University of Alberta

Facial Model Improvement using 3D Texture Mapping Feedback

by

Yongjie Liu

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requiements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta

Fall 2004

Your file Votre référence
ISBN: 0-612-95802-7
Our file Notre référence
ISBN: 0-612-95802-7

Acknowledgements

First, I am most grateful to Dr. Basu, for his patient advice and guidance over these years, and for having given me the opportunity to work on such an exciting project.

I would also like to extend my sincere appreciations to Dr. Li and Dr. Cockburn for serving on my master thesis committee. They gave me plenty of insight advice and suggestions.

Furthermore, many thanks to those researchers, who agree to let me use their pictures in my thesis. My thesis is more expressive with their work.

Most of all, I am very grateful to my parents for their love, support and encouragement all the time. I would not finish this thesis without their deep love and strong support.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

With the evolution of computer technology, 3D computer graphic techniques and their applications are more and more widely used in many fields. Interactive three-dimensional (3D) human facial modeling has a wide range of applications from surgical training to video games. Many of these applications require the deforming of human facial models. Usually, such 3D facial models are represented in terms of primitives, such as points, lines, and polygons. Although complicated and beautiful models can now be rendered with many such primitives by using graphics hardware, how to deform such models to represent desirable human shapes generally, interactively and efficiently is still a problem.

Deformation, or modification of a 3D model, is one of the main techniques used in 3D computer graphics. It basically consists of editing the shapes of 3D models, and providing a general modification of a 3D model, involving operations such as twisting, bending, and stretching. 3D facial modeling is a thriving research field that can yield various important applications, as follows:

- **Virtual reality**. People are able to communicate face-to-face in a virtual world without being at the same place, by simulating facial motions.

- **Facial Animation**. In computer games, 3D animation can be achieved with gradual changes affecting a character's 3D model.
- **Plastic surgery**. Plastic surgery alternatives can be studied and performed with the help of 3D computer graphic techniques. For example, given a generic facial model, it is possible to reconstruct it into various different models, according to external constraints. This will also be useful to many other medical areas.

Generally, the following requirements have to be observed in the previously cited applications: ease of handling the object for the user, quality of deformation/modification result, and speed of the process. A facial model with well-structured information is ideal for further reconstruction and could make future work more efficient compared to a facial model with only raw data. Currently, there exist many approaches to construct a 3D facial model in the virtual world. Two methods are most commonly used: *Laser Scanning* and *Structured Light Scanning*. With Laser Scanning, range images may be obtained by some sensor device, such as a laser scanner. In the range image, the distance of the object surface is known so that points covered by the laser on the surface can be retrieved. Therefore, the 3D object model is specified by the points. A 3D camera-laser "Rotating Arm Scanner" is produced by Zoomage Inc. [3]. This device was used as a texture image scanner in our work. The second common method involves the use of structured light scanning. This type of 3D scanner uses a projector and camera pair to scan and capture an object via structured light patterns. 3D shapes can be processed and calculated after the image is captured

by the scanner. An approach for acquiring facial surfaces is presented in [6] using this technique.

The above methods primarily focus on how to establish or produce a good 3D shape. However, the human face is a very irregular structure, which varies from person to person. It is inefficient and time-consuming to obtain a 3D model for many different people. Different faces share common facial features. For example, the eye contour is approximately an ellipse; the lip contour consists of several curves; and the spatial relationship of facial organs — such as brow, nose, eye, and mouth — is relatively steady. A typical method is to begin with an existing generic facial model without any distinctive features, and modify it gradually to precisely match a specific person's face. Lee *et al.* [34] start with a generic facial model and reconstruct it from information obtained from three head photos (left, front and right). This approach needs to generate a texture image from input images and fits them onto the 3D model, hence extra computation is required and may not be accurate. Simmons *et al.* [53] introduce a method for 3D model creation that "morphs" a canonical anatomically-based model. The canonical model is built on top of a layered articulated segment hierarchy and contains a representation of the creature's skeleton, muscles, and skin. This approach is complicated for the human head model since it requires "morphing" in a bone-muscle-skin hierarchy.

As mentioned previously, our work starts with a generic facial model, which is more feasible and obvious. On the other hand, faces are characterized by their shapes and textures. The facial model appears more realistic after being texture-

mapped. Therefore, in order to enhance the realism of a facial model in the virtual world, a facial model deformable technique and a texture-mapping technique are both proposed to work together to meet people's requirements for interaction, efficiency and speed.

1.2 Objective

Our approach is to reconstruct/deform a generic 3D facial model in order to fit a person's face precisely. Unlike the most popular deforming approach, called Free Form Deformations (FFD), we develop several deformable templates for 3D curves of organs (mouth, eyes, etc.) on the face, and change the shape of the 3D model by modifying the parameters associated with the templates. In the next chapter we explore the reasons for choosing our own deformation method instead of other approaches.

The second objective is to perform texture mapping on a facial model to gain visual feedback of the model reconstruction. This procedure is performed interactively by selecting feature points on both the 3D model and the texture image, and setting up a correspondence between them. Hence, a texture-mapped 3D facial model is obtained and the model may be readjusted later if the matching quality does not meet the user's requirements.

The third objective is to implement the system. This includes scanning the facial texture image, data collection from the 3D facial model, and a Java 3D application for the user to operate on.

1.3 Main Contributions

The overall goal of this research is to build a system for pre-surgery planning or virtual plastic surgery purposes. The main contributions of this thesis are listed below:

Parameterized and deformable 3D curve template for facial organs

In Chapter 3 we present a new method to represent curves in 3D space that has been proposed, developed, and tested. A mathematical representation is used for a parameterized 3D curve, which can be applied to the contour of facial organs such as mouth and eyes. To deform a 3D model, the parameters of the templates must be changed accordingly. Since the facial organ contour is composed using our deformable curves, the organ's shape can be modified via the changes of contour curves.

Provide a novel and simple method for user interaction

We give freedom to the user during the process of facial reconstruction since the user is able to control both the deformation and texture mapping. We implement a straightforward user interface (UI) that is capable of displaying both a facial texture image and a 3D facial model. The texture of interest (TOI) is selected

first and the corresponding feature points on the facial model are associated accordingly. A detailed description is given in Chapter 3.

1.4 Organization

This thesis is divided into five chapters. In Chapter 1, we have introduced the subject of facial model reconstruction (improvement) using 3D texture mapping feedback, and described our objectives and summarized the contributions of this research.

Chapter 2, *Background and Related Work*, presents a detailed look at previous work that has influenced ours or has provided background to the current research. This includes: 3D model deformation techniques, 3D curve representation methods, and texture mapping on the human face.

Chapter 3, *Our Approach*, outlines how our method achieves facial model improvement using 3D texture mapping feedback. We also introduce the mathematical model of the deformable template of the proposed 3D curve. Texture mapping of facial organs, which is related to our work discussed in this chapter as well.

Chapter 4, *Experimental Results*, provides the results obtained using our approach. Chapter 5, *Conclusions and Future Work*, presents our conclusions and suggests possible future work.

Chapter 2

Background and Related Work

This chapter presents the background, and other work related to our research topic. There are several factors that must be considered to achieve our ultimate goal. First, the human face is an extremely complex geometric form; therefore an overview of facial modeling is introduced in the following section. Second, we review some popular approaches for facial model deformation. Because the face exhibits countless tiny creases and wrinkles, as well as subtle variations in color and texture, facial texturing is included at the end of this chapter.

2.1 Facial Modeling

Traditionally, facial models have been built slowly with much effort by manual design and digitization. There are many possible ways, such as using a plastic model [17, 37], and most are time-consuming jobs. Fortunately, a real human face now can be modeled automatically using new technologies, such as 3D scanners. More efficient and accurate methods are currently being developed and used. Basically, they are classified into four categories, which are described in the following subsection.

2.1.1 Laser Scanning

Some sensor devices, such as laser scanners, can yield range images. For each pixel of the range image, the range to the visible surface of the object in the scene is known. Therefore, spatial position is determined for a large number of points on the surface of an object. Many approaches have been proposed for building 3D models using range images. Curless and Levoy [8] present a volumetric method for integrating range images (as many as 70) to generate seamless, highly detailed models of up to 2.6 million triangles. A number of other techniques have been developed for reconstructing surfaces by integrating groups of aligned range images as well. A desirable method requires an efficient and robust algorithm which possesses the following properties: representation of range uncertainty, utilization of all range data, incremental and order-independent updating, time and space efficiency, no restriction on topological type, and the ability to fill holes in reconstruction. Reid *et al.* [48] describe a high-speed scanning triangulation system which captures and displays range data.

The laser scanning technique has been successfully applied to human body measurement. It involves projecting stripes of laser light onto the object of interest, and viewing it from an attached camera. Deformations in the image of the light stripe correspond to the topography of the object under the stripe, and are measured. A typical commercial 3D laser scanning system is the Cyberware® Head and Face Color Scanner (Figure 1) [2]. In operation, the Cyberware® shines a laser onto an object to create a lighted profile. A high-quality video sensor captures this profile from two viewpoints. The system can digitize thousands of

these profiles in a few seconds to capture the shape of the entire object. Simultaneously, a second video sensor acquires color information. Another similar example is the Zoomage® Rotating Arm Scanner (Figure 2) [3], which is capable of generating both a 3D facial model and a 360-degree texture image. This is the 3D scanner used in our work and it will be introduced in detail later.

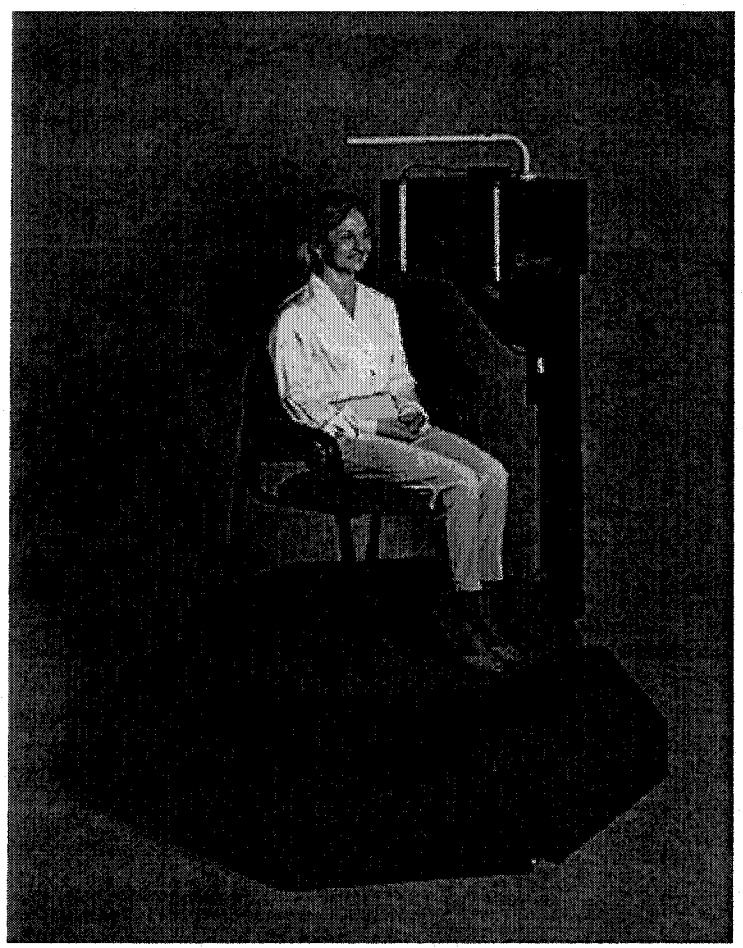


Figure 1: Cyberware® head and face color scanner [2]

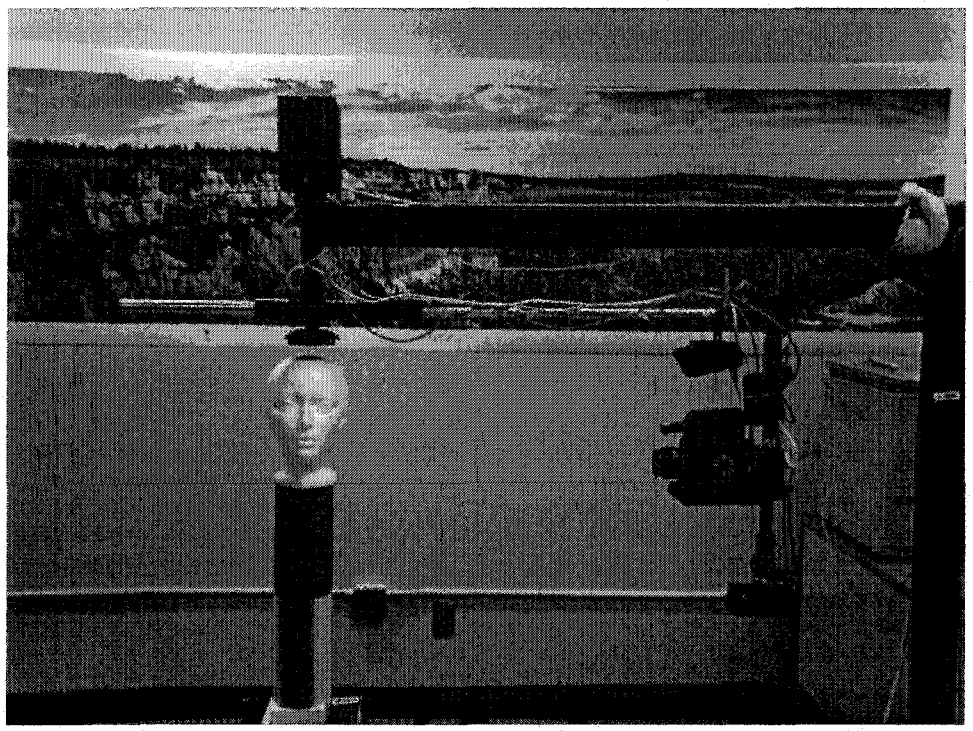


Figure 2: Zoomage® rotating arm scanner

Many researchers utilized laser scanners to construct facial models. The Cyberware® Color Digitizer is used to acquire data from a human face in the work of Lee *et al.* [35]. The scanner rotates 360 degrees around the subject, who sits motionless on a stool as a laser stripe is projected onto the head and shoulders. A range image and a reflection image are obtained after the scan is complete. The authors developed an algorithm that begins with the acquired data and automatically constructs an efficient and fully functional model of the subject's head.

2.1.2 Structured Light Scanning

This type of 3D scanner uses a projector and camera pair to scan and capture an object using structured light patterns. The basic concepts of structured light scanning are: a single image with the projected light pattern is used to recover absolute 3D coordinates. Most of the 3D facial information lies in the resolution offered by normal cameras and projectors. Finally, texture and 3D data can be acquired in near registration by switching the projector on and off.

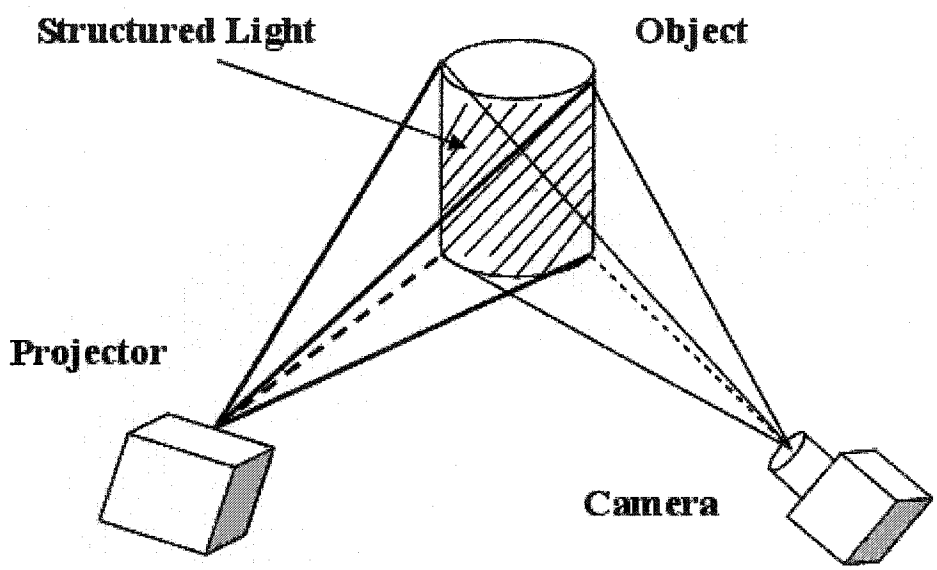


Figure 3: Structured light scanning system

The scanner consists of two common electronic devices: a projector and a digital still camera (Figure 3). The projector is used to project structured light patterns onto the object to be scanned. The digital still camera is used to acquire images of the object under structured lighting. If we know exactly the position of the projector and sensor, pairing the sampled camera pixels with the

corresponding location on the light pattern is sufficient to reconstruct the spatial location of the corresponding point on the object surface. Figure 4 illustrates a human face with projected structured light. The reconstruction method is generally directly related to the characteristics of the pattern adopted. The simplest technique is to shoot a single light plane, which draws a single profile on the scanned surface. Other solutions project multiple planes of light to accelerate the process.

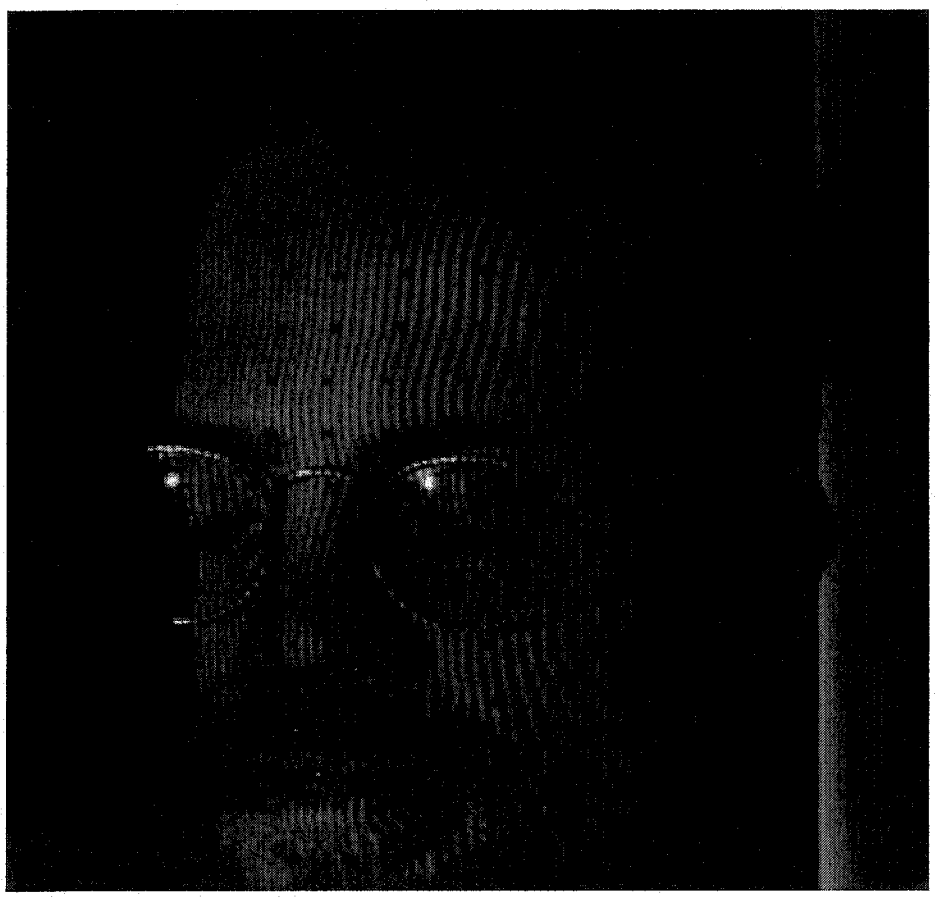


Figure 4: Structured light on a human face [55] (Data source: Courtesy Sen Wang)

A 3D acquisition system based on structured light is presented in [6]. The system was developed to meet the requirement of speed, low cost and resolution. The main disadvantages of a structured light system are the relative bulkiness of the camera-projector head and the limited depth of focus due to optical systems. Proesmans *et al.* [47] demonstrate a good dynamic 3D shape using a slide projector. Other similar systems based on structured light have been developed and tested [41, 49].

2.1.3 Stereoscopy

Stereoscopy, or binocular stereo disparity, can be used to recover 3D geometry. This method uses the geometric relation between stereo images to calculate the depth of the object surface.

Stereo reconstruction is based on the same principle that the human visual system uses for depth recovery. Two images are taken for the same object from slightly different perspectives. The transformations between the two images are used to reconstruct 3D objects. Originally, analogue cameras were used for image capture, followed by a manual evaluation to obtain 3D points from stereoscopic images. The growth of digital technology enabled the development of systems that strongly support this process, both in terms of sensors and of data processing, hence allowing the reconstruction procedure to be made semiautomatic or totally automatic.

A general stereoscopic system requires at least two different views of the scene to be observed. It can be achieved by one moving imaging sensor, or by placing several sensors at different locations. An extended correlation method to

compute 3D shape properties is presented in [9]. The C3D 2020 capture system [1] is a complete scanner developed by the Turing Institute; it produces VRML (Virtual Reality Modeling Language) models using the stereoscopy method.

2.1.4 Model Reconstruction via Feature Modification

All of the above methods focus on how to establish or create a good 3D shape. However, the drawback of these methods is that they only try to recover a 3D shape, but do not provide suitably well-structured information for the model. To get a structure model for medical or plastic surgery purposes, a typical method is to modify an existing generic model with the structured area on the face, such as mouth, nose, and eyes. However, there are many existing approaches to reconstruct a 3D facial mesh from laser-scanned data. An easier way is to create 3D objects from 2D photo information, which can be obtained at a low cost.

Many methods reconstruct a facial model from a few feature points. These feature points are the most characteristic points for recognizing people, whether detected manually or automatically. Other points on the facial model are modified by defined special functions. Lee *et al.* [34] proposed an approach to modify a generic model with some feature points extracted from three head photos (left, front, and right). Texture image is generated by the input image and fit onto the 3D model. The drawback is that extra computation of texture generation is needed, and the results may not be accurate. Simmons *et al.* [53] introduce a method for 3D model creation that "morphs" a canonical anatomically based model. The canonical model is built on top of a layered articulated segment hierarchy and contains a representation of the creature's

skeleton, muscles, and skin. This approach is complicated for the human head model since it requires "morphing" in a bone-muscle-skin hierarchy. Akimoto *et al.* [4], Kurihara and Arai [30], Ip and Yin [23], and Lee *et al.* [33] use interactive semiautomatic or automatic methods to detect feature points and modify a generic model. The drawback of these methods is that there are too few points to guarantee on appropriate shape from a very different generic head or accurate texture fitting.

2.2 Facial Deformation

Because the human face plays an important role in identification, communication, realistic construction, and animation, the modification and deformation of facial models are explored extensively by researchers. The ultimate goal is to describe exactly the human face, movements, and shape to satisfy both structural and functional aspects. However, due to the complexity of the human face, the problem is further compounded with complex interactions and the deformation of different facial features. We discuss major facial deformation techniques and evaluations in this section.

2.2.1 Facial Deformation Techniques

Surface Interpolation with Radial Basis Functions. Surface interpolation is a simple method to deform a facial model [58]. Given a 3D human facial mesh, the positions of the vertex of the mesh is multiplied by a transformation matrix determined by the original and new position of the vertex. Traditionally, we

solve the problem of approximating or interpolating a continuous multivariate function $f(\vec{x})$ by an approximation function $F(\vec{x},\vec{c})$, with appropriate parameter set $\vec{c}$, where $\vec{x}$ and $\vec{c}$ are real vectors ($\vec{x} = x_1 + x_2 + \cdots + x_n$ and $\vec{c} = c_1 + c_2 + \cdots + c_n$). Given an approximation function F and a set of training examples, we will be able to obtain the best approximation of f. The Radial Basis Function (RBF) [46] is often chosen as an approximation function due to its power to handle irregular sets of data in multi-dimensional spaces when approximating 3D smooth surfaces. The name "radial basis function" comes from the radial symmetry of the distance parameters. Some examples of RBFs are the Gaussian function $h(r) = e^{-\left(\frac{r}{c}\right)^2}$, multi-quadrics $h(r) = \sqrt{r^2 + c^2}$, and thin plate splines $h(r) = r^2 \log r$. (Note: a surface defined by an algebraic equation of degree two is called a quadric. Spheres, circular cylinders, and circular cones are quadrics)

Noh *et al.* [39] present an approach to create deformations of polygonal models using RBFs to produce localized real-time deformations. They believe that a face mesh geometry can be locally deformed by a geometry deformation element (GDE). A GDE is the smallest deformation unit defined on the surface of the face. A GDE consists of a control point, the region of influence around the control point, the fixed points that lie in the influence region, and an underlying RBF system (Figure 5). The radial basis function approximation equation takes the form:

$$F(\vec{x}) = \sum_{i=1}^{N} c_i h\left(\left\|\vec{x} - \vec{x}_i\right\|\right)$$

The radial basis function h is multi-quadrics $h(r) = \sqrt{r^2 + c^2}$. An RBF system deforms the facial mesh, based upon the motions of all feature points. The mappings between initial positions and new positions of the feature points are described in terms of the vector coefficients. The rest of the nodes in the influence region are transformed based upon the coefficients.

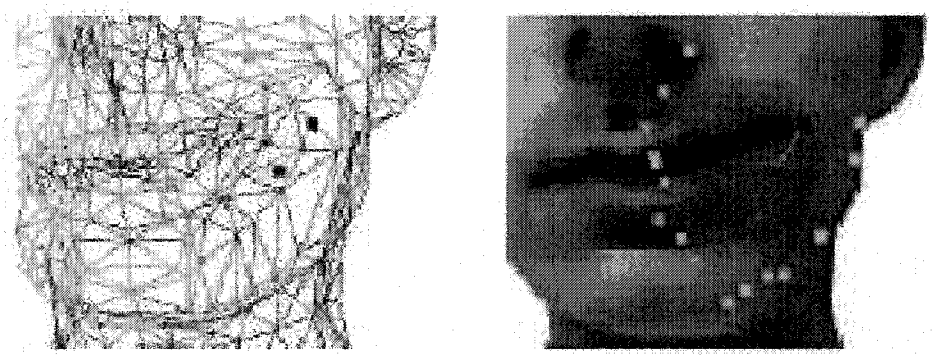


Figure 5: Geometry Deformation Element on the facial surface [39] (Data source: Courtesy Junyong Noh *et al.*)

Lavagetto *et al.* [31] proposed a method to deform facial geometry for MPEG-4. Their method is also based on the use of Radial Basis Functions (RBF) for smooth surface interpolation. To eliminate the limitation of not including an affine transformation, they added a polynomial term to the interpolation function $F(x)$. A remarkable positive characteristic of this method is its capability to work correctly when only a subset of feature points is available. To satisfy the specific needs, monotone functions decreasing to zero, such as Gaussian and

inverse multi-quadrics RBFs are used. There still exists a significant disadvantage with such an approach, due to the global effect produced by the interpolation, which makes it difficult to produce local details. However, this effect can be reduced by forcing the feature points to influence only a limited region of mesh (*i.e.*, interpolation functions associated with the mouth feature points are forced not to interact with those associated with nose and eyes).

MPEG-4 Facial Deformation. MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). The new MPEG-4 standard includes support not only for video and audio, but also for synthetic interactive graphics — in particular, representations of the human face and body. The definition parameters of the human face allow detailed definition of face shape, size, and texture. They represent a complete set of basic facial actions, and therefore support most natural facial expressions. For example, the lips are particularly well defined, and it is possible for people to precisely define both the inner and outer lip contour. In order to represent the Facial Animation Parameters (FAPs) on any single facial model in a consistent way, all facial movements are expressed in the form of Facial Animation Parameter Units (FAPU). According to the MPEG-4 standard, the MPEG-4 decoder is capable of modifying a generic facial model through FAPs, while doing facial animation. When it is time to modify the generic shape and appearance to fit a particular person's face, Facial Definition Parameters are necessary. In contrast to the animation function of FAPs, the Facial Definition Parameters (FDPs) are used to personalize the

generic facial model to a particular face. Normally, the FDPs contain the following:

- 3D Feature Points
- Texture Coordinates for Feature Points (optional)
- Face Scene Graph (optional)
- Face Animation Table (FAT) (optional)

The key component of FDPs is the feature points, and they must always be supplied; the remaining parameters are optional. Feature Points of FDPs are illustrated in Figure 6. In summary, FAPs are used to control facial animation at extremely low bit rates, while FDPs are used to define the shape of the face by deforming a generic facial model, or by supplying a substitute model.

Escher *et al.* [14] present a method to perform facial deformation for MPEG-4. The authors propose an algorithm interpreting the FDPs to handle the deformation of the generic facial model. The implementation attempts to make the generic model fit input parameters. The input includes feature points and texture coordinates. A cylindrical projection is applied on the generic face and the missing points other than the feature points are interpolated. Then the Dirchlet Free Form Deformation (DFFD) method is used to deform the generic facial model. The texture fitting technique used is cylindrical projection matching, and the Barycentric coordinate is utilized in order to interpolate the non-feature points. The method described in this paper is well adapted to the actual definitions of the MPEG-4 SNHC Face Definition Parameters.

A similar approach is presented by Hong *et al.* [19]. They developed a face modeling and animation system, which uses a generic face mesh model and the linear key-frame technique to animate the face model. A target facial model is acquired by the Cyberware® scanner, and the user can utilize this system to interactively customize the generic model by selecting some facial feature points. Facial texture is mapped onto the deformed model to achieve a realistic appearance.

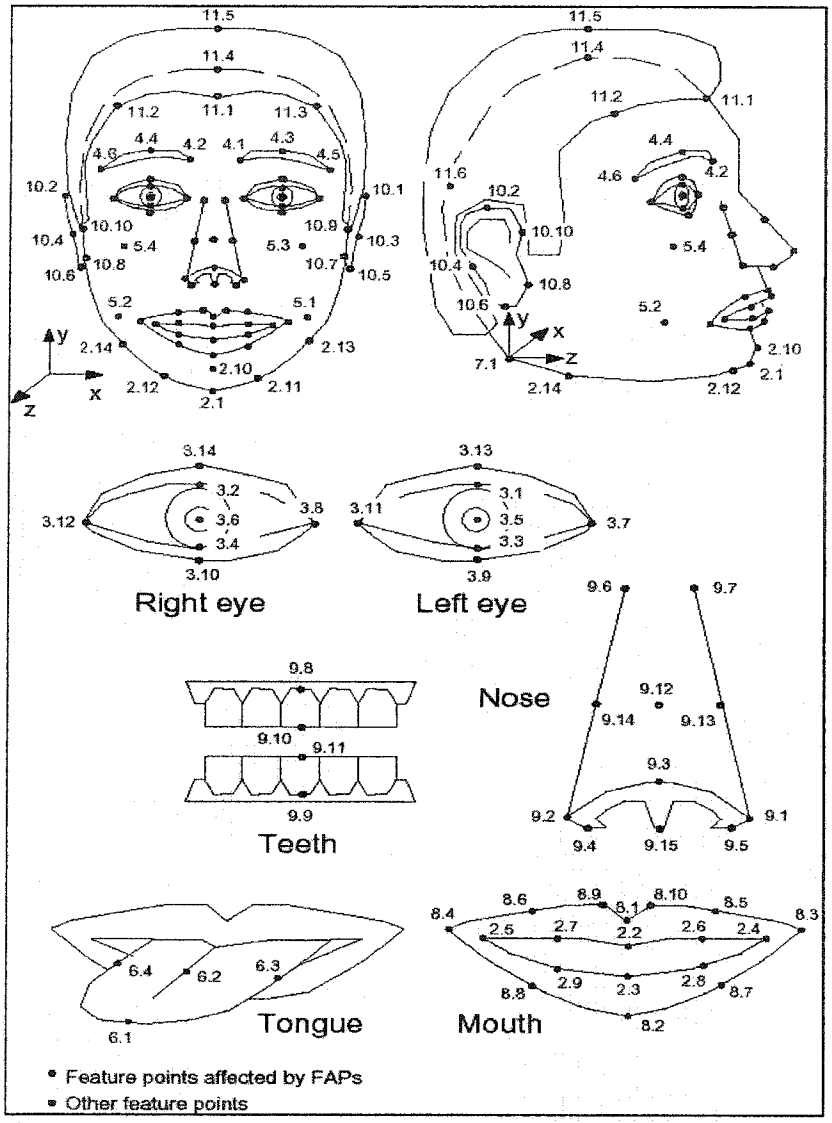


Figure 6: Description of the feature points in the MPEG-4 standard [31]

(Data source: Courtesy Fabio Lavagetto *et al.*)

Facial Action Coding System (FACS). The facial Action Coding System (FACS) is the most widely used and versatile method for measuring and describing facial behaviors. Paul Ekman and W.V. Friesen [13] developed the

ordinal FACS by determining how the contraction of each facial muscle changes the appearance of the face. They studied videos of facial movement in order to identify the specific changes that occurred with muscular contractions then determined how best to deform from one to another. Their goal was to create a reliable means for skilled human scorers to determine the category or categories in which to fit each facial behavior. FACS measurement units are Action Units (AUs), which represent single muscles or clusters of muscles. AUs can present facial changes more precisely and efficiently than muscles. Some sample actions are described by FACS, for example, Inner Brow Raiser, Outer Brow Raiser, and Lid Tightener. Table 1 illuminates the Action Units and their related muscles around the mouth. Each basic action (AU) is a minimal action in the sense that it cannot be broken up into smaller actions. An FACS coder "dissects" an observed expression, decomposing it into the specific AUs that produced the movement. The scores for a facial expression consist of the list of AUs that produced it.

Description	**Facial Muscle**
Upper Lip Raiser	Levator labii superioris
Nasolabial Deepener	Zygomaticus minor
Lip Corner Puller	Zygomaticus major
Cheek Puffer	Levator anguli oris
Dimpler	Buccinator
Lip Corner Depressor	Depressor anguli oris
Lower Lip Depressor	Depressor labii inferioris
Chin Raiser	Mentalis
Lip Puckerer	Incisivii labii superioris and Incisivii labii inferioris
Lip stretcher	Risorius w/ platysma
Lip Funneler	Orbicularis oris
Lip Tightener	Orbicularis oris
Lip Pressor	Orbicularis oris
Jaw Drop	Masseter, relaxed Temporalis and internal Pterygoid

Mouth Stretch	Pterygoids, Digastric
Lip Suck	Orbicularis oris

Table 1: Action Units around the mouth

Platt and Badler [44] present a facial animation system based on the Facial Action Coding System (FACS). They constructed the representation of the face in a bottom-up fashion by starting at the lowest level (*i.e.*, a point on the skin), and gradually building successively higher levels until a sufficiently high level (such as the AU) was obtained. The simplest structure is the point, a 3D location on either the skin or the bone. Arcs connect the points and contain information about elasticity between the points. The simplest structure for organizing the force is the muscle fiber. A muscle fiber consists of a muscle point and the magnitude of the force to be applied at that point. The highest level AU consists of a list of muscles and their relative magnitudes. In this paper, the authors described the basic techniques used to apply a simple muscle pull to the current facial structure. A simulation was implemented to achieve facial animations. The FACS theory is well utilized in Platt and Badler's paper.

Free Form Deformation. Free Form Deformation is a technique for deforming solid geometric models in a free form manner [22, 51]. It can deform surface primitives of any type or degree: planes, quadrics, parametric surface patches, or implicitly defined surfaces. FFD provides a mapping from R^3 to R^3 through a trivariate tensor Bernstein polynomial. The FFD method deforms an object by first assigning to each of its points within the deformation lattice a set of local coordinates. The local coordinate system is defined by a parallelepiped-

shaped lattice of control points with axes defined by the orthogonal vectors s, t, and u, hence setting up the mapping from xyz-coordinates to *stu*-coordinates. Any point X in the (s,t,u) coordinates has the following format

$$X = X_0 + sS + tT + uU$$

A grid of control points P_{ijk} is imposed on the parallelepiped, such that

$$P_{ijk} = X_0 + \frac{i}{l}S + \frac{j}{m}T + \frac{k}{n}U$$

The control points form $i+1$ planes in the S direction, $m+1$ planes in the T direction, and $t+1$ planes in the U direction. The deformed position X_{ffd} of an arbitrary point X can be denoted by a trivariate Bernstein polynomial as:

$$X_{ffd} = \sum_{i=0}^{l}\binom{l}{i}(1-s)^{l-i}s^i\left[\sum_{j=0}^{m}\binom{m}{j}(1-t)^{m-j}t^j\left[\sum_{k=0}^{n}\binom{n}{k}(l-u)^{n-k}u^k p_{ijk}\right]\right]$$

When the control points are moved, the new position of an object point is then determined by a weighted sum of the control points. The weights are functions of the local coordinates originally assigned to the point. Hence, the position change of the control points changes the position of the object point. Generally, the deformation function is a trivariate B-spline tensor product or a Bemstein polynomial. (See Figure 7)

As extensions to the basic FFD, many researchers proposed improved Free Form Deformation methods. Kalra *et al.* [24] presented the option of

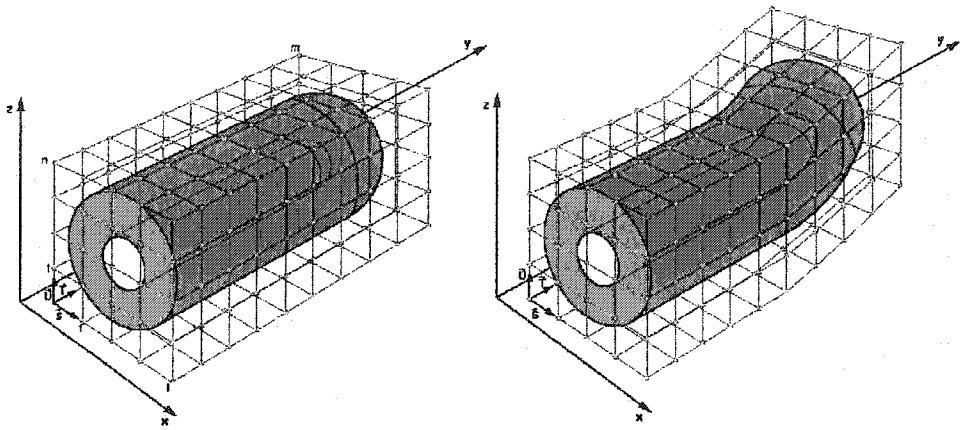


Figure 7: Free form deformation

including a rational basis function in the formulation of deformation. The advantage of using rational FFDs is that it provides one more degree of freedom when manipulating the deformation, by changing the weights at the control points. If the weights at control points are unity, the deformation changes to basic FFD. Another example is the t-FFD approach presented by Kobayashi and Ootsubo [25]. The 't' in t-FFD stands for triangle because they utilize a triangular control mesh instead of volumetric lattice to parameterize the control points. The control mesh consists of a set of triangles with arbitrary topology and geometry. The original shape is deformed by modifying the control mesh. Each point of the model is parameterized based on a local coordinate system attached to each control mesh. The weight for mapping is calculated by an effect function,

using the distance between a point and a triangle. This method is efficient for sophisticated models with many triangulated fragments.

Muscle-based Deformation. In the Facial Action Coding System (FACS) [13], a technique for the extraction of facial parameters is presented. Action Units (AU) are the basic elements for describing facial deformation. Individual muscles or small groups of muscles are defined as Action Units which distort the skin tissue. Computer models of muscles [56] for the human face are constructed to describe the facial model and assist in making facial deformation more efficient. Two types of muscle models are proposed: linear and parallel. Linear muscle has a static bony attachment at one end, while the other end is embedded in the soft tissue of the skin. When the muscle moves, a zone of influence will be affected and moved along, and the points on the skin are affected by the muscle vector contraction. The parameter changes of the muscle model directly lead to the deformation of the facial model. Usually, simple deformation is controlled by single muscle model, and complex deformation could be achieved via the interaction of a group of muscles.

Lee *et al.* use physics-based technique in their realistic modeling for facial animation [35]. They automatically create a dynamic model of facial tissue, estimate a skull surface, and insert the major muscles of facial expression into the model. The muscles spread out below the facial tissue. They also designed a new, synthetic tissue model based on the structure of real skin (Figure 8). There are five distinct layers in the model, which is composed of triangular prism elements. A discrete deformable model is introduced in this paper. This model

has a node-spring-node structure, which consists of the nodal mass m_i, position $x_i(t) = [x_i(t), y_i(t), z_i(t)]^{'}$, velocity $v_i = dx_i / dt$, acceleration $a_i = d^2 x_i / dt^2$, and net nodal forces $f_i^n(t)$. In order to simulate different realistic facial deformations, several muscle force models are proposed: linear, piecewise linear, volume preservation, and skull penetration constraint. In addition to the skin/muscle/skull model, geometric models are also completed for other organs on the face, including eyes, eyelids, and teeth. With the final model, they are able to adapt an individual's face to a generic mesh, and animate/deform the constructed face via muscle actuations.

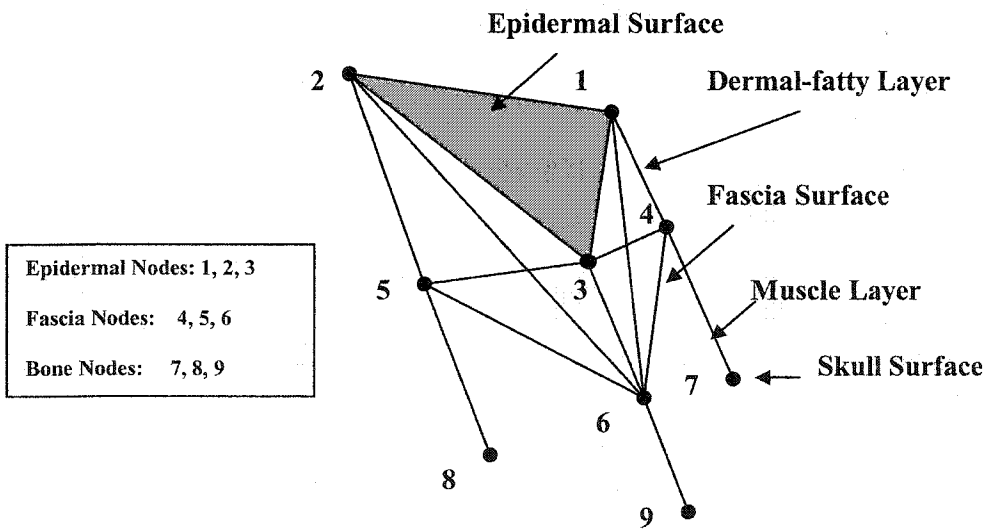


Figure 8: Triangular skin tissue prism element [35]

Finite Element Method. The finite element method is a method for solving an equation by approximating continuous quantities as a set of quantities at discrete points, often regularly spaced into a so-called grid or mesh. It can be

adapted to problems of great complexity and unusual geometry. Celniker and Gossard [7] first used finite element methods to generate primitives that build continuous deformable shapes, designed to support a new free-form modeling paradigm. A deformable shape is calculated by finding a minimum of an energy function. The shape's internal deformation energy E_{def} is determined by its natural resistance against bending and stretching. Let α and β denote the tensors of material, and Ω the parameter domain. The surface energy can then be computed as follow:

$$E_{def} = \int_{\Omega} (\alpha \cdot stretching + \beta \cdot bending) d\Omega$$

$$\alpha = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{bmatrix}, \beta = \begin{bmatrix} \beta_{11} & & \\ & \beta_{22} & \\ & & \beta_{33} \end{bmatrix}$$

If $f(u,v)$ is the external force applied to a surface $w(u,v)$, then let w_u, w_{uu}, w_v, w_{vv} be the first and second order derivatives in the principal directions. The deformation energy E_{def} can be reformulated to

$$E_{def} = \int_{\Omega} \begin{pmatrix} \alpha_{11} w_u^2 + \alpha_{12} w_u w_v + \alpha_{22} w_v^2 + \\ \beta_{11} w_{uu}^2 + 2\beta_{12} w_{uv}^2 + \beta_{22} w_{vv}^2 \end{pmatrix} - 2fw \, d\Omega$$

The main idea of the finite element method is to calculate an approximation of the unknown shape $w(u,v)$ by dividing $w(u,v)$ into patches $w_p(u,v)$, and to use expansions with weighted sums of bases Φ_t, which form the finite elements:

$$w_p(u,v) \approx \sum_t a_t \Phi_t$$

Here $(u, v) \in \Omega_p$ is the local parameter space of a surface patch p. In their paper, Celniker and Gossard develop finite elements for both curves and surfaces. The properties of these geometric primitives have been engineered to support interactive free-form deformation of 3D models.

The finite element method is applied to facial deformation, especially in simulating facial surgery [27]. Facial surface and soft tissue data are both acquired from photogrammetric and CT scans of the individual. After initial data processing, reconstruction, and registration, a finite element model of the facial surface and soft tissue is provided, which is based on triangular finite elements. All the interactive procedures are performed under the guidance of the modeling system which is based on a FEM solver. Koch and Bosshard [26] describe a prototype of a facial expression editor using finite elements. The FEM engine performs the central computation unit of the presented emotion editor.

2.2.2 Evaluation

Direct manipulation of the model. We now discuss the various facial techniques have been briefly reviewed. The triangular or polygon mesh is currently the most popular representation of 3D model. Some of the above methods do not provide direct manipulation on the 3D model mesh, hence making them difficult to implement. MPEG-4 facial deformation needs the context of the MPEG-4 standard and to restrict the facial model in an MPEG-4 environment, or must first be adapted to the MPEG-4 standard. To use the facial action coding system (FACS) for facial deformation, we will have to construct a

relationship between the 3D model and FACS since Action Units (AUs) handle all the facial movement. In the case of muscle-based deformation, one has to build or import muscle models before directly controlling the 3D mesh's deformation. On the other hand, other approaches, *e.g.*, surface interpolation and free-form deformation (FFD), are capable of directly operating on 3D mesh triangles via defined functions or formulas.

The speed issue. Due to the complex structure of the human facial model, the efficiency of a deforming algorithm is one of the most important issues to be considered. Basically, two aspects are commonly studied: user's input and processing speed. Some methods, such as surface interpolation, require the user to select feature points before performing facial deformation. With a limited number of feature points we may still change the positions of other points around those feature points. A similar concept can be found in the MPEG-4 facial deformation technique since it needs feature points and texture coordinates as user's input. Some other methods, *e.g.*, free-form deformation (FFD), can only change the position of points, thus leading to some tedious work if a heavy modification on the facial model is desired. The effect of a difference in workload on user's input is usually obvious; people tend to prefer the methods which require less input. The processing speed of deformation techniques should also be given considerable attention. Most of the methods involve linear computation and approximation. The more detailed the facial modification required, the longer the processing time needed.

2.3 Facial Texturing

Texture mapping is a method in computer graphics to enhance the richness of a 3D object view [18]. Usually, 3D objects consist of a set of polygons defining their surfaces. The more polygons, the more detailed the object could be. In fact, it will be very slow and inefficient to represent a smooth model with a large number of polygons or triangles. Texture mapping is a relatively efficient way to create the realistically-looking complex objects, without changing their topologies. Texture mapping means the mapping of textures onto a 3D surface. The texture image (2D) is mapped onto a surface in 3D object space, which is then mapped to the destination image by the viewing platform. Each point in the space has to be associated with an element on the texture image. This defines the area of texture image mapped onto each polygon of the object. An automatic and straightforward technique is to project the image directly onto the 3D object. If there is not enough of the image to cover the object, the image can be projected two or more times. Many existing image techniques may be used, such as scaling, rotation, and translation.

The main difficulty with the simple method is that it is not suitable for objects with many specific features on their topology, *i.e.*, human facial model. The texture image may have the same features, and the user would like to match the features between object and image. Typical features of the face include eyes, nose, and mouth. This matching is automatically done if the 3D model and texture image come from 3D scanners, *e.g.*, Zoomage® 3D scanner [3] or

Cyberware® scanners [2] because the 3D model and image are both obtained from the same real object. However, this simple projection will not work if there are many differences between the image and 3D object. For example, for a generic facial model, there is no image which can be exactly matched. In such cases, approaches have been proposed in order to map texture image onto general 3D models or a facial model, and match their features precisely. Selected methods are outlined below.

2.3.1 Facial Texturing Techniques

Sannier and Thalmann [50] propose a texture mapping method for direct texture-fitting on the face that is capable of taking into account its important topological features. They developed an interface for direct texture fitting through real-time adjustment. The user specifies facial features by putting marks on a 3D facial model so that features of the texture image fall onto the right position on the object. To determine the texture coordinate for each 3D vertex, the texture coordinates for those marked vertices are known, and the texture coordinates for other vertices are deduced. The steps are as follows: First, the 3D mesh is projected onto the image so that the marked feature points are located on the two-dimensional image. Delaunay triangulation [52] is used for all the marked vertices. For all the projected vertices, it is determined whether they are inside or outside the defined Delaunay region. Next, the polygons of the object belonging to this region are determined as long as the vertices inside the region are known. Finally, the user can interactively set the marked vertices to their correct positions on the texture image. For the marked vertices, the texture coordinates

are figured out by referring to their positions on the picture. For all the other vertices, their new position can be calculated using their Barycentric coordinates. In real time, if a marked point is moved, all the Delaunay triangles using this vertex will change. Accordingly, the positions of points inside the Delaunay triangles must be calculated again. Their texture coordinates are then updated, and a new texturing result is obtained. Our facial texture mapping scheme is partly based on this approach.

Eckstein *et al.* [12] try to solve the problem of texture mapping with hard constraints, which requires precise alignment between feature points in the 3D mesh and specific values on the texture image. The objective is similar to that of our texture mapping phase. Assigning texture coordinates to a 3D mesh can be regarded as a parameterization of the mesh surface, where each 3D vertex is assigned a 2D parameter value. In this method, 3D mesh must be embedded onto the plane with hard constraints first. The authors propose two algorithms for this procedure: constrained embedding and constrained simplification. Since the constraints might have a global effect on the mapping result, and the planar triangulation could also distort the result significantly, a relaxation procedure is employed during the first algorithm to minimize distortions. 3D facial texture mapping can obviously be achieved by their algorithm. Figure 9 illustrates the results and comparisons.

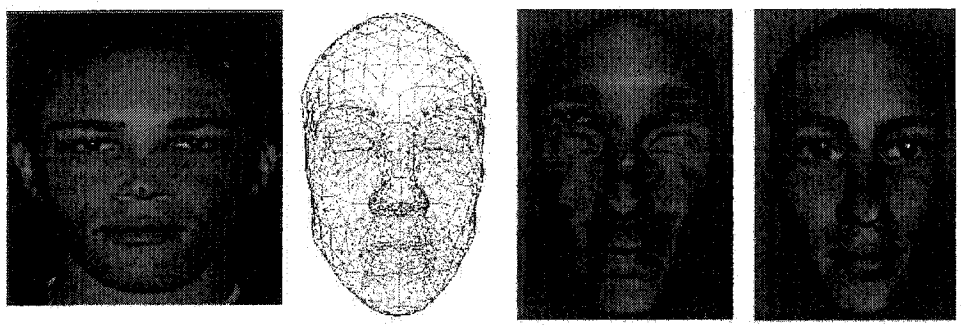


Figure 9: Texture mapping with hard constraints [12] (Data source: Courtesy Ilya Eckstein *et al.*)

Texture mapping using surface flattening [60] is also related to facial texturing. After applying classical scaling [29] to the target 3D mesh, a 2D flattened version of the surface is obtained. Now we have the mapping of every point on the 3D surface to its corresponding 2D flattened point. Given a flat texture image, we can easily map each point from the 2D flattened map to a point on the texture plane. Next, the texture is mapped back to the surface. This technique is straightforward: For each vertex P on the surface, first find the corresponding 2D point in the flattened map; then perform a translation between the 2D coordinates to the texture image coordinates; and finally, use the point's color as texture. Texture mapping can be easily performed after a 3D facial model is flattened by this approach. Figure 10 is an example of face flattening including a 3D facial model with texturing and the flattened texture image of the face.

Figure 10: Example of face flattening [60] (Data source: Courtesy Gil Zigelman *et al.*)

A model fitting method is presented by Pighin *et al.* in their paper [43]. This paper describes an approach for automatically producing 3D facial expressions with a high degree of realism. Their process consists of several steps. First, multiple views of a human face with given expressions are captured, using a camera at different positions. Next, these photographs are digitized and some initial corresponding points on the face are manually marked in the different views (typically, corners of the eyes and mouth, tip of the nose). These points are used to automatically compute the camera parameters (position, focal length, etc.) corresponding to each photograph, as well as the 3D positions of each marked point. The 3D positions are then used to deform a generic 3D face mesh to fit the face of the particular human subject. More marked points are needed in this stage to refine the fit. Finally, one or more texture maps are extracted for the 3D model

from the photos. Hence, the original images can be used to perform view-dependent texture mapping. To produce facial animations, this process is repeated with several different facial expressions, and interpolations are made between two or more different 3D models constructed by this means, at the same time blending the textures at the same time. Since all the 3D models are constructed from the same generic one, there exists a natural correspondence between all points for performing the morph. Thus, the animation can be produced entirely automatically once the different facial models have been constructed, without having to specify correspondence between any of the expressions.

Two texture mapping techniques are introduced in [43]: view-independent texture mapping and view-dependent texture mapping. View-independent texture mapping supports rapid display of the texture face model from any viewpoint, which requires combining several individual photographs into a single texture map. This texture map is constructed on a virtual cylinder enclosing the facial model. The mapping between the 3D mesh and 2D texture image is based on a cylindrical projection. The disadvantage of view-independent texture mapping is that it involves blending together resampled images of the original photographs of the face. The texture mapping result is slightly blurry because of the registration error from the resampling. In order to solve this problem, view-dependent texture mapping is proposed for texturing according to the current view, without blending together several texture images. In view-dependent texture mapping, the facial model is rendered several times, using different

texture images each time; the results are then blended. View-dependent texture mapping has some advantages over view-independent texture mapping. First, it is able to produce more detailed results since it preserves the original texture image without blending. Second, view-independent texture mapping cannot cover some parts of the facial model, whereas this problem does not arise in view-dependent texture mapping if the mesh geometry properly matches the images. However, the drawbacks include slow rendering speed, and the results are more sensitive to the lighting conditions of original photographs.

In addition to the methods that directly use the image from 3D scanners [2, 3], there are others which make use of several photos of a human face. The texture mapping method of Lee *et al.* [34] is designed to generate a texture image from three photos (left side, front, and right side). The boundaries of each of the two photos are detected by the color information or feature points on the face; then, cylindrical projection of each photo is processed. Every two photos are cropped at a certain position, so that the range of the resulting texture image is 360°. The second phase involves mapping a 2D image onto a 3D shape. First, a cylindrical projection is applied to all the points on the 3D head surface. Extra points are added to make a convex hull containing all the points, so that a coordinate of each point is located on the image. Then Voronoi triangulation on the feature points and extra points is processed, and the corresponding Barycentric coordinates of every point in a Voronoi triangle are calculated. The texture coordinates of points on the texture image can be obtained using the texture coordinates of control points and extra points, and their Barycentric

coordinates. Figure 11 illustrates the texture mapping process. This approach is very similar to the method in [50].

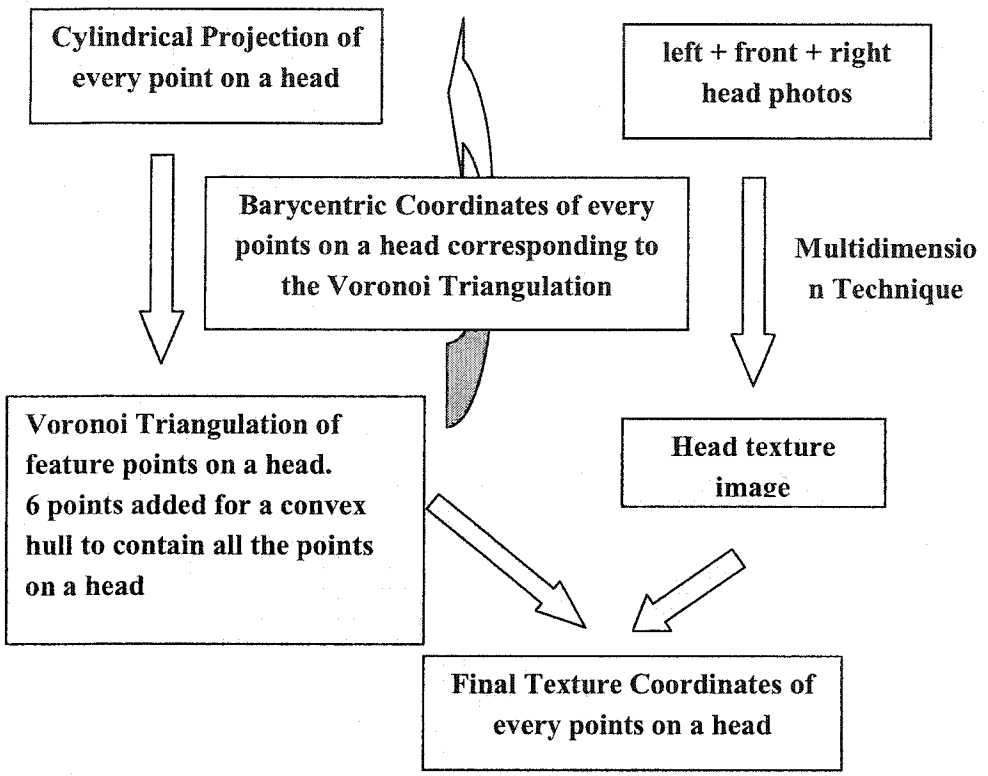


Figure 11: Texture mapping process [34]

2.3.2 Conclusion

Basically, the above facial texturing techniques fall into two categories. The first utilizes cylindrical projections and uses a triangulation approach to surround feature points in a 2D coordinate system. Then, Barycentric coordinates of all other points are calculated based on the triangulation. The texture coordinates of all points, other than the feature points, can be computed by the Barycentric

coordinates and feature points' texture coordinates. Typical examples are [34, 50]. The advantage of this type of method is that the concept is simple and straightforward. The drawback is the relatively heavy computation cost. [12, 60] present another type of approach, which first flattens the 3D surface mesh, then performs texture embedding onto the flattened 2D mesh. The embedding procedure varies according to the different methods. For example, two algorithms are proposed in [12], and Multi-Dimension Scaling (MDS) is used for surface flattening in [60]. These are more efficient because the 3D mesh is flattened with the texture image embedded. The texture coordinates of each point can therefore be directly retrieved without complicated computation.

Generally, texture image generation is not mandatory in all texturing methods since human head texture could be acquired by 3D scanners. Texture generation is required in [34], while most of other approaches directly manipulate the input texture image to achieve realism the of facial model. Because the main goal of our work is to deform the facial model using the feedback of texture mapping, we will focus more on model deformation, and target the easy implementation of texture mapping in real time (the first category mentioned previously). For texture acquisition, we may benefit from the convenience of the Zoomage® Rotating Arm Scanner since a high resolution full, 360 degree view human head texture image can be obtained.

Chapter 3
Our Approach

3.1 Overview

In order to realize the objectives, we start from a generic facial model. To make the model fit a specific person's face, the individual's facial image needs to be captured. For the sake of convenience, we choose a generic human head model, which has 2954 vertices and 3118 polygons (Figure 12). The strongest features of the facial model, such as mouth and eyes, will be modified to match a person's real face as precisely as possible.

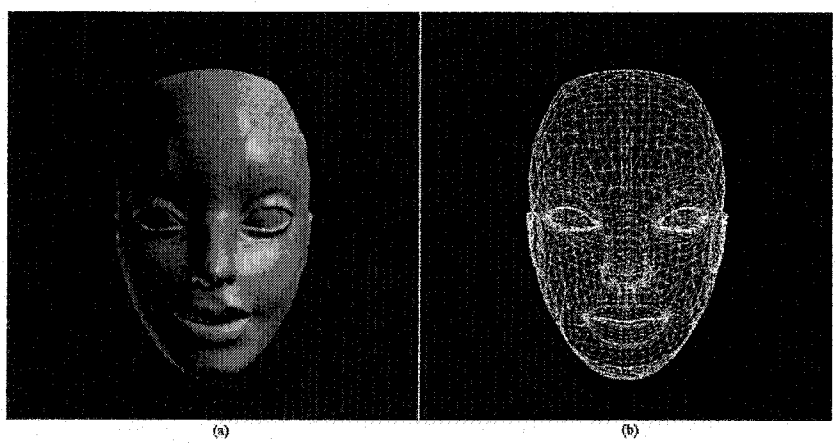


Figure 12: Generic human head model: (a) Solid surface, (b) Wireframe.

In our experiments, we use the visual feedback of texture mapping on the facial model to test whether the model has been deformed to fit a person's real face. The first step is to obtain the facial texture image, using the Zoomage® Rotating Arm Scanner located in our lab. (See Figure 2 for the real scanner, and Figure 13 for its structure). This scanner features a scanning head suspended from a central pivot on a heavy-duty frame. The design permits the scanner to rotate around a person allowing the subject to sit comfortably during the scanning process. The mounted high-resolution camera is capable of acquiring a 360° texture image (see Figure 14 for a sample image).

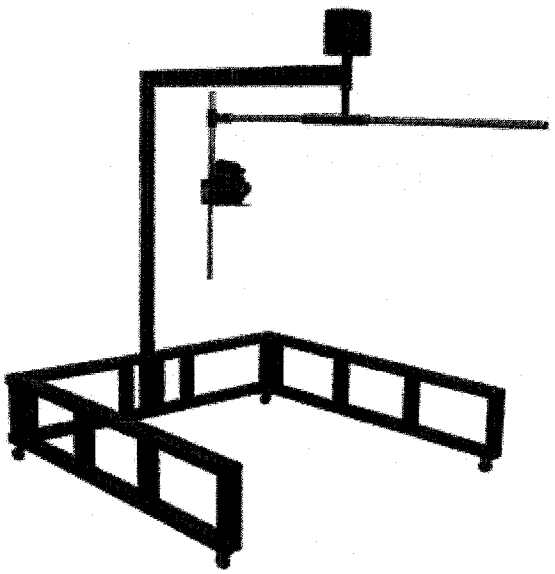


Figure 13: Zoomage® rotating arm scanner

An overview of the procedure is shown in Figure 14. There are three steps: texture acquisition, texture mapping, and model reconstruction. First, the texture image must be obtained before any further steps are taken. The texture mapping

step is divided into two phases: feature points on both the texture and the model are picked out and associated, one by one. Hence, we have a texture area which will be mapped onto the desired feature area of the 3D model in Phase 2. After the correspondence between texture and model is determined, texture mapping is performed. As the result may be distorted or unsatisfactory because of a poor match between an individual texture and a generic model, we will reconstruct the 3D model in order to achieve better matching and a reasonable texture appearance on the model. After this process is complete, the generic 3D model is modified to include the characteristic features, and is finally ready for further processing.

3.2 Facial Model Reconstruction

The ultimate goal of our work is to modify a generic model with the feedback of texture mapping on the 3D model. We consider the points on the contour of mouth, eyes, and other organs on the face, as the feature points.

3.2.1 3D Generic Model and Texture Preparation

The generic model (Figure 12) is created using a triangular mesh without texture mapping on the surface; there are many more polygons (triangles) on the feature areas (eyes, mouth, nose, ears, etc.) than on other parts (*e.g.*, forehead and cheeks). We scale the generic model using 3D scan data from a face. The advantage of using a generic model is that detailed mesh representation in

features areas is possible without the need for accurate 3D scanning with occlusion avoidance.

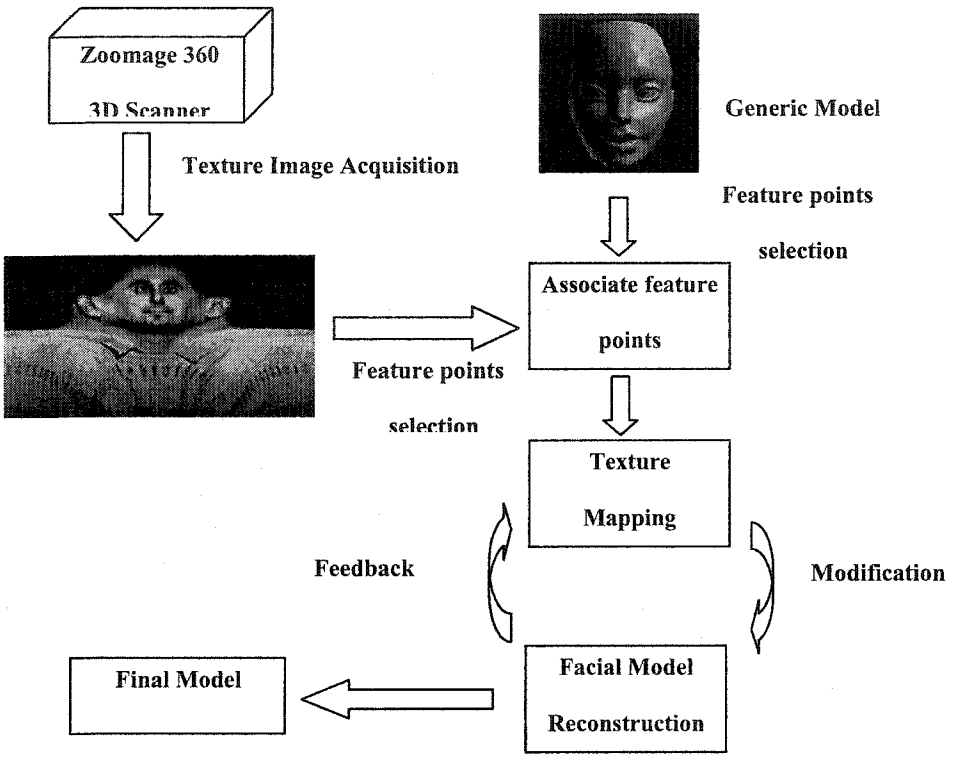


Figure 14: System overview.

For facial texture images, it is quite simple to retrieve the image from the scanned head image (Figure 14). Usually, a typical 360 degree scan of 6.6 mega-pixels takes approximately 125 seconds. It can be performed much faster if we choose to scan only the front part of the head, *i.e.*, 180 degrees. One advantage of using a 360 degree image is that there is no need to generate texture image from several photos at different viewpoints [34, 57]

3.2.2 Feature Point Selection

In order to perform texture mapping on an interesting area, we need to specify some feature points on both the texture and the object. Our method is capable of mapping different texture of interest (TOI) patches onto the same 3D mesh area (eyes, lips, etc.). We also want to interactively define the feature points needed for the texture. We specify the points by putting marks on the texture image so that a featured texture area is obtained. An effective method is to select some control points on the texture image and mark their associated points on the 3D object. Thus, the texture coordinates of the control vertices in 3D space can be interactively determined, and we may deduce texture coordinates for other vertices in the region of interest from the information obtained by the control points.

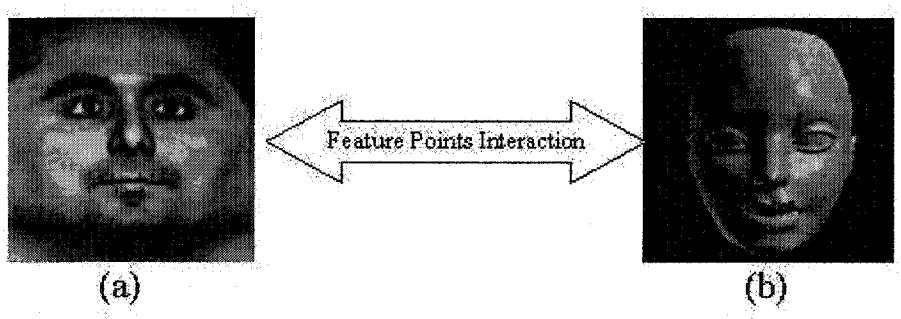


Figure 15: Feature points selection and association on the mouth: (a) Texture image, (b) 3D facial model.

First, we select feature points by clicking on the 3D model. The nearest vertex to a selected position on the mesh is determined and chosen as a feature

point. We use this method to locate all the control points along the contour of an organ on the 3D model. Next, we mark control points on the texture image and extract an area as the Texture of Interest (TOI). Finally, the control points on both the 3D model and TOI are associated, one at a time. After this, we are ready for the texture mapping step. Figure 15 illustrates the process of feature points selection.

3.2.3 Modifying a Generic Face Model

We have reviewed several techniques for deforming/modifying a facial model in Section 2.2.1. A discussion and brief comparison can be found in Section 2.2.2. Since our method is based on a 3D curve template, we will first briefly review curve representations. Our own approach will be presented later.

Curve Representation and Model Deformation

Two of the most commonly used methods for curve representation are triangular mesh and the design tool NURBS. Triangular mesh [5, 20, 21, 32, 54] is easy to construct and manipulate, but requires extensive computation and post-processing, such as mesh simplification [45]. The B-spline surface model is widely used for modeling 3D objects [11, 16, 28, 36]. The non-uniform rational B-spline (NURBS) is adopted as a standard representation of curves and surfaces [42]. Park *et al.* [40] propose an approach for constructing a 3D model and performing parameterization automatically. However, since the definition of the NURBS curve requires control points, knot vectors, and weights, we still have to place many parameters to determine a 3D curve. Therefore, in order to have a simple yet efficient solution, we use parameterized differentiable curves [10]

instead. The advantage of parameterized differentiable curves is that fewer parameters are required, yet the 3D shape is fairly well preserved. The contours of facial organs (mouth, eyes, etc.) can be represented using our method by the combination of several parameterized curves. To reshape the target facial organ, only the curve parameters have to be modified. Therefore our approach tends to be efficient and straightforward.

In Section 3.2.2, the user determines the contour of a facial feature, and then a template is developed to represent the feature contour for further mesh editing. In our previous work, we proposed several 2D templates for facial features [59]. Our method here is based on extending the approach in [59] to represent 3D templates. By using a mathematical representation of each facial feature we can control/modify its 3D shape without losing its intrinsic geometric properties. By contrast, using generic NURBS curves for all features can distort specific shapes of lips or eyes, for example, during editing.

Deformable 3D Curve Template

In [59], the mouth features are represented by an open-mouth template and a closed-mouth template in 2D (Figure 16). Each of the templates consists of several parabolic curves and is defined as:

$$p_i = h_i\left(1-\left(\frac{q}{L_m}\right)^2\right) \tag{1}$$

Where, the parameter h_i describes the height of the upper and lower lips. The width of the mouth L_M is calculated as the distance between the two corners of

the mouth. To present an open-mouth feature, four deformable curves are needed, with five parameters $h_i, i = 1,2,\ldots,4$, and the width of the mouth L_M. A similar idea is applied to the closed mouth.

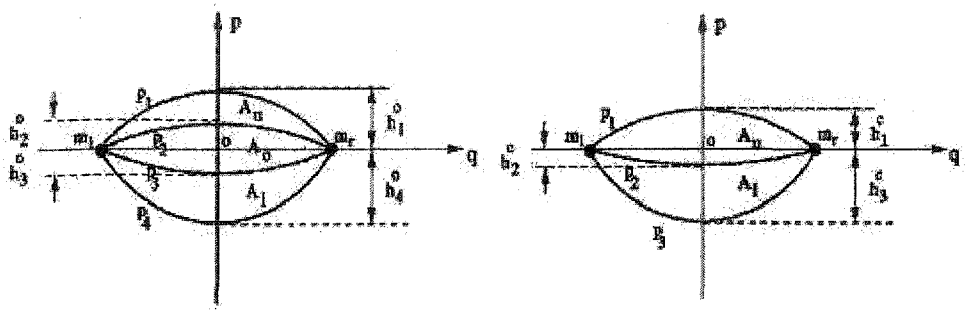


Figure 16: Deformable template for open mouth (left) and closed mouth (right) in 2D [59]

We extend this concept (Equation (1)) to 3D space. When we project a 3D curve onto the XY and XZ planes, the projections still have the shape of parabolic curves. Hence, we have the new representation of mouth shape in 3D as

$$x = t$$

$$y = w_i\left(1-\left(\frac{t}{L_M}\right)^2\right) \tag{2}$$

$$z = h_i\left(1-\left(\frac{t}{L_M}\right)^2\right)$$

where, the parameter w_i describes the height of the curve's projection on the XY plane and h_i describes the height of the curve's projection on the XZ plane. L_M is the width of the mouth. With this parameterized 3D curve representation, the

user can edit the mouth shape in 3D space by changing the values of w_i and h_i. Now, we are able to describe mouth shape by a deformable template using Equation (2). The open mouth is represented by five 3D curves. The closed-mouth is represented by four 3D curves, since the two middle curves merge into one.

Figure 17 illustrates different views of a set of deformable curves generated by our template in 3D space. We have generated three curves by using the same width parameter L_M and various values of w_i and h_i (refer to Equation (2)). The resulting curves can be easily adapted to represent the contours of a human mouth in 3D.

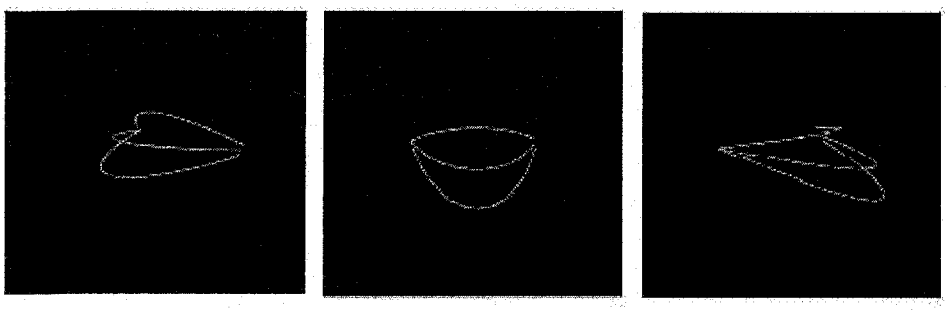


Figure 17: Deformable curves in 3D space

Mouth Representation and Modification

The open mouth can be formed by five curves (Figure 18), all of which are parameterized by our deformable template. This representation can accommodate both an open mouth and a closed mouth. Figure 18 shows an open-mouth template; the closed mouth is obtained by merging middle curves 1 and 2 into

one single curve. Since the lip contour is a combination of five different curves, we need to construct two partial curves (upper 1 and 2) and three complete curves (middle 1, 2, and lower). To achieve the objective of shape modification, the associated curve parameters must be adjusted.

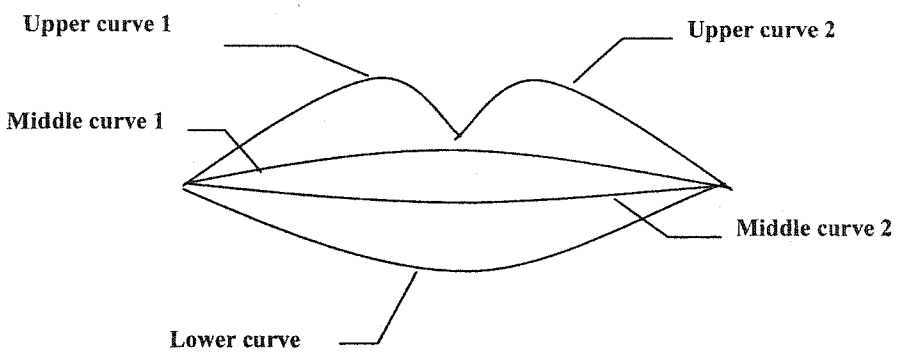


Figure 18: Mouth representation by deformable templates

Eye representation and modification

The contours of eyes on a generic head model can also be represented, using our deformable curve template. Here, we present only the open-eye case (Figure 19), which combines two curves generated by Equation (2). The shape modification for eyes can be performed in a manner similar to that for the mouth.

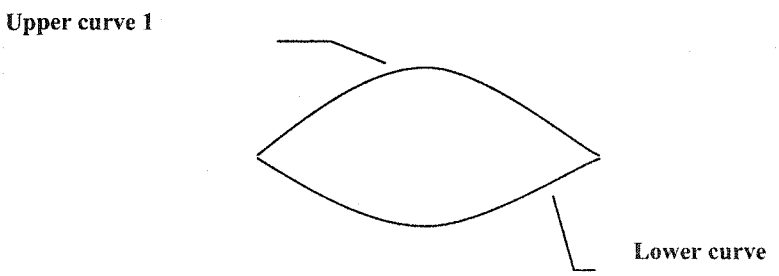


Figure 19: Eye representation by deformable templates

3.3 Facial Texture mapping

3.3.1 Texture Mapping on a 3D Model

We adopt the texture-fitting methodology from [50]. The first step is to apply the marked feature vertices on a 3D model to the texture image. This requires the 3D mesh to be "converted" into a two-dimensional version. There are currently many methods targeting this problem with surface flattening [12, 15, 60]. Here, we choose the simple yet efficient approach of cylindrical projection. Cylindrical projection is preferable for a region corresponding to a volumetric surface (the head, for example). After projection of all the vertices of the facial model into 2D, the region to be textured must be defined, so a Delaunay triangulation [52] is performed based on all the marked feature vertices. This 2D set of triangles roughly defines the region to be textured. Then, for each projected triangle of the object, it must be determined whether it is inside the Delaunay region. One solution is to use the Barycentric coordinate [38] of each projected vertex. By comparing the area determined by the Barycentric coordinate to the known area of a Delaunay triangle, we are able to tell if the vertex is inside or outside the triangle. After the region to be textured is determined, the user can interactively set the marked feature vertices to their correct positions on the texture image (the positions have already been specified during the feature point selection procedure, refer to Section 3.2.2). For all the marked vertices, we calculate their

texture coordinate simply by referring to their position on the texture image. For all the other vertices in the textured region, their position on the texture can be calculated using their Barycentric coordinates.

The texture mapping procedure can be done in real time. If the user moves a feature point, all the Delaunay triangles using this point will change, and the positions of the 2D vertices inside these triangles will have to be recalculated. Therefore, the Barycentric coordinates of the affected vertices are modified, followed by an update of the texture coordinates of these vertices in 3D space.

3.3.2 Texture Mapping Feedback

Our approach is based on user interaction and model improvement is controlled by texture mapping feedback. When the user changes the deformable template's parameters (or moves the feature points to new positions), the texture mapping on the featured area will be updated immediately by this motion. Hence, the user may make a comparison between a mapping result and the original texture image. There is no specific criterion to complete the process of shape adjustment, and the user determines when (s)he is satisfied with the texture mapping on a modified generic model. Experimental results in the next section describe implementation details of our strategy.

3.4 Implementation

In this section, the details of our implementation will be described. First, we give an overview of the system implementation. Then some important modules of the system are explained. Finally, a brief conclusion is given.

3.4.1 Overview

The goal of the implementation is to verify and visualize the concept of our proposed method for the improvement of a 3D model. Basically, the user interacts with both the texture image and the 3D facial model, so our implementation mainly focuses on how to manipulate the 3D model and the texture image.

For the user's convenience, the system simultaneously displays three windows on the computer screen. They are: the menu and toolbar window, texture image display window, and 3D model display window. Figure 20 depicts a screen capture of the system.

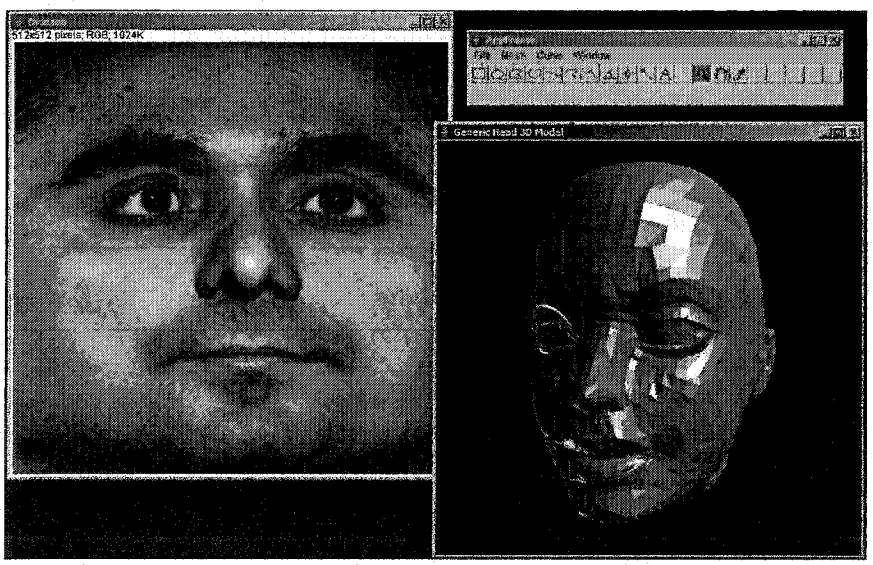


Figure 20: Interface of the system

To understand the system structure better, we include a figure for illustration (Figure 21). It shows the basic structure of the system we have developed.

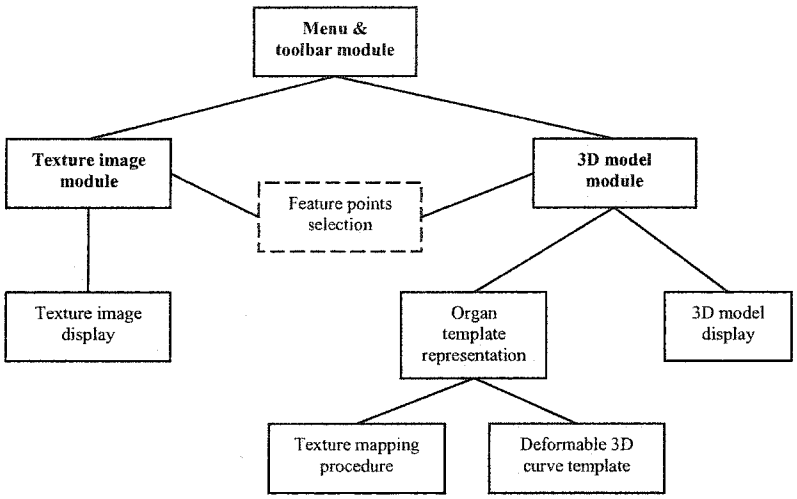


Figure 21: Basic structure of the system

The detailed description of each module is covered in the next subsection. The menu and toolbar module controls the communication between the texture image module and 3D model module, *e.g.*, setting up the correspondence of the selected feature points. The basic function of the texture image module is to display the texture image and provide capabilities to let the user select feature points on the image. The 3D model module is the core module, which consists of two major sub-modules: "organ template representation" and "3D model display." The fundamental element of an organ (mouth and eye) template is a set of deformable 3D curves. The 3D model module includes all the necessary functions and procedures for modifying the generic facial model using our approach. First, it is

capable of displaying the 3D model on the screen. Second, it performs the most important tasks – model deformation and texture mapping.

We choose Java to write the code of the system program. There are three major reasons why we made the decision. (a) Java is platform-independent and can be easily deployed on the web, so that our system can run on any operating system, as well as on the web. (b) The Java 3D API provides a set of object-oriented interfaces that support a simple, high-level programming model to build, render, and control the behavior of 3D objects and visual environments. With the Java 3D API, we can generate high-quality, scalable, platform-independent 3D graphics into applications and applets based on Java technology. (c) Java is an object-oriented programming language, so we can expand our program with less effort later.

3.4.2 System Description

The implementation details are explained, especially in the form and data structure and algorithm description. We focus on the related parts of our work, *e.g.*, the 3D model loading procedure, curve representation class, and texture mapping procedure.

3D facial model module

Before we explain our approach to load the 3D model for display and manipulation, the 3D model data file format is first introduced. Since we use a generic 3D head model, all the data about the model is known. The 3D model is represented by triangular mesh and stored in a single text file. The file has the following format (Figure 22): the first line is the number of vertices on the 3D

head model; the second line denotes the number of triangles on the 3D mesh; the vertex's coordinates start from the third line. Each line has three floating numbers, which are the X, Y, and Z coordinates respectively. After the coordinate of each vertex is listed, the mesh triangles are defined in each line, starting with the number of vertices, and each vertex's order number appearing at the end.

```
2953
5870
0.770299 -0.387685 2.683797
0.931153 -0.466347 2.661932
0.943918 -0.383871 2.700418
0.777934 -0.350620 2.683289
.....
.....
3 2943 2894 2893
3 2943 2893 2942
3 2948 2947 2939
3 2948 2939 2938
.....
.....
```

Figure 22: 3D model data file

We create a customized Shape3D class "OriginalMesh" to load the 3D model's data, which includes a triangle array. All the functions for handling the geometry and appearance of the 3D model itself, such as feature point selection and changing point's position, are defined in this class. When the 3D model object is created, the data of the vertices and the triangles are retrieved from the text file and the 3D mesh is immediately constructed. Initially, the 3D head model is only rendered with single color and light. We can apply texture on the

model surface later. The scaling function is also included, since sometimes we have to adjust the height or width of the head model to fit the person's facial texture better (Figure 23).

Figure 23: Model scaling

In order to select feature points on a 3D model, we provide a method for user to click at various positions on the 3D model. The distance from each point on the 3D model to the selected position is calculated, compared, and the nearest one is chosen as the desired point.

Texture image module

The texture image module is an essential part of the system. Generally, it has the following purposes: (i) To load the texture image and display it in the texture image window. (ii) To allow user to select feature points on the texture image. (iii) Zoom in/out of the texture image. The image loading process starts from the menu of the control module, and can take various types of image format, *e.g.* JPEG, GIF, and BMP. The feature points selection on texture image is quite simple and straightforward. First, the user presses "Segmented line selections" button on the toolbar to enter segmented line drawing mode (for user's

convenience, the segmented lines are used to track the order of each feature point). Then, left click of the mouse within the texture image window allows marking the feature point. Finally, right click of the mouse indicates the last feature point to finish the feature point selection process. The position of a selected point can be changed by clicking on a point and dragging it to a new location. Our system has two modes for the feature points selection on texture image and 3D model. One is to finish marking all the feature point on either the texture image or the 3D model, and marking on the other one later in the same order; another option is to mark one point and immediately mark its corresponding point on the counterpart. Figure 24 illustrates the window at the time when all the feature points have been marked.

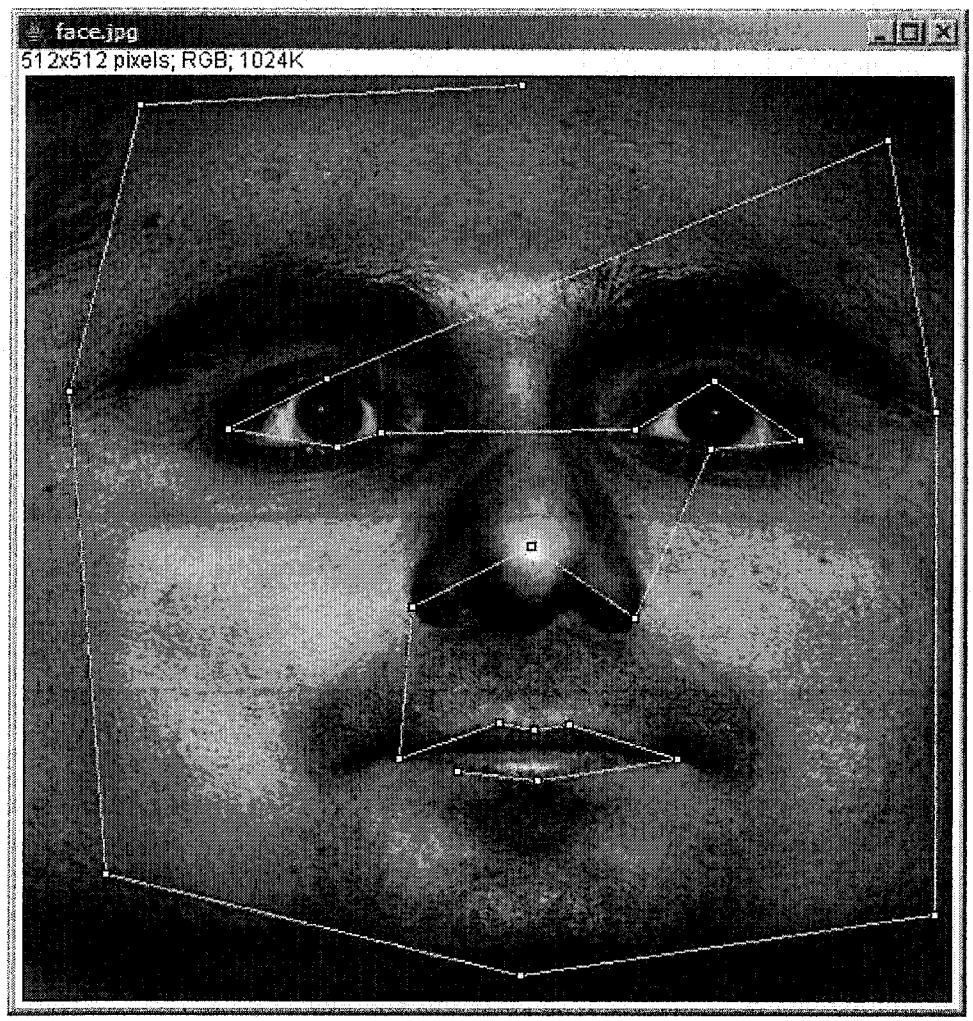


Figure 24: Feature point selection on texture image

Deformable 3D curve template

We developed a dedicated class "ParaCurve" for the deformable 3D curve template. Each instance of the class represents a deformable 3D curve in space. Before we use the deformable template to parameterize the facial organs (mouth and eyes), the parameters in Euqation (2) must be known. For the user's convenience, default configuration files, which contain predefined parameters,

are created. "ParaCurve" class will communicate with "OriginalMesh" class to modify the model's shape according to the input parameters. Basically, "ParaCurve" accepts the parameters and formats them based on the requirement; "OriginalMesh" does the actual work to change the shape of a 3D model.

Patch of interest module

One of the most important tasks of our implementation is to perform texture mapping on the featured area. It is relatively easy to obtain the featured texture of interest (TOI) from a texture image (see texture image module). The remaining work of texture mapping consists of determining the area of interest on a 3D mesh to be textured and calculating the texture coordinate of each vertex inside the featured 3D area. The method we use can be found in Section 3.3.1, therefore only implementation details are discussed here.

Two important classes are introduced: "Patch" and "Delaunay." The objective of the "Patch" class is to represent a featured area on the 3D mesh, receive texture coordinates from the texture image module, and pass necessary information to the "OriginalMesh" class to perform texture mapping. Since only a limited number of feature points are selected, we still have to find out the exact triangles to be textured on the 3D mesh. The "Delaunay" class is used to construct Delaunay triangulation based on the selected feature points. In our system, we define a triangle to be inside a featured area only if all the three vertices of this triangle is inside the featured area (constructed by the Delaunay triangulation).

```
/*
 * Triangulation class.  A triangulation is represented as a
set ofpoints and the edges which form the triangulation.
*/
class Delaunay {
    static final int Undefined = -1;
    static final int Universe = 0;
    int nPoints;
    RealPoint point[];
    int nEdges;
    int maxEdges;
    Edge edge[];
    //triangle index array
    public int [][] Tri = new int[500][3];
    public int nTri = 0;//number of triangles

    Delaunay(int nPoints) {
      // Allocate points.
      this.nPoints = nPoints;
      this.point = new RealPoint[nPoints];
      for (int i = 0; i < nPoints; i++)
            point[i] = new RealPoint();
      // Allocate edges.
      maxEdges = 3 * nPoints - 6;//Max number of edges.
      edge = new Edge[maxEdges];
      for (int i = 0; i < maxEdges; i++)
            edge[i] = new Edge();
      nEdges = 0;
    }
    ......
    ......
}
```

Figure 25: Fragment of the Delaunay class source file

Since the calculation of texture coordinates requires the assistance of Barycentric coordinates, we use Barycentric coordinates to solve the point-in-triangle problem in order to reduce computation time. Figure 26 depicts how we calculate the Barycentric coordinates of a point relating to a triangle. Each vertex's texture coordinate can be obtained by its Delaunay triangle and the triangle's three vertices' known texture coordinates.

```
private double[] Barycentric(Point2d p, Point2d q[]) {
  double[] w = new double[q.length];
  int j, prev, next;
  double weightSum = 0;
  double lenSq;
  int n=q.length;
  double ox = p.x, double oy = p.y;
  double area = this.area2(q[0],q[1],q[2]);
  //For each vertex q[j] of Q:
  //Grab the previous and next q's and compute the barycentric weight
  for(j = 0; j < n; j++)
  {
    if (p.equals(q[j])) {//if the point is one of the triangle's
    points
      for (int i=0;i<n;i++) {
            if (i==j)
                  w[i] = 1d;
            else
                  w[i] = 0d;
      }
      return w;
    }
    else {
      prev = (j+n-1)%n;
      next = (j+1)%n;
      w[j] = area2(p,q[prev],q[next]);
      weightSum += w[j];
    }
  }
  // Normalize the weights
  for(j = 0; j < n; j++) {
      w[j] /= weightSum;
  }
  return w;
```

Figure 26: The computation of Barycentric coordinates.

Other than the above functions, the "Patch" class also sets the relation between the texture feature points and the 3D model feature points.

3.4.3 Conclusion

The skeleton of our implementation for the system is described in this section. The system provides a friendly user interface and simple operation method. By

using the Object Oriented (OO) advantage of the Java language, we are able to divide the entire system into several small modules. Each module holds its own members and methods to solve specific problems. Future modules or methods could be easily added to our system. Because the current system is developed as a Java applet, it is also simple to install it on the web.

Chapter 4

Experimental Results and Evaluation

In this chapter, we will explore some experimental results based on executing our program. To study the effectiveness of the interactive nature of the system, a user evaluation is performed and explained in the second sub section.

4.1 Experimental Results

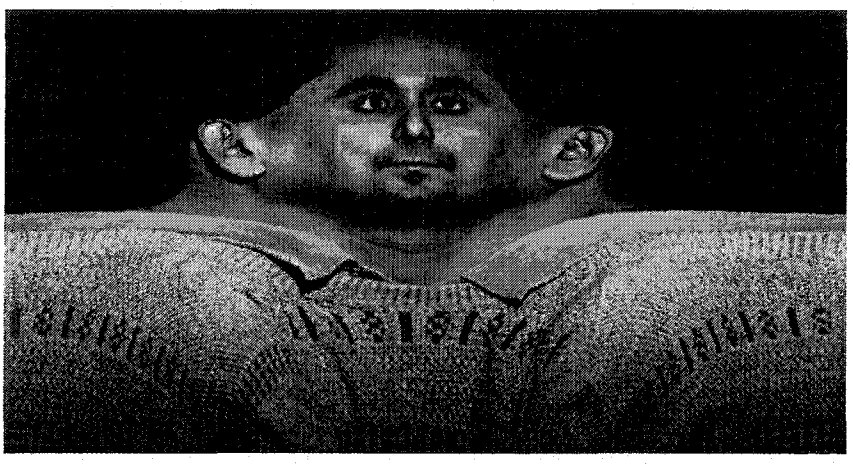
We tested our technique on a generic head model with a facial texture image acquired using the Zoomage® 3D scanner. The scanner creates a 360 degree texture image of a human head and we work only with the face region showed in Figure 27 (b). Figure 29 gives the texture mapping result without interactive matching with the facial model. The feature points on both the 3D model and texture are selected interactively by a user and the correspondence between them is set up accordingly (Figure 28) (note that only the lower lip is showed here; the feature point selection for the upper lip is similar). Interactive texture mapping without using a parameterized mouth contour is showed in Figure 29, and demonstrates a mouth texture mapped to the 3D mesh using the proposed 3D parametric model.

Comparing the left column (without parametric mouth model) and right column (with parametric mouth model) in Figure 30, we see that the mouth is reconstructed much more accurately by the parameterized curve, and the texture is mapped correctly; the curves in the right column are much smoother than those in the left column. In our implementation, parameters are modified by selecting the target curve and dragging the mouse to an appropriate position. However, the texture mapping result depends greatly on the texture feature point selection and the correspondence to the appropriate point on the 3D model. When the result no longer shows any improvement, it may be necessary to redo the feature point selection and correspondence.

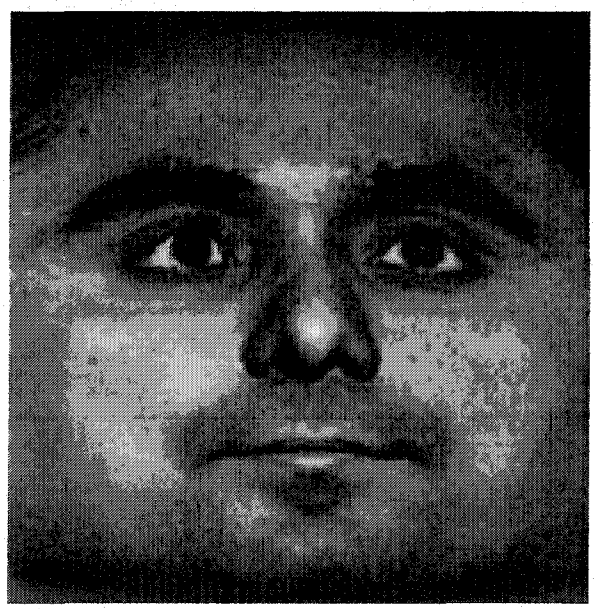
We performed this experiment in real time, *i.e.*, gradually changing the parameters of the deformable templates while viewing the results. We find that the shape of a facial organ may be deformed immediately after each change of the parameters. If the parameter values are increased or decreased, the contours of the organ change from one shape to another, without noticeable delay. Therefore, our technique appears to be efficient for real-time 3D model deformation or animation.

In the experiment for the mouth, we selected 7 feature points for the lower lip, which is formed by two parameterized curves. The upper lip consists of three parameterized curves, and contains 12 feature points. By this configuration, we obtained the results in Figure 30. The texture mapping result is still acceptable without great distortions and errors, although we may obtain more accurate results by adding more feature points on the model and texture image. This could

help the user make better decisions based on the texture mapping feedback, so that the generic facial model can be fitted to a specific person's face more precisely.



(a) Head texture image from scanner



(b) Facial texture image

Figure 27: Texture image

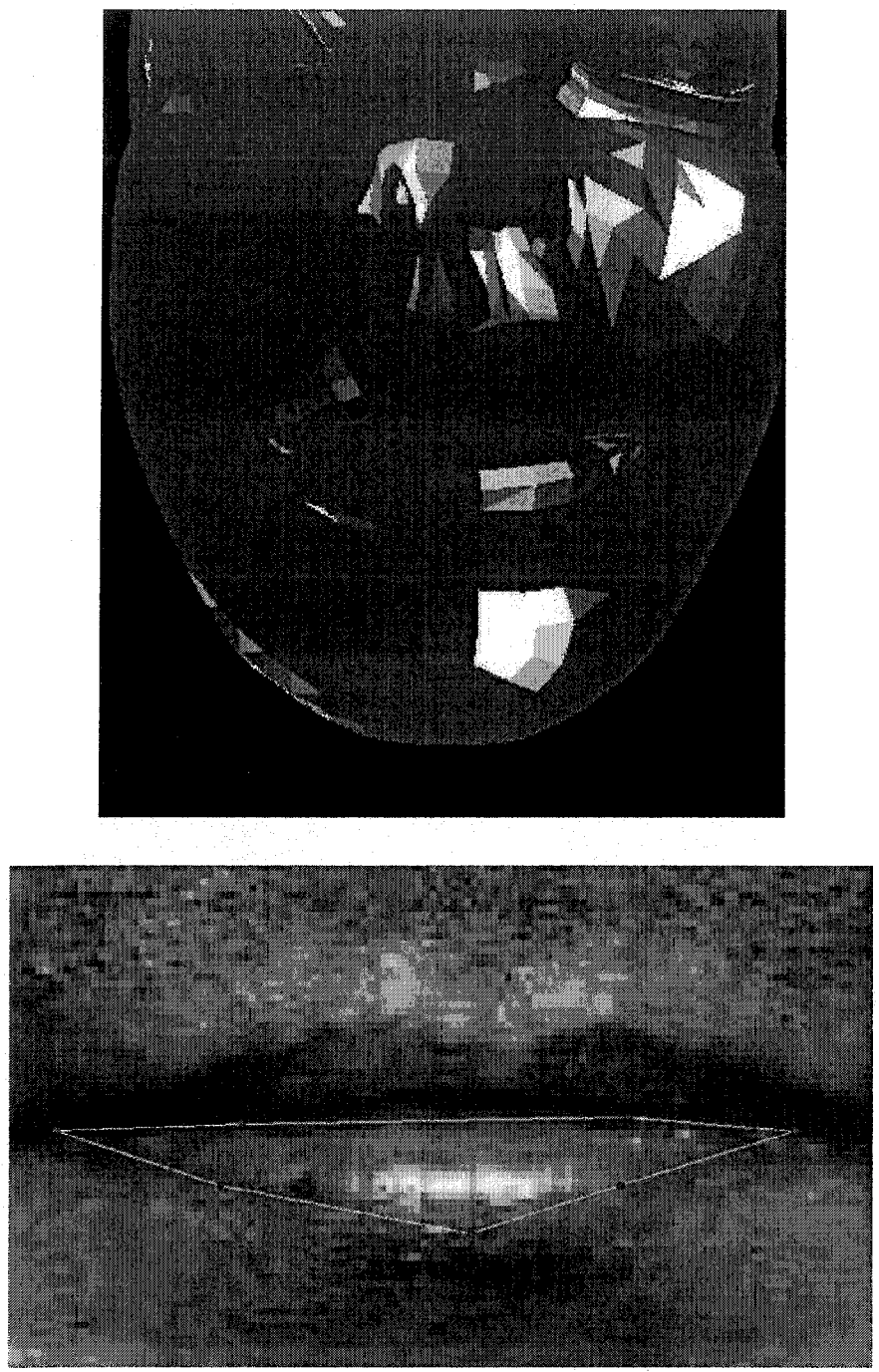


Figure 28: Feature point correspondence (lower lip)

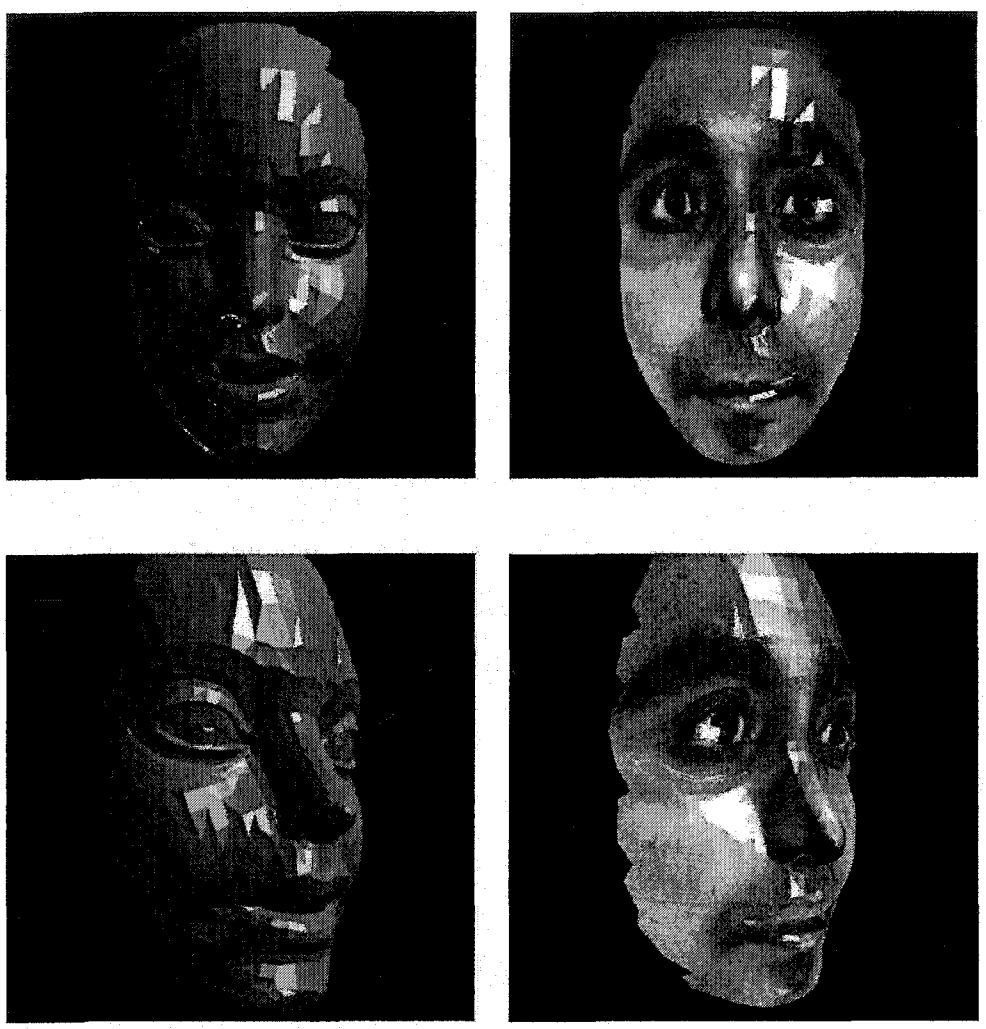


Figure 29: Texture mapping without interactive modification

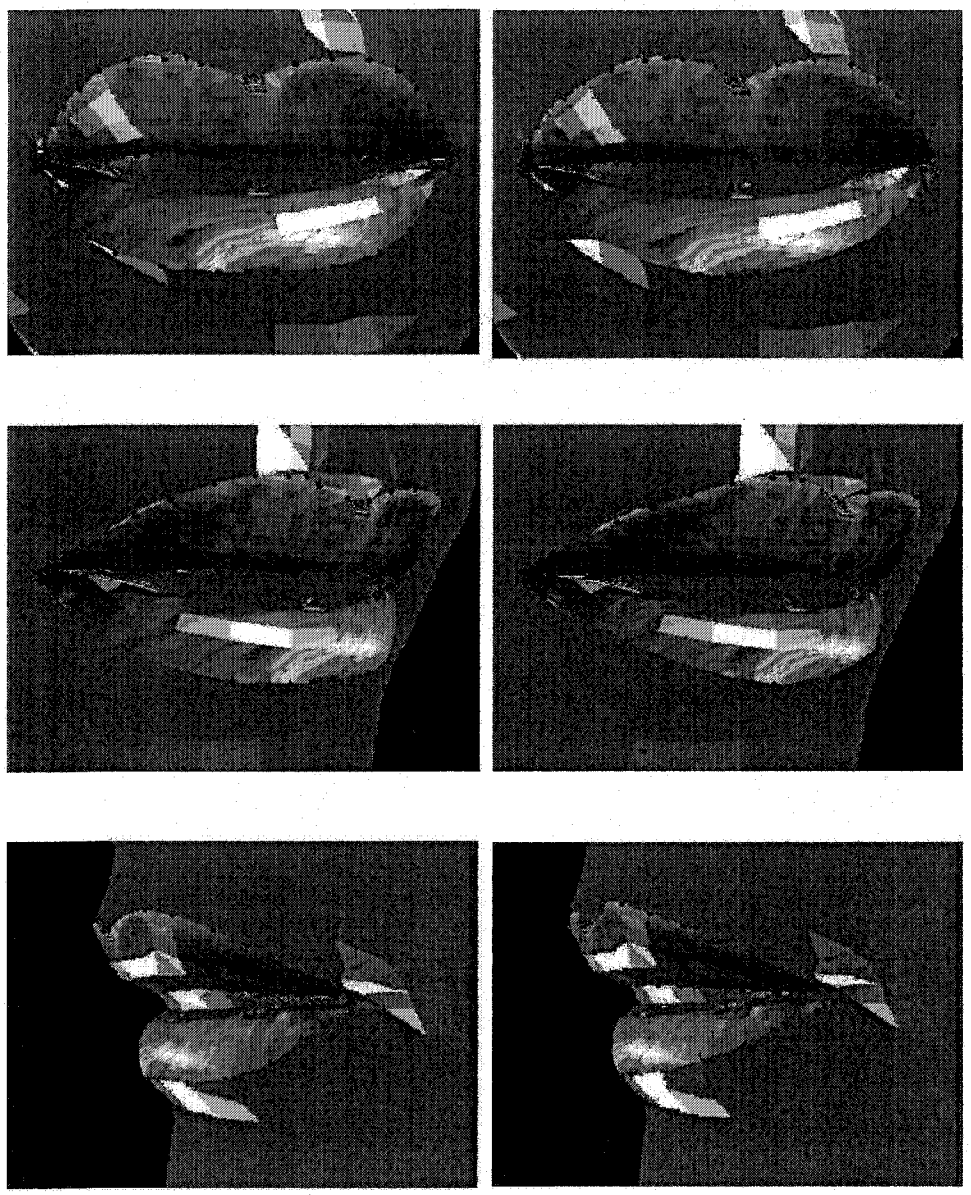


Figure 30: Mouth reconstruction using feedback

Similar experiments have been done on the eyes as well. The results are shown in Figure 31 (left column: before parameterization; right column: after the

curves have been parameterized). Since a human's eye is much simpler than a human's mouth, the improvement might not be as much as that for the mouth.

All of the above experiments focus on a single organ (mouth and eye) and does not show the result with the whole facial texture mapped on. To achieve more realistic results, we tested the system by mapping the entire facial texture on the 3D model and adjusting the mouth, eyes, and nose to their correct positions. Three texture images are used during the experiments. Two of three have open eyes and one has closed eyes. Some inevitable shortcoming can be observed, *e.g.*, the mismatch and distortion of the closed-mouth texture and the open mouth 3D model. This happens because of the limitation that our system is currently not capable of changing a closed mouth model to an open one. The other similar limitation is that we are unable to change an open-eye model to a closed one. In Figure 33, Figure 35 and Figure 37, the upper pictures show the results without improvement, and the lower ones always represents results after improvement.

From the results with the whole facial texture image we may find that it is not the right approach if we only want to generate a perfect model with precise texture image mapped on. Our approach cannot lead to results as good as professional 3D scanners. Our initial goal is to allow a 3D facial model to fit a person more precisely. There are two reasons why we start with a generic 3D model instead of the 3D model generated by 3D scanners. First, current 3D scanner cannot accurately model ear, hair etc. as generic ones can. Second, we

propose an approach that allows a user (possibly over the internet) to interactively build 3D face models using any face images.

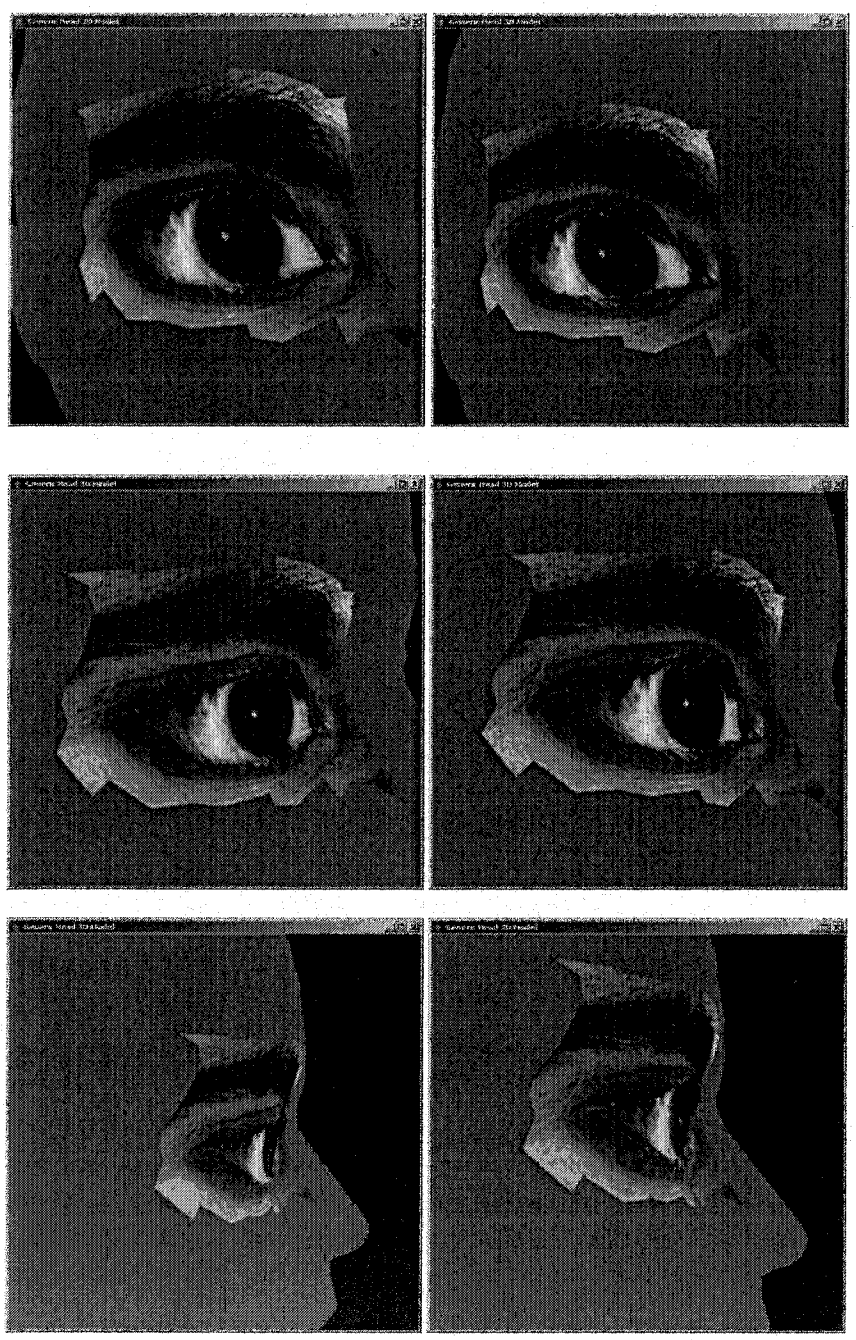


Figure 31: Eye reconstruction using feedback

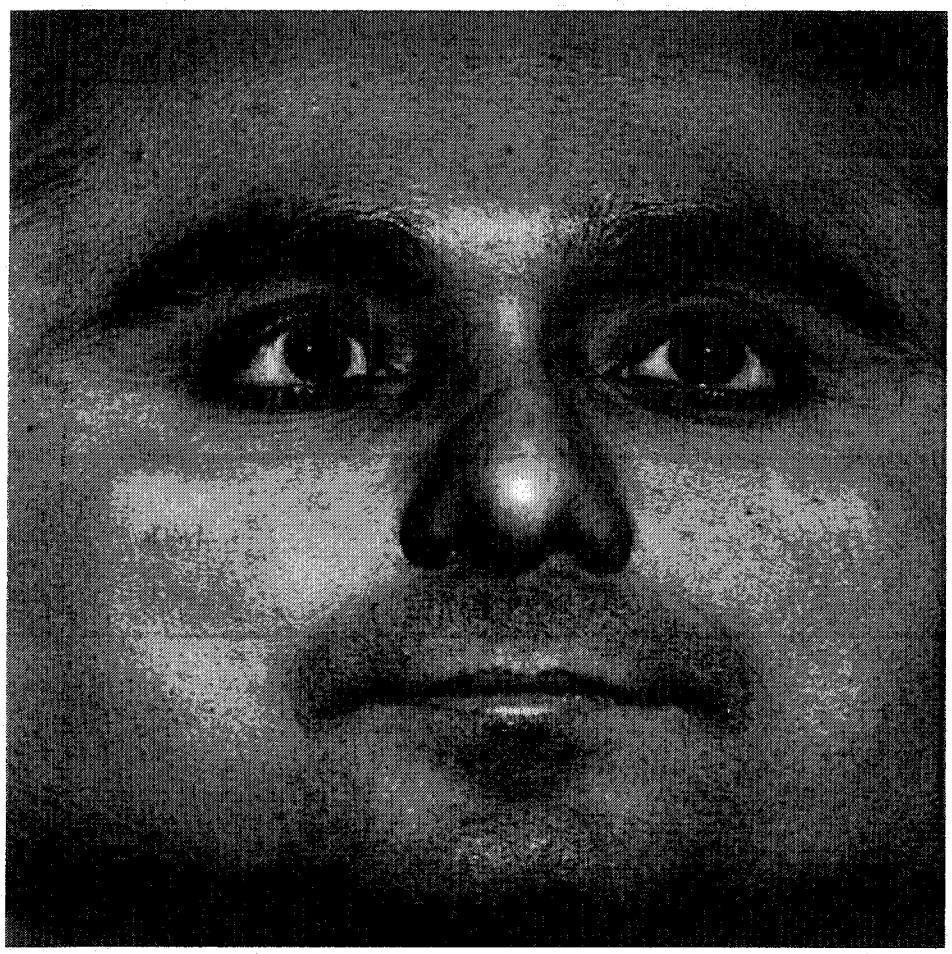


Figure 32: Texture image used in Figure 33

Figure 33: Model improvement with a whole facial texture mapping (1)

Figure 34: Texture image used in Figure 35

Figure 35: Model improvement with a whole facial texture mapping (2)

Figure 36: Texture image used in Figure 37

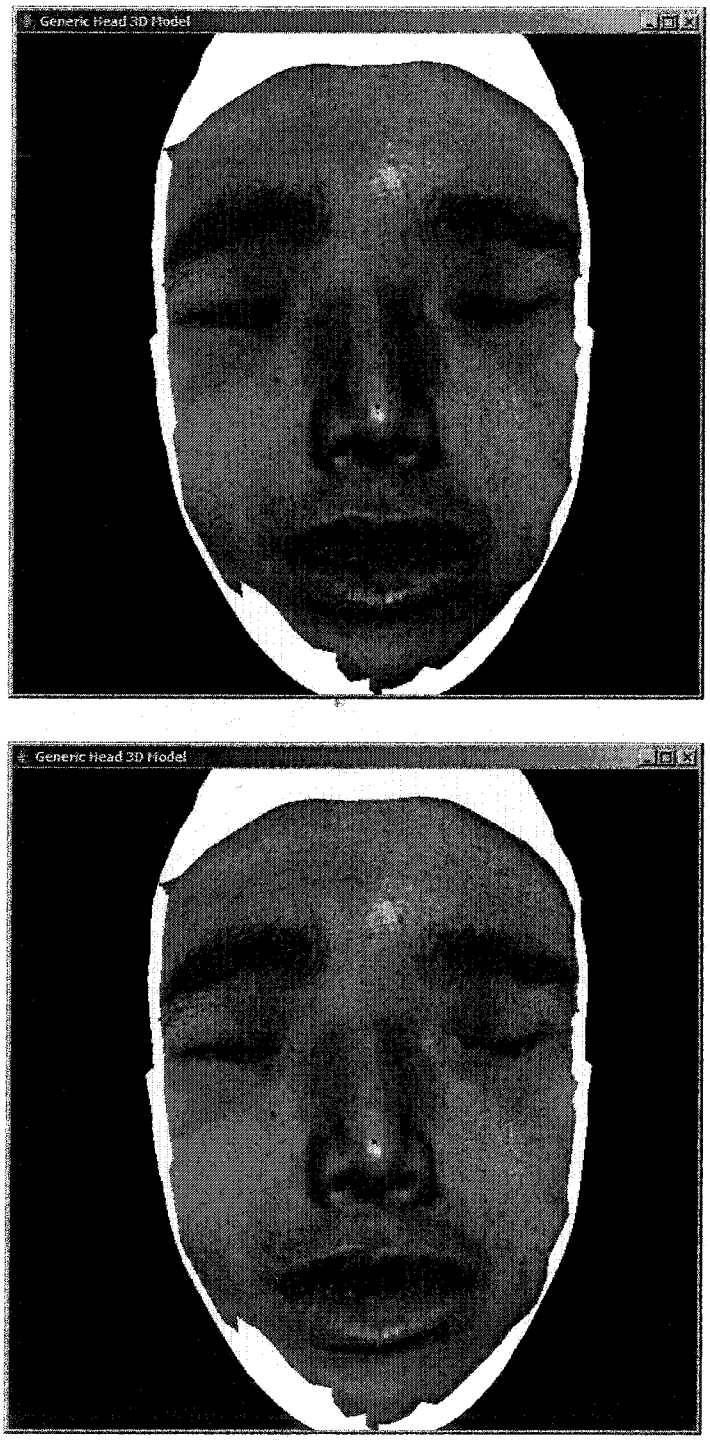


Figure 37: Model improvement with a whole facial texture mapping (3)

4.2 Evaluation

Because our technique is based on user feedback, it is necessary to evaluate the effectiveness of the system from several aspects. We used ten judges to evaluate our implementation and give their evaluation marks on four aspects: ease of operation, texture mapping quality, speed, and visual improvement after using curve templates. The evaluation mark has five levels, *i.e.*, 1, 2, 3, 4, 5, which denote very unsatisfied, unsatisfied, neutral, satisfied, and very satisfied, respectively. The user can give any valid float point marks, *e.g.*, 3.5.

The judges performed evaluations independently and separately, on the same PC located in our lab. The PC's configuration was as follows: Pentium 2.4G Hz CPU, 1 GB RAM, 80 GB hard drive, 64 MB video RAM, and 19 inch CRT monitor. We recorded the evaluation marks and their average, which are shown in Table 2. In conclusion, our system has good performance on both ease of operation and speed. The textures mapping quality and model improvement are also acceptable by most of the users. This verifies that the ideas in this thesis have their advantages, and the implementation can be considered successful.

Items / Judge No.	Ease of operation	Texture mapping quality	Speed	Improvement after using curve templates
1	5	4	5	5
2	4	5	5	5
3	4	5	5	4
4	5	5	5	4
5	3	5	5	4
6	5	4	5	5
7	5	4	5	4
8	4.5	5	5	4
9	5	4	5	5
10	5	4	5	4
Average	4.55	4.5	5	4.4

Table 2: Evaluation marks

Chapter 5
Conclusions and Future Work

5.1 Conclusions

Motivated by the need for an efficient yet simple solution for human facial model deformation, we presented a way of reconstructing the human face model, especially for organs on the face such as the mouth and eyes. Texture mapping is used for feedback in order to achieve a better reconstruction. The input texture image was obtained from a 3D scanner.

The process of facial model improvement is interactive. The techniques proposed here include:

- Retrieving a facial texture image from a Zoomage® 3D scanner.
- Deformable 3D curve templates for modification of the 3D model.
- Interactive texture mapping of the feature facial organs.

The main theoretical contribution of this thesis lies in the development of deformable templates for facial organ models. Given a set of specific parameterized curves, the mouth and eyes can be represented by the template, and modified for further processing. The texture mapping phase makes it very convenient for the user to specify feature points, both on the 3D model and the texture image, thereby setting up the correspondence between them. We have

taken advantage of previous research on interactive texture mapping and 3D modeling, and demonstrated a solution for facial model reconstruction.

5.2 Future Work

One of the immediate goals is to extend the current system to include a tool to assist in reconstructive surgery. This should have a friendly user interface, and should integrate the texture image input and various output formats. An important emphasis of any further work will be how to develop representations for other complex facial organs, such as the nose, ears, and other areas of interest on the face.

One planned future work concerns how to determine the model deformation for areas adjacent to an organ contour. Currently, we have not considered the area except for the contour in the organ representation template. Thus, we can only change the organ shape in a limited degree of freedom; for example, we cannot change a big mouth into a small one. However, the organ's shape can be modified to any desired size if we know how to deform adjacent regions.

Bibliography

[1] Exhibition on the 10th and 11th September 1996 at the Industrial Exhibition of the British Machine Vision Conference.

[2] http://www.cyberware.com , Cyberware Inc., Monterey CA, USA.

[3] http://www.zoomage.com, Zoomage Inc., Edmonton AB, Canada.

[4] T. Akimoto, Y. Suenaga, and R. S. Wallace. Automatic creation of 3D facial models. *IEEE Computer Graphic Application*, 13(5):16-22, 1993.

[5] N. Amenta, M. Bern, and M. Karmvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 415-421, 1998.

[6] C. Beumier and M. Acheroy. 3D facial surface acquisition by structured light. *International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging*, 15-17:103-106, 1999.

[7] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques,* pages 257-266, 1991.

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. volume 30, pages 303-312, 1996.

[9] F. Devernay and O. D. Faugeras. Computing Differential Properties of 3-D Shape from Stereoscopic Images withoug 3-D Models. In ICVPR, pages 208-213, 1994.

[10] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, N.J. Prentice Hall, 1976.

[11] M. Eck and R. Hoppe. Automatic reconstruction of B-spline surface of arbitrary topological type. *Computer Graphics*, 30(Annual Conference Series):325-334, 1996.

[12] I. Eckstein, V. Surazhsky, and C. Gostsman. Texture mapping with hard constraints. In EuroGraphics, volume 20, pages 95-104, 2001.

[13]P. Ekman and W.V. Friesen. *Manual for the Facial Action Coding System.* Consulting Psychologists Press, Palo Alto, 1978.

[14]M. Escher, I. S. Pandzic, and N. Magnenat-Thalmann. Facial deformation for MPEG4. In *Computer Animation*, pages 138-145, 1998.

[15]M. S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometry Design*, 14(3):231-250, 1997.

[16]D. R. Forsey and R. H. Bartels. Surface fitting with hierarchical splines. *ACM Transaction Graphics*, 14(2):134-161, 1995.

[17]M. Gezi. The new face of animation. *Computer Graphics World*, 21, November 1998.

[18]P. S. Heckbert. Survey of texture mapping. *Proceedings of Graphics Interface*, pages 207-212, 1986.

[19]P. Hong, Z. Wen, T.S. Huang, and H. Shum. Real-time speech-driven 3D face animation. In *First International Symposium on 3D Data Processing Visualization and Transmission*, pages 713-716, 2002.

[20]H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 71-78, 1992.

[21]H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 19-26, 1993.

[22]W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 177-184, 1992.

[23]H. H S Ip and L. Yin. Constructing 3D individualized head model from two orthogonal views. *The Virtual Computer*, 12:254-266, 1996.

[24]P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann. 3D interactive free form deformation for facial expressions. Computer Graphics, 1991.

[25]K. G. Kobayashi and K. Ootsubo. t-ffd: free-form deformation by using triangular mesh. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, pages 226-234, 2003.

[26] R. M. Koch and A. A. Bosshard. Emotion editing using finite elements. *Computer Graphics Forum*, 17(3):295-302, 1998.

[27] R. M. Koch, M. H. Gross, F. R. Carla, S. F. Von Buren, G. Fankhauser, and Y.I.H. Parish. Simulating facial surgery using finite element models. In *SIGGRAPH'96*, pages 421-428, 1996.

[28] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 313-324, 1996.

[29] J. B. Kruskal and M. Wish. Multidimensional scanning. Beverly Hills, Calif:Sage Publications, 1978.

[30] T. Kurihara and K. Arai. A transformation method for modeling and animation of the human face from photographs. *Computer Animation,* pages 45-58, 1991

[31] F. Lavagetto, R. Pockaj, and M. Costa. Smooth surface interpolation and texture adaptation for MPEG-4 compliant calibration of 3D head models. *Image and Vision Computing Journal,* 18(4):345-354, 2000

[32] A. W. F. Lee, W. Sweldens, P. Schroder, L. Cowsar, and D. Dobkin. Maps: multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 95-104, 1998.

[33] W. Lee, P. Kalra, and N. Magnenat-Thalmann. Model based face reconstruction for animation. In *Multimedia Modeling (MMM'97), World Scientific,* pages 323-338, 1997.

[34] W. Lee, E. Lee, and N. M. Thalmann. Real face communication in a virtual world. In *Virtual Worlds,* pages 1-13, Springer, 1998.

[35] Y. Lee, D. Terzopoulos, and K. Walters. Realistic modeling for facial animation. *Computer Graphics,* 29(Annual Conference Series):55-62, 1995

[36] W. Ma and J. P. Kruth. Parameterization of randomly measured points for least square fitting of b-spline curves and surfaces. *Computer Aided Design,* pages 663-675, 1995.

[37] N. Magnenat-Thalmann and D. Thalmann. The direction of synthetic actors in the film rendez-vous montreal. *IEEE Computer Graphics and Applications, 7:7-19, 1987.*

[38]M. Meyer, A. Barr, H. Lee, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools,* 7(1):13-22, 2002

[39]J. Noh, D. Fidaleo, and U. Neumann. Animated deformations with radial basis functions. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology,* pages 166-174. 2000.

[40]I. Park, Il Dong Yun, and Sang Uk Lee. Automatic 3-D model synthesis from measured range data. *IEEE Transactions on CSVT,* 10(2):293-301, 2000.

[41]J. Park, G. N. DeSouza, and A. C. Kak. Dual-beam structured-light scanning for 3-D object modeling. In *Proceeding of Thrid International Conference on 3-D Digital Imaging and Modeling,* pages 65-72, 2001.

[42]L. Piegl and W. Tiller. *The NURBS Book.*New York: Springer-Vering, 1997.

[43]F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques,* pages 75-84, 1998.

[44]S. Platt and N. Badler. Animating facial expressions. *ACM Computer Graphics,* 15(3):245-252, 1981.

[45]E. Pojar and D. Schmalstieg. User-controlled creation of multiresolution mehses. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics,* pages 127-130. 2003.

[46]M. J. D. Powell. Radial basis functions for multivariate interpolation: a review. *Algorithms for the Approximation of Punctions and Data,* 1987.

[47]M. Proesmans and L. Van Gool. Reading between the lines: method for extracting dynamic 3D with texture. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology,* pages 95-102. 1997.

[48]G. T. Reid, S. J. Marshall, R. C. Rixon, and H. Stewart. A laser scanning camera for range data acquisition. *Journal of Physics D: Applied Physics,* 21:S1-S3, 1998.

[49]C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. A low cost 3D scanner based on structured light. In *EuroGraphics 2001 Proceedings,* volume 20(3), pages 299-308, 2001.

[50]G. Sannier and N. Magnenat-Thalmann. A user-friendly texture-fitting methodology for virtual humans. In *Proceedings of the 1997 Conference on Computer Graphics International,* page 167. 1997.

[51]T. W. Sederberg and S. r. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual conference on Computer Graphics and Interactive Techniques,* pages 151-160. 1986.

[52]R. Sibson. Locally equiangular triangulation. *The Computer Journal,* 21(2):65-70, 1992

[53]M. Simmons, J. Wilhelms, and A. Van Gelder. Model-based reconstruction for creature animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation,* pages 139-146, 2002

[54]M. Soucy and D. Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. CVGIP: *Image Understanding,* 63(1): 1-14, 1996

[55]S. Wang. Feature extraction in project: Real-time structured light scanner. *Seminar of Computer Graphics Group in Computer Science Department, State University of New York at Stony Brook,* 2003.

[56]K. Waters. A muscle model for animation three-dimensional facial expression. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques,* pages 17-24. 1987.

[57]J. Yan and H. Zhang. Realistic virtual face and body synthesis. In *International Workshop on Machine Vision Application,* pages 28-30, November 2000.

[58]B. C. Yin and W. Gao. Radial basis function interpolation on space mesh. *SIGGRAPH97, Virtual Proceeding,* page 150, 1997.

[59]L. Yin and A. Basu. Generating realistic facial expression with wrinkles for model-based coding. *Computer Vision and Image Understanding,* 84(2): 201-240, 2001.

[60]G. Zigelman, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics,* 8(2): 198-207, 2002.