geometric modeling.

Implicit surfaces can be used to model the surface geometry of the face using a collection of analytical surfaces. An implicit surface [7] is defined by a function

$F(x, y, z)$ assigning a scalar value to each (x, y, z) point in space. The set of points satisfying $F(x, y, z) = 0$ defines the wanted implicit surface. For example, we can describe a sphere of unit radius centered at $(1, 1, 1)$ with the implicit function $F(x, y, z) = (x-1)^2 + (y-1)^2 + (z-1)^2 - 1 = 0$.

We can blend and constraint properties of implicit surfaces to allow creation of models that would be difficult to build with other techniques. However, using implicit surfaces is not a preferable technique for facial animation, since it is difficult to interact with a collection of them, and also it will take more time to manipulate and display due to computation requirements.

Parametric surfaces are three-dimensional surfaces that are defined by bivariate parametric functions. Three functions of two parametric variables are used; one function for each of the spatial dimensions based on quadric or cubic polynomials. Most popular of these surfaces, referred to as tensor-product parametric surfaces, are defined in terms of control values and basis functions, such as the B-splines, Beta-splines, Bézier patches and non-uniform rational B-spline surfaces(NURBS) [3]. We can express these surfaces as

$$S(u, v) = \sum_i \sum_j V_{i,j} B_{i,k}(u) B_{j,m}(v), \tag{3.1}$$

where $S(u, v)$ is the parametric surface, $V_{i,j}$ are the control values, and $B_{i,k}(u)$ and $B_{j,m}(v)$ are the basis functions of polynomial orders k and m, respectively. A matrix representation can be used for this formulation, as in

$$S(u, v) = [u]\ [B_u]\ [V]\ [B_v]^T\ [v]^T, \tag{3.2}$$

where $[u] = [1\ \ u\ \ u^2\ \ \ldots\ \ u^{k-1}]$ and $[v] = [1\ \ v\ \ v^2\ \ \ldots\ \ v^{m-1}]$. We choose the parameters u and v to vary over the interval $[0.0, 1.0]$ for each surface patch, and different basis functions B_u and B_v are used according to the type and order of the surface.

Waite [75] and Nahas et al. [49] used bicubic B-spline surface modeling techniques to model faces with relatively few control points. However, using few control points means sacrificing from the detail in the model, and also places like creases become difficult to implement as they require defeating surface continuity properties. Wang's Langwiedere facial animation system was implemented using hierarchical B-spline surfaces, where hierarchical refinement was employed to add local surface detail including the interior of mouth and the eyes and the eye sockets [76]. Facial expression animation is controlled using simulated muscle functions to manipulate the hierarchical surface control points.

Polygonal surfaces can also be used to represent the facial model's surface. Since modern graphics workstations are adept at displaying polygonal surfaces, polygonal representation of facial models with modest complexity can be updated in near real time. Essentially, all facial models are displayed using polygonal surfaces, in form of regular polygonal meshes or arbitrary networks of connected polygons; even the techniques discussed above are approximated with polygon surfaces for display. The model we use in the thesis is also represented with a polygon mesh.

While modeling with polygons, the designer should come up with a mesh that will allow shape changes in the face. In order to keep the polygons planar when the shape changes, mostly triangles are used. Modeling with required detail for being able to form facial expressions is needed, which can be achieved by using more detail in places with high curvature. The areas with more control points will also require more detail to behave well for expression animation. For rendering to be true, the creases in the face like the under-eye region should match with edges of polygons; and it would give a better effect of shading discontinuity if the vertices at the creases have different normals for opposite sides of the crease. Since the face is almost symmetric, the models are built for one half; the other half of the face is formed by reflecting the first one.

In order to determine the surface shape for a face model, one of the following data acquisition techniques is generally employed:

- three-dimensional surface measuring,
- interactive surface sculpting,
- assembling faces from components, and
- creating new faces by modifying existing faces.

The techniques for three-dimensional measuring are using digitizers, photogrammetric techniques or laser scanning techniques. The three-dimensional digitizers are used to locate positions of certain points in space with mechanical, electromagnetic or acoustic measurements. It may be time-consuming, and to be able to have accurate results to measure vertices for a polygon mesh on the face, the face shape should not change, so using sculptures will be preferred.

The photogrammetric method is capturing facial surface shapes and expressions photographically, by taking multiple simultaneous photographs each from a different point of view. The face to be photographed may be painted with lines of the polygon mesh structure wanted, whose locations are then defined accordingly from the different-view photographs.

Laser-based surface scanning devices like Cyberware's [31] produce a very large regular mesh of data values in a cylindrical coordinate system. Surface color data can also be captured simultaneously with the surface shape data. The details of these techniques and parameterized face models are given in [56].

3.3 Face Modeling in our System

The polygonal face model used in our work is the face model used in Keith Waters' facial animation system [77]. The polygonal face data is stored in a format called "Simple Mesh Format" (smf) [25]. In this format, first the vertices in the mesh are listed in each line starting with a 'v' character and giving the x, y and z coordinates of the vertex. After the vertex list, lines starting with an 'f'

character begin, which correspond to the faces (polygons) in the mesh. Each line lists the three indices for the vertices that the triangle has.

The facial animation system first reads in the vertices into an array of type `vertex` structure, where each element has floating point numbers (x, y, z) for the coordinates. The structure also has fields for (x, y) texture coordinates, the (x, y, z) coordinates for newly calculated position of the vertex after deformation of the face according to the new facial parameters, the (x, y, z) values for the normal on that vertex, and the list of polygons the vertex is a part of.

Then, the data for triangular faces forming the face model is read. There are 256 vertices and 439 polygons listed in the file. This mesh forms only one half of the face, which we reflect to get the full facial mask. The vertical line dividing the face into two is taken to be common to both halves. This is for calculating the normals for the vertices on this line correctly to give a smooth look. The resulting model consists of a facial mask from the throat up to the top of the forehead, extending until ears from left and right. Figure 3.6 gives a wireframe display of the model.

The system also reads the data for facial muscles defined in a text file, which lists the names of muscles, the locations for head and tail of the muscles, and related values like influence zone.

3.3.1 Muscles

The muscle model used in the system is first described by Keith Waters [77]. In the model, the facial muscles are modeled as linear muscles. A linear muscle deforms the mesh like a force and can be modeled with parameters determining its effect as described in Chapter 6, where we will also explain how the muscles are used for animating the face.

After reading the muscle values, we initialize them with the list of two edges the head and the tail is closest to, in order to determine the new places when the face model is calibrated. There are nine symmetric pairs of muscles defined on

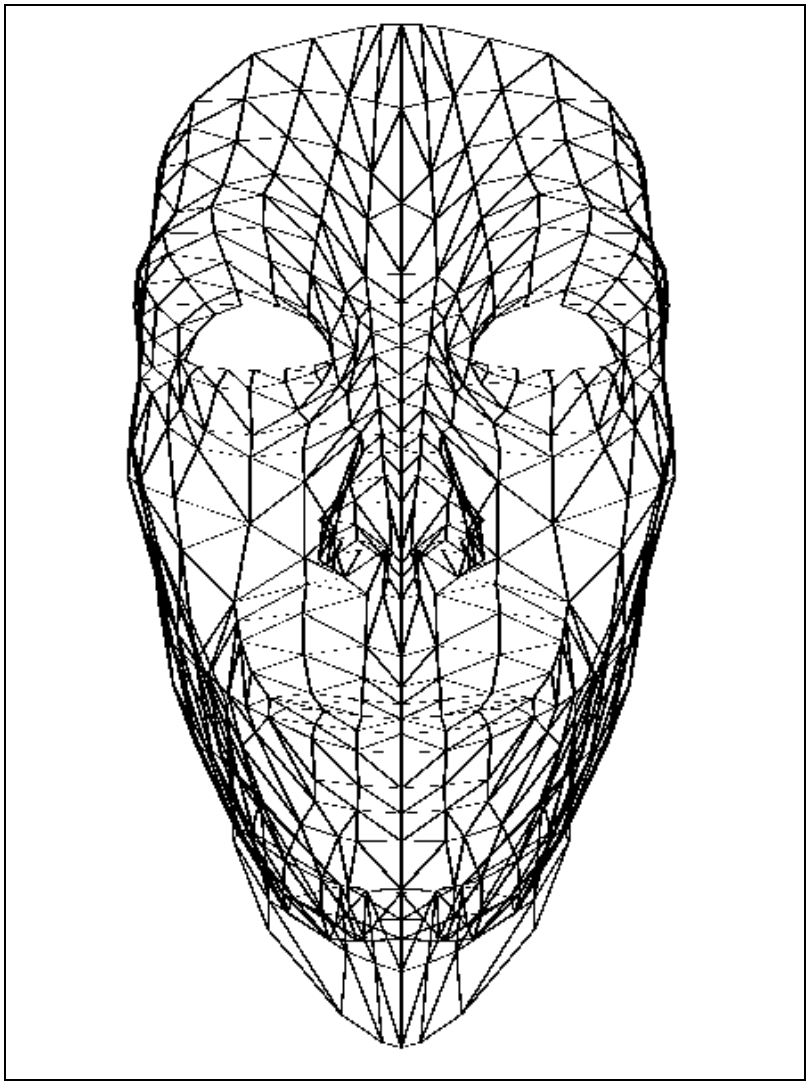


Figure 3.6: Wireframe display of the face model.

the face model. Figure 3.7 gives the facial muscles structure.

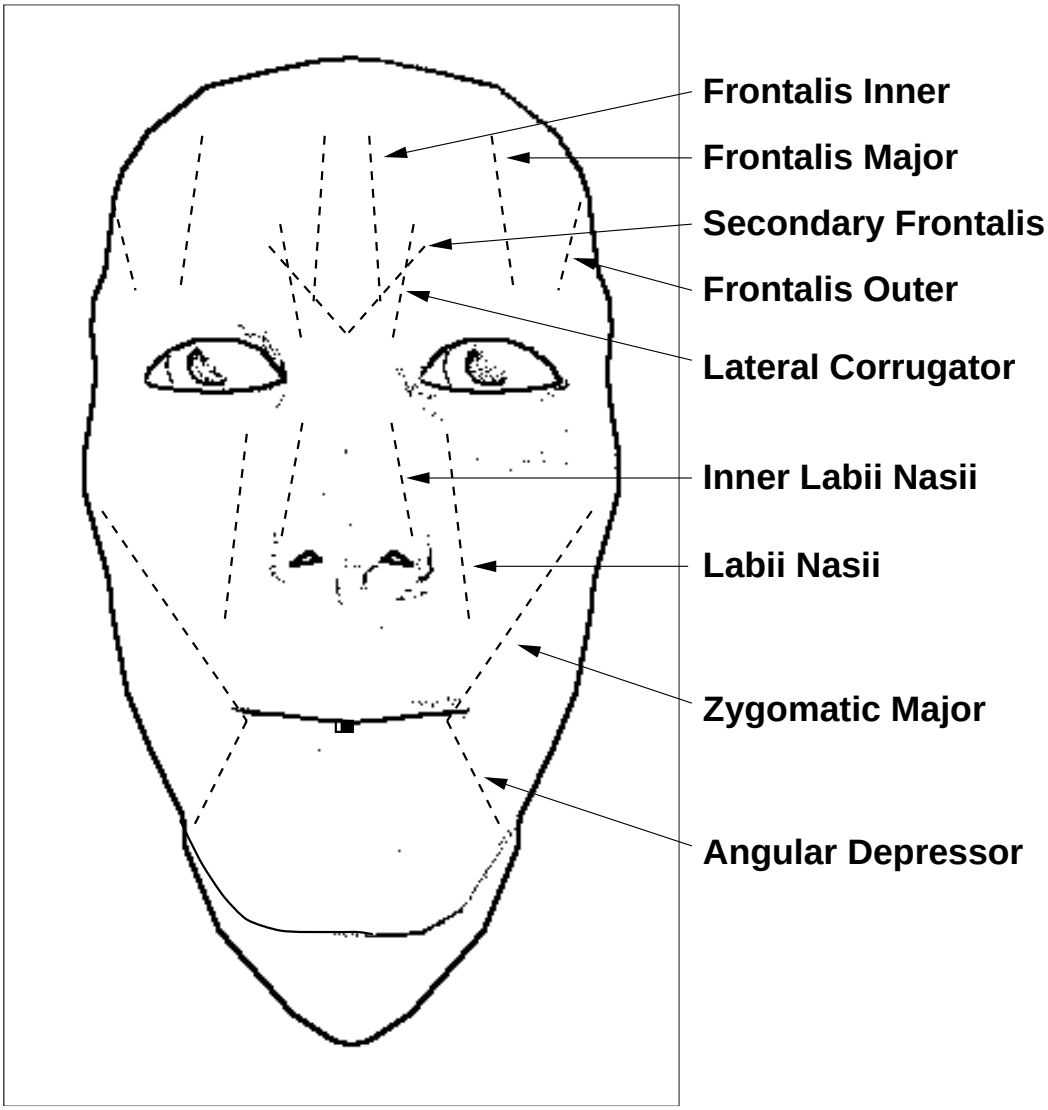


Figure 3.7: Facial muscles.

The muscles used in the face model are briefly described in the sequel.

1. *Zygomatic Majors:* This muscle pair arises from the malar surface of the zygomatic bone and inserted into the corner of the mouth. Technically, it elevates the modiolus and buccal angle, meaning that it elevates the corner of mouth which is an action that occurs in expressions like smiling and happiness.

2. *Angular Depressors:* This pair is also connected to the corner of the mouth, but elongates down to the bottom part of the chin. When contracted, they pull the corners of the lip down. The sadness expression is modeled with contraction of these muscles.

3. *Labi Nasi:* This pair is placed on the cheek with one end connected to the top of the lip. Their contraction causes to pull the lip upwards, which may be a part of disgust expression on the face.

4. *Inner Labi Nasi:* This pair is closer to the nose on the cheek, and behaves similar to the Labi Nasi pair. The origin is on the sides of the nose, and the insertion point is close to the bottom side of the nose. When contracted, the sides of the holes of the nose move upwards.

5. *Frontalis Outer:* This pair of muscles are found on the upper side of the eyebrows close to the outer side on both right and left part of the face. The outer side of eyebrows is raised with contraction of these muscles. When the face will be deformed to have a surprise expression, one of the muscles activated is this pair.

6. *Frontalis Major:* This muscle is also above the eyebrows, but in the central region. The effect is raising the eyebrow, when it is contracted. Surprise and fear expressions include activation of this muscle.

7. *Frontalis Inner:* This pair is at the same level with other Frontalis muscles, but is on the top of the nose closer to the center of the face above the eyebrow. Contracting this muscle will pull the inner part of eyebrow upwards, as in the fear expression.

8. *Lateral Corigator:* These muscles connect the top of the nose to the inner side of the eyebrow. They are close to the Frontalis Inner muscles, but they have a more horizontal orientation and are vertically at a lower level. Contracting this muscle wrinkles the eyebrows, which will give an anger expression to the eyebrow part of the face.

9. *Secondary Frontalis:* This last pair is placed near the Lateral Corigators, but is positioned more vertically. When they are contracted, the inner parts

of the eyes raise. The fear expression includes such a muscle activation.

These facial muscles are enough to model most of the facial expressions. However, if we want to have more realistic facial expressions, we can define new ones similarly, and activate to get finer results. For example, the orbicularis oris surrounding the mouth can be simulated by putting a few linear muscles as in Figure 3.8, which will be useful if we want to do speech animation [72].

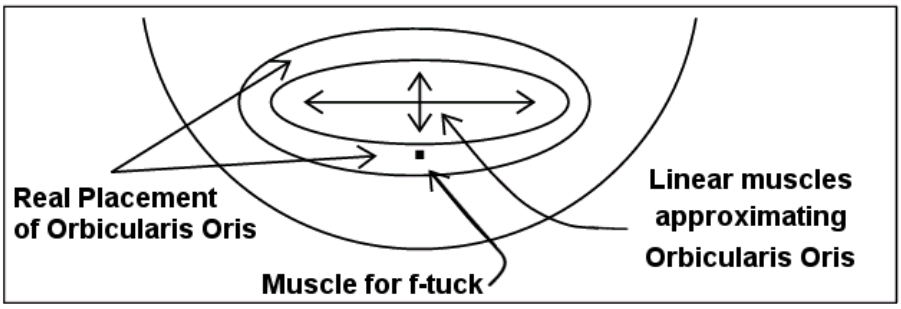


Figure 3.8: Simulation of orbicularis oris using linear muscles.

3.3.2 Eyes

The eyes are modeled as hemispheres in our model, similar to Keith Waters' model. However, since our facial model is calibrated with new parameters, the shape of the eyelids change, which may cause some parts of the hemisphere to be visible in front of the eyelids. To avoid this, we calculate the place of the hemisphere center as the center of the eye, and scale it in x, y and z directions to fit it behind the eyelid without causing such defects.

The muscle-based polygonal face model we used is adequate to simulate facial expression animation, and look like the synthesized person's face with extracted parameters and texture. However, it does not involve the complete head, so is missing the hair and ear parts, which takes much from its believability. However, we can still parameterize it as a generic face to represent new faces by deformation. The reality of the face will be higher when the model involves the whole head, since it will include more features that will help the viewer to recognize the synthesized image.

Chapter 4

Facial Feature Extraction

Feature extraction is the determination of the parameters that will define the face in order to deform a parameterized face model to build the face shape corresponding to the desired face. This is a part of our facial animation system, where we use two orthogonal pictures of a face to get the parameters, which are applied to our model. The deformation process is a subject of the next chapter. This chapter will first introduce the MPEG-4 Standard for facial animation, which sets the frame for parameters to define a face. Then, background of this subject mentioning some studies realized for similar works will be presented. Finally, the technique used in the thesis for extraction of the facial features is explained, followed by ideas which could further be used to improve the process.

4.1 MPEG-4 Standard for Facial Animation

Parameter set design is an important concept, where certain points should be under consideration. The first important point is, of course, having a set that will represent any face such that when two faces differ, the parameters representing them should also differ; and when we are given the parameter sets we should be able to design the corresponding faces that will look similar to the original faces at least in terms of geometric topology. An ideal and complete parameter set will

be able to represent any facial topology with any expression.

The MPEG-4 Standard is an effort to come up with certain rules to define any visual scene including any object. Apart from the conventional concept of accepting the audio/visual scene as composed of a sequence of rectangular video frames and associated audio track, the standard takes a scene as a set of Audio-Visual Objects (AVOs). The sender performs the encoding of the parameters that define these AVOs, and transmits them via a single multiplexed communications channel. Then the decoder extracts the information to build the corresponding AVOs to form the scene.

The Moving Picture Experts Group (MPEG) has built the standard in a period of over five years with some modifications to update to new versions on the proven success of the digital television, interactive graphics applications with synthetic content and the web having distribution of and access to content. It provides standardized technological elements that will realize integrating the production, distribution and content access. The main aspects of MPEG-4 are

- coding objects forming a picture independently,
- the ability to use these objects to compose the scene interactively,
- combining graphics, animated objects and natural objects in one scene, and
- transmitting scenes in higher dimensions.

A detailed explanation of the MPEG-4 Version 2 Standard for a Face and Body Animation object (FBA) can be found in [12], and a more detailed version for the MPEG-4 Standard overview is in [48]. In this thesis, we are interested in the Facial Definition and Animation Parameters defined by the standard.

An animated face can be rendered using the "facial animation object" of the MPEG-4 Standard. In order to control the shape, texture and expressions of the face, Facial Definition Parameters (FDPs) and/or Facial Animation Parameters (FAPs) can be used. The standard also defines the bitstreams via which the

related parameters can be transmitted. But only the FDPs are of interest to us for the feature extraction. We want to have a set of parameters that will define the face we want to construct, so that we can deform our parameterized model to build the corresponding topology.

When constructed according to the FDPs, the face rendered will have a neutral expression, which can be changed using animation as talked over in Chapter 6. MPEG-4 supplies the FAPs for this purpose, over a stream which will change the expression on the constructed face in time causing the animation. Our system does the animation using Keith Waters' muscle-based animation approach [77]. Still, the program could be developed to be incorporated with a MPEG-4 FAP stream decoded into respective muscle state vectors to realize the animation. Figure 4.1 displays the facial definition parameters of MPEG-4 Version 2.

4.2 Background on Feature Extraction

Many researches have been reported in the domain of feature extraction, since extracting the recognizable elements in a picture, or more generally image understanding, is an area that has very wide application. We may need to understand an image for localizing an anatomic structure in it [69]. However, we are interested in recognizing human faces. Even that will have wider application areas. We may be trying to track a face in a video and detect its pose as in [46], which may be part of a complex system that could include facial animation. Studies that try to extract facial expressions with reasoning as in [63], models for visual speech feature extraction [43], face and gender determination [81] are only a few among numerous works under research. However, still the methods in such applications can have uses in feature extraction from faces for parameterization, too. Many studies like [62, 64, 74] give algorithms to identify lips in a face, or shape of lips in order to help visual speech applications, which may be useful for extraction of lip features in a facial animation system.

There are many studies for detecting and extracting features from faces. A

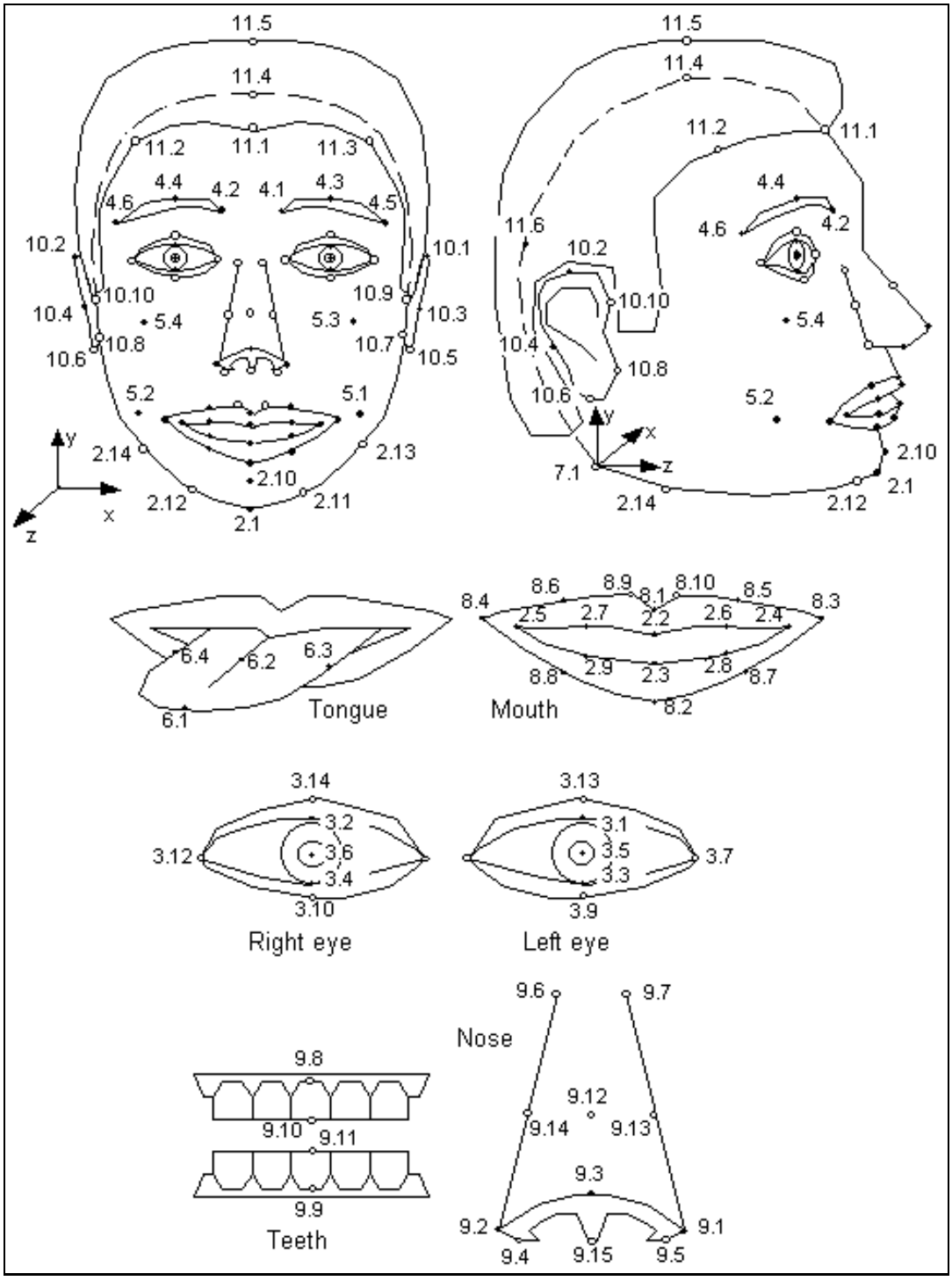


Figure 4.1: The MPEG-4 Face Definition Parameters [48].

method called active contour models and balloons is explained in [15]. Elastic bunch graph matching to extract the features is described in [82]. Using eigen-templates for face recognition is reported in [67]. Turk and Pentland review alternative approaches for face recognition in [71]. Mainly there are two approaches for facial recognition. The first is computing a set of geometric features from pictures of faces. Kanade started such an approach with his pioneering work in [32]. The approach tries to determine the relative position and other distinctive parameters like eyes, nose, mouth and chin even in pictures of very coarse resolution. The other important technical approach to face recognition is using template matching. In this approach, templates of a face are used to match to that in the picture with varying parameters as in [2]. In [9], the two approaches are compared in terms of results and performance.

Next, we will give a brief description of a technique named snakes that can be employed in template matching. This technique could be incorporated into our system for improving the feature extraction process. Various other researchers make use of snakes in feature extraction of either facial or other kind of images [52, 68, 70].

4.2.1 Snakes

Snakes are used to track edges in an image. We can define a snake as an energy minimizing spline under guidance of external constraint forces and image forces like edges. When placed nearby edges in an image, the snakes are attracted towards the edge, finally localizing on it. Therefore, we could use this active contour model to identify certain features in the face, when we first roughly locate the place where they will be initially placed. A snake will always minimize the value for total energy which is designed to be minimal when aligned with a contour in the image. We also incorporate internal spline forces to control smoothness of the contour while we give weights to the other terms in energy function to guide the snake as we want it to behave.

The snakes algorithm is first developed by Kass, Witkin and Terzopoulos

as active contour models [33]. However, as Amini et al. later pointed out, the algorithm has problems like numerical instability and also points have a tendency to gather on strong portions of the edges [1]. They further proposed a dynamic programming algorithm that was more stable and could include hard constraints, but was very slow. With n being the number of control points in the snake spline and m being the size of neighborhood a point is allowed to move in a single iteration, the algorithm had a complexity of $O(nm^3)$. Later Williams and Shah came up with an $O(nm)$ algorithm using a greedy approach [79]. It also improved the older algorithms and used more evenly spaced control points that brought a better and more accurate estimation of curvature of the edges with inclusion of a continuity and curvature term into the energy functional.

We can represent a contour as a vector $v(s) = (x(s), y(s))$, s being the parameter to denote the arc length. The snake algorithm is mainly defining an energy functional and finding the best localization of the snake spline that will minimize the energy. The original energy functional can be formulated as

$$E_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 (E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds. \tag{4.1}$$

In Equation 4.1, the term E_{int} is the internal energy of the curve arising from bending or discontinuities. Image forces like the edges, lines and terminals are represented in term E_{image} term. The external constraint forces are gathered in E_{con} term.

Williams and Shah Algorithm allowed a contour to converge to an area of high image energy meaning the edges by controlling the first and second order continuity. The energy functional to be minimized is modified to

$$E = \int (\alpha(s)E_{cont}\beta(s)E_{curv} + \gamma(s)E_{image})ds. \tag{4.2}$$

In Equation 4.2, E_{cont} and E_{curv} correspond to E_{int} of Equation 4.1. E_{image} is the term measuring edge strength or intensity in the image. As in the previous

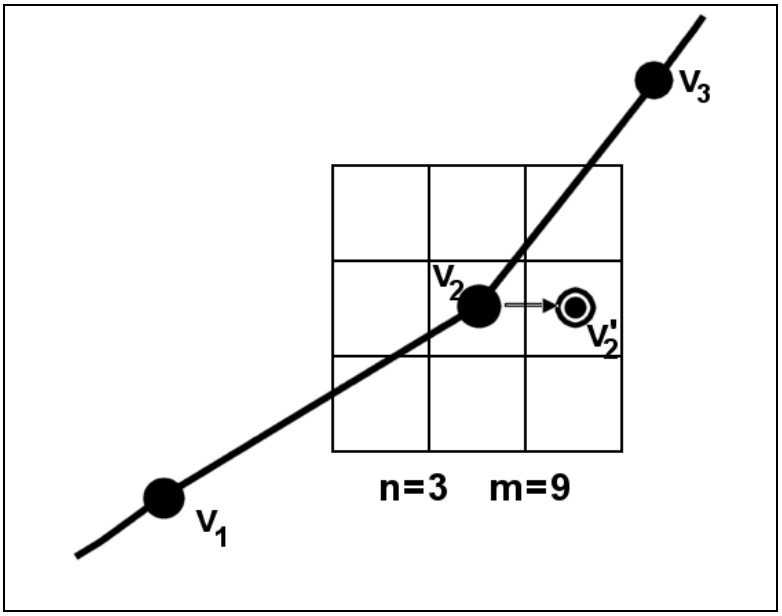


Figure 4.2: Computation of the new location of a control point in an iteration.

algorithms, the contour's new location is calculated iteratively until the minimal energy is reached. In each iteration, a new location is found for each point. To do this, the energy function is computed for the points in the neighborhood of a point (including the point itself) that will be relocated. The neighborhood with the minimal energy is set to be the new place for that point.

In Figure 4.2, if we are computing the new location for v_2 and v_2' has the minimal energy of the nine neighborhood points, we move v_2 to v_2' for this iteration.

The study chooses α to be 1 and γ to be 1.2 so that image gradient determining whether the pixel in the image is an edge is given more importance over the continuity term. β is taken as either 0 or 1, which is decided according to whether that location is assumed to be a corner or not.

To avoid the contour control points to shrink for minimizing distance between them, E_{cont} term is adjusted which provides an even spacing to have the contour behave for satisfaction of first order continuity. The algorithm uses the distance as $d - |v_i - v_{i-1}|$, the difference of average distance between all control points and the distance between the current and previous points. This forces points spaced with a distance close to average distance to have minimum energy value. The d

value is re-computed in each iteration, and the distance value is normalized to [0,1] by dividing it to the largest value in the neighborhood formed of candidate points the point can move.

The curvature term of the energy functional is also normalized similarly, and is computed as $|v_{i-1} - 2v_i + v_{i+1}|^2$. This is reasonable enough as the first term, E_{cont}, that already spaces the points relatively evenly.

The E_{image} term computes the image force taken as the gradient value for this algorithm. The gradient value is between 0 and 255. Given the gradient value of a point as mag, and the minimum and maximum of such edge strength values in its neighborhood , the normalized edge strength term is computed as $(min - mag)/(max - min)$, which will have a negative value to minimize the term on edges since a point on an edge will have larger gradient value. When the gradient values are very close, we adjust the minimum to be 5 less then the maximum to prevent large differences in this term's value through areas where gradient value is uniform.

The curvature value is computed at each iteration for each point, and β is set to 0 for the value which is a curvature maximum, and left as 1 for the rest, in order to give feedback to the energy minimization process. We compute the curvature as

$$\left| \frac{\vec{u}_i}{|\vec{u}_i|} - \frac{\vec{u}_{i+1}}{|\vec{u}_{i+1}|} \right|^2, \tag{4.3}$$

where $\vec{u}_i = (x_i - x_{i-1}, y_i - y_{i-1})$ and $\vec{u}_{i+1} = (x_{i+1} - x_i, y_{i+1} - y_i)$. For avoiding corners to form before the corner is near an edge, we check the gradient value to be above some threshold. The computation of curvature is followed by a nonmaxima suppression to decide for the corner points of next iteration as those curvature maxima points having curvature value above a threshold. The greedy algorithm developed by Williams and Shah is given in Figure 4.3.

A good set of values for the thresholds determined after experiments is 0.25 for $threshold_1$ to determine whether to set β to 0, 100 for $threshold_2$, a small value

```
Initialize α_i, β_i to 1 and γ_i to 1.2 for all i
do
/* loop for relocating points */
   for i=0 to n
      E_min = MAXIMALVALUE
      for j=0 to m-1
         E_j = α_i E_cont[j] + β_i E_curv[j] + γ_i E_image[j]
         if E_j < E_min then
            E_min = E_j
            j_min = j
      move point v_i to location j_min
      if j_min is not the current location,
         increment ptsmoved by 1
/* this part determines where to allow */
/* corners in the next iteration */
   for  i=0 to n-1
      Calculate curvature using Equation 4.3
   for  i=0 to n-1
      if ( (c_i > c_{i-1} and c_i > c_{i+1})
            /* curvature is larger than neighbors */
            and (c_i > threshold_1)
            /* curvature is larger than threshold 1 */
            and (mag(v_i > threshold_2)) )
            /* edge strength is above threshold 2 */
            then β_i = 0
until ptsmoved < threshold_3
```

Figure 4.3: Williams and Shah Algorithm.

between 2 and 5 for $threshold_3$ to decide whether we have converged according to the number of points that moved in that iteration. As explained, the snakes algorithm will locate a defined contour on a nearby edge using this method. Therefore, this can be used in extraction of facial features, after we have roughly found out the places for the features to allow us define and initialize the contour. This can be used, for example, to locate nostrils above the lips in a frontal view image of the face, or to decide on feature points like nose top on a profile image. Below we describe a similar algorithm, named as *deformable templates*, to locate features that can be fit to a template on the face.

4.2.2 Deformable Templates

The deformable template matching algorithm to be presented in this section is a work of Alan L. Yuille [83], for determining facial features in order to define the face for a facial recognition system. The approach tries to extract the facial features using a priori shape knowledge and spatial relationships between them.

A deformable template is designed in two stages:

1. We define a geometric model of the template for the shape of the feature.
2. We specify the imaging model for how the template will appear in the image and decide on the matching criterion to guide the template's interaction with the image.

The current edge detector algorithms will not be able to reliably locate the salient features of the face, therefore a guidance for the matching criteria is required to bring more accuracy to the process. The deformable template approach of Yuille, based on Fischler and Elschlager's work [24], determines some templates according to the expected shape of the feature as a geometric model which can change its parameters using the image data to match to it dynamically. The location and parameters of the template is determined by minimizing an energy function using intensity, edges, peaks and valleys in the image.

First, the image is preprocessed to get the valleys, edges and peaks in the fields:

$$\Phi_v(x) = -I(x) \qquad \Phi_e(x) = \nabla I(x) \cdot \nabla I(x) \qquad \Phi_p(x) = I(x). \tag{4.4}$$

$I(x)$ can be taken as the intensity value for a point x on the image. The valleys are the points which have low intensity; peaks have high intensity in the image, and the edges can be determined by calculation of gradients in image intensity. The next section will briefly mention about Sobel operator [51] for determining gradients and edges in an image as used in the thesis for localization of feature regions. The fields in Equation 4.4 take their largest values at low intensity, edges and peaks, respectively.

4.2.2.1 The Eye Template

The geometric template for the eye can be designed as described below. This shape is determined after experimentation of different eye shapes.

- To represent the boundary between the white region of the eye and the iris, a circle of radius r, centered on a point x_c is used. The interior region of this circle will be guided to match to the low intensity valleys in the image.
- Two parabolic sections are employed for matching to the boundary of the eyes, using edge information. It is centered on x_e, has width $2b$, maximum height a for the boundary above the center, maximum height c for the boundary below the center, and θ as the angle of orientation.
- Two points are used to be attracted to peaks in the image standing for centers of white regions of the eyes. One is labeled as $x_e + p_1(cos\ \theta, sin\ \theta)$ and the other is $x_e + p_2(cos\ \theta, sin\ \theta)$, with x_e being center of the eye and θ being the orientation angle as described before.

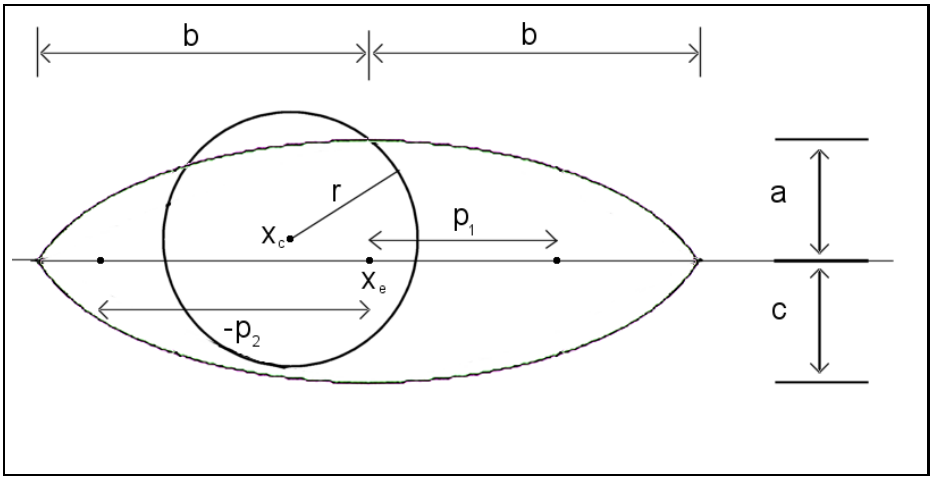


Figure 4.4: Geometric shape of the eye template.

- White region of the eye takes place in the region between the bounding contour and the iris. The area tends to move towards the large intensity values, the peaks.

The components of the eye template are held together by

- the forces that keep the iris center and eye center close,
- the forces that make width $2b$ to be around 4 times the iris radius, and
- the forces that keep the center of white regions to be roughly midway from center of eye.

Figure 4.4 displays the geometric shape of the eye template that can be defined using the nine parameters x_c, x_e, p_1, p_2, r, a, b, c, θ.

To help in calculating places for points when the orientation changes, two vectors are defined:

$$e_1 = (cos\ \theta, sin\ \theta)\ \ and\ \ e_2 = (-sin\ \theta, cos\ \theta). \tag{4.5}$$

Now, we can represent a point x as (x_1, x_2) where $x = x_1 e_1 + x_2 e_2$. The top half of the parabola can be represented as $x_1 \in [-b, b], x_2 = a - a \cdot x_1^2/b^2$. Similarly, the bottom half of the boundary is the region where $x_1 \in [-b, b], x_2 = c - c \cdot x_1^2/b^2$.

A potential energy function to be minimized can be given as

$$E_c = E_v + E_e + E_i + E_p + E_{internal}. \tag{4.6}$$

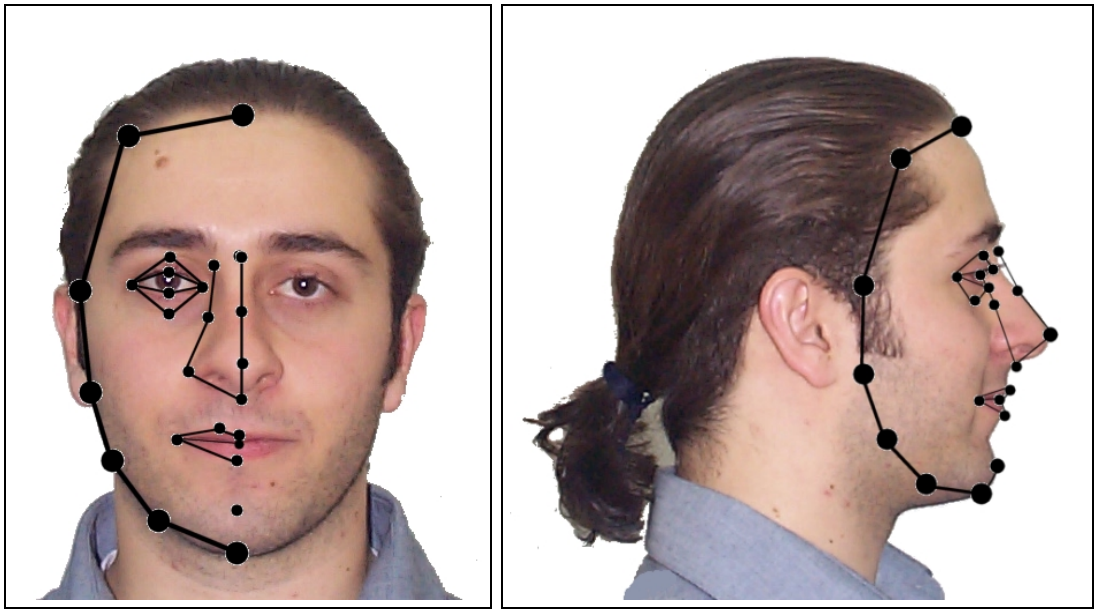
This function will make the algorithm converge and give a measure of the goodness of matching of the eye template. The details of the formulation of this energy function, as well as the refined versions of the function that will make the process act more robustly, can be found in [83].

This algorithm is useful for locating the places of features that can be defined with a fine geometrical model. It might be good to incorporate the method to develop feature extraction in addition to the technique used in the thesis that is to be described in the next section. However, the method alone may not be adequate since every single feature defined in MPEG-4 FDPs may not be adjusted to be extracted on some deformable template, where heuristics and reasoning may come into application to complete the rest of the process.

4.3 Feature Extraction Process in our System

The features we use to deform our model contains a subset of the MPEG-4 facial definition parameter set since our model is not a complete head, but just the facial mask. However, the set consists of the parameters that can be handled by being extracted from two orthogonal photographs of a head and applied to the parameterized face model to construct the fitting topology. Figure 4.5 gives the facial features we used on a face image.

We implemented feature extraction process partially. The system currently determines rough positions of the feature elements. In fact, image understanding is so error-prone that the best algorithms can fail in some ways due to the quality



(a) (b)

Figure 4.5: The facial features used for deformation process: (a) frontal view, and (b) lateral view.

of images, noise, lighting conditions, or some other elements in the image. Even when the images are taken under ideally defined conditions with predefined lighting, no shadows, perfectly recognizable background, etc, an unexpected wrinkle on the face can mislead the algorithm which may cause the rest of the algorithm to behave unexpectedly. Therefore, the algorithm we used may also be misleaded by shadows or shiny regions due to abnormal lighting.

The photographs we experimented were frontal and profile head photographs which did not have other elements on the face like mustache and beard and eyeglasses. The profile images had the faces looking to the right side of the image. Hard shadows, especially on profile images, did cause the algorithm to think some features to appear on the shadows of the face. Some further processing could try to avoid this, but we should accept that shadows are unexpected because incorporating some shadow ignoring algorithm may be forcing the program to ignore parts that look like shadows. Therefore, image understanding is a very difficult process, since the computers do not work as the human brain, which recognizes, stores and retrieves images in a way that is not exactly known.

The algorithm we employed to roughly locate facial features on face interprets the intensity levels and edges at different threshold levels by reasoning to make decisions. Before explaining this process step by step, we first briefly describe how we calculated the intensity values and detected edges in the images.

For all operations in feature extraction, we used the grayscale value for pixels to represent the intensity value of the image. The calculation for this is simply adding some contribution of each component in RGB to get the grayscale value for that pixel, which is a good approximation to separate the grayscale component from hue and saturation components. Setting each of the RGB components to this new value will produce the grayscale image, since the result found is the luminance component and the color will be a tone of gray when the R, G and B are equal. The formula is like

$$Grayscale = 0.212671\,R \;+\; 0.715169\,G \;+\; 0.072169\,B. \tag{4.7}$$

Equation 4.7 also represents that the human eye is most sensitive to green, then red, then blue. Other methods can be found in [10], but the best result is achieved with Equation 4.7.

The edges in an image can be detected with different algorithms, among which are gradient based edge detection with Roberts, Sobel or Prewitt filter operators, Laplacian based edge detection, Canny's edge detection method. An edge operator is a neighborhood operation which determines the extent to which each pixel's neighborhood can be partitioned by a simple arc passing through the pixel, where pixels in the neighborhood on one side of the arc have one predominant value and pixels in the neighborhood on the other side of the arc have a different predominant value. The edge detection method we used in the thesis is gradient based employing the Sobel operator.

Gradient-based edge detection calculates a gradient value for each pixel where the gradient will be high in regions with a fast change of grayscale intensity value. It is assumed that edges are the pixels with a high gradient.

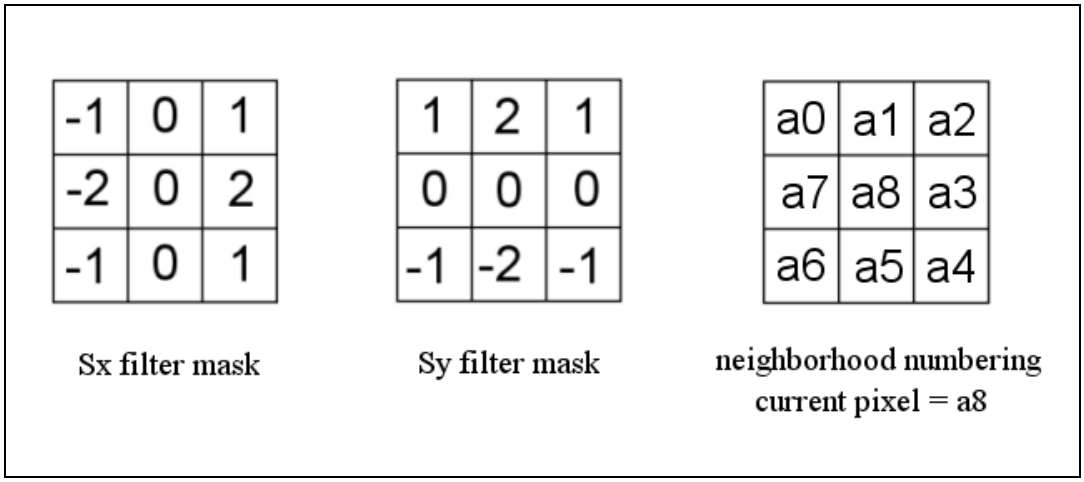


Figure 4.6: Sobel filter masks.

For gradient calculation, we use the grayscale image that can be built by using the Equation 4.7. Then, we use the Sobel operator, which is a spatial filter convolved with the image to calculate the gradient. Figure 4.6 gives the Sobel spatial operator for x direction, Sobel spatial filter for y direction, and the labeling of pixels in the 3x3 neighborhood window of a pixel.

Convolution is an operation on two dimensional signals, calculated as

$$y(n_1, n_2) = T[x(n_1, n_2)] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) \cdot h(n_1 - k_1, n_2 - k_2), \qquad (4.8)$$

where $h(n_1, n_2)$ is the impulse response of a linear shift-invariant system T. We can simply think this as taking a window and wandering it on each pixel in the image to calculate the result for each pixel as some operation between the convolved window and matching neighboring window of the pixel. Then, convolution of this filter amounts to adding the corresponding multiplication of the filter window with the neighborhood window, which will give the formulae in Equations 4.9 and 4.10 for the calculation of S_x and S_y for each pixel in the image.

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6), \tag{4.9}$$

$$S_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4), \tag{4.10}$$

where $c = 2$ in the Sobel filter. This operator places an emphasis on pixels that are closer to the center of the mask. The gradient value is computed as the magnitude of the gradient using the two partial derivatives, like $M = \sqrt{(S_x^2 + S_y^2)}$.

Rest of the edge detection process is determining on a threshold to decide on the pixels which will be taken as part of edges. The average value of gradients amongst all pixels can be a threshold, to tell that all pixels having gradient value above the average value is an edge pixel. To view the result, the edge pixels are shown as black, and the rest as white in our program. The threshold can be changed to emphasize only the stronger edges. We can also use some reasoning on gradients to decide on edge pixels, like first building the edge map then examine it to ignore some small edge pieces. We, for example, employ a Median filter further to avoid noisy edges that look like dots in the results. The nonlinear Median filter is calculated by replacing each pixel value in the image with the median of the pixel values in the 3x3 neighborhood of that pixel.

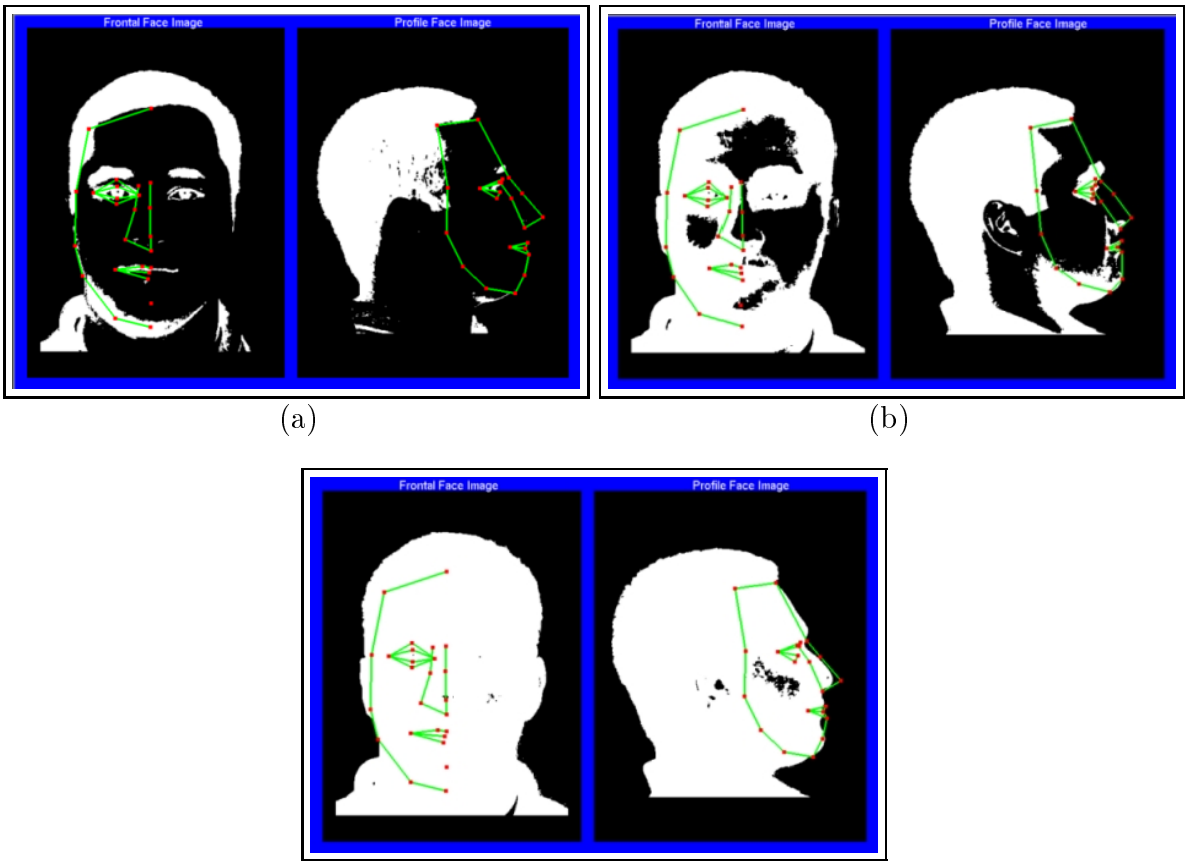
4.3.1 Feature Extraction on the Frontal Image

For extraction of the features on the frontal face, we use different images that are built from the grayscale form of the original image. The first one is the intensity image which can be considered as the peaks image, where a pixel is black if the grayscale intensity value is above 85, white otherwise. The number 85 is found after experimentation over the facial images used as example in the thesis. 85 is a value lower than the average intensity value. A typical face photograph as used in the thesis will have a minimum intensity value around 20, maximum around 250 and average around 145. In such an image, the background is lost into black pixels, and the face is outlined as a white region inside which the eyes, nose and the lips seem as white regions and the rest as black. The other image is the edge

map of the first image, with a threshold of 4. Since the first image is a black and white image, the gradients at the edges are high, so 4 is a good threshold for edge detection in the second image. Other images we used are intensity images with different thresholds like one with slightly above the average intensity value in the image and edge maps of them when required. In this image, the head is a white region in the inside, too, and the background is black. Figures 4.7 and 4.8 show intensity peak and edge images with different thresholds on a sample face.

In the algorithm, first we wander on the pixels from the top to bottom to get the highest point, which is the top of the head. This point is also the top of the head in the x direction, since the bumps due to hairs are removed in the image, and the head's elliptic shape has the middle top point in the top point of the image. Next, we use a symmetry-based approach to locate the left and right side of the head on the intensity image with threshold 85. We check a region from the head top we found to half of the rest of the image downside. We sum the number of the white pixels in the y direction for each x value. We see that the results show a symmetrical distribution, with maximums being at the left and right points of the head. We also can decide that the middle point between these two maximums is the symmetry axis for the head.

Then, we build an intensity image with threshold slightly lower than 40, which is also decided by experimentation. After that, we get the edge map for this image. In this image, the eyes are dominant inside the facial region. A similar edge pixel counting on this image starting from the symmetry axis to the left of the image will have high values on the eye region, and the left side of the head region. We decide on the rough area of the eyes as the region with high values closer to the symmetry axis. The left side of the head is the starting of high values to the left of the eye region. The eye region is bounded from left and right using the start and end of high values in this region. Then, we refine the eye region with top and bottom boundaries by counting the number of edge pixels in the horizontal direction inside the left and right boundaries. The values become small between the eye top and eyebrows. They also tend to get small below the eyes, where we define as the below of eye boundary.



(a) (b)

(c)

Figure 4.7: Intensity peak images with different thresholds on a sample face: (a) low threshold, (b) average threshold, and (c) high threshold.

(a) (b)

Figure 4.8: Edge images with different thresholds on a sample face: (a) low threshold and (b) high threshold.

Now, we go down the intensity picture with threshold 85, starting from the eye level, counting the white pixels in x direction in the region between left of the eye and the image symmetry axis. This helps us to locate the mouth and the bottom for the facial region as the bottom of the chin. The nose region usually behaves unexpectedly in the image so we decide the nose bottom to be roughly at 7/8th of the distance between the mouth and the top of the head. The remaining feature points on the chin and cheeks are also located by estimation using the points we already found out. Figure 4.9 shows the final results on a sample image for this stage of the extraction process.

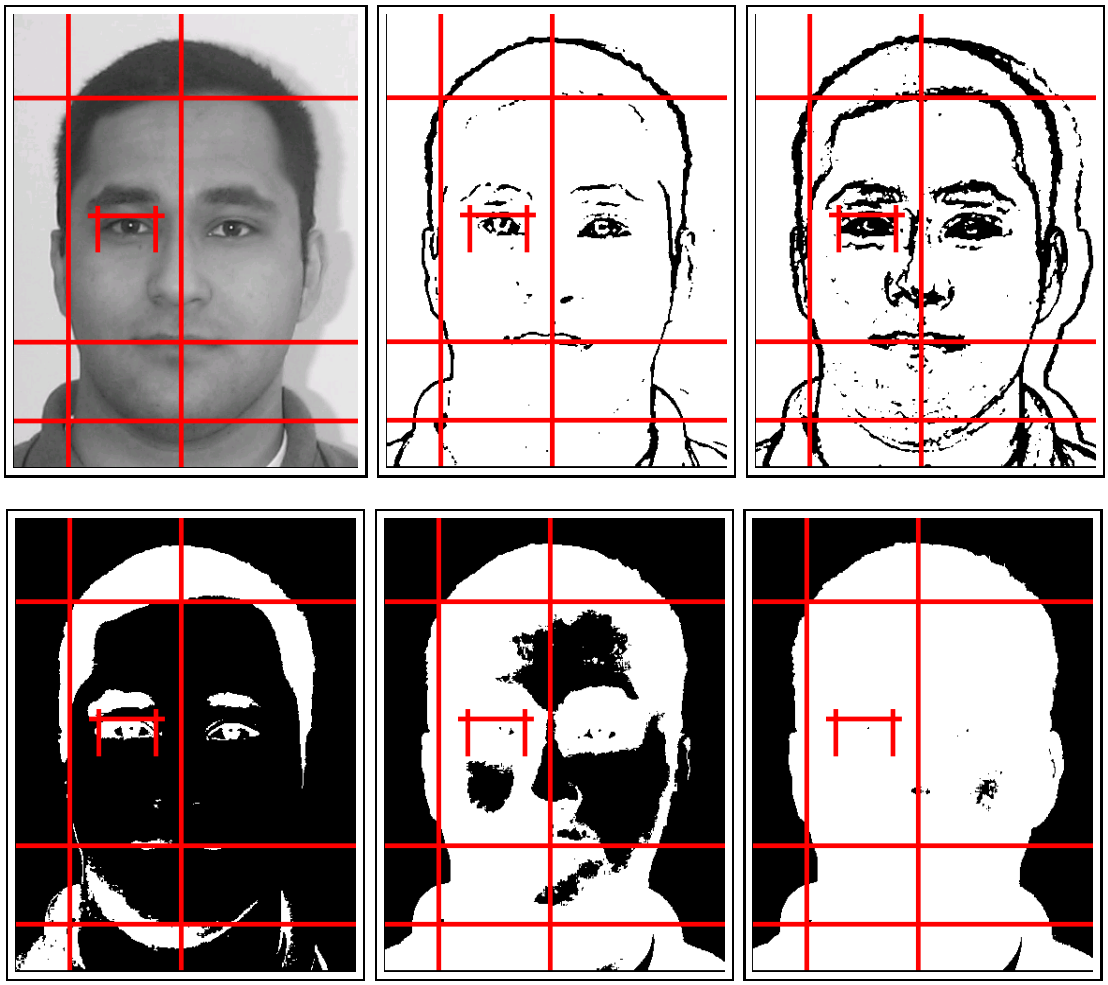


Figure 4.9: Final results of the frontal feature extraction process.

4.3.2 Feature Extraction on the Profile Image

For feature extraction in the profile image, we use a similar approach with the intensities and edges. The profile image has a face figure facing to the right of the image. We locate the top of the head similarly, by looking at the intensity image with threshold 85. Since the hair bumps are removed at this level, we get a rough location of the top of the hair region. Next, we search for the nose region. It is simply the rightmost point in the middle region of the image within the high threshold intensity image displaying a white silhouette of the head. The only problem is that when there is a shadow of the face on this region, the algorithm will be mistaken by the shadow.

Next, we look for the bottom of the chin. This region is below the nose top, and where the throat region will start. Therefore, we see that the throat region starts where the intensity image will have white points in the right to be farthest from the right boundary of the image below the nose. The chin is roughly where the distances are at average level above the throat beginning. We find the nose bottom similarly by going down the nose top to an average level.

To locate the eye region, we use the shape of the nose. The nose ends up in the top where a curve occurs at the region it is connected to the forehead. What we do is to let a point move upwards on the boundary of the nose in the high threshold intensity image, until such a region is encountered, which we decide by checking whether the point is starting to move to right after reaching the minimal point. In a low threshold intensity image, around this region only the eyes are visible, whose boundaries we find to get the eye region. We also locate the left part of face, when we go to the left starting from the eye region.

Next, we look for the forehead top. We see that, in a low threshold intensity image, we encounter this point when we go above the eye region, for heads carrying hair. We can use some reasoning to calculate it with a ratio on the forehead boundary to have it work for pictures with very short or no hair. We go to left at this point to locate the left of the forehead region. Final results of the feature extraction process on a sample profile image are given in Figure 4.10.

Using similar approaches and more example images, we can develop a better algorithm to more accurately locate the regions. An approach using color, shape and symmetry based cost functions can be found in [65]. We could further implement a snake and/or deformable template based approach to refine the locations for some features. But no matter what algorithm is used, image understanding is error-prone and may fail under different conditions. Our facial animation system, therefore, uses manual determination of features, and just employs automatic extraction to roughly initialize the feature mask on the image and leave the rest to the user.

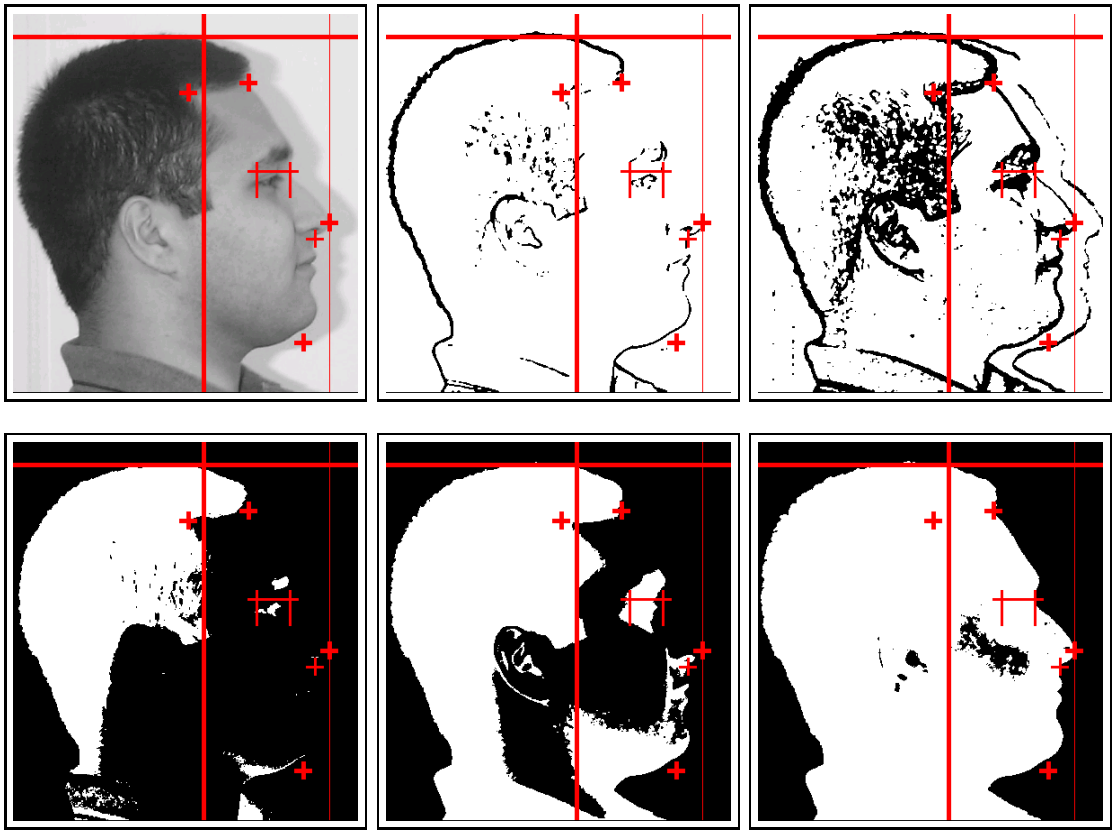


Figure 4.10: Final results of the feature extraction process on the profile image.

Chapter 5

Deformation of the Face Model

This chapter provides the description of the parameters we use to define the face, and how to deform the generic face model according to these parameters. Recently, MPEG-4 became a standard for face definition and animation. The parameterized models since then were mostly compatible with the MPEG-4 parameter set, like [21].

The model for different faces can be either obtained by some of the methods as told in the face modeling chapter, or a generic model can be deformed using the parameters to build the corresponding face model. This second one is the method we use in the thesis. First, we extract the facial features as described in the previous chapter. Then we use these values to set the values of the parameters for our parameterized model. In fact, the parameters are almost the ones we extract from the face photographs, which are almost a maximal subset of MPEG-4 parameters for the facial mask model we have. The generic model we use is Keith Waters' model. Although we use such a parameter set for deformation of the face model into corresponding face, the animation is done using the muscle-based animation technique of Waters [77].

5.1 Background Work on Modeling Individual Faces

Kurihara and Arai's work [35] is an example of creating faces from a single canonical face. They use a 3000 vertex model digitized with a laser scanner from a mannequin model. Control points are chosen among these vertices, and once control points are set, the remaining vertices are arranged by interpolation using the control points.

Another method to get new faces from existing faces is described in [44]. In this technique, local deformations are applied to portions of the face. In the study, Magnenat- Thalmann et al. define five ways to select region of deformation and four ways to apply transformation to the regions. According to them, the local region to be selected can be specific vertex numbers, or vertices within a bounding box, or vertices within a cylindrical slice, or vertices with a specific color value, or set operations between one of these methods. Once the vertices of interest are selected, the new locations for the vertices can be computed as either moving them towards a reference vertex according to specific percentage of the displacement vector from the reference vertex to each vertex, or a percentage of a specific reference vector for translation can be applied to all vertices, or the new locations can be found by scaling them according to their distance from a specified plane, or a variation factor can be used to decide on how to transform parts of the region of vertices. The local deformations can also be based on ellipsoidal volumes and warping functions as in [18].

Another method used for deformation is FFD, standing for free form deformation as described in [66]. The idea is to define a cube of control points in which we embed the face model. Then the cube is deformed in some way to deform the model inside. The method can further be applied using different shapes of control points, which may give better results as the shape will look like the face and the control points will be placed well. The FFD method is combined with Dirichlet surfaces [17] to produce more flexible control point volume to get a better deformation [39].

As it is seen, many studies bring different approaches as solution to the deformation process, which have points in common. Of course, the best method could be having the model be built using a laser scanner from the face of the person. But, what we want is a faster and easier way that will require less resource. So, using a generic model detailed enough to represent any face with a well-defined parameter set and application of deformation accordingly is preferred in facial animation. We employ such a method in the thesis. Although the face model we use may not have very great detail, the parameters and local deformations as we will mention in the next chapter will give good enough results.

5.2 Face Deformation in our System

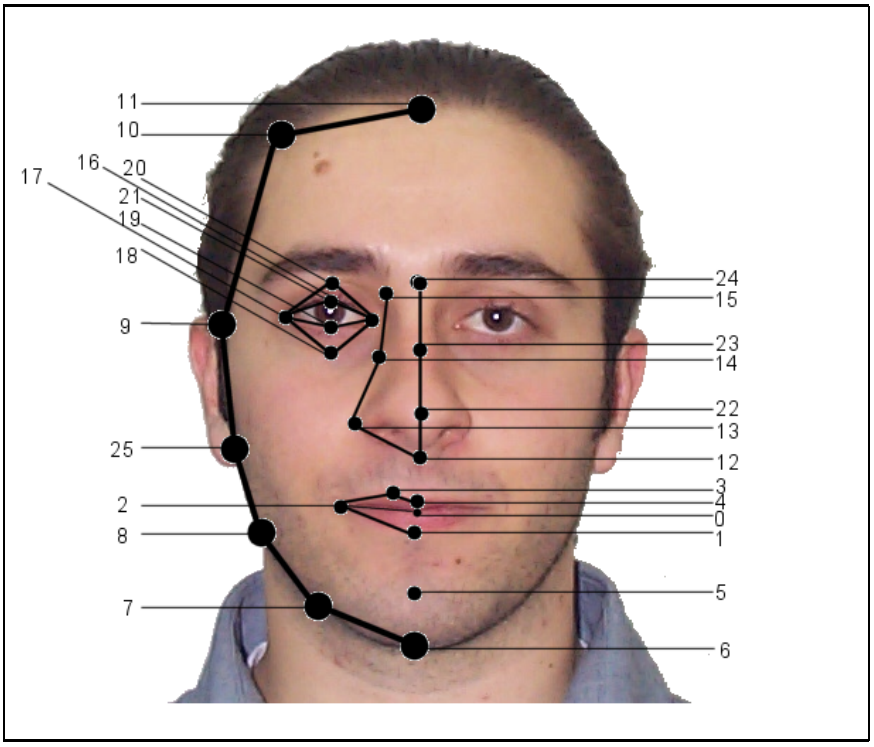
We deform Keith Waters' face model [77] as a generic face model by using the parameter set explained in this section. x and y coordinates of the vertices are adjusted using the frontal parameters extracted from the frontal image of the input face, then z coordinates are arranged according to the profile image features.

In the deformation process, we first deform the mouth part, then the nose part, the eyes, the chin, the forehead part and then arrange the cheek part by filling between the deformed parts.

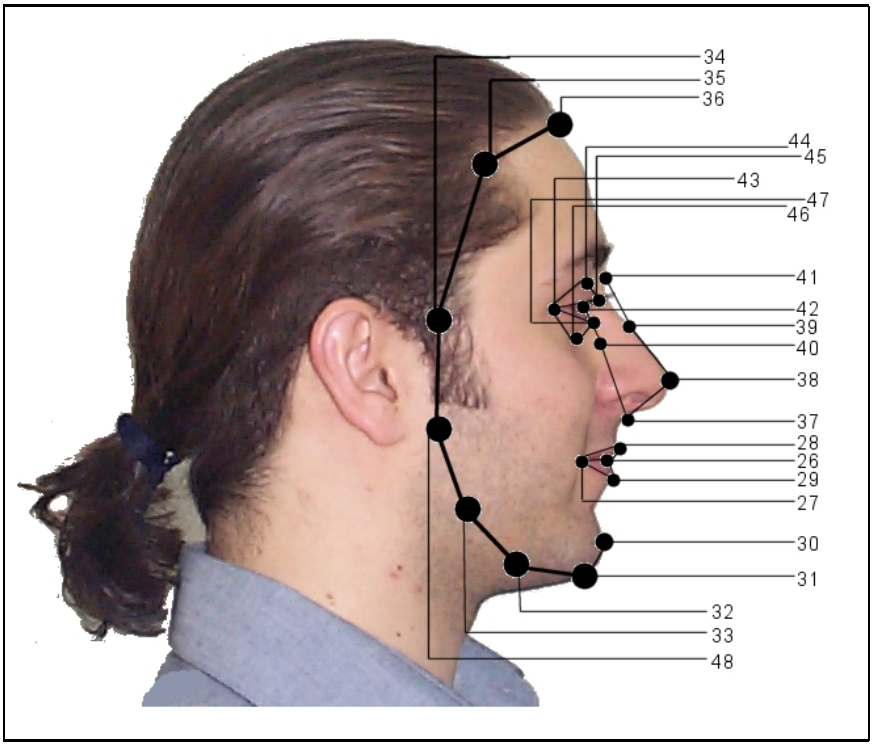
The numbering of facial features extracted from the images is presented in Figure 5.1. In this section, we explain the calculation of the parameters using these feature points identified with numbers on them. Vertices in the left half of the face are calculated and then mirrored according to $x = 0$ to have the other half.

5.2.1 Mouth

The frontal mouth is deformed by the parameters mw, mtx, mty, mtc, mb, mnb(cf. Figures 5.2 and 5.3). mw stands for *mouth width*. This is the horizontal



(a)



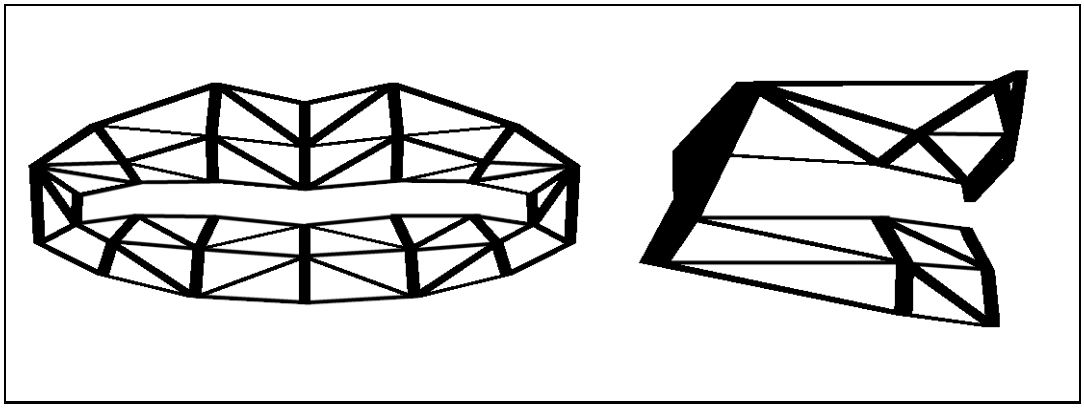
(b)

Figure 5.1: The numbering of the facial features: (a) frontal, and (b) lateral view.

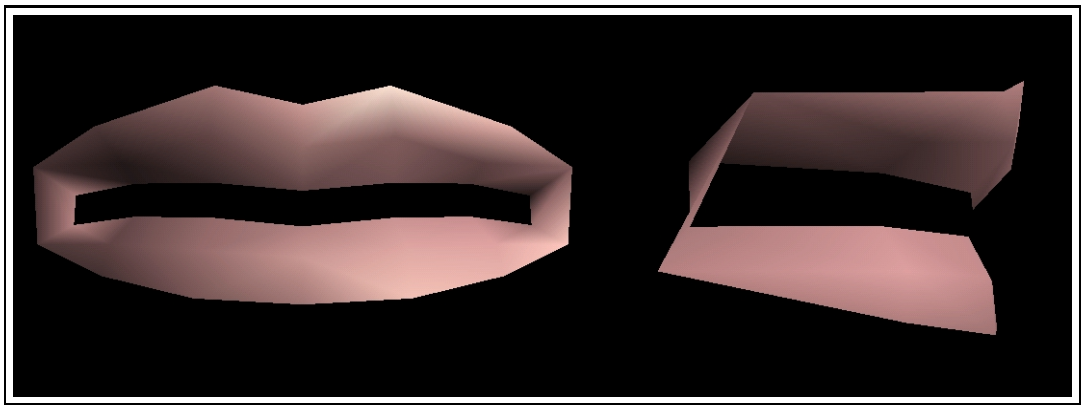
difference between feature points 0 and 2 in terms of the number of pixels. For normalization, we multiply each parameter with $1.572/mw$ to have the mouth width in the face coordinates as 1.572, which is the mouth width in the generic model. We first subtract the upper left half of the vertex coordinates from that of the lip's left corner (vertex 87) in order to translate them according to the left part of the mouth. The x coordinates of the vertices 77, 78 and 79 are scaled according to mw so that x coordinate difference for vertices 77 and 87 is 1.572. mtx is the difference between x coordinates of the vertices 79 and 80, which is the difference between features 3 and 4. Vertices 81 and 82 are placed according to the differences from vertex 80, that is they are translated to new location of vertex 80. Vertices 83, 84 and 85 are interpolated according to the difference between vertices 85 and 87. The mtc parameter is calculated as the vertical difference between features 0 and 4 and it is used to locate the y coordinate for vertex 79. y coordinate of vertex 78 is adjusted with the same scale. mty, the vertical difference between 3 and 4, is used to calculate the new y coordinate for vertex 80 by scaling, and vertex 81 with the same scale. The same scale is used for y coordinates of vertices 83 and 84, too.

Then, the z coordinates are arranged. Pmw is the width in z direction, which is the difference between features 26 and 27. It is the difference in z direction between vertices 87 and 77, and vertices 82 and 85 are scaled in z accordingly. $Pmtz$ is the z difference between the features 26 and 28, corresponding to the vertices 77 and 79. The scale for this is used for z scale of vertices 78 and 79, as well as vertices 80 and 81, and with vertices 83 and 84. After this, deformation of the left upper lip is completed. Therefore, we translate it so that vertex 87 is on (-1.572, 0, 0).

Next, we deform the lower left lip. First, they are translated such that vertex 9 will take place on (0, 0, 0). The mw parameter is used to scale x coordinates for vertices 0, 3 and 6. Then, the vertices 1 and 2 are translated according to vertex 0; vertices 4 and 5 according to vertex 3; and vertices 7 and 8 according to vertex 6. Vertex 10 is translated according to vertex 9. The mb parameter is the vertical difference between features 0 and 1, corresponding to the vertices 0 and 2. This scaling is used for y coordinate calculation of vertices 1 and 2, 4 and



(a)



(b)

Figure 5.2: The frontal and lateral view of the mouth: (a) wireframe, and (b) smooth shaded.

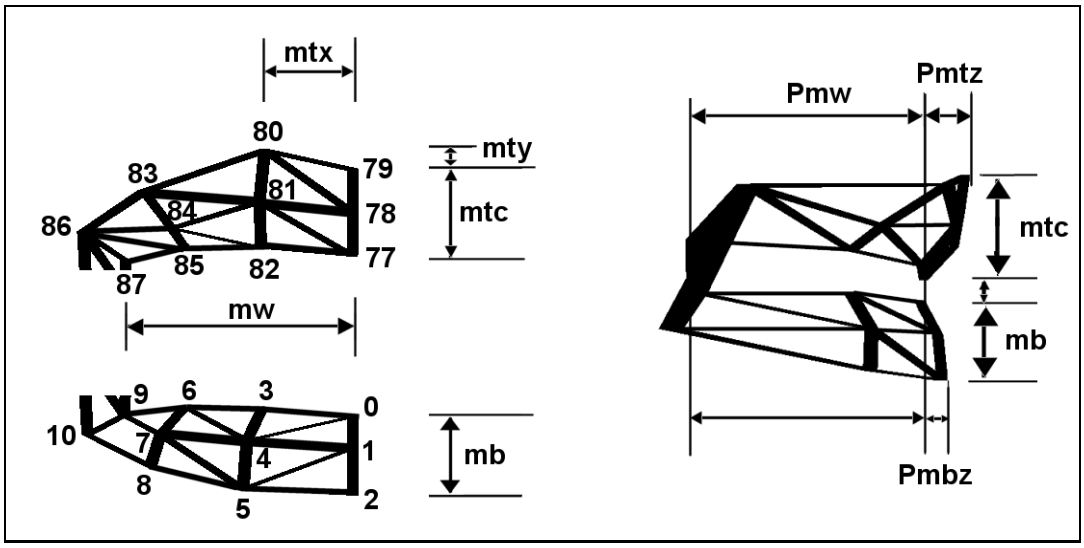


Figure 5.3: Mouth region parameters.

5, 7 and 8.

Then, Pmw is used for scaling z coordinates of vertices 0, 3 and 6. The $Pmbz$ parameter is the difference in z coordinates for vertices 0 and 2, corresponding to the difference of x coordinates in profile image for features 26 and 29. The z coordinates for vertices 1 and 2 are scaled and translated accordingly on the vertex 0. Similar transformation is done for the vertices 4 and 5 on vertex 3, vertices 7 and 8 on vertex 6.

Finally, having deformed the lower left lip, we translate the vertex 9 to match approximately the new location of the vertex 87. Figure 5.4 shows deformed mouth shapes according to different parameters.

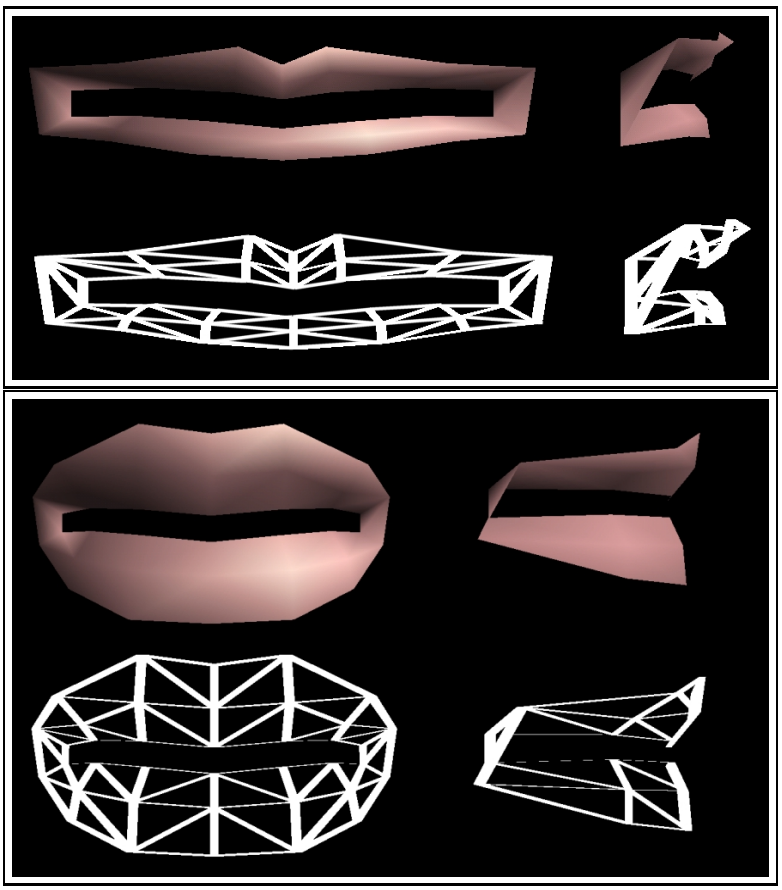
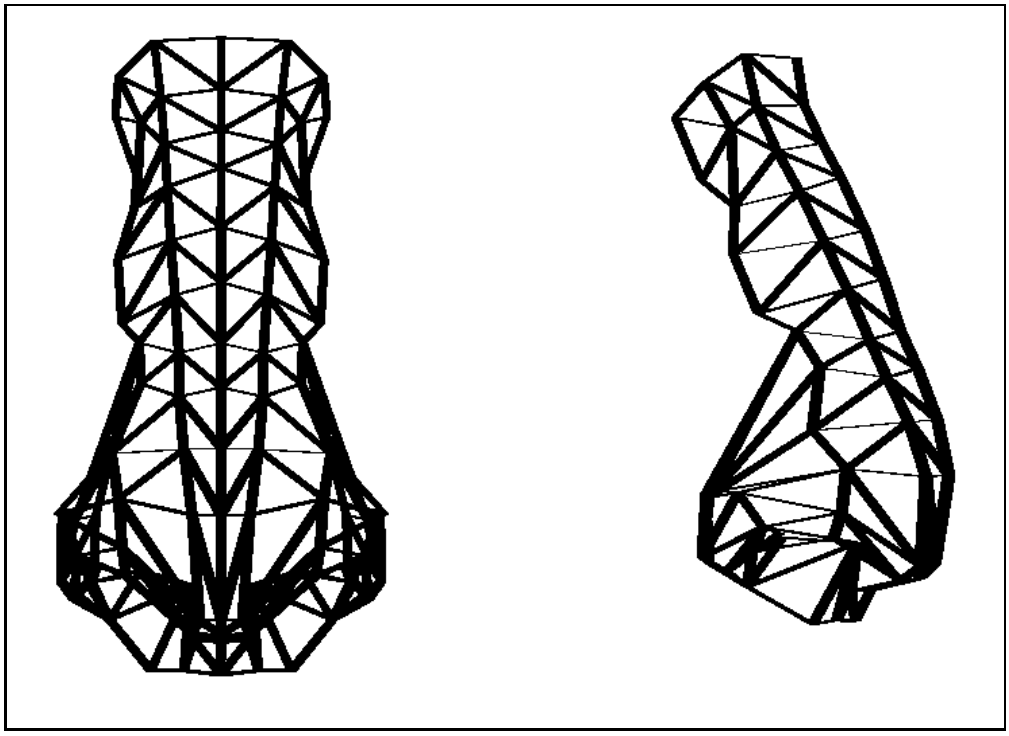


Figure 5.4: Examples of deformed mouth shapes.

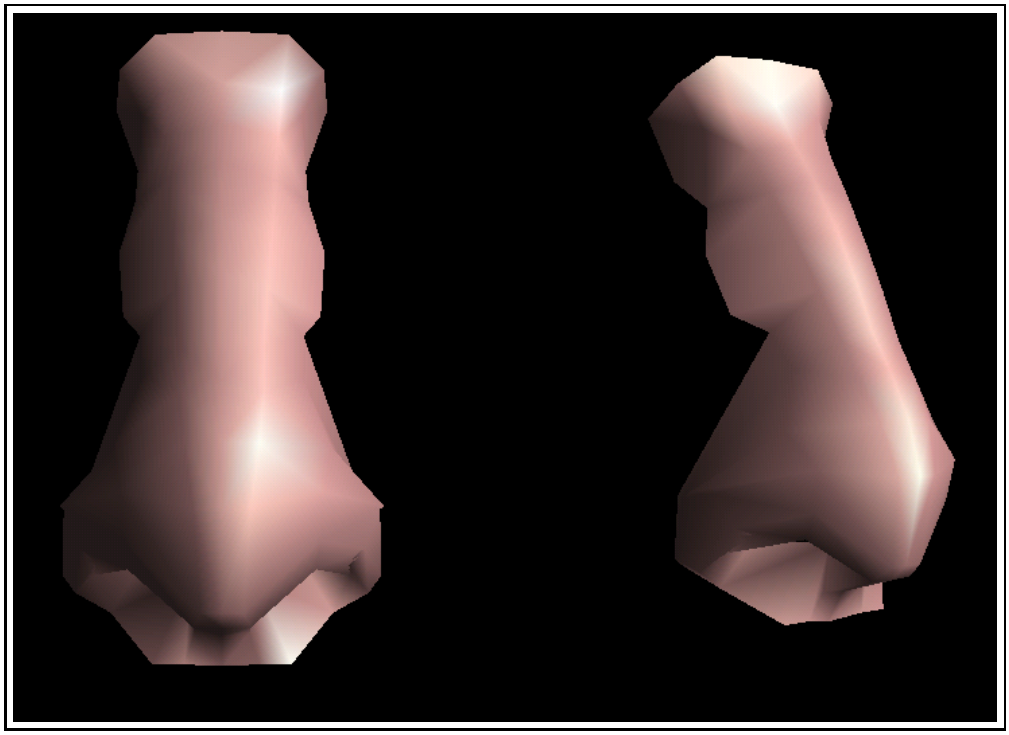
5.2.2 Nose

The parameters for the bottom of the nose are nt, $Pntz$, nh and nsy (cf. Figures 5.5 and 5.6). The bottom of this part is taken as vertex 92, which corresponds to feature 12 on the frontal face. The tip of the nose corresponds to vertex 121 and feature 22. nt is the vertical difference between vertices 92 and 121, which is the number of pixels in y direction for features 12 and 22. The y coordinate of vertex 121 is arranged according to the nt parameter, and z coordinate is arranged according to $Pntz$, which is the difference in z direction between vertices 92 and 121, the x difference in features 37 and 38 of profile face image. The nh parameter decides on the width of the nose hole, setting the x coordinate of vertex 110 according to the feature 13. The y coordinate for vertex 110 is determined by the nsy parameter, as the difference between the y coordinates of vertices 92 and 110, corresponding to features 12 and 13. The z coordinate for vertex 110 is computed by translating it according to vertex 92 since the difference in shape will not elevate it in z direction. Vertex 92 and 121 has 0 as the x coordinate because our model is assumed to be symmetric. After vertices 92, 121 and 110 are located, the remaining vertices in the nose bottom are interpolated according to these three vertices with the required scaling, translation and rotation transformations. Finally, we translate all to the new place of vertex 92, which has x coordinate as 0, y coordinate computed according to mnb as the vertical difference between vertex 79 and vertex 92 corresponding to features 4 and 12, and the z coordinate computed according to Pnz as z difference of vertices 79 and 92 corresponding to the horizontal difference in the profile image features 28 and 37.

The middle nose is deformed according to the parameters nmc, $nmsx$, $Pncz$, $Pncw$ (cf. Figure 5.6). The vertex 135 is relocated with x coordinate being 0, y coordinate being scaled according to nmc, and z being scaled according to $Pncz$. nmc is the vertical difference between vertices 121 and 135, corresponding to the features 22 and 23. $Pncz$ is the z coordinate difference for the same two vertices, corresponding to the horizontal difference in the features 38 and 39 of the profile image. Vertex 137's x coordinate is adjusted according to parameter $nmsx$, as the difference between x coordinates of the vertices 137 and 135, corresponding



(a)



(b)

Figure 5.5: The frontal and lateral view of the nose: (a) wireframe, and (b) smooth shaded.

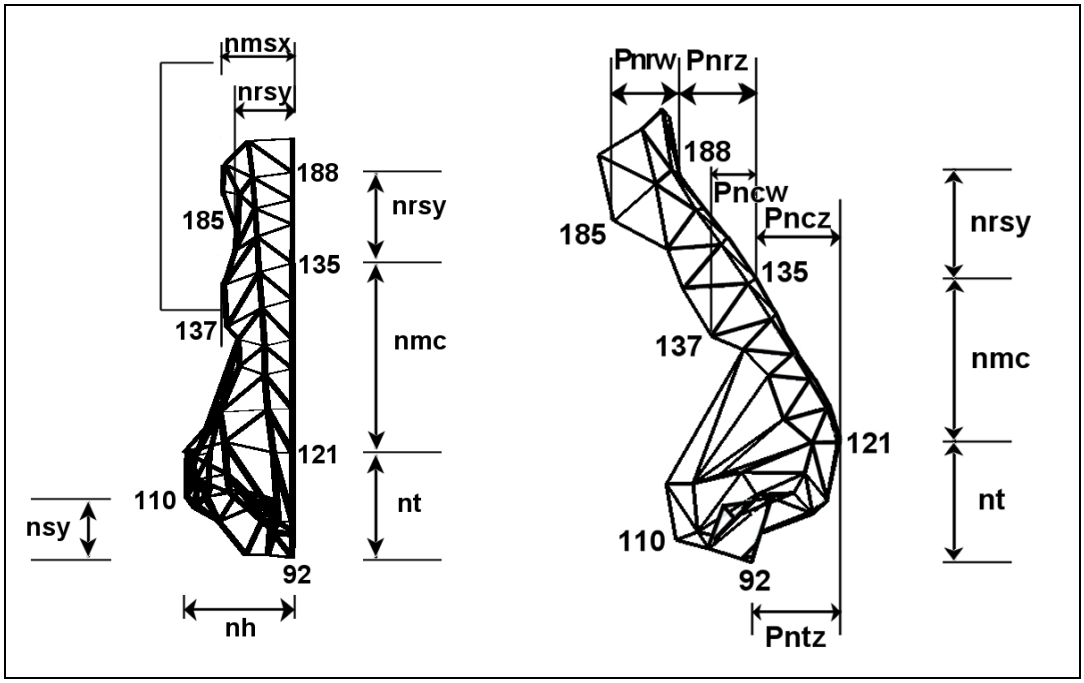


Figure 5.6: Nose region parameters.

to the horizontal difference of the features 14 and 23; and the y coordinate is found by scaling according to the middle nose height and translating according to vertex 135; the z coordinate is located according to $Pncw$ parameter, which is z difference for the vertices 135 and 137 corresponding to x difference of features 39 and 40 in the profile image. Once the vertices 135 and 137 are located, the points are translated according to vertex 121, and then the in-between vertices are located using transformations.

The parameters for deforming the upper side of the nose are $nrsx$, $nrsy$, $Pnrz$ and $Pnrw$ (cf. Figure 5.6). Vertex 188 is the top point for the upper part of the nose. Its x coordinate is 0, y coordinate is found by $nrsy$ parameter, which is the vertical difference between the vertices 135 and 188, corresponding to the features 23 and 24. The z coordinate of vertex 188 is computed according to $Pnrz$ and vertex 135. $Pnrz$ is the difference in x in the features 39 and 41 of the profile image. Vertex 185's x coordinate is calculated according to $nrsx$ parameter with respect to vertex 188. This is the horizontal difference between the features 15 and 24. The z coordinate of vertex 185 is found by $Pnrw$ parameter and the new location of vertex 188. The $Pnrw$ parameter is the horizontal difference between

profile image's features 41 and 42. Once the vertices 188 and 185 are relocated, the remaining vertices' locations are calculated by transformations according to the vertices 135, 137, 185 and 188. Figure 5.7 shows deformed nose shapes according to different parameters.

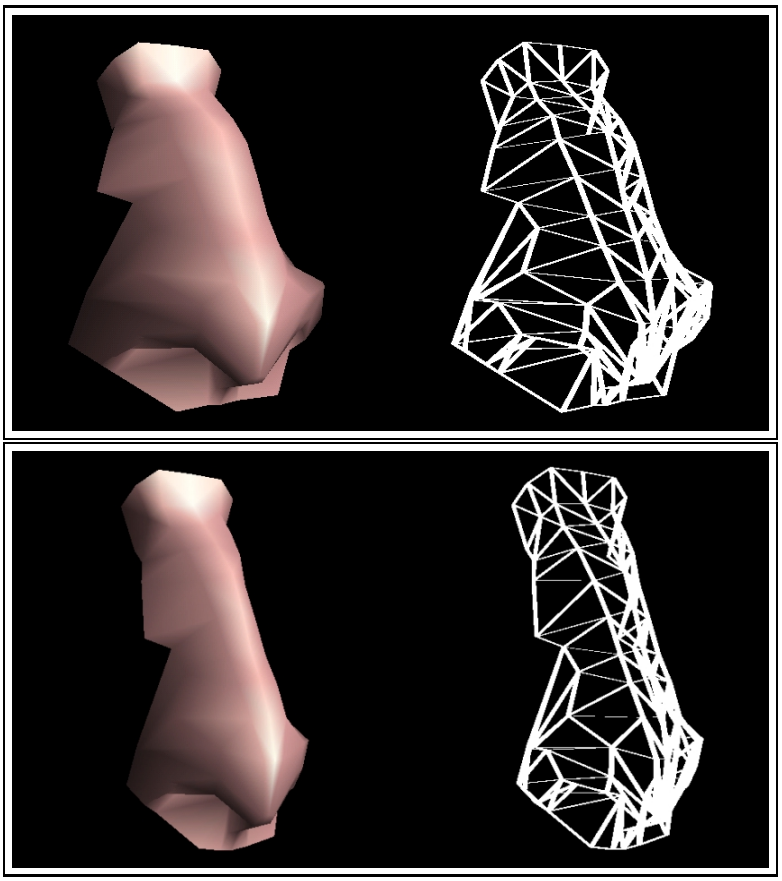
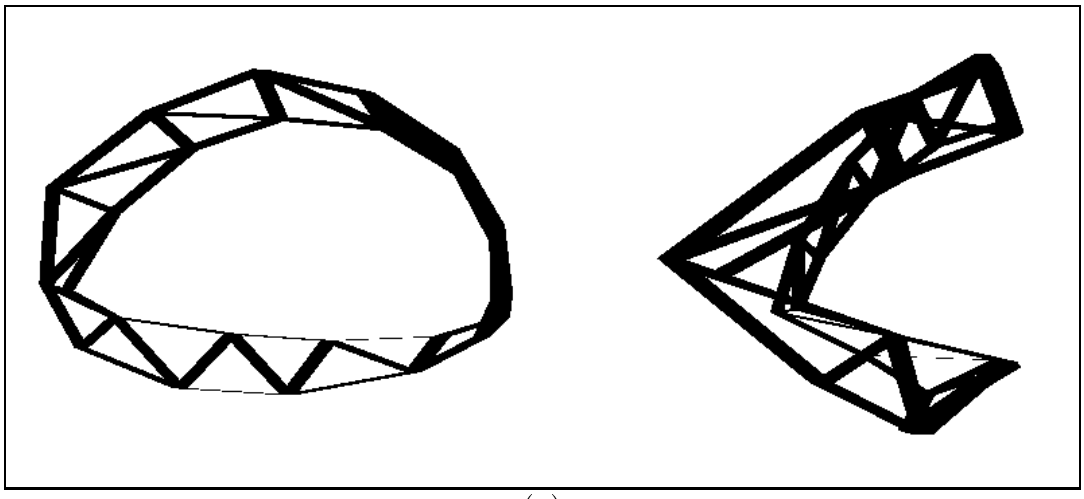


Figure 5.7: Examples of deformed nose shapes.

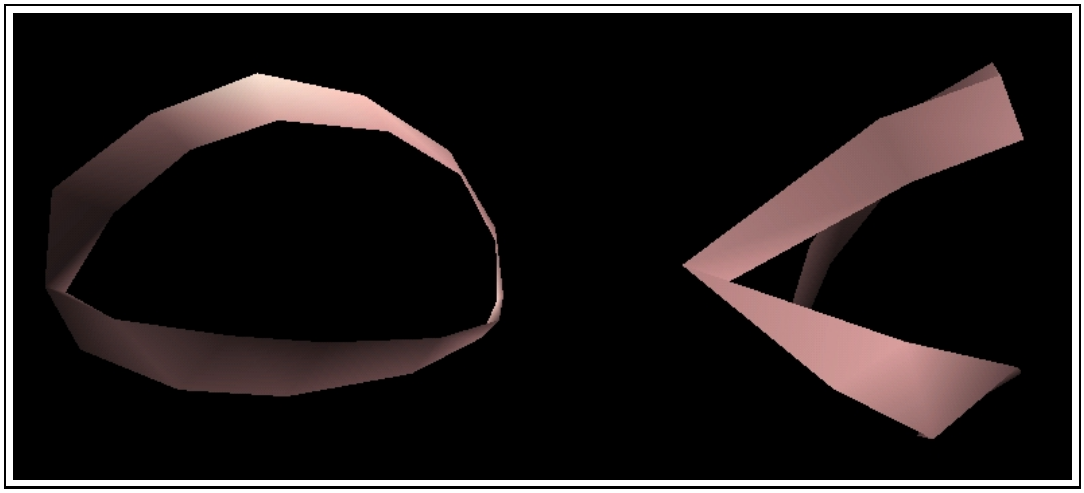
5.2.3 Eyelids

The parameters related to eyelids are eye_angle, enx, eny, Pez, ew, $eltw$, $elbw$, ety, eby, $Petw$, $Pebw$, $Peltw$, $Pelbw$ (cf. Figures 5.8 and 5.9). First, all vertices of the eyelids are translated so that vertex 178 is at $(0, 0, 0)$. Vertex 196 is relocated in y and z using the parameters ety and $Petw$. ety is the vertical difference of the top of the top boundary of eyelid with left corner of the eyelid, the features 20 and 17, respectively. $Petw$ is the z difference of the vertices 178 and 196,

corresponding to the x difference of the features 43 and 44. The x coordinate of vertex 196 remains the same. When we locate the vertex 196, we rotate the upper boundary of upper eyelid according to new position of this vertex.



(a)



(b)

Figure 5.8: The frontal and lateral view of the eyelids: (a) wireframe, and (b) smooth shaded.

eltw is vertical difference of features 20 and 21, used to calculate the new y coordinate of vertex 247 using the new y coordinate of vertex 196. *Peltw*, being the horizontal difference between the features 43 and 45 in the profile image, is for calculation of the new z coordinate of vertex 247 according to vertex 178. In this way, we locate vertex 247 and we rotate the other vertices on the bottom boundary of the upper eyelid.

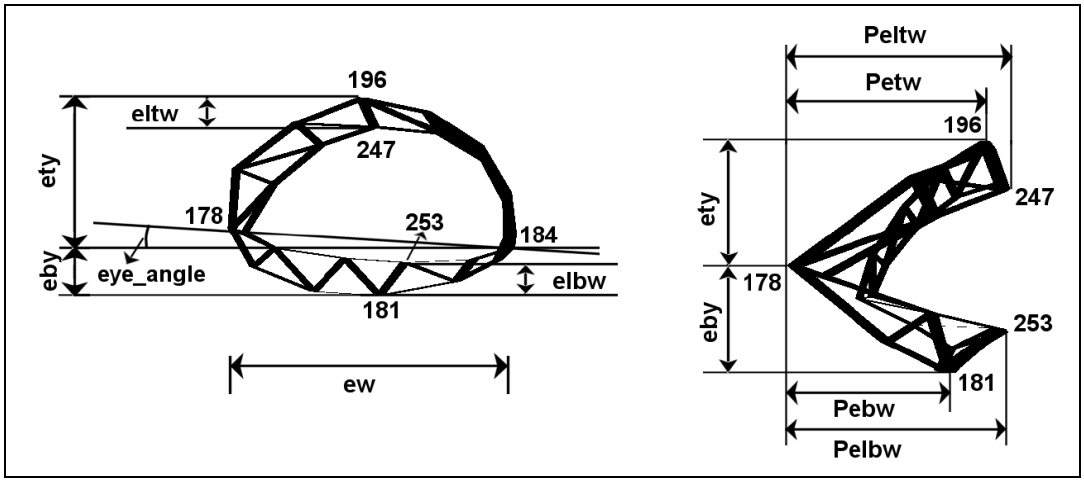


Figure 5.9: Eyelid parameters.

In the lower eyelids, vertex 181 is relocated while the x coordinate remains the same, the y coordinate is found by eby according to vertex 178, and the z coordinate is found by $Pebw$ according to vertex 178. eby is the vertical difference between the features 17 and 18. $Pebw$ is the horizontal difference between the profile features 43 and 46. Rotation of the remaining vertices on the bottom boundary of the lower eyelid transforms them to their new positions.

Vertex 253's new x coordinate is the same as before; y coordinate is computed by the parameter $elbw$, which is the vertical difference between the features 18 and 19, according to vertex 181. The z coordinate of vertex 253 is computed with respect to the z coordinate of vertex 178, using the parameter $Pelbw$ corresponding to the horizontal difference between the features 43 and 47 in the profile image. The other vertices on the upper boundary of the lower eyelid are rotated to make vertex 253 pass from its new location.

With the y and z coordinates of the eyelid vertices determined, we now rotate the eyelid around the vertex 178 according to the eye_angle value. Then, we scale the x coordinates according to the ew parameter, which is horizontal difference between the features 17 and 18, corresponding to the vertices 178 and 184. The enx parameter is the horizontal difference of features 24 and 16, eny is the vertical difference between them and Pez is the horizontal difference between the features 41 and 43. We translate the vertices according to the new location for vertex 178.

The left side of the face at the eye level is adjusted according to the few parameter, which is the horizontal difference between the features 9 and 17. The x coordinate of vertex 171 and vertices between 171 and 178 are found accordingly. The z coordinates are computed similarly according to the horizontal difference of features 34 and 43.

The upper part of the eye region corresponding to eyebrows is located by translation according to the located upper boundary of the upper eyelid. Although we have parameters by and bz to scale the y and z coordinates for this part, extracting these features from photographs is difficult, so we do not change these parameters while deforming. Figure 5.10 shows deformed eyelid shapes according to different parameters.

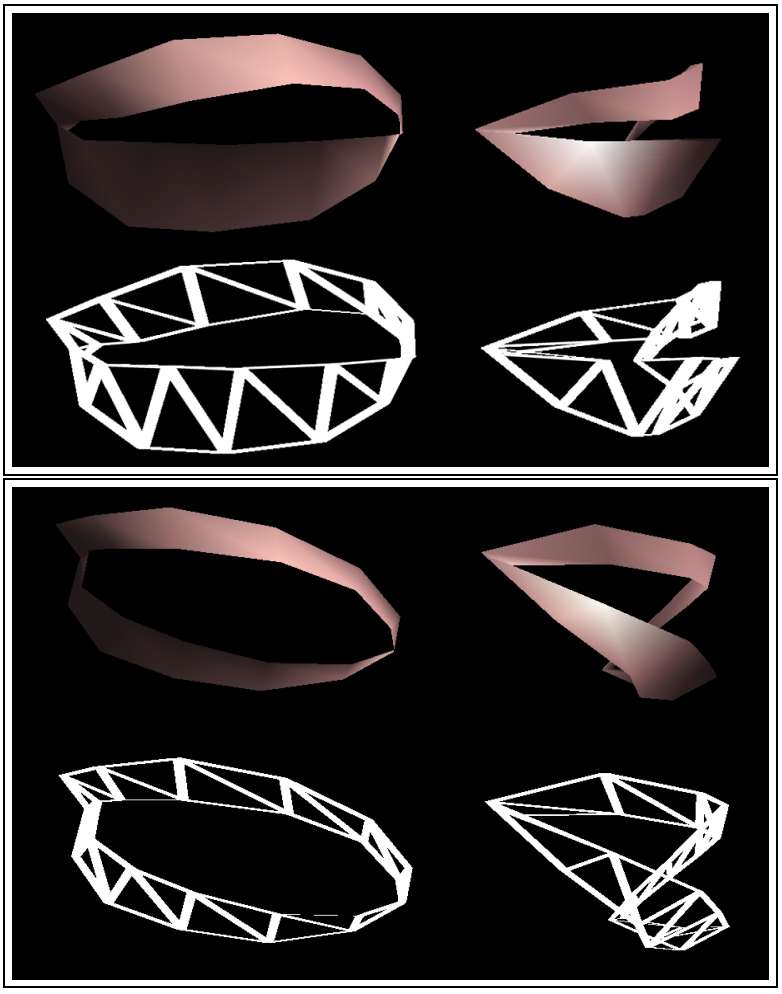
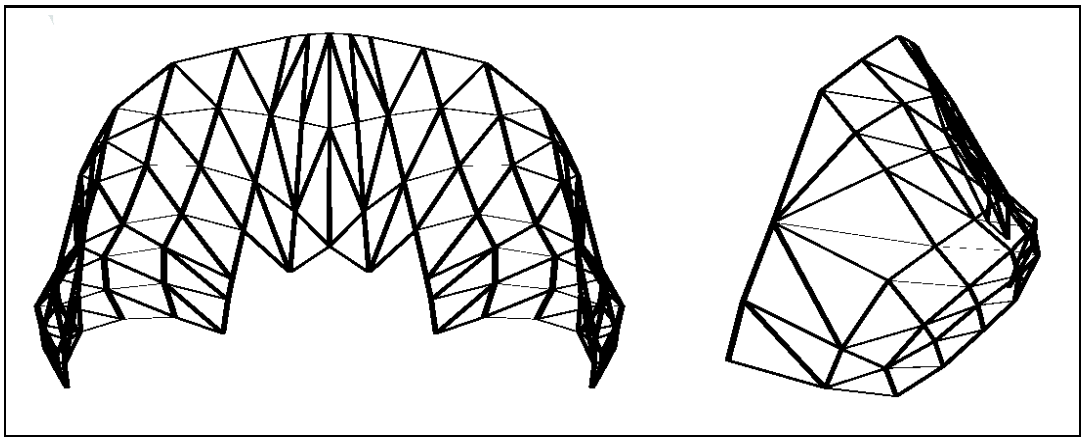


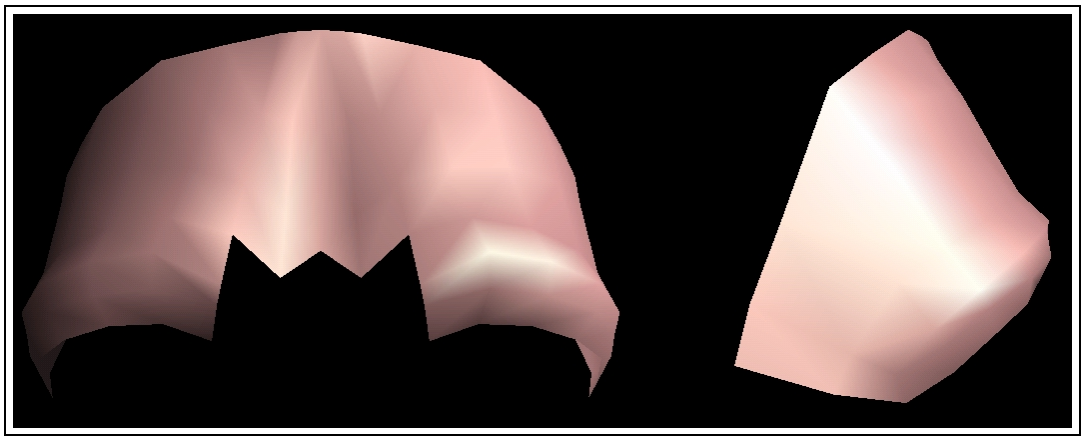
Figure 5.10: Examples of deformed eyelid shapes.

5.2.4 Forehead

The forehead is deformed using the ftx, fty, ftz, fhw and fhd parameters (cf. Figures 5.11 and 5.12). We deform the upper boundary of the forehead between vertices 236 and 242. Features 10 and 35 correspond to vertex 236, and features 11 and 36 correspond to vertex 242. The vertices in this boundary are translated to set the vertex 242 to (0, 0, 0). Then, the vertex 236 is located according to parameters ftx, fty and ftz. ftx is horizontal difference between features 10 and 11, fty is the vertical distance between features 10 and 11, and ftz is the horizontal distance between features 35 and 36. When vertex 236 is located with this new position, the remaining vertices are scaled accordingly.



(a)



(b)

Figure 5.11: The frontal and lateral view of the forehead: (a) wireframe, and (b) smooth shaded.

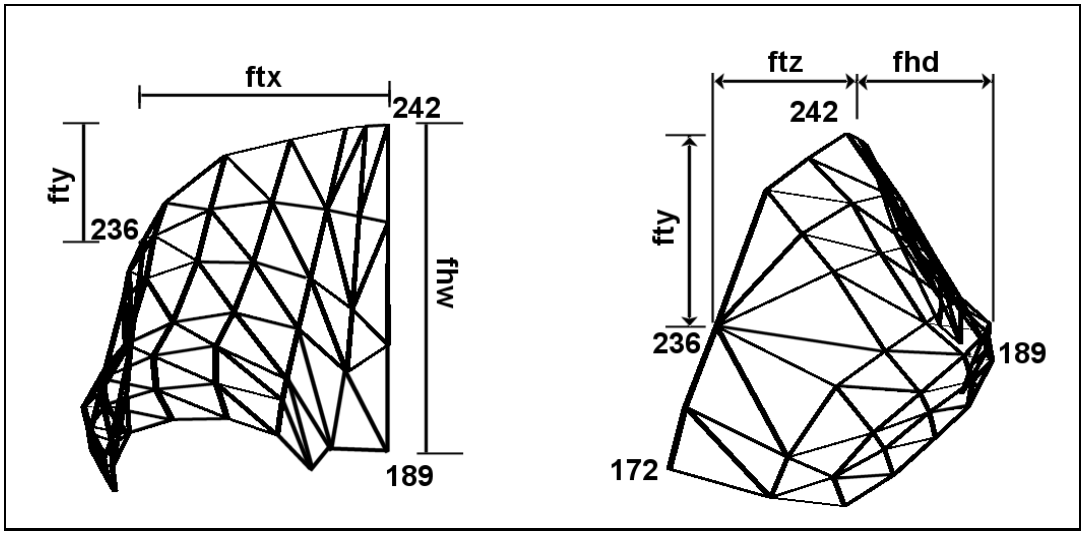


Figure 5.12: Forehead region parameters.

Then, vertex 242 is located, and the rest of that boundary vertices are translated accordingly. The new vertex 242 location has x coordinate as 0, y coordinate computed according to fhw and vertex 188, z coordinate computed according to fhd and vertex 188. fhw is the vertical difference between features 24 and 11, and fhd is the horizontal difference between features 41 and 36. When the upper boundary vertices are deformed and translated according to new place of vertex 242, the inbetween vertices are relocated with interpolation using scaling, translation and rotation. Figure 5.13 shows deformed forehead shapes according to different parameters.

5.2.5 Chin

First of all, the vertices in the chin are translated as if vertex 2 is on (0, 0, 0), but the vertices 2, 5, 8 and 10 are not modified since their locations are calculated while deforming the mouth. The vertex 2 is used as a reference point for transformation of other vertices.

Vertex 24 has x coordinate 0, y coordinate is computed using $mbcc$ with respect to vertex 2, and z is computed using $Pccz$ with respect to vertex 9. $mbcc$ is the vertical difference of features 1 and 5. $Pccz$ is the horizontal difference of

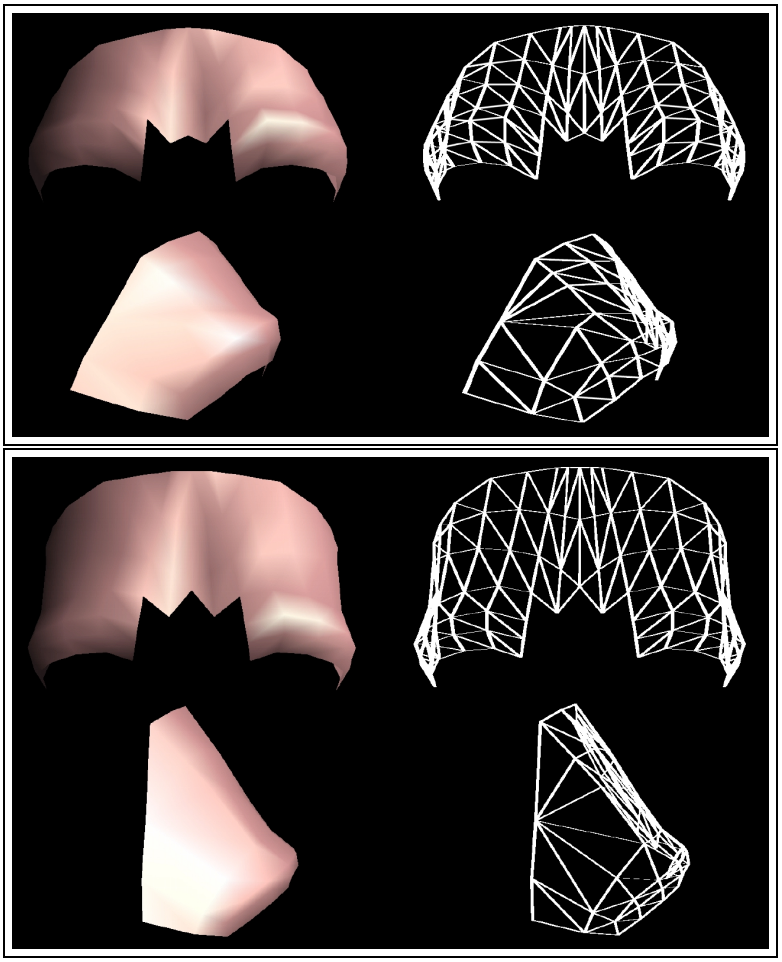
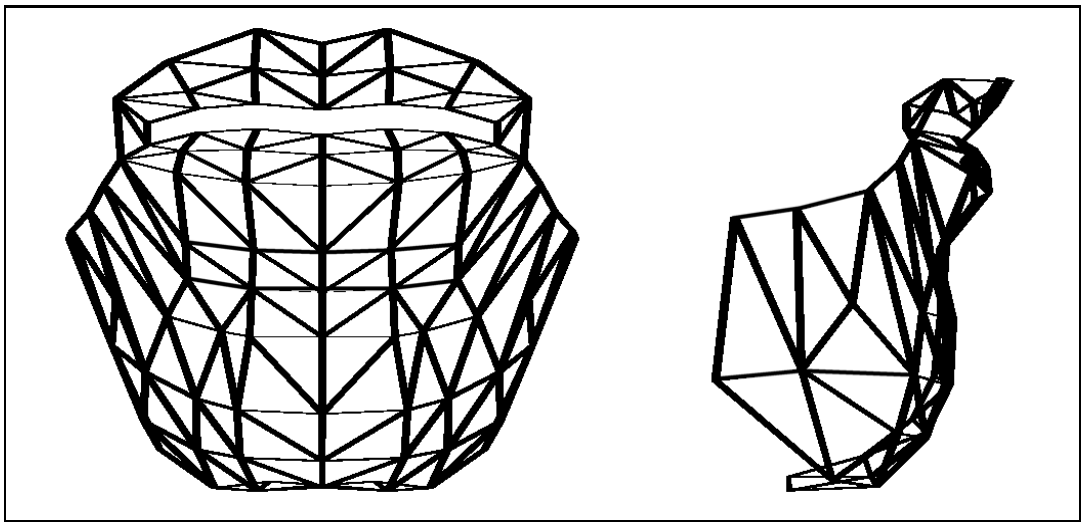


Figure 5.13: Examples of deformed forehead shapes.

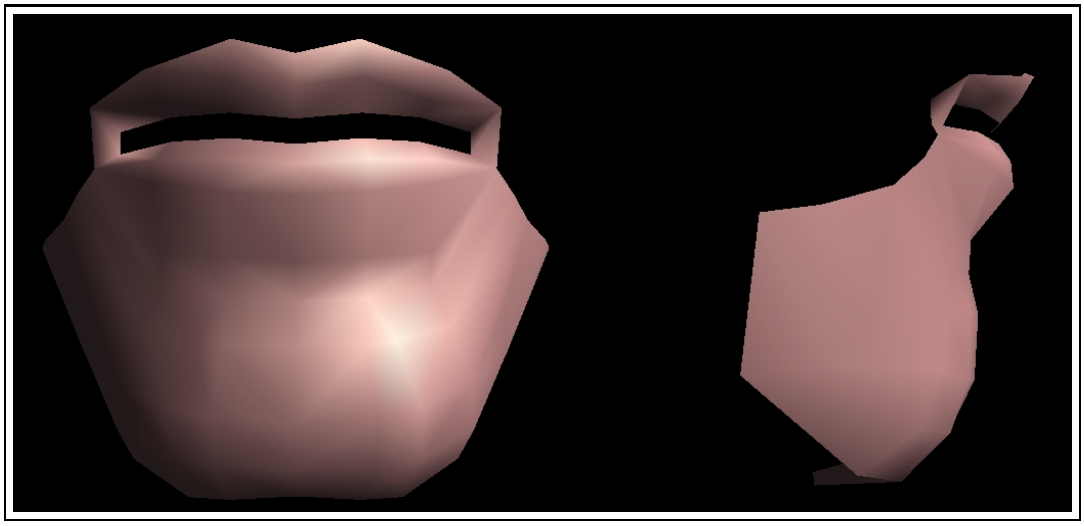
features 27 and 30. Similarly, vertex 50 has x coordinate as 0, y coordinate is calculated using $mbcb$ parameter with respect to vertex 2, and the z coordinate is calculated using $Pcbz$ with respect to vertex 9. $mbcb$ is vertical difference of features 1 and 6, $Pcbz$ is the horizontal distance between features 27 and 31 (cf. Figures 5.14 and 5.15).

Vertex 33 is calculated using the parameters $c2x$, $c2y$, and $Pc2z$, according to the vertices 2, 50 and 9, respectively. $c2x$ is the difference between features 1 and 8 in horizontal direction, $c2y$ is the vertical distance between features 6 and 8, $Pc2z$ is the horizontal distance between features 27 and 33. Similar to vertex 33, the coordinates of the vertex 42 is calculated using the parameters $c1x$, $c1y$ and $Pc1z$. Vertices 2, 50 and 9 are used in these calculations, respectively.

$c1x$ is the difference between features 1 and 7 in horizontal direction, $c1y$ is the vertical distance between features 6 and 7, $Pc1z$ is the horizontal distance between features 27 and 32.



(a)



(b)

Figure 5.14: The frontal and lateral view of the chin: (a) wireframe, and (b) smooth shaded.

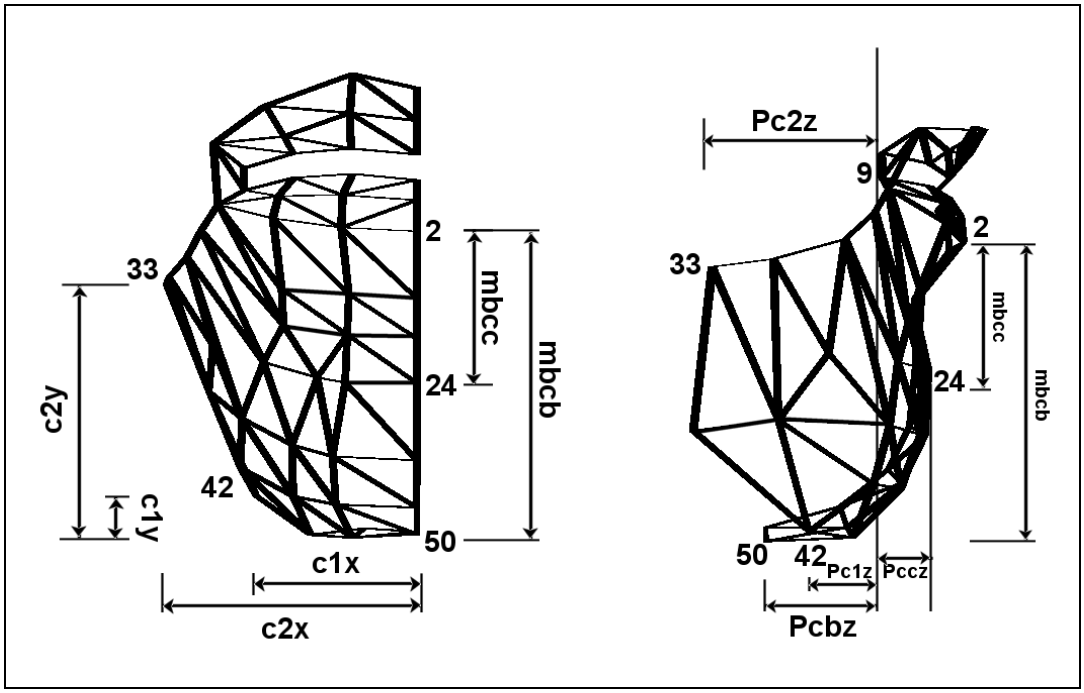


Figure 5.15: Chin region parameters.

Once we have located the vertices 24, 50, 33 and 42, which are the control points in the chin region, the rest of the chin vertices are interpolated accordingly with respect to the region they take place among the control points. The rest of the vertices at the level of vertex 33 and below, that are not on the chin, which can be named as throat region, are translated according to chin vertices, and scaled when required. Figure 5.16 shows deformed chin shapes according to different parameters.

5.2.6 Cheeks

The parameters kx, ky and kz, as well as the new locations of the rest of the face define the new shape for the cheek (cf. Figures 5.17 and 5.18). The parameters kx, ky and kz are extracted from the images using the features 25 and 48. x and y difference is taken according to feature 12, and the z difference according to feature 37. A fraction of these values are used to define the new location for the vertex 142 with respect to the vertex 92. Having located vertex 142 to define

shape of the cheek, we fill the gaps remaining using translation and interpolations using the control points on other regions like the eyes, chin and nose. Figure 5.19 shows deformed cheek shapes according to different parameters.

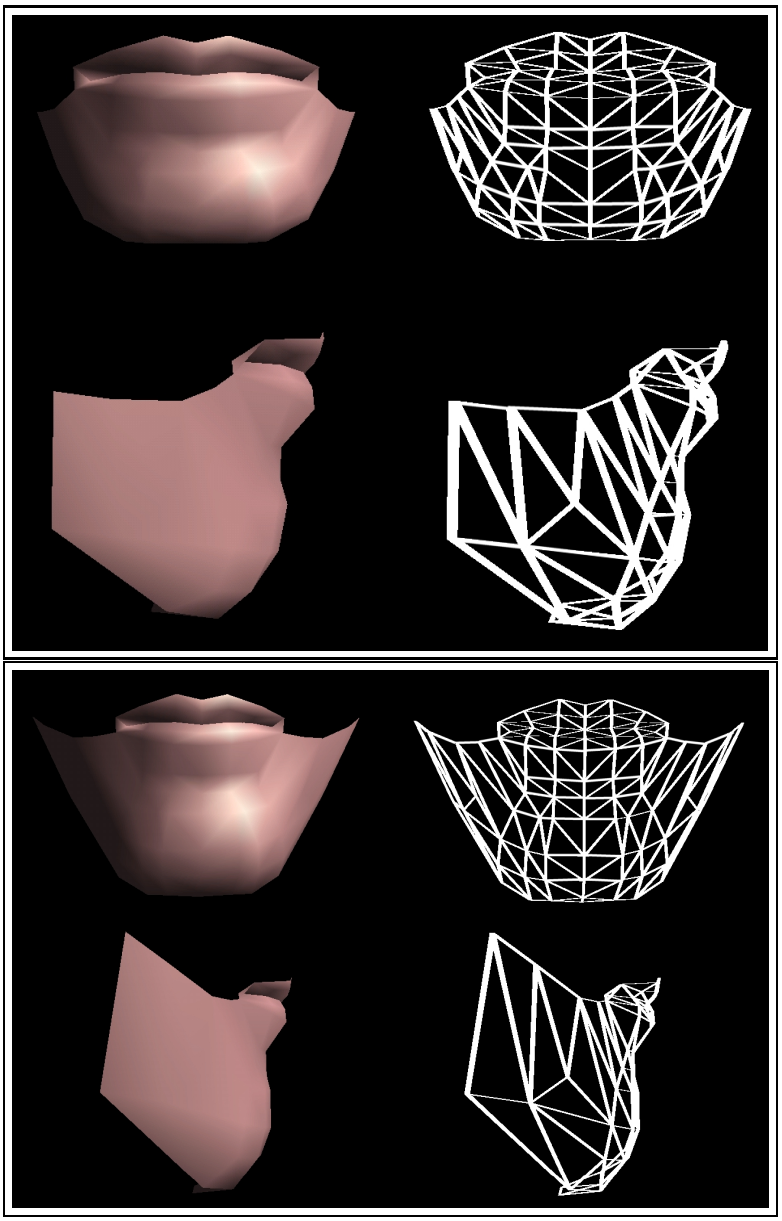
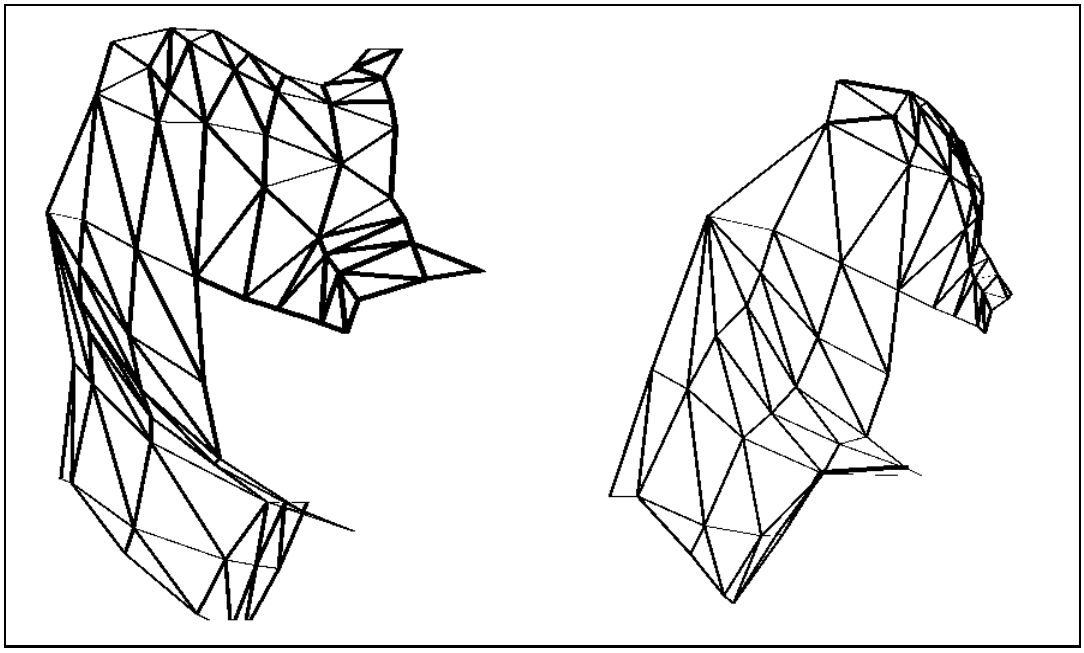
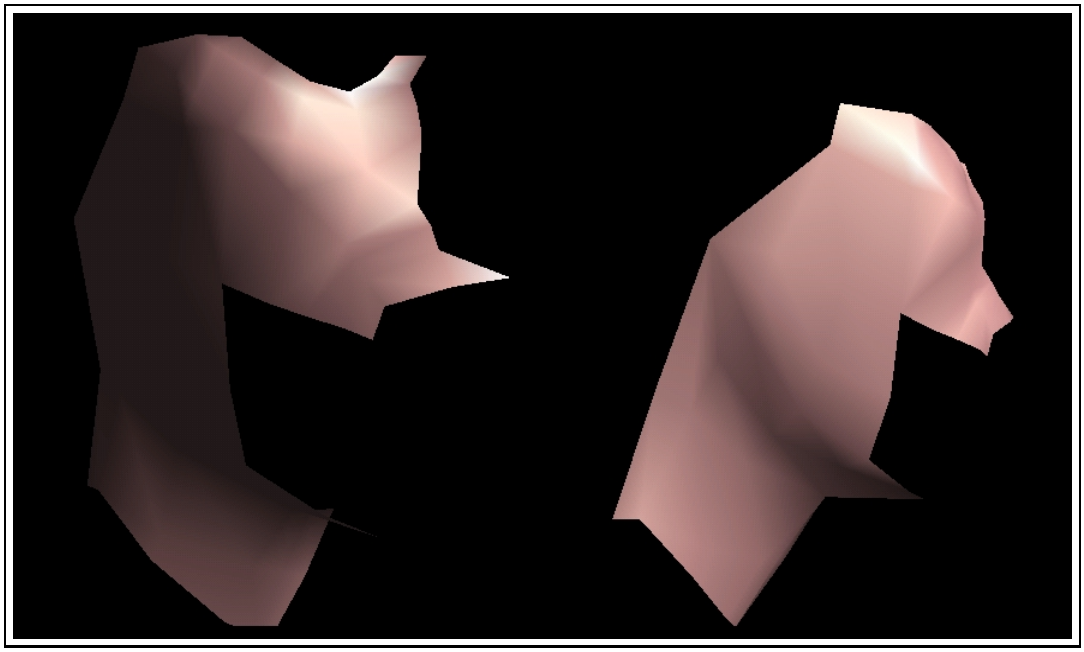


Figure 5.16: Examples of deformed chin shapes.



(a)



(b)

Figure 5.17: The frontal and lateral view of the cheek: (a) wireframe, and (b) smooth shaded.

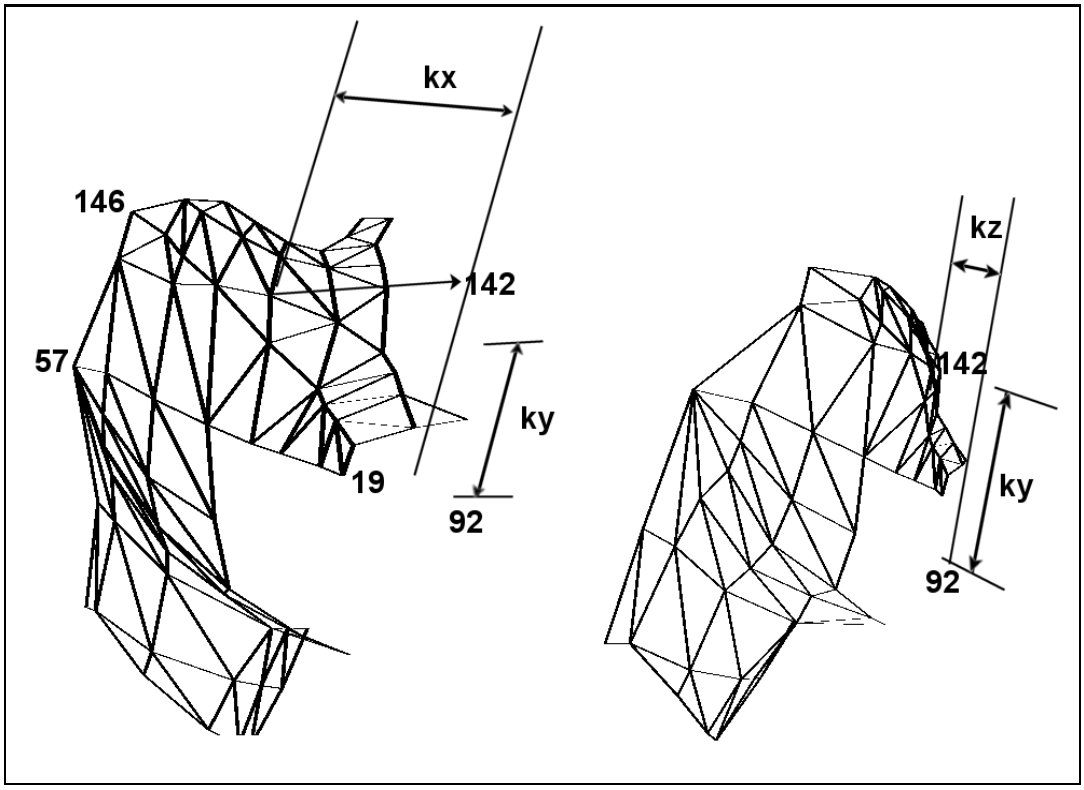


Figure 5.18: Cheek region parameters.

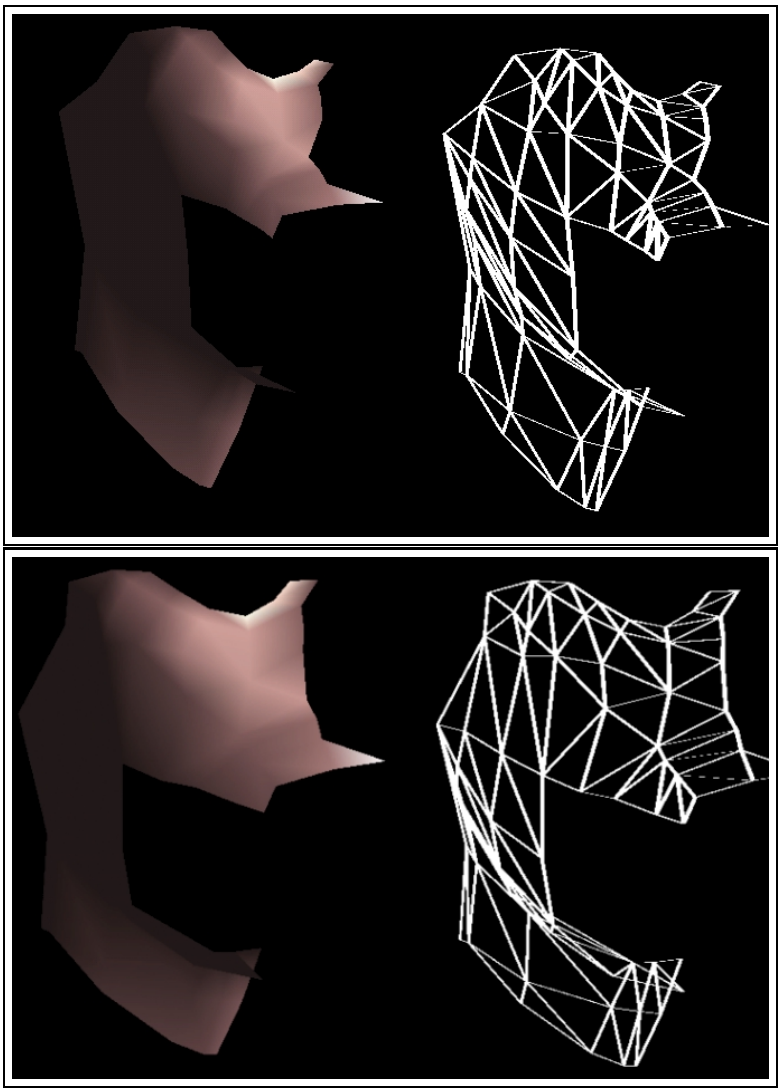


Figure 5.19: Examples of deformed cheek shapes.

5.2.7 Muscles

The muscles of the face are input to the system as part of the generic model from a file. Then, we initialize them to locate their positions in terms of the vertices they take place between. Once we calibrate the face model, we calculate new positions for the muscles by interpolating them according to the vertices they take place between.

As it is seen, the parameterized face model we used is good enough to represent most facial geometry. The deformation process gives good results as long as the parameters are not too abnormal. Figure 5.20 shows examples of faces with muscles obtained by deforming the generic face model.

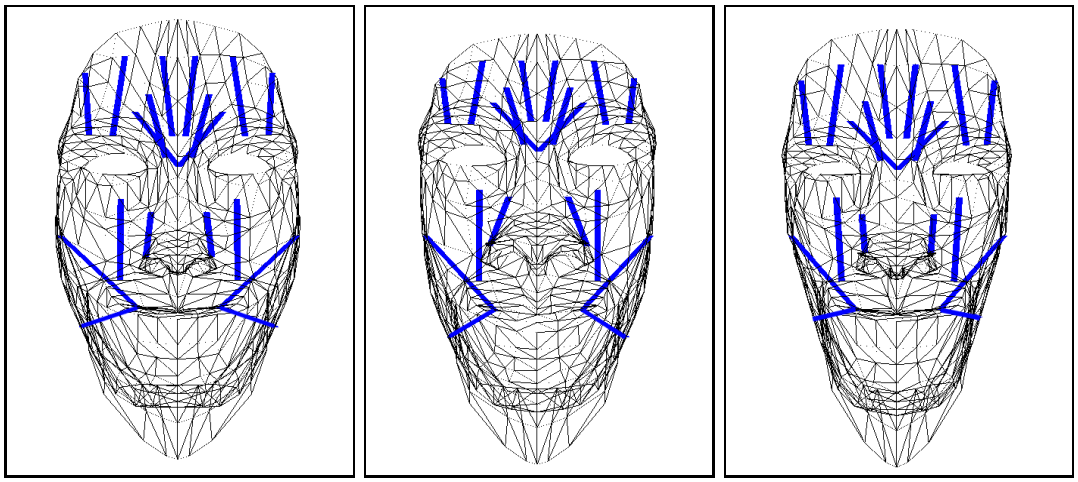


Figure 5.20: Examples of deformed faces with muscle vectors.

Chapter 6

Rendering and Animation

6.1 Rendering

Rendering is one of the most important issues for implementing a realistic facial animation system. Since believability and reality of the output is determined by the final output image synthesized to represent the face, it should be rendered correctly to serve its aim. However, no matter how detailed the model is, there can be flaws that will give away the truth that it is a synthesized image, as the brain is very good at keeping and recognizing images. In fact, we can say that no such study has passed the Turing test for facial animation, so that any watcher of the photograph and synthesized image of a face could tell which is real and which is synthesized.

Current graphics processors are highly developed; a graphics adapter produced for a PC can render up to about 20 million triangles per second. Therefore, it is not very unlikely to have facial animation models playing at 60 frames per second with detailed enough models. There can be problem in speed if only the CPU needs to be extensively used for producing each scene.

The facial animation system we developed uses OpenGL [50] to render the facial model produced, and texture mapped model could be played with about

100 fps on a Pentium III-500 MHz CPU with 256 MB RAM and 3Dfx Voodoo3 2000 graphics processor with 8 MB graphic memory. The frame rate could even be increased with modifications to the code to optimize operations done for each frame produced. A greater fps can be reached by first producing screens and storing them in the memory, and then playing them on the screen. Since the model we used had only 878 polygons to render, and could produce 100 fps, we did not need to further optimize the code for better performance.

The model produced after deformation of the generic face according to the parameters obtained from the user interface can be displayed either in wireframe mode, smooth shaded mode, or texture mapped mode. The wireframe mode renders the triangle edges with lighting enabled. The lighting environment as well as the color for the facial surface material is defined with some variables defined in a text file. The smooth shaded mode uses OpenGL smooth shading to create the output. When the model is initially produced, we calculate the normals for each vertex. At first, the normal for each triangular polygon is calculated using the three vertices for that triangle. If the vertices of the triangle are A, B, and C each being a 3D vector, the normal of that triangle is calculated as the cross product $|B - A| \times |B - C|$.

Using this normal to render each triangle will produce a flat shaded output, where the triangles will be easily seen. Therefore, the normal for each vertex is calculated as an average normal of the surrounding polygons. When the triangles and vertices were being input to the system from the file, we also keep a list with each vertex that will have the index for the triangle that vertex is a part of. Then, the normal for a vertex is the mathematical average of the normals of the polygons in the polygon list of that vertex.

When each vertex's normal is calculated, rendering the face model smoothly is by just giving the normal for the vertex as well as that vertex coordinate to the OpenGL system. In the smooth shaded mode, OpenGL uses Gouraud shading model [28] to render the points inside the polygon. Gouraud shading takes the normals for each vertex and calculates the color value for each of them, then it interpolates the color values of other points in the polygon linearly. According to

Gouraud shading, the color for a vertex is

$$vertex\ color = emission + ambient_{global} + \sum lightsources, \tag{6.1}$$

where $emission$ is the light coming from self-illuminating objects, $ambient_{global}$ is the global ambient component which is computed as multiplication of a global ambient light term with the ambient material property at the vertex, and $\sum lightsources$ is the contribution of the light sources in the environment as the sum of the light coming from each of them. This calculation is done for each primary color, namely red, green and blue.

With the texture mode, each vertex is assigned a texture coordinate to be mapped while rendering. When the three vertices of the triangle are given texture coordinates, the color of the polygon points are determined by the mapped place of those points in the texture image. The texture coordinates for the vertices of our model are simply determined by the x and y coordinates of the vertices. First, we calculate the minimum and maximum x and y values among all the vertices in the model. Then, the texture coordinate for a vertex becomes $(tex_x, tex_y) = ((vertex_x - min_x)/(max_x - min_x), (vertex_y - min_y)/(max_y - min_y))$. At this point, the texture image needs to be considered. The image should be such that when the texture coordinates are mapped to it, the mapped place should be the right place of the face and be detailed enough to have a good look.

In fact, the texture image we used in our system is not very suitable. It is produced from the frontal face image by taking the part between the facial feature points. The result will be true in this case, when the output is viewed as a frontal face. However, the polygons on the lateral sides has z coordinates which increases the area for the polygon. Therefore, the area mapped on the image will be much smaller than the polygon's real area, causing the texture to be mapped on it to have a low resolution. A study producing almost right texture maps will be explained in the next section, which is left as a further development to implement. Indeed, the function preparing the texture image implements a part of this method, but the output will not look right as it will require further image

processing and projection to be applied to the pictures, so it is turned off in the code. Without processing the lateral image, the union of it with the frontal image will not look right since the lighting conditions will differ much causing a part of the skin to shine while other part has different lighting. Therefore, we just use the frontal image for texture, and leave the rest as a further development. Figure 6.1 displays an example output model in the three modes.

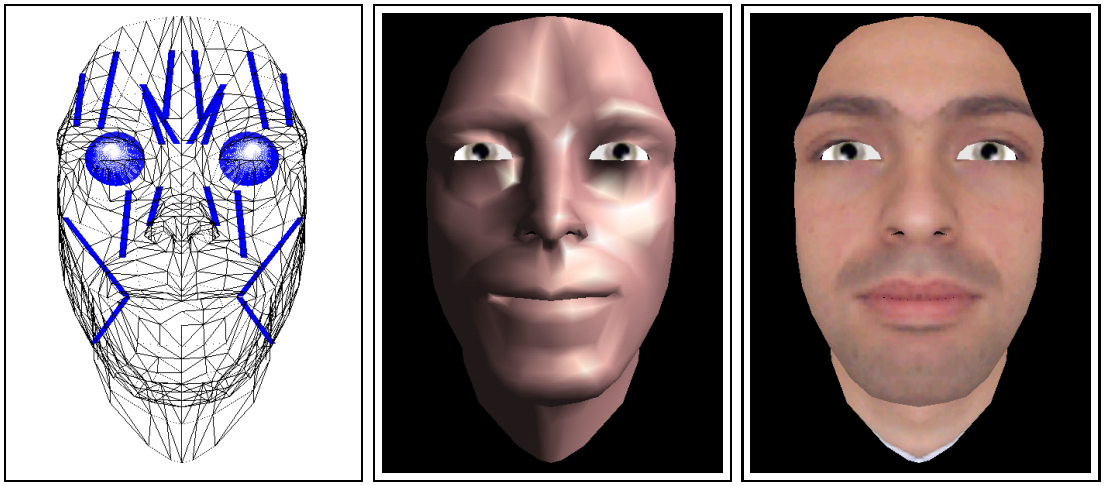


Figure 6.1: A face rendered with three modes: (a) wireframe, (b) Gouraud shaded, and (c) textured.

6.1.1 Face Texture Generation

To produce a more realistic texture image, [38] implements a method that will be described in this section. A good texture image will have greater resolution for the mapped polygons. Using laser range scanners will provide with the best texture data, however the model we implemented tries to build the texture from two orthogonal photographs of the face. Front view of the face is kept as it is in order to have the highest resolution for the front side of the face. We need to use the side view image for the part that is to the left of the eyes and mouth for higher resolution in these parts. We then unite these two parts and take the mirror to complete the other part of the face. Building texture for left side of the face, then reflecting this will be good when there are defects with the texture of a

nonsymmetrical face, however this will ignore the different facial texture that may occur on the other side. Therefore, the algorithm to be explained uses the frontal face as it is and unites the transformed left side to the left, and the reflected form of this side picture to the right to get to the final texture image. To prepare the side view image for concatenating with the frontal face, facial feature lines are drawn on the face and transformed to match frontal and side feature lines as shown in Figure 6.2. Once the face feature lines are transformed, two frontal and profile faces are ready to concatenate as in Figure 6.3.

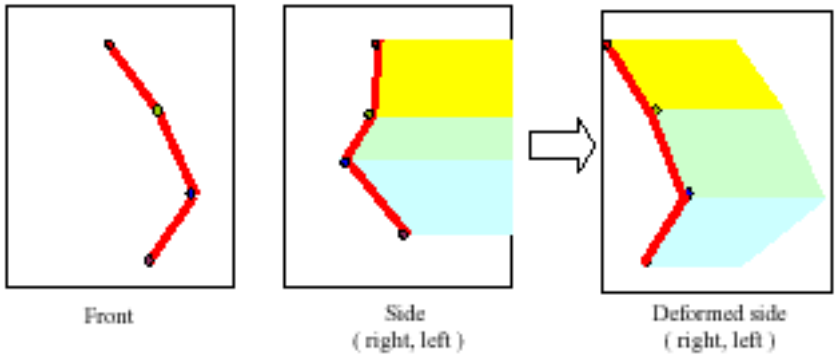


Figure 6.2: Transformation of the sideview feature lines [38].

Figure 6.3: Transformed feature lines on the face photograph [38].

Our facial animation system implemented the described method for facial texture generation. However, the output is not correct due to the different lighting conditions on the two photos. Lee and Magnenatt-Thalmann talks about a multiresolution technique in [38] and [11] to overcome this problem. Briefly, this

method will produce Gaussian and Laplacian images using reduce and expand operators to use in pyramid decomposition and merging of the images. The technique is useful for removing the boundaries between the two orthogonal images due to different illuminations on them. Once the texture image is built, the texture coordinates need to be fit to the image. For this, using simply the x and y coordinates as we did will not work. If the face pictures are transformed and united, we need to find the projection of the vertices (at least some control points) on two orthogonal planes which represent the two photographs. When we do this, we will have the point where each vertex should map, and the output texture mapping will look good.

6.2 Animation

Once we have the model of the face, the facial animation system should apply further deformations to the face in order to synthesize new images that will represent various expressions. Much work has been carried out in this field, to come up with techniques that will produce facial expressions on the model. We can define facial animation as facial expression animation, since a face can be animated by having it represent an expression like smiling, or pronouncing some word. Therefore, the word expression will cover any movement on the skin of the face to reflect the internal emotion of the human to be animated as well as accompanying deformations of the skin and muscles and bones to realize some action like speech. Our system uses the technique developed by Keith Waters [77], to define the animation by parameterizing it using states of the muscles defined on the face. The detail of the work will be presented later in this chapter, after some general knowledge on facial animation.

6.2.1 Animation Techniques

One of the elusive goals of computer animation is to synthesize realistic facial expressions. Since the facial structure is a complex collection of muscles and skin

tissue, that is difficult to model completely. Studies have been carried out to approximate the physical behavior of the face structure for about thirty years in the domain of computer graphics, which will manipulate the surface representing the face to make it have the desired expressions. Among the techniques which approximate the realistic expression formation are *interpolation*, *performance-driven animation*, *direct parameterization*, *pseudomuscle-based*, and *muscle-based animation*.

Interpolation is a widely-used technique in computer animation, where a key-framing approach is employed to specify certain frames of the animation and letting the system to build the in-betweens by interpolation of the expressions between the key-frames. [19] describes six universal expressions on the face, which are happiness, anger, sadness, fear, surprise, and disgust. These facial expressions are presented in Appendix A.1. In an animation with interpolation, these expressions and a blending of them could be used as key-frames, and in-betweens may be generated with interpolating the values of parameters of the facial model accordingly. The performance-driven animation technique obtains inputs from interactive input devices to provide parameters for facial animation. Examples include real-time systems where animator wears data gloves and body suits equipped with sensors to correctly transmit the state of expressions as input to the animation system. Direct parameterized approach [54] employs local interpolations, geometric transformations and mapping techniques to manipulate the features of the face.

Pseudomuscle-based techniques [45, 66] simulate muscle actions using geometric deformation operators without simulating the facial skin tissue behavior. A better approach is the muscle-based approach of Platt and Badler [61]. A similar muscle-based animation technique, which is the one used in our work, is developed by Waters [77], to deform the skin tissue model by simulating the muscles using a mass-and-spring model.

Various researchers worked on coding expressions of the face. One of the earliest attempts is that of Hjortsjo [30], named as the Mimic Language. His work develops a model to investigate and systematize the muscular activities

that create the diverse facial expressions. Another study is by Ekman and his colleagues [20], which describes a facial action coding system (FACS). The system defines the most basic facial muscle actions and their effect on facial expression. There are the set of all possible basic action units (AU) performable by the human face, which are minimal actions that cannot be divided into smaller actions. The facial expressions in facial animation are realized by simulation of sets of these action units. The final effort in definition of facial expressions is the MPEG-4 standard for facial animation. MPEG-4 defines facial animation parameters to describe movements of the face [37].

Keith Waters' model of simulating facial expressions [77] defines 9 pairs of muscles on the face, listed previously in the face modeling chapter, and simulates facial expressions with activations of these muscles. Deforming the skin tissue in the influence zone of the muscle performs muscle activation. Of course, this system is an approximation to the physical behavior of the face muscles since the real structure of the face is very complex. Waters primarily defines three types of muscles on the face: linear, sphincter, and sheet. The linear and sheet muscles act like linear muscle vectors, whereas sphincter muscles perform elliptical contraction.

Similar to a facial muscle composed of smaller muscle fibers, which originates from a point and inserts into another, the muscle vectors have head and tail points that also define its direction. However, the computer model of a muscle is represented by a single vector. The direction of the muscle is toward a point of attachment on the bone, and the magnitude of displacement is described according to the muscle spring constant and the tension created by a muscular contraction. The linear muscle will pull the surrounding skin towards the node of attachment on the bone, whereas a sphincter muscle will squeeze the skin tissue toward an imaginary center. The muscle model tries to compute the displacement of surface nodes in its influence zone according to spring constant and tension of the muscle.

The muscles used in our model are linear muscles. We now explain the working principles of the linear muscle simulation for animation purposes. Details of linear, sheet and sphincter muscles can be found in [56, 77].

The skin on the face is assumed to be a mesh of springs that can be deformed under the tension of muscles until a certain point, as it simulates the elastic skin tissue whose visco elastic behavior fades when the tension reaches to some value. Therefore, the muscle vector model act as forces deforming the polygonal mesh topology of the face according to some factors and constraints. To compute the effect of a muscle vector contraction on a vertex node, we take into consideration the tension of the muscle, the elasticity of the skin which can be modeled as spring constants of the region of mesh, the distance of the vertex to the muscle, and the zone of influence of the muscle. The following parameters are associated with each linear muscle (cf. Figure 6.4):

Influence zone: this is the arc of circle inside which the muscle's effect will appear. It is usually some value about 35 to 65 degrees.

Influence start (fall start): this is the tension value, after which point the muscle's influence will be effective.

Influence end (fall end): when the muscle tension value reaches this point, contracting it will not deform the skin since the model assumes skin will lose its elasticity after a point.

Contraction value: this is the muscle tension value.

The direction of the muscle vector determines the force direction that the muscle will apply.

In Figure 6.4,

- p is a point in the mesh,
- p' is its new position after the muscle is pulled along the vector v_1v_2,
- R_s and R_f represent muscle fall start and fall finish radii, respectively,
- $a1$ represents the maximum zone of influence, typically between 35 and 65 degrees,

- D is the distance of p from muscle head and
- $a2$ represents the angle between the muscle vector v_1v_2 and the vector defined by the node to relocate and the muscle head, v_1p.

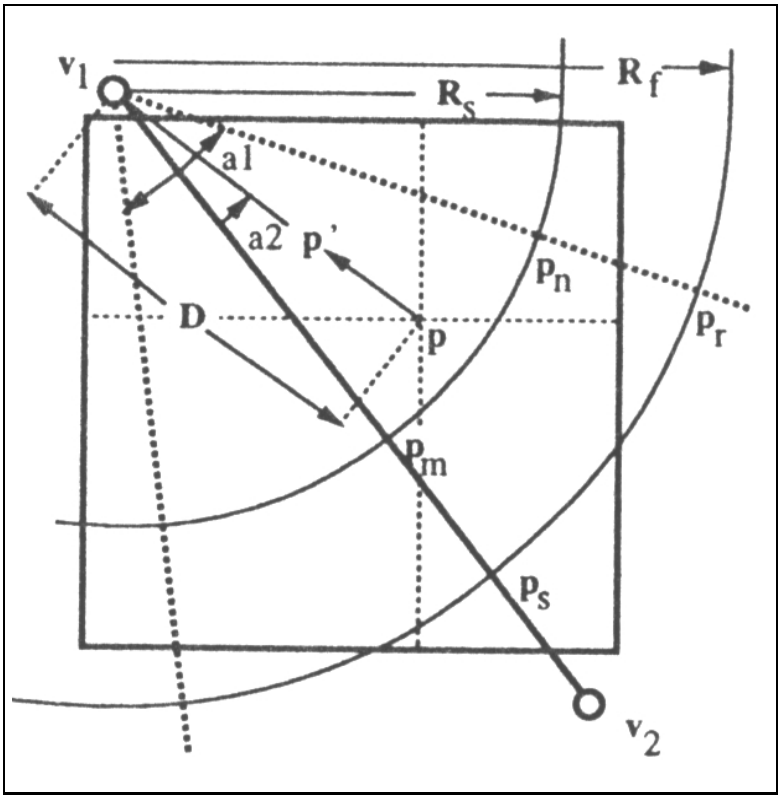


Figure 6.4: Parameters of a muscle [56].

Note that v_1 and v_2 are not necessarily placed as a node of mesh because muscles are thought as forces that can be anywhere in the space. This muscle representation is for 2D but it can easily be adapted to 3D by applying the same rules to the third dimension.

The displacement of a vertex p to a new position p' within the segment $v_1p_rp_s$ towards v_1 along the vector pv_1 is calculated as

$$p' = p + akr\frac{pv_1}{\|pv_1\|} \tag{6.2}$$

where new p' location is a function of an angular displacement parameter $a =$

$cos(a2)$. We can represent $\|pv_1\|$ as $D = \|v_1 - p\|$, and r is a radial displacement parameter calculated as

$$r = \begin{cases} cos((1-D)/R_s) \;\; for \;\; p \;\; inside \;\; sector \;\; (v_1 p_n p_m p_1) \\ cos((D-R_s)/(R_f - R_s)) \;\; for \;\; p \;\; inside \;\; sector \;\; (p_n p_r p_s p_m) \end{cases} \tag{6.3}$$

and k is a fixed constant representing the elasticity of the skin as a spring constant.

6.2.2 Animation in Our System

We use the muscle-based approach of Keith Waters to define facial expressions on the face. A key-framing technique is employed to produce animations. Each key-frame sets the states of each of the muscle defined on the face as well as some other data like jaw rotation angle and rotation for the head, and cosine interpolation of these values is used to generate the inbetweens. Cosine interpolation is suitable for approximating the viscoelastic behavior of the skin [56]. It is the most suitable interpolation technique when the model is more complex and small differences between two frames can be seen better with the model that has enough detail. This is because the muscles' acceleration and deceleration can be represented as a cosine function of time. In this type of interpolation, a cosine interpolation function is used to calculate the time of displaying an inbetween. This scheme is also known as acceleration and fits well to the facial animation. The display time for an inbetween frame i is given by

$$tB_i \;=\; t_1 + \Delta t(1 - cos(\theta)), \tag{6.4}$$

where $\theta = \frac{i\pi}{2(n+1)}, \quad 0 < \theta < \frac{\pi}{2}$ and $i = 1, 2, 3, \ldots, n$ for n inbetweens.

The user can easily generate the keyframes of an animation by using the buttons and interpolate them to generate a facial animation. The animations are saved to a file in the form of expression parameters and orientation parameters for each keyframe and they are interpolated to generate inbetweens. An example animation file is given in Figure 6.5.

```
5
1:
0.0 4.0 4.6
1.1 1.0 0.0 0.0 0.8 0.9 0.2 0.2 0.1 0.3 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ;
5:
2.2 0.1  0.2
0.0 0.0 0.7 0.6 0.2 0.0 0.1 0.0 2.0 2.0 1.8 1.9 1.2 1.1 1.4 1.3 0.2 0.3 ;
```

Figure 6.5: An example animation file.

In this format, the first integer is the number of total frames in the animation to be formed. Then, each keyframe is described by the frame number which is followed by a colon, and 21 floating point values for the parameters. The first two parameters are the values of rotation in x and y axes, respectively. The third is the value of the jaw rotation angle, which should be between 0.00 and 4.00. Next, the contraction values for the 18 muscle vectors are listed in the order found in the expression parameters. When two consecutive frames in the file do not have consecutive frame numbers, inbetween frames are interpolated using cosine interpolation.

The user interface of the system has a pane where OpenGL is used to paint, an option menu, a slider, three radio buttons and four buttons. When first started, the system asks for the filenames of the frontal face image, and the profile image of the face to be modeled. Then, the files are read and processed for feature extraction to locate the initial features in the pictures. Then, the user is let to point, choose and move the feature points with the mouse. The option menu in the left top can be used to display the images in grayscale, colored, edge detected with a user-specified threshold, and with intensity peaks with a user-specified threshold. Once the feature points are located, the user should press the 'Build Model' button to deform the generic model according to the feature parameters. The result is initially displayed in wireframe mode. The display mode can be selected using the radio buttons at the right top. The middle two buttons are for opening an animation file, and playing it on the screen with inbetweens interpolated, respectively. Figure 6.6 displays the user interface, and still frames from the animations generated by the system are presented in Appendix A.2.

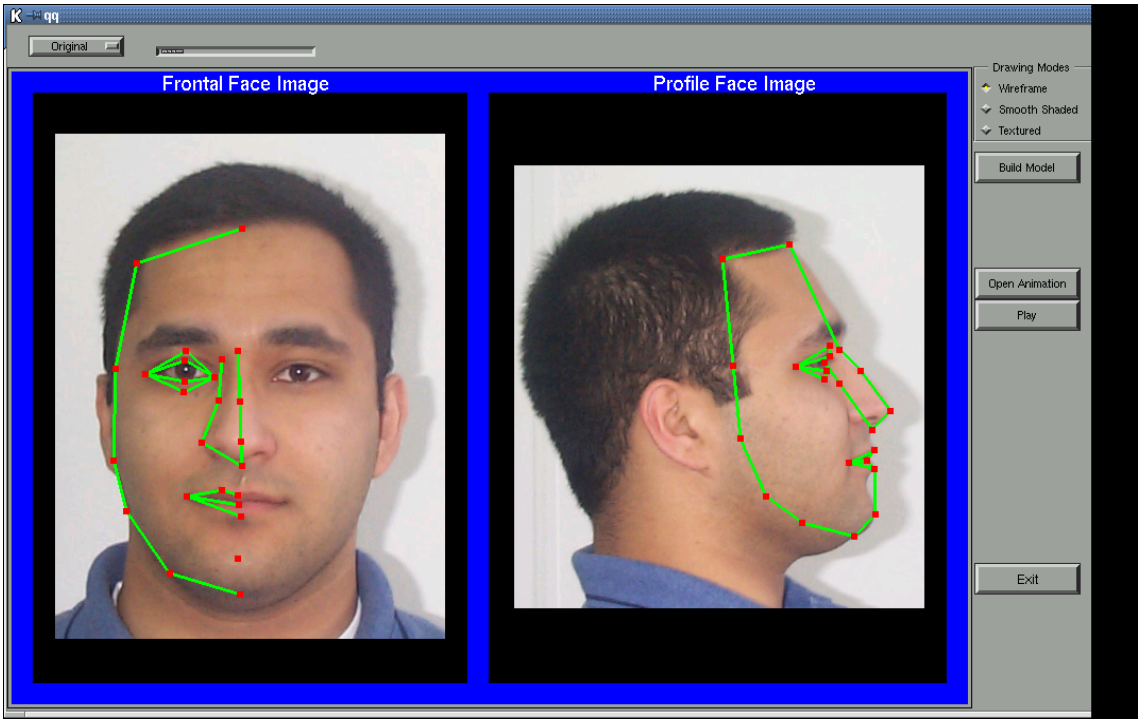


Figure 6.6: The user interface of the system.

Chapter 7

Conclusions and Future Work

A facial modeling and animation system we developed is explained in this thesis. The face model used is a muscle-based face model developed by Keith Waters [56], with 512 vertices and 878 triangular polygons. The model is animated using contraction of the 18 muscles defined on the face, with the muscle-based animation technique mentioned in the same work of Waters.

The main emphasis of the thesis work is to generate personalized face models by deforming a generic face model according to the two orthogonal pictures of a face, and then animate the individualized face models generated in this manner. For this purpose, we let the user edit the places of the facial features, which are compliant with the MPEG-4 face definition parameters, on these pictures. Then, the system builds the corresponding face with local deformations. The user can animate the generated personalized face model.

Sample executions of the animation system show that the system will perform well in terms of speed for facial animation. A rate of 100 fps can be reached with the model we have used. Improvements to the system can be done using the techniques explained in the related sections of the thesis. We list some of the ideas for improvements and further developments in the next section.

7.1 Future Work

As explained throughout the thesis, a facial animation system is composed of many parts each of which can be examined thoroughly. Even a single part can need great amount of research and labor to develop methods and implement. The system we have implemented consists of methods we have come up with united with existing techniques partly or fully incorporated into the system.

Although the face model we used is a facial mask and it is adequate for our purposes, it would be better if a model of the complete head is used in the system with teeth, tongue and better eye models. This would require additional work in all parts of the system, but surely will increase the realism of the animations.

Using a complete head model will increase reality, and using a more detailed model will synthesize each part of the system better with no doubt. With more vertices on the model, the wrinkles and creases of the skin can be modeled well; the shading will produce better and more believable results, and simulation of muscle contraction will produce more flexible expressions on the face. However, this will bring cost to the system in terms of space and time. The space will not be of high consideration due to high availability of RAM that will be enough for detailed polygonal models, but time will be needed to be optimized since we need to produce the facial images with desired expressions and display them on the screen with an acceptable frame rate, typically at least 30 fps. Therefore, while using such a model, an optimum choice in terms of detail should be made to produce wanted results. Even, techniques like adjusting level of detail when the model is displayed far away which will make details useless to compute, or techniques to avoid computation of unseen parts or unwanted effects could be employed to ease the work of CPU for production of a scene. Furthermore, the hair can be modeled using techniques mentioned in [56]. Or a study can be realized to build the facial model with extracting some parameters from the pictures which will define the color and shape of the hair. This is a really difficult problem that is open to research. To increase reality, at least, using the hair as a texture will be a good start. Still, the time consideration will be important as well as space in a system with hair animation, which may need to model hairs as

a number of strands whose lighting calculation can be complex.

With a more complex model, we will need to define more feature points to use as control points in deformation of the face. The MPEG-4 defines an adequate number of parameters to define even the whole body, however, we may need to use more parameters for special purposes that will ease especially the deformation process. Of course, we will need to use better techniques to automate the feature extraction step, which will most probably be a very time-consuming process, needing to be optimized in that sense, too. The system can implement various methods to obtain the feature point values like inputs from hardware like laser scanner, orthogonal or some other number of photos, different photogrammetric methods, which will bring flexibility to the system in terms of forming a better model. Different number of feature points can be defined for some faces, and the system can be designed to compute the rest of the points using anthropometrical data and interpolation. Furthermore, the user may be let to decide on the level of detail for the output deformed face model. This will be useful when the system can be used to animate either simple faces for simple computer animation or more complex ones for believable animation like video conferencing that will further require detailed speech animation. Even in the case of using two orthogonal photographs for feature extraction, the system can use more complex image understanding methods that are trained over a large range of faces, to minimize the errors in determining the facial features. The system should be able to correctly segment the image to decide on boundaries for hair region, facial region, and background. Methods that would guess the lighting conditions, remove noise from images, adjust thresholds automatically for determining edges and other image processing methods can be incorporated into the system to detect the features well.

The feature extraction is highly related to the deformation process, in that, the parameters required are obtained as a result of feature extraction process. As well as improving automatic feature extraction, it would be fine to prepare a better user interface for manual edition of feature points on the input pictures. Constraints can be built in order to avoid the user from placing feature points in an undesired way. The user interface can even be changed to form the texture

map image automatically, or with some manual manipulation at some points. A more functional program can also display an overview of the built model so that the user can refine the parameters for more realistic modeling. Providing such feedback in building the model will help produce more realistic output. With an improved extraction of features and greater number of control points, the deformation process will need to be designed more carefully that will deform smaller local regions. We may employ union of the many techniques that can deform the polygon mesh that will build realistic model suitable to the wanted parameters. When parts of the face are modeled according to the parameters, interpolation between the other regions could be improved. When the model is more detailed, the system can have the functionality to model the wrinkles and creases.

The rendering part can also be improved. The current system sets the material for the facial mask to have certain properties and lets OpenGL to render it using Gouraud shading [28] under the specified lighting conditions. We may have a better shading in this case by arranging the material properties differently in different regions of the face to produce more realistic faces. Of course, using texture mapping will be the best of all for reality of the synthesized image. As mentioned previously, our system already requires a better texture mapping technique, so the method mentioned in [38] can be implemented as a first future work.

Once we implement a better face model with better rendering, we may need to simulate the animation process in more detail. For this, we may define new muscles on the face that can be of different shapes and implement activation of them. This will require an examination of the facial anatomy to produce a better muscle model simulation. For example, we may add muscles around the mouth for doing speech animation. Another future work can be incorporation of different techniques for animation to form facial expression on our parameterized model. Using more parameters for a single expression will require us to change the data animation files have, or we could design the system to recognize different kind of animation files. We can further implement procedures to form output files in various video compression formats. The system could be integrated with a speech

animation module as described in [72], where a text input in some format will be used to compute the parameters for required expression on the face and animate the face accordingly.

With a similar approach, we can build a MPEG-4 compatible animation system, that will decode data transmitted over the network as specified in the standard and build and animate the model accordingly. Furthermore, the system can be flexibly designed to work with different generic faces and feature sets.

Finally, we can build an animation editor interface to form animation files by adjusting the parameters for key-frames. A more complex user interface could be interaction with the model on the OpenGL window. The mouse could be used to select muscles or vertices on the face model, and manipulated with deformations.

As it is seen, further improvements for such a facial animation system are almost limitless. We may come up with ideas that will make the system more flexible, more realistic and more functional. However, time constraints will restrict us in adding such improvements to the system. The current technology may be inadequate for realizing some parts of the methods used. The CPUs can still be slower to produce fast results when in each frame it needs more operations to synthesize the required facial image. Modeling techniques can be away from exactly simulating the real life. Therefore, the result may probably be just a better approximation to the real life, leaving the facial animation Turing test as described previously not passed. The synthesized image can still be distinguished from the real face when viewed by the human eye. However, the researches done in this will certainly help ease some part of human life or produce more realistic outputs within entertainment business.

Bibliography

[1] A. Amini, S. Tehrani, and T. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Proc. of the Second International Conference on Computer Vision*, pages 95–99, Florida, 1988.

[2] R. Baron. Mechanisms of human facial recognition. *International Journal of Man Machine Studies*, 15:137–178, 1981.

[3] R. Bartles, J. Beatty, and B. Barsky. *Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, Los Altos, CA, 1987.

[4] P. Bergeron and P. Lachapelle. Controlling facial expressions and body movements. In *Advanced Computer Animation, SIGGRAPH 85 Tutorials*, volume 2, pages 61–79, New York, 1985.

[5] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. of the IEEE International Conference on Computer Vision*, pages 374–381, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[6] A. Blake and M. Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *ACM Computer Graphics (Proc. of SIGGRAPH'94)*, volume 28, pages 185–192, 1994.

[7] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[8] S. Brennan. Caricature Generator. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1982.

[9] R. Brunelli and T. Poggio. Face recognition: features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1062, 1993.

[10] C. Bunks. *Grokking the GIMP*. New Riders Publishing, 2000.

[11] P. Burt and E. Anderson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[12] T. Capin, E. Petajan, and J. Ostermann. Efficient modeling of virtual humans in MPEG-4. In *Proc. of IEEE International Conference on Multimedia and Expo (II)*, pages 1103–1106, New York, 2000.

[13] H. Chernoff. The Use of Faces to Represent Points in N-dimensional Space Graphically. Technical Report NR-042-993, Office of Naval Research, Washington, DC, 1971.

[14] C. Choi, H. Harashima, and T. Takebe. Highly Accurate Estimation of Head Motion and Facial Action Information on Knowledge-based Image Coding. Technical Report PRU90-68, I.E.I.C.E.J., 1990.

[15] L. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):211–218, 1991.

[16] X. Deng. *A Finite Element Analysis of Surgery of the Human Facial Tissue*. PhD thesis, Columbia University, New York, 1988.

[17] A. DeRose. Composing Bézier simplexes. *ACM Transactions on Graphics*, 7:198–221, 1988.

[18] S. DiPaola. Extending the range of facial types. *Journal of Visualization and Computer Animation*, 2(4):129–131, 1991.

[19] P. Ekman. The argument and evidence about universals in facial expressions of emotion. In *Handbook of Social Psychophysiology*, pages 143–164. H. Wagner and A. Monstead (eds.), John Wiley, Chichester, New York, 1989.

[20] P. Ekman and W. Friesen. *Manual for the Facial Action Coding System.* Consulting Psychologists Press, Inc., Palo Alto, CA, 1977.

[21] M. Escher, I. Pandzic, and N. Magnenat-Thalmann. Facial deformation for MPEG-4. In *Proc. of Computer Animation'98*, pages 56–62, 1998.

[22] I. Essa and A. Pentland. A Vision System for Observing and Extracting Facial Action Parameters. Technical Report 247, Massachusetts Institute of Technology, Perceptual Computing Section, Cambridge, MA, 1994.

[23] G. Faigin. *The Artist's Complete Guide to Facial Animation.* Watson-Gutill Publications, New York, 1990.

[24] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22(1):67–92, 1973.

[25] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *ACM Computer Graphics (Proc. of SIGGRAPH'97)*, pages 209–216, 1997.

[26] E. Gasper. Getting a head with hyperanimation. *Dr Dobb's Journal of Software Tools*, 13(7):18, 1988.

[27] M. Gillenson. *The Interactive Generation of Facial Images on a CRT Using a Heuristic Strategy.* PhD thesis, Ohio State University, Computer Graphics Research Groupi, Columbus, OH, 1974.

[28] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623–629, 1971.

[29] D. Hill, A. Pearce, and B. Wyvill. Animating speech: an automated approach using speech synthesis by rules. *The Visual Computer*, 3:277–289, 1988.

[30] C.-H. Hjortsjo. *Man's Face and Mimic Language.* Lund, Sweden, 1970.

[31] *4020/RGB 3D Scanner with Color Digitizer.* Cyberware Lab., Inc., 1990.

[32] T. Kanade. Picture Processing by Computer Complex and Recognition of Human Faces. Technical report, Kyoto University, Dept. of Information Science, 1973.

[33] M. Kass, A. Witkin, and D. Terzopoulus. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[34] R. Koch, M. Gross, F. Carls, D. von Büren, G. Fankhauser, and Y. Parishi. Simulating facial surgery using finite element methods. In *ACM Computer Graphics (Proc. of SIGGRAPH'96)*, pages 421–428, 1996.

[35] T. Kurihara and K. Arai. A transformation method for modeling and animation of the human face from photographs. In *Proc. of Computer Animation'91*, pages 45–58, Tokyo, 1991. Springer-Verlag.

[36] J. Lander. Read my lips: facial animation techniques. *Game Developer Magazine, CMP Media Group*, June 1999.

[37] W. Lee, M. Escher, G. Sannier, and N. Megnenat-Thalmann. MPEG-4 compatible faces from orthogonal photos. In *Proc. of Computer Animation'99*, pages 186–194, Geneva, Switzerland, 1999.

[38] W.-S. Lee and N. Magnenat-Thalmann. Head modeling from pictures and morphing in 3D with image metamorphosis based on triangulation. In *Proc. of Modelling and Motion Capture Techniques for Virtual Environments (CAPTECH'98)*, pages 254–267, 1998.

[39] W. S. Lee and N. Magneneat-Thalmann. Fast head modeling for animation. *Image and Vision Computing*, 18(4):355–364, 2000.

[40] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *ACM Computer Graphics (Proc. of SIGGRAPH'95)*, pages 55–62, 1995.

[41] J. Lewis and F. Parke. Automated lip-synch and speech synthesis for character animation. In *Proc. of Computer/Human Interface+Graphics Interface'87*, pages 143–147, 1987.

[42] W. Lorensen and H. Cline. Marching cubes: High resolution 3D surface construction algorithm. In *ACM Computer Graphics (Proc. of SIGGRAPH'87*, pages 163–169, 1987.

[43] J. Lüttin, N. Thacker, and S. Beet. Locating and tracking facial speech features. In *Proc. of International Conference on Pattern Recognition*, Vienna, Austria, 1996.

[44] N. Magnenat-Thalmann, H. Minh, M. deAngelis, and D. Thalmann. Design, transformation and animation of human faces. *The Visual Computer*, 5:32–39, 1989.

[45] N. Magnenat-Thalmann, N. Primeau, and D. Thalmann. Abstract muscle actions procedures for human face animation. *The Visual Computer*, 3(5):290–297, 1988.

[46] M. Malciu and F. Preteux. A robust model-based approach for 3D head tracking in video sequences. In *Proc. of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'2000)*, pages 169–174, Grenoble, France, 2000.

[47] S. Morishima and H. Harashima. A natural human-machine interface with model-based image synthesis scheme. In *Proc. of Picture Coding Symposium*, pages 319–322, Tokyo, Japan, 1991.

[48] MPEG-4 Overview, `http://drogo.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm`, 1999.

[49] M. Nahas, H. Huitric, and M. Sanintourens. Animation of a B-spline figure. *The Visual Computer*, 3(5):272–276, 1988.

[50] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley, Reading, MA, 1993.

[51] W. Niblack. *An Introduction to Digital Image Processing*. Prentice Hall, 1986.

[52] J. Park and J. Han. Contour matching: a curvature-based approach. *Image and Vision Computing*, 16:181–189, 1998.

[53] F. Parke. Computer generated animation of faces. In *Proc. of ACM National Conference*, volume 1, pages 451–457, 1972.

[54] F. Parke. A parameterized model for facial animation. *IEEE Computer Graphics and Applications*, 2(9):61–70, 1982.

[55] F. Parke. ACM SIGGRAPH Course Notes 26: State of the Art in Facial Animation, 1990.

[56] F. Parke and K. Waters. *Computer Facial Animation.* A.K. Peters, 1996.

[57] A. Pearce, B. Wyvill, and D. Hill. Speech and expression: A computer solution to face animation. In *Proc. of Graphics Interface'86*, pages 136–140, 1986.

[58] S. Pieper. More than Skin Deep: Physical Modeling of Facial Tissue. Master's thesis, Massachussetts Institute of Technology, Media Arts and Sciences, 1989.

[59] F. Pighin, J. Hecker, D. Lischinksi, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *ACM Computer Graphics (Proc. of SIGGRAPH'98)*, pages 75–84, 1998.

[60] S. Platt. A System for Computer Simulation of the Human Face. Master's thesis, University of Pennsylvania, Philadelphia, PA, 1980.

[61] S. Platt and N. Badler. Animating facial expressions. *ACM Computer Graphics (Proc. of SIGGRAPH'81)*, 15(3):245–252, 1981.

[62] R. Rao and R. Mersereau. Lip modeling for visual speech recognition. In *Proc. of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, 1994.

[63] M. Reinders, M. van der Goot, and J. Gerbrands. Estimating facial expressions by reasoning. In *Proc. of the ASCI'96 Conference*, pages 259–264, Lommel, Belgium, 1996.

[64] J.-L. D. S. Horbelt. Active contours for lipreading-combining snakes with templates. In *Proc. of the 15th GRETSI Symposium on Signal and Image Processing*, Juan les Pins, France, 1995.

[65] E. Sabert and A. Tekalp. Frontal-view face detection and facial feature extraction using color, shape, and symmetry-based cost functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:669–680, 1998.

[66] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. In *ACM Computer Graphics (Proc. of SIGGRAPH'86)*, pages 151–160, 1986.

[67] T. Shakunaga, K. Ogawa, and S. Oki. Integration of eigentemplate and structure matching for automatic facial feature detection. In *Proc. of AFGR'98*, 1998.

[68] K. Sobottka and I. Pitas. Segmentation and tracking of faces in color images. In *Proc. of International Conference on Automatic Face and Gesture Recognition*, pages 236–241, Killington, Vermont, 1996.

[69] W. Sorgel and B. Girod. Deformable templates for the localization of anatomical structures in radiologic images. In *Bildverarbeitung fur die Medizin*, pages 49–53, 1999.

[70] G. Tunalı. Animating Facial Images with Drawings. Master's thesis, Bilkent University, Dept. of Computer Engineering and Information Science, 1996.

[71] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognition Neuroscience*, 3(1):71–86, 1991.

[72] B. Uz, U. Güdükbay, and B. Özgüç. Realistic speech animation on synthetic faces. In *Proc. of Computer Animation'98*, pages 111–118, 1998.

[73] M. Vannier, J. Marsh, and J. Warren. Three-dimensional computer graphics for craniofacial surgical planning and evaluation. *ACM Computer Graphics*, 17(3):263–273, 1983.

[74] M. Vogt. Fast matching of a dynamic lip model to color video sequences under regular illumination conditions. In *Speechreading by Human and Machines*, volume 150, NATO ASI Series F, pages 399–407, 1996.

[75] C. Waite. The Facial Action Control Editor, FACE: A Parametric Facial Expression Editor for Computer Generated Animation. Master's thesis, Massachusetts Institute of Technology, Media Arts and Sciences, Cambridge, MA, 1989.

[76] C. Wang. Langwidere: Hierarchical Spline Based Facial Animation System with Simulated Muscles. Master's thesis, University of Calgary, Calgary, 1993.

[77] K. Waters. A muscle model for animating three-dimensional facial expression. In *ACM Computer Graphics (Proc. of SIGGRAPH'87)*, pages 17–24, 1987.

[78] P. Weil. About Face. Master's thesis, Massachusetts Institute of Technology, Architecture Group, Cambridge, MA, 1982.

[79] D. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(1):14–26, 1992.

[80] L. Williams. Performance driven facial animation. In *ACM Computer Graphics (Proc. of SIGGRAPH'90)*, pages 235–242, 1990.

[81] L. Wiskott, J. Fellous, N. Kruger, and C. von der Malsburg. Face recognition and gender determination. In *Proc. of the Int. Workshop on Automatic Face and Gesture Recognition*, pages 92–97, Zurich, Switzerland, 1995.

[82] L. Wiskott, J.-M. Fellous, N. Kruger, and C. von der Malsburg. Face Recognition by Elastic Bunch Graph Matching. Technical Report IR-INI 96-08, Institut fur Neuroinformatik, Ruhr-Universitat Bochum, Germany, 1996.

[83] A. Yuille. Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 3(1):59–70, 1991.

[84] A. Yuille, D. Cohen, and P. Hallinan. Feature extraction fram faces using deformable templates. In *Proc .of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'89)*, pages 104–109, Los Alamitos, CA, 1989. IEEE Computer Society Press.

Appendix A

Sample Face Models with Expressions

A.1 Universal Facial Expressions

Figure A.1 presents universal facial expressions on the face model.

A.2 Facial Expressions on Individual Faces

Figure A.2 presents different expressions on individual faces. Here, the face models are rendered with texture mapping.

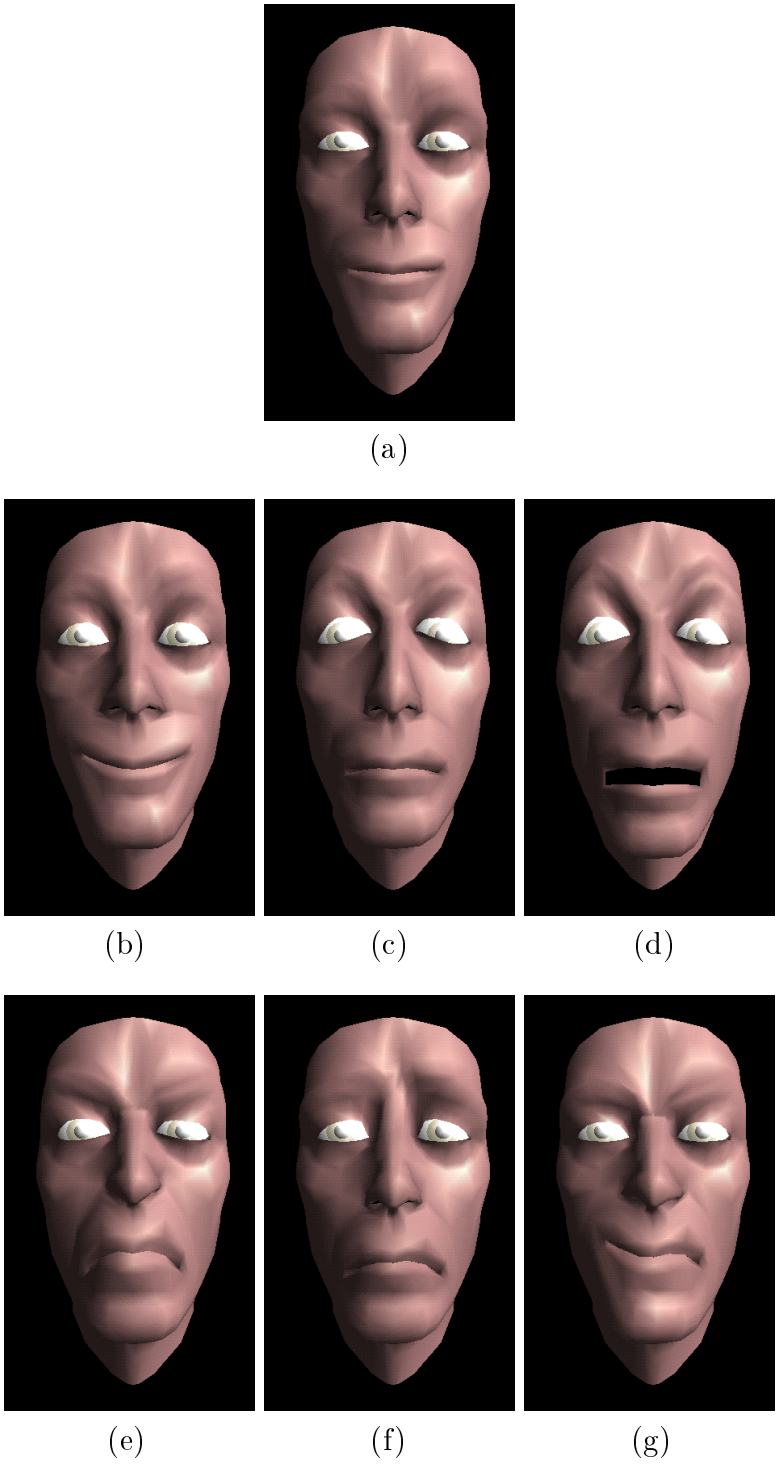


Figure A.1: Universal facial expressions: (a) neutral, (b) happiness, (c) surprise, (d) fear, (e) anger, (f) sadness, and (g) disgust.

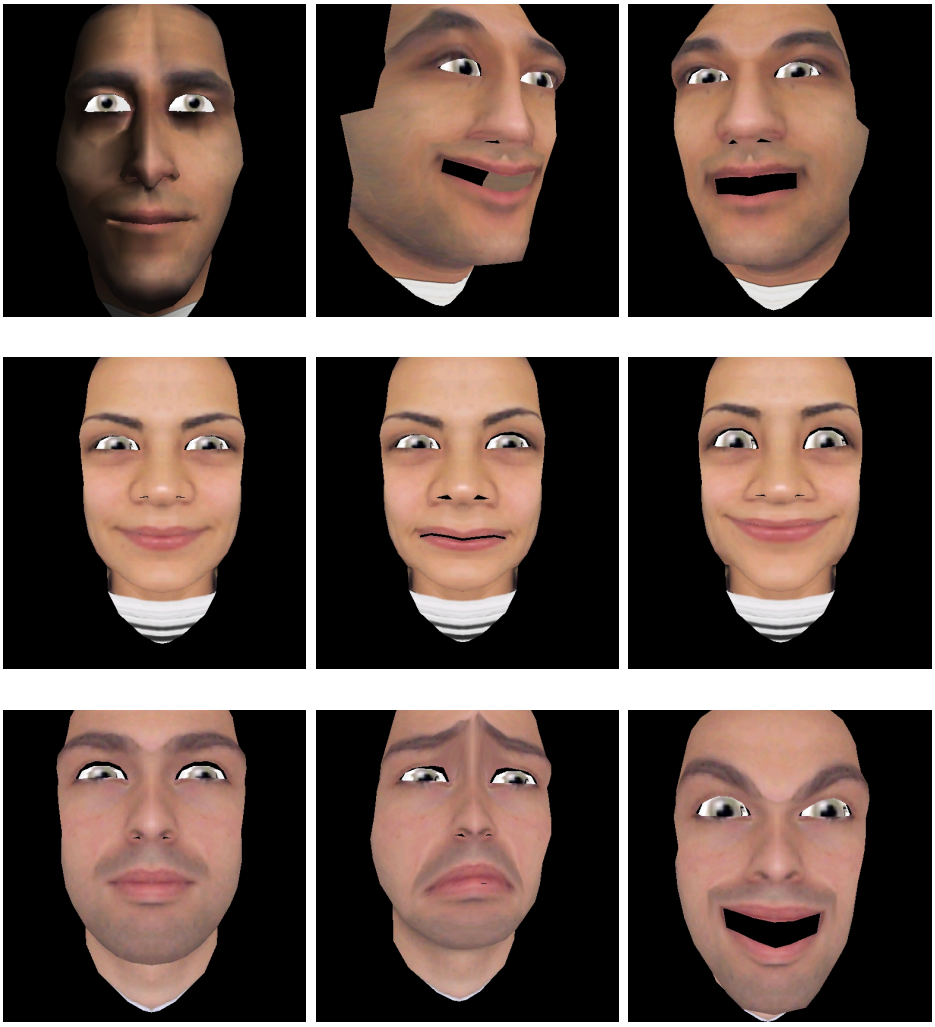


Figure A.2: Different expressions on individual faces.