uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Lijia Zhu

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Example-based Natural 3D Expression Generation for Virtual Human Faces

TITRE DE LA THÈSE / TITLE OF THESIS

Wonsook Lee

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Jochen Lang

Anthony Whitehead

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Example-based Natural 3D Expression Generation for Virtual Human Faces

Lijia Zhu

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering

University of Ottawa

Ottawa, Ontario, Canada

Dec 2006

Your file Votre référence
ISBN: 978-0-494-25848-4
Our file Notre référence
ISBN: 978-0-494-25848-4

Abstract

This thesis proposes a methodology which produces natural facial expressions for 3D human face models by making use of a databank. Firstly, guided by the specified feature point motions, an approach is introduced for generating facial expressions by blending the examples in an expression databank. The optimized blending weights are obtained implicitly by Genetic Algorithms (GA). Secondly, a consistent parameterization technique is shown to model the desired structured human face by processing the raw laser-scanned face data. Based on this technique, we construct a facial motion databank consisting of consistent facial meshes. Thirdly, rather than producing novel facial expressions for a different subject from scratch, an efficient method is presented to retarget facial motions from one person to another. Finally, these techniques are combined together to reconstruct the 3D facial motions (surface motions) based on the motion capture data (feature point motions).

Acknowledgements

Numerous people deserve recognition for their parts in my Master's studies. First of all, I would like to express special thanks to my thesis supervisor, Prof. Won-Sook Lee, for her guidance, support and encouragement to complete my master studies.

I appreciate Andrew Soon for his contribution to my Master's research with the help for RBF and subdivision systems. Special credit goes to my lab manager Francois Malric for his patience and kindness for allowing us to scan his various face models during the research. I also appreciate the following volunteers for allowing me to scan their faces for this document: Prof. Emil M. Petriu, Andrew Soon and Javier Mora. I extend my thanks to other research colleagues, Ding Cai and Pengcheng Xi, for their cooperation and discussion during this work. I also express my appreciation to Li Zhang and Steven M. Seitz in the *Graphics and Imaging Laboratory* of *University of Washington* for allowing us to use their animation data for our research.

I acknowledge *Materials and Manufacturing Ontario* for funding the research. *University of Ottawa* and *Ontario Ministry of Training, Colleges and Universities* provided me with funding during my master studies, for which I am also very grateful and appreciative.

Special thanks go to my parents, Shouxun Zhu and Yinfei Mi, for their endless understanding, encouragement and support. Finally, I thank Xi Xiong for her love.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

Virtual humans with high quality natural facial expressions are essential for computer game and film industry. Although the face is just a small portion of a person, it varies from person to person and it hints the internal emotion of the subject. Moreover, people are very sensitive to recognize, distinguish and reading faces. Consequently, modeling and animating human face usually require higher level of detail and realism than doing other real-world objects.

Traditionally, animators tediously crafted the human face model for producing facial animation. Although nowadays some 3D commercial animation tools (e.g. MAYA and 3Ds MAX) are available, these tools require tedious adjustment for optimizing animation parameters. Since Parke's pioneering works ([Parke 1972] and [Parke 1974]), many approaches have been proposed to model and animate human faces more automatically (Thorough survey can be found in [Parke 1996]). In 1999, MPEG (*Moving Picture Experts Group*) developed the *MPEG-4* standard for animation field (Detailed survey in section 2.3.7). The *MPEG-4* standard specifies a neutral face model and a number of feature points (FPs) as reference points for face shape definition as well as face animation. One of the important advantages of *MPEG-4* facial animation is that facial animation is more easily controllable with a set of feature points rather than all surface points of the face model. Here feature points are the prominent points on the face that capture the outstanding features of the face such as corners of eyes and corners of lips etc. Some other works may not use MPEG-4 FPs, but are also generating motions for all points of the facial mesh based on feature point

movements. Commonly, with the limited number of feature points (e.g. currently 84 feature points are adopted in the MPEG-4 standard), it is not easy to create high quality facial expressions. E.g., it is difficult to produce sophisticated wrinkles on the human face. In order to produce more natural-looking high quality 3D expressions, it is crucial to specify motions for all surface points of the facial mesh.

1.2 Problem statement

This thesis addresses the issue of developing a scheme for generating natural 3D facial expressions. The requirements for such a scheme are:

- The scheme should be easily controllable with the facial feature points instead of all the surface points of the facial mesh.
- The resulting facial expressions should be natural-looking.
- For any person, we shall model the corresponding face accurately and capture its prominent facial features correctly.
- It should be able to transfer generated expressions from one subject to another, making it possible to re-use the expressive animation data.

1.3 Proposed solution

This section briefs the solution to the problem stated in section 1.2. The detailed solutions will be shown in the corresponding chapters. Firstly, guided by the specified feature point motions, the optimal blending surface is found out of an expression databank which is a bank of diverse expressions of a human face. Genetic Algorithms (GA) are utilized to find the optimized blending weights to mix given examples in the databank. Secondly, we use a laser scanner to capture various expressions. A parameterization

technique is used to model the human face by processing the raw laser-scanned face data consistently. For this technique, a generic model is prepared in advance. This generic model is then adapted to the raw laser-scanned data. Consequently, given different laser-scanned data, the resulting facial meshes have the same structure inherited from the generic model. Using this technique to process various raw laser-scanned data, a facial motion bank can be constructed. And since this consistent parameterization technique is adopted, facial motions can be easily retargeted from one face model to another because they share the same structure.

1.4 Document organization

This document consists of seven sections. The rest of this document is organized as follows. Chapter 2 presents a state of the art in 3D facial modeling and animation. The surface-point guided surface blending approach and the GA technique to find the optimized blending weights are discussed in depth in Chapter 3. Chapter 4 describes the consistent parameterization technique to process the raw laser-scanned face. Our facial motion databank is also constructed in Chapter 4. An efficient technique for transferring facial motions from a source model to a target model is shown in Chapter 5. In Chapter 6, previously described techniques are combined together to reconstruct the facial motions based on the motion capture data. This thesis concludes with Chapter 7 which summarizes the key contributions made, discusses the limitations and suggests future research. After the Reference list, three published papers arising from this thesis are listed at the end of this document.

Chapter 2 State of the Art in 3D Face Modeling and Animation

This chapter presents a review of the state of the art in topics which are closely related to the research presented in this thesis. Section 2.1 outlines different surface representations for modeling. Section 2.2 covers various ways to build 3D human face models. Diverse approaches for producing 3D facial animation are presented in Section 2.3.

2.1 Surface representation

Modeling is the process of creating 3D surfaces for the real world objects. There are two commonly used surface representations for face modeling: *Polygon* and *Spline* surfaces. *Polygon surfaces* represent or approximate objects using polygons. A group of polygons, connected to each other by shared vertices, is generally referred to as an element. Each of the polygons making up an element is called a face[1]. On the other hand, S*pline surfaces* use bicubic or quadrilateral surface patches to model the object ([Parent 2002]*)*. One popular example of *spline surfaces* is *Nonuniform Rational B-spline* (NURBS).

Each surface representation has special properties and benefits. Compared with *polygon surfaces*, *spline surfaces* have the benefit of low data complexity when generating smooth surfaces. On the other hand, *polygon surfaces* are simpler and more flexible. When

[1] http://en.wikipedia.org/wiki/Polygonal_modeling

we model human faces, *polygon surfaces* have some advantages over *spline surfaces* ([Parent 2002]). Firstly, when a small addition to one local face area is needed for better representing a facial feature, extra polygons can be inserted into this area easily. On the other hand, spline patches are one connected unit. Thus, the entire surface has to be modified. Another benefit of using *polygon surfaces* is that we can work on isolated areas while hiding the rest. However for *spline surfaces*, this is normally not possible since the spline patches run throughout the mesh.

Consequently, in this document, *polygon surfaces* are used for face modeling and animation. In particular, triangle meshes are used here because they are computationally easier than higher order polygons.

2.2 3D face modeling

The first problem confronting an animator to produce facial animation is to model the human face and make it suitable for animation. A good facial model must be capable of capturing the distinguishing features of the subject. Traditionally, skilled animators worked on the facial modeling process by crafting sculptures manually (sometime only neutral face and other times as well as ones with expressions) and then digitize them for animation. Later on, digital sculpturing methods are introduced with the user-friendly interface to remove the need of real-world sculpture. However as these approaches are tedious and time-consuming, many more automatic approaches for 3D face modeling are presented.

2.2.1 Digitized sculpture

The first 3D face modeling approach is an extension of the traditional sculpture work. After the sculpture of the desired subject is crafted with clay, wood, or plaster, the physical

sculpture is digitized into the computer, most often using a mechanical or magnetic digitizing device. A 2D coordinate grid is drawn on the surface of the physical sculpture, and then the polygons can be digitized on a polygon-by-polygon basis ([Parent 2002]). For example, in Figure 2.1, when people want to build the mesh information on the sculpture, they use a specialized equipment to get 3D curve information from the coordinate gird drawn on the 3D surface.

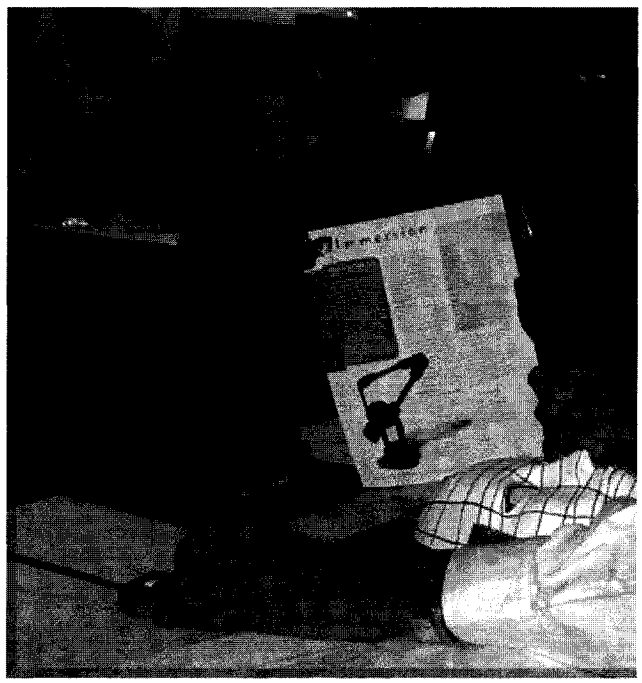


Figure 2.1 : The specialized equipment is used to get 3D curve information on a face mold.

Another method to digitize the physical sculpture is to use a laser-scanner. A laser beam is projected onto the sculpture and is used to calculate distance to the sculpture surface. With a laser scanner, a very accurate digitized model can be created. In the recent movie "Lord of the Rings", the Gollum character was first brought to life in the form of plaster model. Then the plaster model was laser-scanned for digital work.

Photographs of the subject can also be used to facilitate this digitizing process. Magnenat-Thalmann *et al.* ([Magnenat-Thalmann 1987]) extract the 2D coordinate information from the photographs of the subject and then they generate 3D coordinates of the face models based on 2D information. However, laborious work is needed to achieve matching between different photographs.

Instead of preparing the physical sculpture in advance for digitization, we can even emulate the traditional physical sculpting process using a user-friendly interface. For example, LeBlanc *et al.* ([LeBlanc 1991]) present an interactive system *Sculptor* to emulate the traditional physical sculpting process. With *Sculptor*, the user is able to modify and assemble triangle meshes. In later work, Sannier and Magnenat-Thalmann ([Sannier 1997]) show a tool called *TextureFit* which adds a texture map to the virtual human. A photograph of a soldier from the *Terracotta Army* is used for generating the texture map for the face model. They sculpture the virtual soldier using *Sculptor* and *TextureFit.*

2.2.2 Interpolation

Parke ([Parke 1974]) uses bilinear interpolation to create various facial shapes by changing the conformation parameters of a generic face model. The conformational parameters are distances between facial features. These parameters distinguish one subject from another. It is difficult to tune the parameters for each new face model. Lee *et al.* ([Lee WS 1998] and [Lee WS 1999]) use the linear interpolation technique on given faces to generate a new face with texture information. Using this technique, a population of virtual human face can be constructed easily. Blanz and Vetter ([Blanz 1999]) present a method to learn a morphable face model by applying *Principal Component Analysis* (PCA) to vector representations of the shape and texture of 3D prototype faces. Novel faces can be generated

by tuning blend weights of the prototype faces. In their work, a 3D face database consisting of about 200 laser-scanned face models is constructed. A great deal of time is required to compile this face database to bring the prototype faces into full correspondence and to estimate parameters for the new face.

2.2.3 Generic model adaptation with 2D photographs

Using 2D photographs has the benefit of not requiring the presence of the physical model once the photograph has been taken. So this approach has wide applications for video conferencing and compression ([Parent 2002]). Usually the orthogonal photos (front and side view) of the subject are adopted. This approach normally prepares a generic face model in advance. Feature detection step is usually needed in order to detect the feature point positions (e.g. eye, nose and mouth corners) in the input photos. The 2D facial feature information extracted from the orthogonal photos is then used to calculate 3D feature positions.

Kurihara and Arai ([Kurihara 1991]) require the user to interactively define a small number of control points on the input photos. Then the other feature points are adjusted using interpolation. Since too small number of control points is defined, the generated model does not fit the input face very well. Ip and Yin ([Ip 1996]) propose a method to automatically extract facial features from photographs using a local maximum-curvature tracking algorithm. The input facial photos are blended using linear interpolation and texture-mapped onto the 3D head model. Their method works well with Asian looking, but not with Caucasian looking people. Lee and Magnenat-Thalmann ([Lee WS 2000]) provide a semi-automatic feature point extraction method with a user interface for interactive correction if it is required.

The reconstructed 3D face can be animated immediately with given expression parameters embedded in the pre-defined generic model.

Since a camera is very common, this approach has the advantage of using cheaper and more convenient equipment. Although it is easy to catch characteristic facial regions on the photographs by specific algorithms, it is still difficult to capture non-characteristic points on the face. Consequently, the resolution of the resulting face model is limited, especially making it difficult to model subtle wrinkles on the face.

2.2.4 3D range data

In order to capture the human face in high resolution, 3D range data is utilized in face modeling. Usually, specialized hardware is required and cost range depends on the technology and level of accuracy needed. Different types of 3D range data are discussed in the following sub-sections.

2.2.4.1 Stripe generator

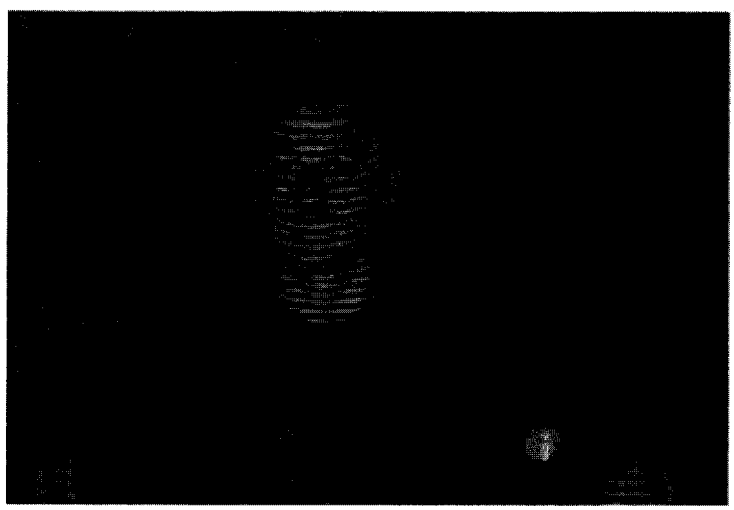


Figure 2.2 : Illusion of the light pattern projected onto the human face with the scanning device *Capturor* of *InSpeck ([D'Apuzzo 2006])*.

This technology is based on the projection of light patterns. Usually, this system is composed of a pattern projector and a light sensor. More complex systems use more light sensors. A light pattern (usually in the form of stripes) is projected onto the human face (Figure 2.2). A light sensor (e.g. a digital camera) captures the changes in the pattern. Stripes on the face surface are measured and then the shape of the subject can be determined. Everything happens mostly within one second so that the human face can be digitized easily. The advantages of this technology are its relatively low cost and its speed.

Proesmans and Van Gool ([Proesmans 1997]) project square grid patterns onto a moving face and analyze the captured video to construct face models. Recently, Zhang *et al.* ([Zhang L 2004]) use a capture system which employs synchronized video cameras and structured light projectors to record videos of a moving face. Then they propose a method that goes from video sequences to high-resolution animated face models.

2.2.4.2 Laser scanner

Laser scanning technology uses the laser light source to project one or more thin and sharp laser beams on the human face. The laser scanner samples the subject at regular intervals to create unorganized set of surface points to represent the subject in very high resolution. For example, there is the 3D scanning services available in *XYZRGB*[2] which is capable of capturing human face detail in the order of about 100 micron.

---

[2] http://www.xyzrgb.com

Although a laser scanner is good at capturing human face with very high resolution, the raw laser-scanned face data can not be directly used for facial animation. Firstly, holes and noise usually exist in the raw data which make the distracting artifacts appear. Secondly, the raw surface data is so dense and irregular that it can not be easily used for optimal model construction and animation.

A common approach to tackle these drawbacks is to pre-define the generic model with all the necessary structures. This technique shares the same concept as Section 2.2.3 where 2D photographs are used as input. Here the 3D laser-scanned range data is used instead of 2D data, making it capable of capturing higher resolution face data. The benefit of this approach is that all the generated face models inherit the same structure or even animation control parameters from the pre-defined generic model. It is easier to animate the derivatives of the generic model rather than the raw laser-scanned data.

Much research has been done on adapting the generic model to the raw laser-scanned data. Lee *et al.* ([Lee Y 1995]) develop the algorithms that automatically construct functional models of the human face from laser-scanned range and reflectance data. They label the facial feature points on the generic model prior to the adaptation step. They apply the modified *Laplacian* operator to the captured range map, thus detecting feature points in the map. The generic model with pre-labeled features is then adapted to the range data. Zhang *et al.* ([Zhang Y 2004]) use a set of measurements between the specified feature points on the generic model and the scan data as references. Then a global adaptation is carried out to align the generic model to the scan surface. A local adaptation then deforms the geometry of the generic model to fit the vertices to the scan surface. Jeong *et al.* ([Jeong 2002]) generate multi-resolution head models from scan data. A generic control model is served as the

starting point, a hierarchical representation of the model is then generated by repeated refinement using subdivision scheme and measuring displacements to the input scan data.

2.2.4.3 Silhouette

Face shape can also be reconstructed from several silhouette images of the subject. Zheng ([Zheng 1994]) reconstructs a 3D facial model from sequences of contours. A sequence of images is taken as an object rotates. Then he calculates the 3D convex hull based on those silhouette images. After that, he builds the object model with the estimated 3D hull. This approach has a drawback since it approximates the object only based on the silhouette information. In particular, for a face, the silhouette images do not provide all the information of the face. The quality of the resulting model is thus limited.

In order to solve this problem, recently Lee *et al.* ([Lee J 2003]) present a novel method to obtain the 3D shape of a human face using a sequence of silhouette images as input. They utilize a face database consisting of 138 faces. They approximate the input silhouette images by linear interpolating the eigen-heads, which are obtained by a Principal Component Analysis (PCA) of faces in the database. The resulting face model whose silhouette images match very close to the input silhouettes.

2.2.4.4 Stereoscopy

Figure 2.3 : Stereo camera images and reconstructed 3D surface points in [Nagel 1998].

Another way of obtaining 3D range data is to utilize the stereo images and use the geometric relation over stereo images to estimate the surface depth. For example, Nagel *et al.* ([Nagel 1998]) use two stereo cameras to take stereo images from two different angles. Then from the images, they calculate the depth map to describe the distances of the surface points to the camera. This depth map can then be transformed into a set of 3D points representing the surface of the person. Figure 2.3 shows the input images for the stereoscopic and the estimated 3D surface points. In their work, the generic model is predefined with muscle structures. Then they adapt the generic model to the estimated 3D surface point data. After that, they animate the face with the predefined muscles.

2.3 3D face animation

2.3.1 Direct parameterization

Parke ([Parke 1972] and [Parke 1974]) proposes a parametric approach to generate a wide range of facial expressions. He designs the parameters with two categories: *conformational* and *expressive*. The *conformational* parameters distinguish one subject from another and these parameters are used when modeling the face (e.g. nose length/width, eye

size and separation). The *expressive* parameters are related with the facial expressions (e.g. the upper-lip position, mouth corner position).

The direct parameterization approach is almost hard-coded for generating face animation. Tedious manual work is required to tune those parameter values. In addition, for every new face model, the same tedious work has to be performed again. More automatic approach is expected.

2.3.2 Physics-based approach

This approach is inspired by the concept of anthropotomy, where the face is changing due to the interaction of internal muscles and skin tissues. Face animations are generated by simulating the physical behaviors of the face.

Terzopoulos *et al.* ([Terzopoulos 1990]) use a mass-spring model to represent the facial mesh in three layers (skin, tissue, and muscles attached to bones). Elastic spring elements connect each mesh node and each layer. Face animation is generated by propagating muscle forces through the mesh. Extensive computation is involved in their work.

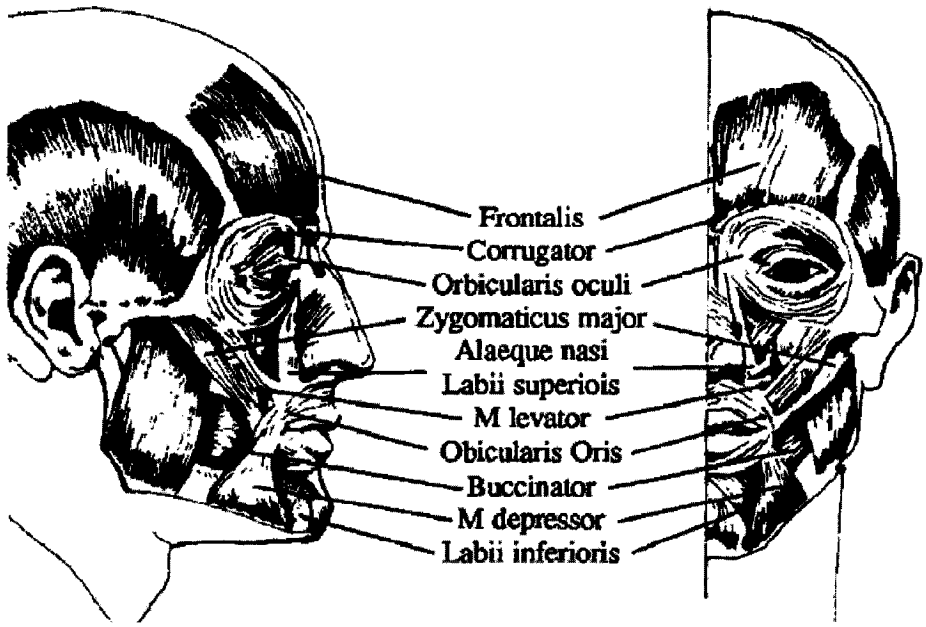


Figure 2.4 : The major muscles in the face shown in [Waters 1987].

Waters ([Waters 1987]) uses a vector-based muscle model instead. The major muscles in the face are shown in Figure 2.4. Then according to the major facial muscles, he designs pseudo-muscle consisting of the vector direction, an origin, and an insertion point. Facial animation is produced by changing the contraction values of the embedded muscles under the facial mesh.

Commonly, for physics-based approach, physical structures must be manually placed and tuned under the facial mesh. Tremendous computation is usually required. It is not very intuitive to control the animation parameters through such physical models.

2.3.3 Surface deformation

Facial animation can also be generated by performing the surface deformation on the given facial mesh. This deformation is usually controlled by a set of control points. *Free-Form deformation* (FFD) is a technique for deforming volumetric objects by manipulating

control points arranged in 3D cubic lattices ([Sederberg 1986]). The object is embedded in an imaginary and flexible control box which contains a 3D grid of control points. As the control box is deformed into arbitrary shapes, the embedded mesh deforms accordingly. The disadvantage of FFD is that the parallelepiped lattices restrict the type of deformations that can be achieved. In order to overcome this limit, Hsu *et al.* ([Hsu 1992]) propose the method of *Direct manipulation of Free-Form Deformation* (DFFD) which is an extension of FFD. They remove the constraint on the position and topology of control points. *Rational Free Form Deformation* (RFFD) is a further extension of DFFD which gives one more degree of freedom on the weights assigned to the control points. DFFD is just a special case of RFFD where all control points are given the same weight. For RFFD technique, different weights can be assigned to the control points. Kalra *et al.* ([Kalra 1992]) describe interactive facilities for simulating abstract muscle actions using RFFD. The particular muscle action is simulated as the displacement of the control points of the control-unit.

FFD and its extensions are more computationally efficient and intuitive than the physics-based approach. However, they are hard to produce smooth and natural facial animation results since they are limited to the control lattices. In order to tackle this drawback, *Radial Basis Functions* (RBF) networks have been used for facial animation by many researchers. RBFs are a powerful and efficient mathematical tool for surface approximation ([Powell 1987]). For RBF deformation approach, control points are defined on the object mesh firstly. Once surface approximation is established, the parameters of RBFs are updated to reflect any changes to the control points. RBFs can then approximate the positions of the points other than the control points. Some researchers (e.g. [Fidaleo 2000] and [Noh 2000]) use a small set of control points and utilize RBFs to deform the facial mesh locally to produce facial expressions.

RBF deformation method produces more natural results than FFD deformation does. Compared with the physical based approach, it is much easier and more intuitive to control facial animation. However, it is still difficult to produce subtle wrinkles on the facial skin with only a set of control points.

2.3.4 Performance-driven animation

Facial animation can be driven by the facial motion data captured from a live actor's performance. Williams ([Williams 1990]) tracks expressions of a live actor and then he maps 2D tracked data onto the scanned 3D face model. In his work, motion is captured using a single video camera and a mirror which generate the multiple views required for vision reconstruction. Since Williams' pioneering work, performance-driven facial animation has been widely studied. Guenter *et al.* ([Guenter 1998]) capture the geometry and texture information on the live actor's face from video streams and produce life-like facial animation based on the captured markers. Some researchers combine performance-driven animation and physics-based approach together. For example, Terzopoulos and Waters ([Terzopoulos 1993]) estimate the dynamic facial muscle contractions from video sequences of expressive human faces. They use deformable contour models to track facial motions in video images. Then the tracked muscle contractions are used to animate the physically based model.

The aforementioned works utilize video sequences to drive facial animation. They reconstruct 3D motion capture data based on 2D video images. However, it is difficult to capture detailed motions of facial skin from the video sequences. Recently, with more

advancing optical motion capture technology, higher resolution and more consistent facial motions can be captured. For example, for a VICON[3] motion capture system, the subject is surrounded by an array of high-resolution cameras. Each of these cameras has a ring of LED strobe lights around the lens. A set of reflective markers is attached to the subject. When the subject performs the action, light from the strobe is sensed by the camera lens and thus creating a video signal. The VICON software suite is then used to process the raw video data. With the VICON software suite, 3D motions from the real actor can be reconstructed automatically after combining 2D data from each camera.

Using such an advanced optical motion capture system, subtle facial motions from the performer can be captured. For example, Havaldar ([Havaldar 2006]) uses two hundred cameras to track eighty markers on the subject's face. Then he creates sophisticated expressions based on the captured facial action units. In the recent successful movie "*The Lord of the Rings: the Two Towers*", facial expressions and body movements from real actors are captured with a high-resolution motion capture system and then used to animate the virtual actor *Gollum* ([Scott 2003]). Motion capture is considered to be a very powerful tool as it greatly improves the efficiency of the animation process.

2.3.5 Interpolation

In section 2.2.2, the interpolation technique is used to generate novel face models by interpolating the faces in the database. Interpolation can also be used for generating facial

[3] http://www.vicon.com/

expressions between different expression poses of the same person. This can be traced back to Parke's pioneering work ([Parke 1972]) where an interpolation function specifies smooth motion between two key-frame expressions. Facial animation can then be produced by interpolating two selected key expressions. To generalize this a bit more, a weighted sum of two or more selected facial expressions can be used in which the weights of the linear combination sum to one ([Parent 2002]). By varying the weights, a wide range of facial expressions can be produced with very little computation. Pighin *et al* ([Pighin 1998]) fit a generic face mesh to a collection of photographs of an individual's head. Then they produce novel expressions by interpolating different expression models with the same underlying structure.

Recently, many researchers combine performance-driven approach with interpolation technique to drive facial animation. Kouadio *et al.* ([Kouadio 1998]) present an animation system that captures facial expressions from a performance actor. They animate the human face based upon a bank of 3D facial expressions. A linear combination of the key expressions in the bank is used to produce facial expressions. In their work, interpolation weights are obtained by minimizing the *Euclidean* distance between corresponding feature points in the face model and live markers. Chuang and Bregler ([Chuang 2002]) produce facial animation with the interpolation of motion capture data. In their work, interpolation weights are calculated based on the least square solution. Chai *et al.* ([Chai 2003]) show that facial actions can be created from a preprocessed motion capture database and that a user can control the animation by acting out the desired motions in front of a video camera. They develop a facial tracking system to extract animation control parameters from video. Then they reconstruct facial motions by linear interpolating the K closest examples in the database.

In general, interpolation technique provides an intuitive way to control facial animation. Especially when the expressive animation data is available, the resulting facial expression has the same quality as the expression models in the database. Recently, performance-driven approach has been combined with interpolation technique which greatly improves the efficiency of animation process.

2.3.6 *Facial motion retargeting*

Facial motion retargeting technique efficiently re-uses existing animation data. Gleicher ([Gleicher 1998]) shows that the body motions of one character can be transferred to another. Inspired by this idea, Noh and Neumann ([Noh 2001]) propose the expression cloning technique to clone the motion vectors of the source face model onto the target model. Blanz *et al.* ([Blanz 2003]) present a system that transfers expressions across individuals, based on a common representation of different faces and facial expressions in a vector space of 3D shapes and textures. This space is computed from 3D scans of neutral faces and expression faces. Pyun *et al.* ([Pyun 2003]) show an example-based approach for cloning facial expressions between models while preserving the characteristic features of the target model. However they require the animator to prepare many example key-models for both source and target models. Na and Jung [Na 2004] describe the system for transferring expressions between source and target model. Their approach consists of two steps: *base mesh retargeting* and *detail mesh retargeting*. The example-based approach is adopted in the base level where lower resolution models are used, so their approach requires fewer examples than [Pyun 2003] does. Then they increase the resolution of the models in the *detail mesh retargeting* step where the normal mesh technique is used to hierarchically transfer the normal offsets in the source onto the target.

Generally speaking, facial motion retargeting approach is a very useful tool to re-use the existing animation data. With this technique, facial animations generated by other approaches can be transferred to the novel face models efficiently.

2.3.7 MPEG-4 facial animation

2.3.7.1 Overall view of MPEG-4 face animation

By proposing the Facial Action Coding System (FACS), Ekman et al. ([Ekman 1978]) started the definition of a standardization system to be used for facial synthesis. In order to deconstruct facial expressions into atomic facial movements, they define 44 basic action units (AUs). The deconstruction is closely linked but not identical to muscle activation. Combinations of independent AUs are capable of generating a wide range of facial expressions.

More recently, MPEG-4 integrates synthetic and natural content in multimedia communications. There are diverse applications for communication and entertainment. It is important to make the standard parameters to communicate between different remote sites. The *Face and Body Animation Ad Hoc Group* (FBA) has defined the standard in detail for human faces and bodies. The MPEG-4 standard for facial animation came into being in 1999. Pandzic and Forchheimer ([Pandzic 2002]) give a very detailed introduction and discussion on this standard. MPEG-4 specifies a face model in its neutral status, a number of *feature points* (FPs) on this neutral face as reference points and a set of *face animation parameters* (FAPs), each corresponding to a particular facial action deforming a face model in its neutral state. Deforming a neutral face model according to some specified FAP values at each time instant generates a facial animation sequence.

2.3.7.2 MPEG-4 face model in neutral state

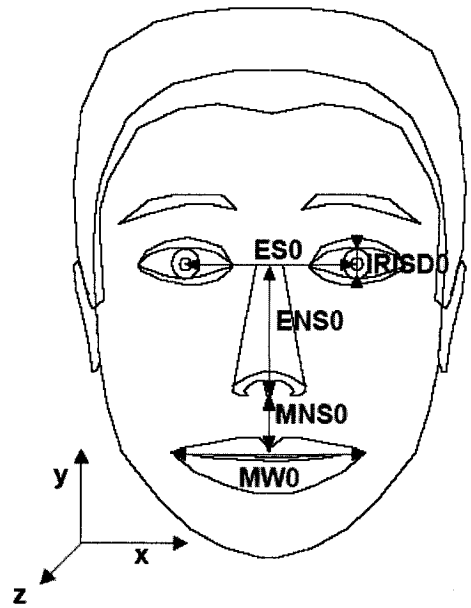


Figure 2.5 : A neutral face model specified in the standard ([Pandzic 2002]); Distances between the key features are used to define FAPU.

As the first step, MPEG-4 defines a generic face model in its neutral state (Figure 2.5) by the following properties ([Pandzic 2002]):

- Eye gaze is in the direction of z-axis
- All face muscles are relaxed
- The mouth is closed; Lips are in contact; the line of the lips is horizontal and at the same height at lip corners

Since the FAPs are required to animate faces of different sizes and proportions, the FAP values are defined in *face animation parameter units* (FAPU). ES0, IRISD0, ENS0, MNS0, MW0 shown in Figure 2.5 are FAPUs defined in the MPEG-4 standard ([Pandzic 2002]). The FAPU are computed from spatial distances between major facial features on the model in its neutral state. ES0 is the eye separation; IRISD0 stands for the Iris diameter; ENS0 is eye-nose separation; MNS0 is mouth-nose separation; MW0 stands for mouth width.

The FAPU is used for normalization between different faces and it allows interpretation of the FAPs on any facial model in a consistent way.

2.3.7.3 MPEG-4 feature points and face animation parameters

MPEG-4 specifies 84 *feature points* (FPs) on the neutral face (Figure 2.6). These FPs are for providing spatial references when defining *face animation parameters* (FAPs). FPs are organized in groups such as cheeks, eyes and mouth. The locations of these FPs have to be known for any MPEG-4 compatible face model. Some FPs are not affected by FAPs (e.g. feature points on the side hairs). However, they are required for defining the shape of the face model ([Pandzic 2002]).

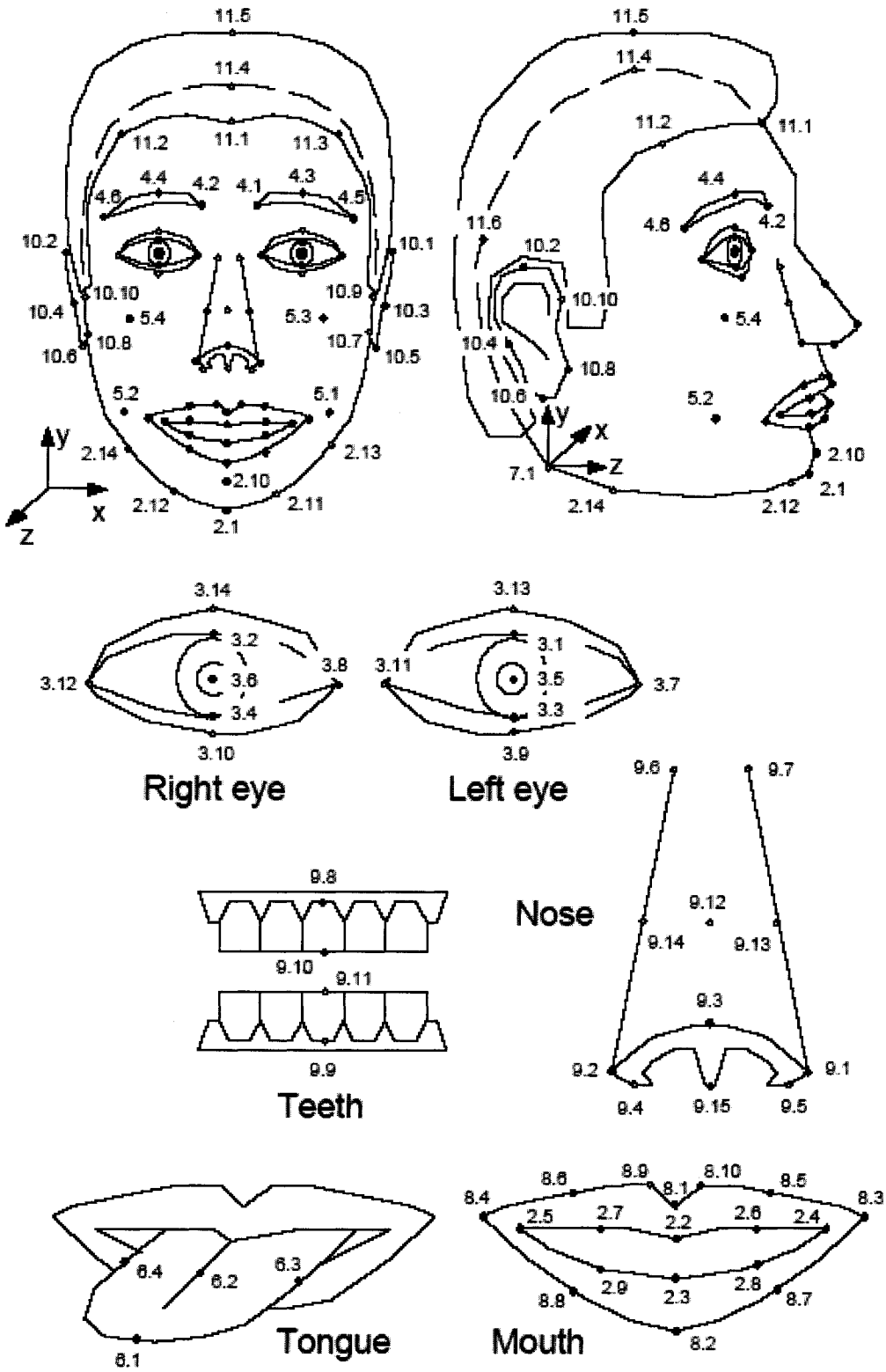


Figure 2.6 : MPEG-4 feature points defined in the standard ([Pandzic 2002]).

The FAPs are designed based on the study of minimal facial actions and are closely related to muscle actions ([Pandzic 2002]). The FAP set contains high-level and low-level parameters. The high-level parameters are visemes and expressions. FAP 1 is a visual correlated to a phoneme. Totally 14 static visemes are defined (Table 2.1) in the MPEG-4 standard. In order to allow for co-articulation of speech and mouth movement, transitions from one viseme to the next are defined by blending the two visemes with a weighting factor. FAP 2 is defined for expressions including the six primary facial expressions (joy, sadness, anger, fear, disgust, surprise). FAPs 3 to 68 are low-level parameters which are used to represent the displacement of some feature points according to a specific direction. For example, FAP 33 corresponds to the vertical displacement of left middle eyebrow; FAP 54 is for displacing the outer right lip corner horizontally. The combination of all deformations resulting from these displacements generates various expressions. A facial animation then is a variation of these expressions over time ([Pandzic 2002]).

Viseme #	phonemes	example
0	none	na
1	p, b, m	put, bed, mill
2	f, v	far, voice
3	T,D	think, that
4	t, d	tip, doll
5	k, g	call, gas
6	tS, dZ, S	chair, join, she
7	s, z	sir, zeal
8	n, l	lot, not
9	r	red
10	A:	car
11	e	bed
12	I	tip
13	Q	top
14	U	book

Table 2.1 Visemes and related phonemes described in the standard ([Pandzic 2002]).

2.3.7.4 MPEG-4 face animation implementations

MPEG-4 provides a feature-based standard for facial animation, making it easier to control the shape and motion of the face. Systems using such a standard can have good compression for facial animation. Given motions of the facial feature points, it is the developer's freedom to decide how to move the surface points. Ostermann ([Ostermann 1998]) uses a piece-wise linear interpolation function for each FAP to produce the desired result. Significant tuning work is required in order to achieve good results. Some MPEG-4 works provide a way of specifying how a FAP is interpreted in a precise manner. A look up table called *facial animation table* (FAT) is used to map feature point motions onto all surface point motions. The FAT actually extends the definition of each FAP into any desired level of detail ([Pandzic 2002]). Mani *et al.* ([Mani 2001]) transfer the FAT from a source face model to a target face model. In their work, additional adjustment of B-spline weights is required to increase the correctness of the mapping. Although users can specify detailed control for all surface points in FATs, significant manual tuning process is still unavoidable.

In the MPEG-4 standard, for a high-level FAP, it usually decomposes the facial expression into several low-level parameters. For example, according to the MPEG-4 standard ([Pandzic 2002]), FAP 2, the primary expression *anger, is* decomposed as follows: the inner eyebrows are pulled downward and together; the eyes are wide open; the lips are pressed against each other or opened to expose the teeth. Another example of FAP restriction is discussed by Garchery *et al.* ([Garchery 2005]). They mention that for some areas such as lips, the work could be very tedious because there are 21 FAPs controlling such a very small lip region. In order to avoid tedious specification for each FAP, Garchery *et al.* ([Garchery 2005]) present a surface deformation method to calculate surface point motions based on the

feature point motions. Their work is developed based on the earlier work by Kshirsagar *et al.* ([Kshirsagar 2001]). Kshirsagar *et al.* ([Kshirsagar 2001]) compute regions influenced by each of the FPs and the corresponding weights for deformation for all the vertices in the influence region. Then from the displacement of the FPs, they calculate actual motions for all surface points of the facial mesh.

Chapter 3 Facial Expression Generation Based on Feature-point Guided Surface Blending

As discussed in the previous chapter, many techniques are hard to create natural bulging, folding, or wrinkles on the facial skin. While the feature-point based approach provides easier control for facial animation, it brings the issue of how to generate natural-looking expressions with merely controlling a few key points on the face.

To tackle this issue, it is better to have additional information on how the model should move (i.e. how all surface points should move). Mapping the feature point motions into an available expression databank would be a suitable solution. Especially when an expressive animation data is available, it would be very beneficial to make use of that data and interpolation is probably the most suitable animation technique for this case. Given a set of key facial expressions from an existing animation database, a blend shape model can be constructed by considering every facial expression as a linear combination of key expressions. By varying the blending weights, a wide range of facial expressions can thus be produced. This technique is known as morphing target animation.

While the interpolation technique has been widely used in popular commercial animation tools such as *3Ds MAX* and *MAYA*, the weights for interpolation require significant manual adjustments by animators. As introduced in section 2.3.5, different techniques are proposed by researchers to calculate blending weights. In this chapter, a novel approach is proposed to find the optimized blending weights via Genetic Algorithms (GA). GA is inspired by the process of natural evolution ([Goldberg 1989]). Essentially, GA's problem solving approach is to evaluate possible solutions at each generation, and then the

offspring solution is generated to update the population pool. Given the evaluation function, the fitter individuals are encouraged to survive and they contribute more to the population. As a result, the optimal solution progressively improves during the process of evolution.

This chapter is organized as follows. In section 3.1, an expression databank is prepared. The feature points for the neutral face model are also defined. The GA technique for finding the optimized blending weights is proposed in section 3.2. The experiment results are shown in section 3.3. Section 3.4 explains the enhancements for the initial system.

3.1 Preparation of the expression databank

L. Zhang *et al.* ([Zhang L 2004]) use a capture system which employs synchronized video cameras and structured light projectors to record videos of a moving face. Then they propose an approach that goes from video sequences to high-resolution animated face models. The resulting 3D meshes illustrate high-quality facial animation and in addition the meshes are consistently parameterized.

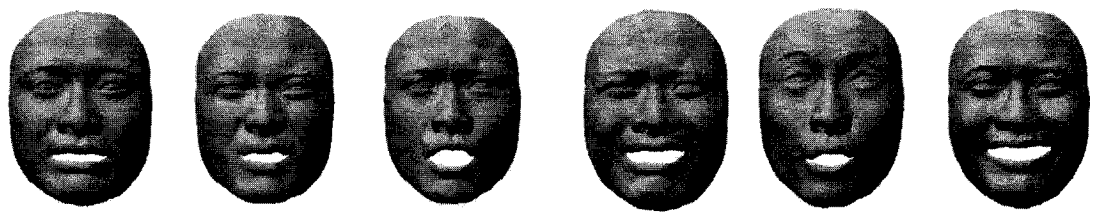


Figure 3.1 : Examples of expression models in the databank.

In this chapter, we make use of the existing animation data produced by L. Zhang *et al.* ([Zhang L 2004]). There are a total of 384 animated models in their data. They all share the same structure. In this chapter, thirty key models from their data are used to construct the

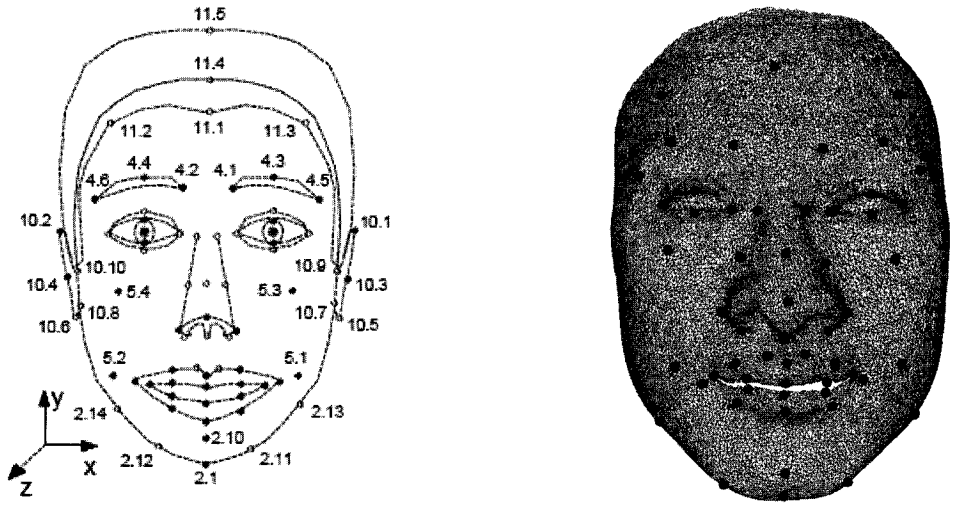
expression databank. Examples of expression faces are shown in Figure 3.1. Given a set of expression models in the databank, the interpolated expression model E can be expressed as:

$$E = \sum_{i=1}^{n} w_i E_i \tag{3.1}$$

where w_i is the weight assigned to the specific expression face E_i in the databank, and n is the number of expression models in the databank. Based on this databank, a wide range of facial expressions can be produced by varying *wi*.

For constructing the expression databank, we can just deposit each key expression face model in the expression databank. However, in order to improve the efficiency of accessing this databank, one optimization step is performed here. Since those expression models all share the same structure, it is not necessary to save the structure information for each expression model separately. Firstly, the neutral face model has to be kept in the databank. And then for each point of a given expression face model, we just compute the difference between this point and the corresponding point of the neutral face model. This difference vector is the motion vector for that point with respect to that expression face model. By doing so, we can represent each expression face model as an array of motion vectors (using the neutral face model as the reference).

For defining FP positions in the neutral face model, we respect the MPEG-4 standard. MPEG-4 defines a total of 84 FPs (Figure 3.2(a)). The feature point set we defined (Figure 3.2(b)), which includes 56 FPs, is the subset of MPEG-4 feature point set. Some MPEG-4 FPs are excluded which are in the areas of the tongue, tooth, ear and hair because the expression models in the databank do not include those parts. Some FPs are also excluded where they are too close to each other (e.g. some points in the eye area).



(a) (b)

Figure 3.2 : Feature point sets (a) MPEG-4 feature point set; (b) Feature points defined here.

3.2 GA optimization for finding blending weights

```
t = 0;
initialize a population P(t) of expression faces;
while (not termination condition) do
   randomly select 3 expression faces from P(t);
   evaluate three picked expressions;
   generate offspring and update P(t);
   t = t+1;
end while
```

Figure 3.3: Our GA structure.

In this section, the goal is to find the optimized blending weights for interpolating expression models in the databank. The basic structure of the proposed GA technique is listed in Figure 3.3. The population of expression faces is initialized with the previously constructed databank. The population size for our GA is 30. The population size remains constant during the overall GA process.

In each generation, three expression faces from the population pool are selected randomly. If two expressions are evaluated at a time, the GA process will converge more slowly. By using the number three, we will get rid of bad genes in the expression databank faster. On the other hand, more than three faces could be evaluated in each generation. GA will converge faster. However it may lead to the local maximum solution. Therefore we decide to evaluate three expression faces at a time.

Now we need to design a fitness function to evaluate randomly selected three expression faces. The purpose of the evaluation is to determine how well the selected expression face matches the input (i.e. specified feature point motions).

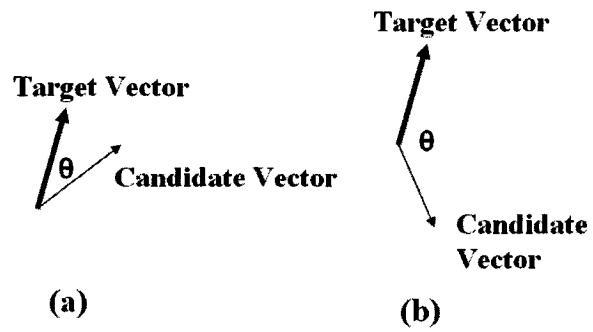


Figure 3.4 : Illustration for explaining fitness function.

Suppose that there are a total of *M* feature points with the non-zero motion vector in the input. First, let us have a look at any one FP with the non-zero motion vector (Figure 3.4). Suppose that the *target vector* is the FP motion vector which is specified in the input requirement and the *candidate vector* is the vector for that FP for the selected expression face. Obviously, the *candidate vector* in Figure 3.4(a) is more qualified than that in Figure 3.4(b) according to the input requirement (i.e. specified feature point motions). It is better to have a

smaller angle θ value between those two vectors. The angle θ can be expressed in terms of the target vector *Va and the* candidate vector *Vb* :

$$\cos\theta = \frac{V_a \bullet V_b}{|V_a| \times |V_b|} \tag{3.2}$$

It hints that it is better to have a larger value of $cos\theta$ for each feature point. We can sum up $cos\theta_i$ values for each FP with the non-zero motion vector in the input requirement. Then we normalize the total sum with the M value (Because the maximum total sum is: $M*cos0^0 = M$). Our fitness value F can thus be given by:

$$F = \frac{\sum_{i=1}^{M} \cos\theta i}{M} \tag{3.3}$$

where M is the number of FPs with the non-zero motion vector. The best value F that can be achieved is 1. However, 1 is just the ideal value. Our aim is to produce optimal value which is as close to 1 as possible. The closer the value is to 1, the greater the similarity between the target expression according to the input requirement and the selected expression face.

This fitness function is used to evaluate previously selected expression faces. The individual with the worst fitness value in this tournament is not allowed to survive in the next generation. It is removed from the population and replaced with the offspring of two winners. There are various crossover strategies to produce the offspring from parents. The crossover operator plays a central role in GA ([Herrera 1998]). The simplest crossover we used is called Midpoint crossover where the offspring lies exactly on the middle of two parents. Suppose that P_1 and P_2 are the parents, the offspring of them is expressed as:

$$Offspring = 0.5 \times P_1 + 0.5 \times P_2 \tag{3.4}$$

For Line crossover (a.k.a. Flat crossover), the offspring, which is located in a random position between the parents (P_1 and P_2), is given by:

$$Offspring = t \times P_1 + (1-t) \times P_2 \tag{3.5}$$

where $0 \leqslant t \leqslant 1$.

BLX-α crossover is an extension of Line crossover. The offspring can be calculated based on the parents (P_1 and P_2) in the following way:

$$\Delta = \alpha (P_2 - P_1); \tag{3.6}$$

$$Offspring = t(P_1 - \Delta) + (1-t)(P_2 + \Delta)$$

where $0 \leqslant \alpha \leqslant 1$ *and* $0 \leqslant t \leqslant 1$.

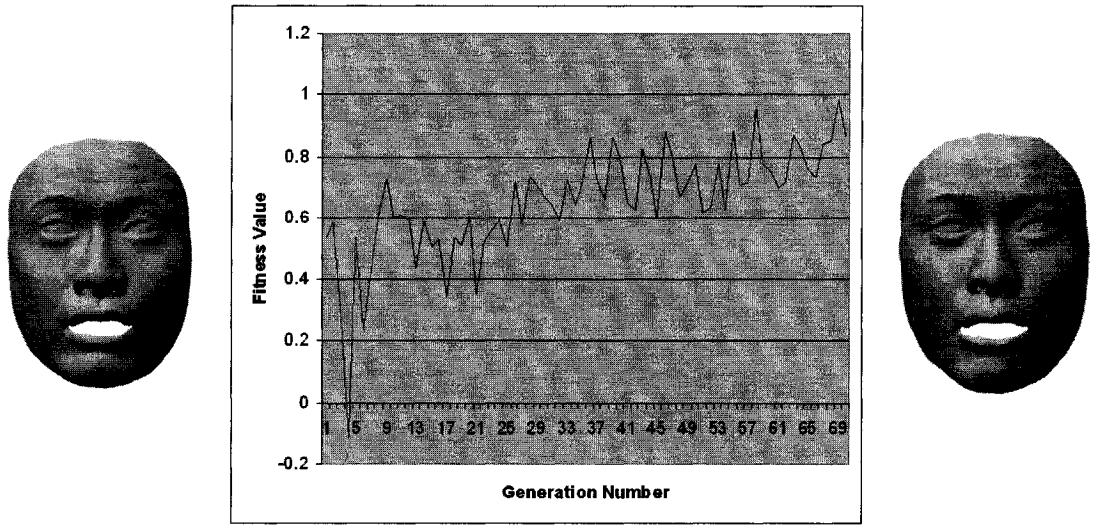
Midpoint crossover is the special case of Line crossover when $t = 0.5$ in equation (3.5), and Line crossover is also the special case of *BLX-α* crossover when $\alpha = 0$ in equation (3.6). Each of these three crossovers is capable of generating the offspring from previous two winners in the tournament and leading to the optimal solution finally. More experimental results with these three crossovers will be discussed in section 3.3.

However, one more step is needed to adjust the offspring. Previously, in the fitness function, only vector directions of FPs are considered. Now we need to adjust FP vector lengths of the offspring to meet the input requirement (i.e. specified feature point motions). We sum up the lengths of the motion vectors for the FPs in the input. Also we sum up vector

lengths for the corresponding FPs in the offspring. Then the scale between those two values is used to adjust motion vectors for all surface points of the offspring.

Notice that if the vector's direction and length are evaluated simultaneously in the evaluation function, we may fall in the dilemma to decide which of the following two cases is better: a) better vector direction but worse vector length; b) worse vector direction but better vector length. By treating the vector's direction and length separately as proposed, we ensure that we won't be in such a dilemma.

Finally, the offspring is used to replace the worst individual in the previous tournament. The original population now comes to the next generation. We iterate such process again and again until N generations are processed. It is our GA termination condition. N =70 is chosen here. It is decided by the population size and our strategy to refine the population. How the population converges to the optimized solution can be visualized in Figure 3.5(b) where X-axis denotes the generation number and Y-axis shows the fitness value of the generated offspring. We keep track of the best individual over all generations in the whole GA process. When GA process meets the termination condition, the best individual expression face so far is returned as the final surface blending result (Figure 3.5(c)). Guided by the feature point motions in Figure 3.5(a), motions for all surface points are calculated by blending examples in the databank. The optimized blending weights are calculated implicitly via GA. The generated surface blending result shown in Figure 3.5(c) is quite similar to the target expression in Figure 3.5(a).



(a) (b) (c)

Figure 3.5 : Illustration of GA evolution process (a) Original model from [Zhang L 2004] but is not an example in the expression databank; (b) Sample experiment result with Line crossover; (c) The surface blending result.

Essentially, for our GA technique, the blending weights are assigned to each expression model in the databank implicitly over the generations. The expressions which are less relative to the input requirement (i.e. specified feature point motions) are assigned less weights. The more qualified expression faces are assigned more weights and are accumulated to contribute to the final surface blending result.

3.3 GA experiments

There are a total of 384 animated models provide by Zhang *et al.* ([Zhang L 2004]). The animation data can be found in their website[4]. Each of these models is made of about 46K triangles. We have tested our feature-point guided surface blending approach with 20 randomly picked models which are not in our expression databank but from their original data. We merely use the motion vectors of the feature points from those models to drive our test. The test goal is to see whether the similar result as the original data can be generated. The experiments are done on a 2.60 GHz AMD Opteron PC with 2.0 GB of RAM. The GA process is terminated when 70 generations are evaluated. The computation time is about 35 seconds.

	Avg. of Avg. Best Fitness Value	Avg. of STD
BLX-0.2	0.962	0.0068
Midpoint	0.960	0.0057
Line	0.970	0.0075

Table 3.1 : GA experiments with various crossovers.

In the experiments, we have tried various crossovers with those test models. For each crossover, we run the GA process ten times for each model. The experiment results are

[4] http://grail.cs.washington.edu/projects/stfaces

shown in Table 3.1. *Avg. of Avg. Best Fitness Value* stands for the average of the average best fitness value. *Avg. of STD* is the average standard deviation of the fitness value for that crossover. For the BLX-α crossover operator, we tried parameter α with various values: 0.1, 0.2, 0.3, 0.4 and 0.5. According to our experience, $\alpha = 0.2$ is chosen for its better performance over others. The performances of three crossovers (Midpoint, Line and BLX- α) do not vary too much. No crossover dominates the others. Usually, the ideal operator depends on the model itself. We assume that experiment results are distributed normally. By the property of the standard deviation, two standard deviations away from the average account for roughly 95% of the data. We can get the approximate performance range of the fitness value for each crossover from Table 1. Compared with Midpoint crossover (0.949 ~ 0.971) and BLX-0.2 crossover (0.948~ 0.976), Line crossover (0.955 ~ 0.985) performs a little bit better. Therefore in the rest of the document, Line crossover is chosen.

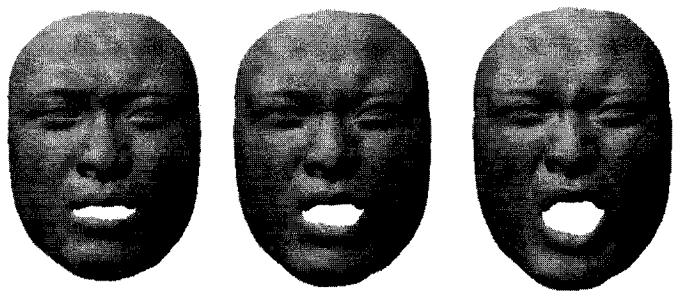
Notice that our GA process terminates when 70 generations are generated. If we permit more generations in the evolution process, BLX- α may perform better because BLX-α is trying to explore the solution space in a wider way. However, the computation time will increase if more generations are processed. The choice of the crossover operator is a compromise between accuracy and convergence speed.

Theoretically speaking, increasing the number of expression face models in the databank would lead to more accurate results. However more examples mean longer time for the overall GA process. It is also a compromise between accuracy and speed.

3.4 Enhancements

3.4.1 Emphasize level control for expression

In order to produce facial expressions of various intensities, emphasize level parameter c is used to control the generated surface blending results. The input FP motion vectors are multiplied by the value c before starting the GA process. Figure 3.6 shows different generated results by varying parameter c. As we can see, larger c value leads to more exaggerated expression.



(a) (b) (c)

Figure 3.6 : Emphasize level control parameter *c*. (a) *c* =0.7; (b) *c* =1.0; (c) *c*=1.4.

3.4.2 Face decomposition for variety of expression

Our surface blending result of the initial approach is limited to the symmetrical expressions since the example expressions in the databank are all symmetrical. i.e. the left and right halves of the facial expressions are symmetrical. Using the initial system, we may never get asymmetric results if the expressions in the databank are all symmetric. According to psychology research, a face can be split into several regions that behave as coherent units to exhibit facial expressions ([Ekman 1975]). Based on this idea, many researchers (e.g.

[Kouadio 1998] and [Fidaleo 2004]) split the face into several regions for better control of animation. Normally, they divide the face region from two to ten sub-regions.

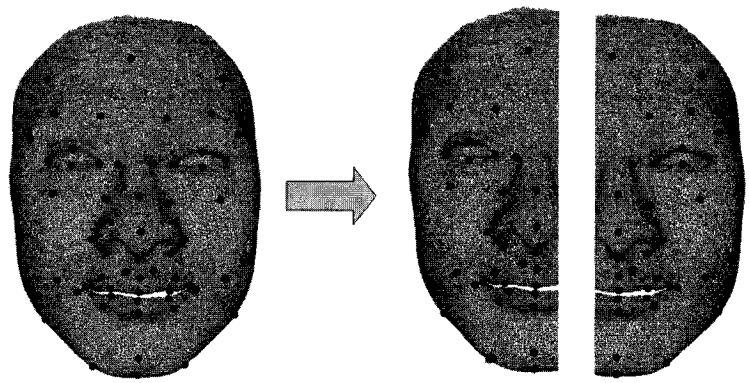


Figure 3.7 : The face is decomposed into two regions.

We firstly split a face into two parts: left and right faces (Figure 3.7). Two sub-regions overlap each other. It offers the possibility of producing asymmetric expression. We apply GA for left and right FPs separately. Then left and right blending face results are combined together to produce a synthesized facial expression. However, special treatment is needed in the joint part where artifacts may appear. One possible solution is to apply a 3D smoothing filter for each point on the joint part (i.e. average motion vectors of the surrounding points). It works fine when the left and right face expressions do not differ too much. When the difference between the left and right faces is big, unnatural expression result may appear. Kouadio *et al.* ([Kouadio 1998]) mention that points surrounding the borders should be included in both regions, and their new positions in each half expression are weighted accordingly.

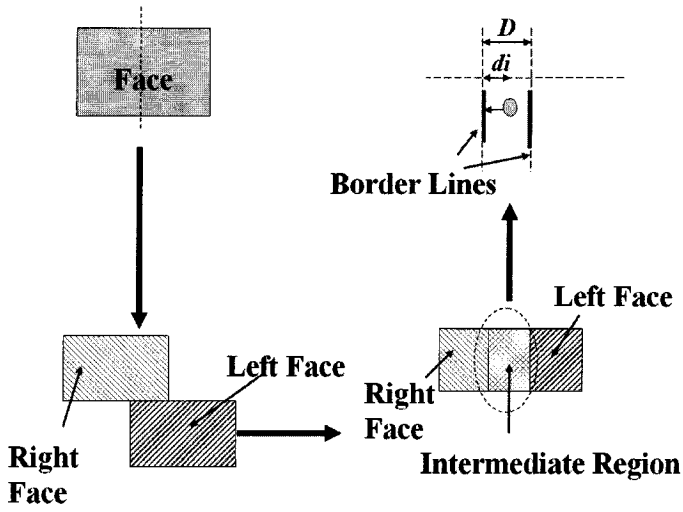


Figure 3.8 : Illustration to explain the strategy to remove the artefacts around the joint part for two sub-region decomposition.

Inspired by their idea, the following technique is used to tackle this issue (Figure 3.8). The left and right faces overlap each other around the central line. The intermediate region is designed for blending the animation vectors of the left and right faces with the following distance-related weight assigning function. The formula for calculating the motion vector of the i^{th} point in the intermediate region is given as:

$$V_i^{Intermediate} = (1 - \frac{di}{D}) \times V_i^{Right} + \frac{di}{D} \times V_i^{Left} \qquad \textbf{(3.7)}$$

where *di* is the distance of that point to the border line; *D* is the width of the intermediate region; V_i^{Left} is the motion vector for that point in the left face region and V_i^{Right} is that in the right face region. The left and right faces overlap each other in the intermediate region. They both contribute to the intermediate region. With this distance-related weight assigning function, there are no artifacts around the border lines in the results. Figure 3.9 shows some asymmetric expression results by dividing the face into left and right regions.

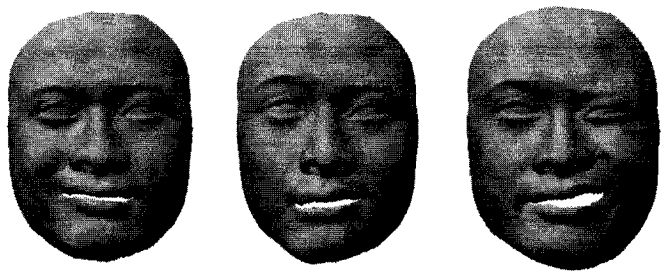
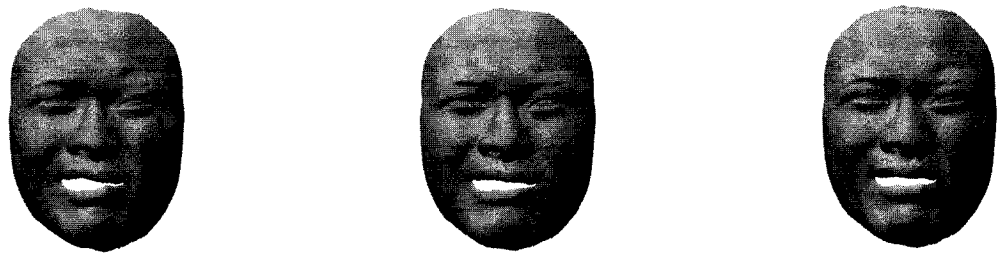


Figure 3.9 : Asymmetric expression results achieved by dividing the face into two sub-regions.

We also perform experiments to explore how the amount of overlapping affects the result. Figure 3.10 (a)-(c) shows the result with 10%-30% overlapping respectively. Notice that the percentage of overlapping means the amount of overlapping. E.g. 10% overlapping means that 10% of the whole width of the face in the left/right half face which is close to the face decomposition boundary also contributes to the calculation of the right/left half face. From Figure 3.10, firstly, we can see that the result is more close to the symmetric expression with higher overlapping percentage value. Secondly, when the left and right face expressions differ significantly (Figure 3.10 (a)), with 10% overlapping, the result is not so natural. However, for this case, with higher percentage value (e.g. 20%), the result becomes quite natural. We decide to pick 20% as the default percentage value for controlling overlapping. However, since the percentage value is quite depends on the goal expression itself, the user has the choice to control the percentage value for overlapping.



(a) (b) (c)

Figure 3.10 : Experiments to show how the amount of overlapping affects the final result. (a) 10% overlapping; (b) 20% overlapping; (c) 30% overlapping.

Next, we divide the face into four sub-regions (Figure 3.11). For each half face, it is further divided into upper and lower parts. Each quarter region is controlled by the corresponding FP sub-group. Given motion vectors of FPs in the input, we decompose it into four sub-goals. We apply GA to search solutions in the expression databank with four sub-goals separately. Then four optimized GA sub-solutions are combined together to produce the synthesized facial expression. Similarly, the previously described technique for removing the artifacts in the joint parts is also applied.

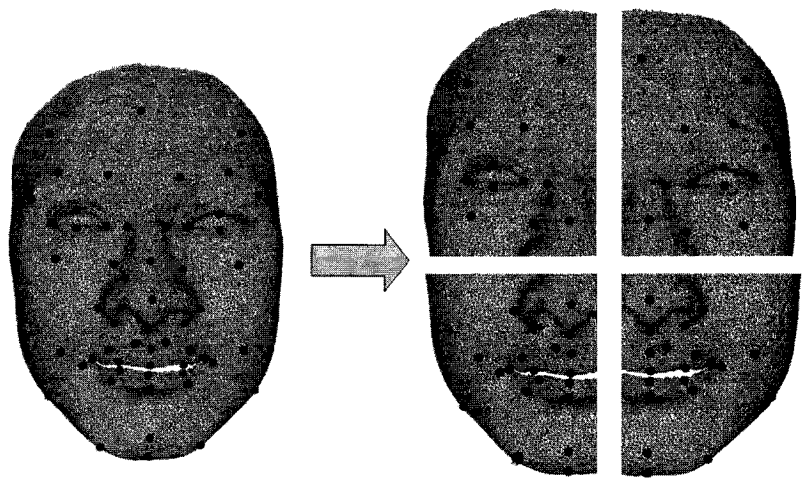


Figure 3.11: The face is decomposed into 4 sub-regions.

For the ground truth models we tested in the previous section, we conduct the experiments with Line crossover for 10 randomly selected models. Using the initial approach, we have the average best fitness value 0.960. By decomposing the face into two regions, average best fitness value is 0.972. The computation time increases from 35 seconds to 70 seconds on previously mentioned PC. With the four sub-region scheme, we have the value 0.981. The computation time is 140 seconds. It shows that, by dividing the face into several regions, we can improve the results by finding several optimal sub-solutions via GA. However the computation time increases. So it is the trade-off between accuracy and speed.

Figure 3.12 shows our facial expression results produced after making the aforementioned enhancements to our initial feature-point guided surface blending approach. A wide range of facial expressions can thus be produced from the existing databank.

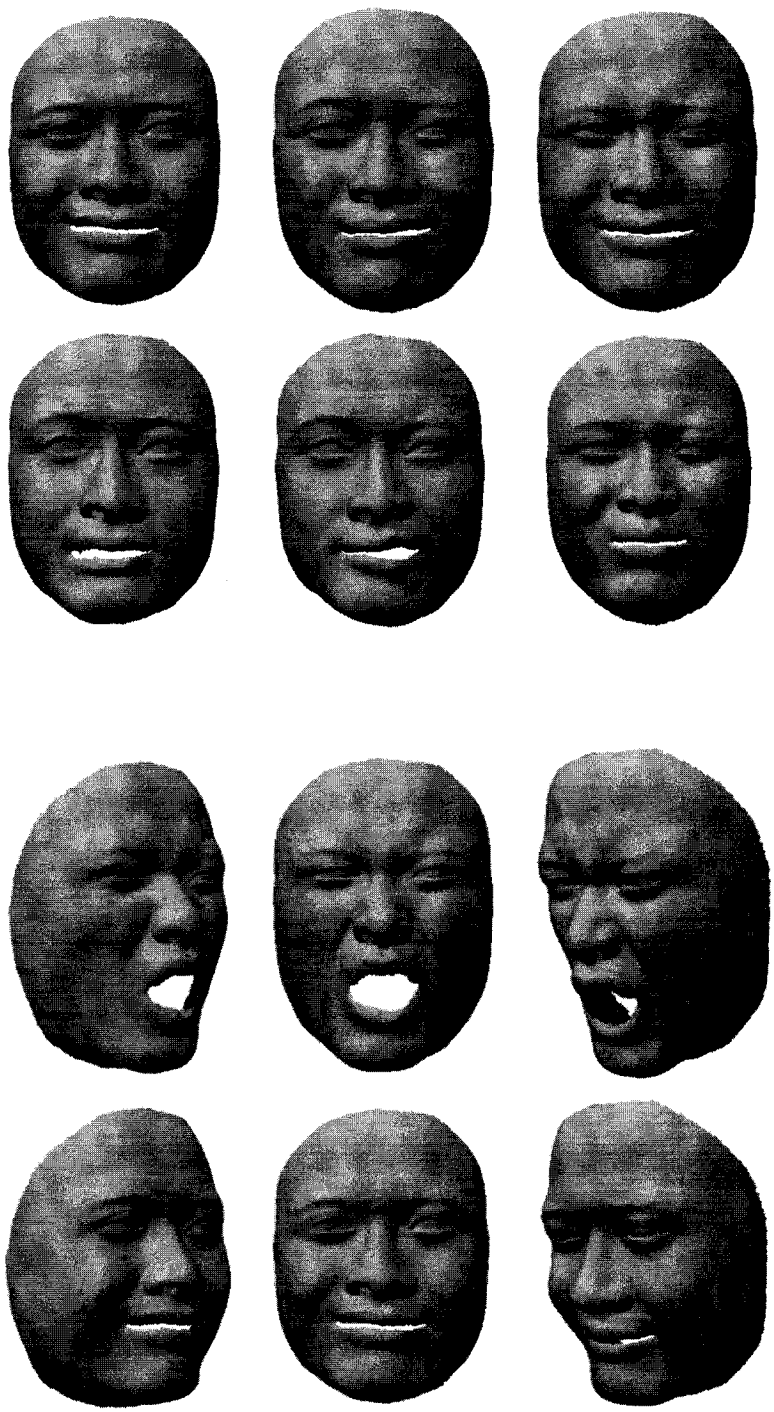


Figure 3.12 : Facial expression results achieved using feature-point guided surface blending approach.

3.4.3 GA speed optimization

Currently, our GA process is limited to the running speed. In this sub-section, some techniques are used to speed up the whole GA process.

Our first step is to optimize the databank. Previously, for each expression face model, the motion vectors on the MPEG-4 feature points are calculated when initializing the GA population pool. And this calculation is done in the runtime. We can speed up such a process by pre-computing the motion vectors on the MPEG-4 feature points for each expression model during the offline period. Therefore for each expression face model, we deposit the motion vectors for both surface points and MPEG-4 feature points into the databank. This offline process saves the running speed greatly.

Secondly, we observe that in the GA evolution process, in each generation, actually only MPEG-4 feature point motions are involved in the calculation. The surface points are only needed in the final calculation step where various expression models in the databank are blended. In the previous version, all surface points of the two parents are involved in the calculation when processing each generation. In our optimized version, in each generation, we just keep track of the following information:

- The index numbers of the two parent expression face models;
- The coefficients for blending these two parents;
- The index number of the offspring;
- The MPEG-4 feature point motion vectors for this offspring.

Let us explain the meaning of the index number. Suppose that there are a total of N models in the population pool, then the individual expression face model with the index number i is the i^{th} model in this pool.

By tracking the aforementioned information when processing each generation, we can trace back to the upper generation when the GA process terminates. By doing so, we can get the coefficients for the original expression models in the databank finally. The calculation which involves all the surface points is done at this time only. Consequently, the GA speed is improved.

After adopting previously described optimizing techniques, the computation time for the initial GA system (without decomposition) improves to eight seconds on the previously mentioned PC. For the two-region decomposition scheme, the computation time is ten seconds. For the four sub-region scheme, the computation time is sixteen seconds.

Chapter 4 Consistently Parameterized Laser-scanned Faces

In the previous chapter, we generate various facial expressions for a specific face model by blending the examples in an expression databank. However, up to now, we are limited to the face models in the databank. In other words, we only can generate facial expressions for one person which is the one in the given databank. Our next goal is to generate facial expressions for novel face models. First problem confronting us is how to model a novel human face accurately. This chapter provides a solution for this issue.

Today, there are many commercial laser-scanners specialized for capturing human faces. For example, the *Cyberware™ 3030* laser scanner [5] is an advanced technology for digitizing the human head model. In operation, this scanner (see Figure 4.1) casts a laser beam on an object to create a lighted profile. Meanwhile, a high-quality video sensor captures this profile from two viewpoints. The scanning process captures an array of digitized points, with each point represented by x, y, and z coordinates for shape and 24-bit RGB coordinates for color. In this document, a *Cyberware™ 3030* laser scanner is utilized for our research. The shape information obtained in scanning is more focused in this document rather than the color information.

[5] http://www.cyberware.com/products/scanners/ps.html

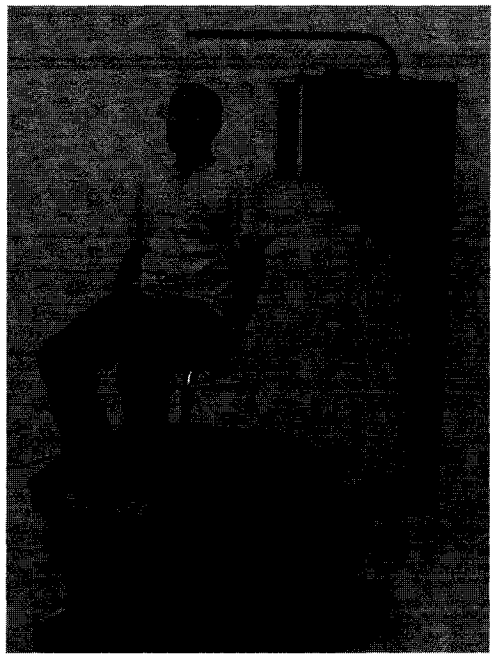


Figure 4.1 : *Cyberware*^{TM} *3030* laser scanner.

Although it precisely captures the shape of the human face, the raw laser-scanned data has many drawbacks as discussed in Section 2.2.4.2. It is difficult to operate the unorganized and inconsistent laser-scanned data directly. This chapter is intended to parameterize the raw laser-scanned data consistently. After the consistent parameterization process, all laser-scanned faces shall share the same structure. Model *A* and model *B* are said to have the same structure if they have the same vertex topology, the same number of vertices, and the same relation between vertices. *The same relation between vertices* means that for any vertex in the model *A,* the corresponding vertex in model *B* has the same meaning. For example, if the vertex 1563 represents the nose tip in model *A*, then the vertex 1563 in model *B* should also represents the nose tip point as well.

The main idea of this chapter extends the method proposed by Lee and Thalmann ([Lee WS 2000]) and the system implementation is also done by modifying the system

developed for [Lee WS 2000]. In the previous system, the inputs are the 2D orthogonal photos of the subject. In this chapter, the raw laser-scanned 3D face data is used as input.

This chapter is organized as follows. Section 4.1 prepares a generic model. The facial feature points in the laser-scanned data are detected in section 4.2. Then a three-step consistent parameterization technique is presented. Firstly, in section 4.3, the generic model is deformed using the RBF technique. Secondly, section 4.4 increases the vertex count of the deformed generic model. Thirdly, previous resulting facial mesh is further refined in Section 4.5. More results of neutral consistent facial meshes are shown in section 4.6. By adopting such a consistent parameterization technique, finally a facial motion databank is constructed in section 4.7.

4.1 Preparation of the generic model

The goal of using a generic model is to use a common geometric structure to model the human faces. All of the resulting structured models are derivatives of this generic model. Consequently, consistency is achieved among the resulting face models.

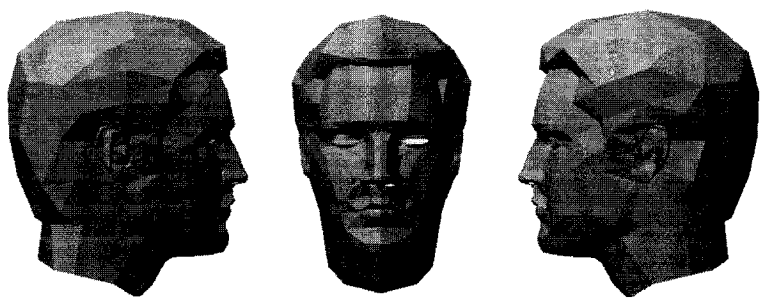


Figure 4.2 : Generic model (1,485 triangles).

Figure 4.2 shows the generic model used which comprises of 1,485 triangles. This generic model derives from the one used in [Lee WS 2000]. Previous generic model in [Lee WS 2000] has holes in its eye ball position and mouth region. However, those areas are

usually solid in our raw laser-scanned data. Therefore some minor changes in those areas are made in order to make the generic model more suitable for our laser-scanned face data.

This generic model also contains predefined feature point information which is the subset of the generic model's vertices. These feature points represent the most characteristic points of the human face model. Totally, 172 feature points are defined here. More feature points are defined here than the MPEG-4 standard since most of the MPEG-4 feature points are in the front face region. However, here we need a few more feature points in the side hair, neck, top and back head regions.

4.2 Feature point detection

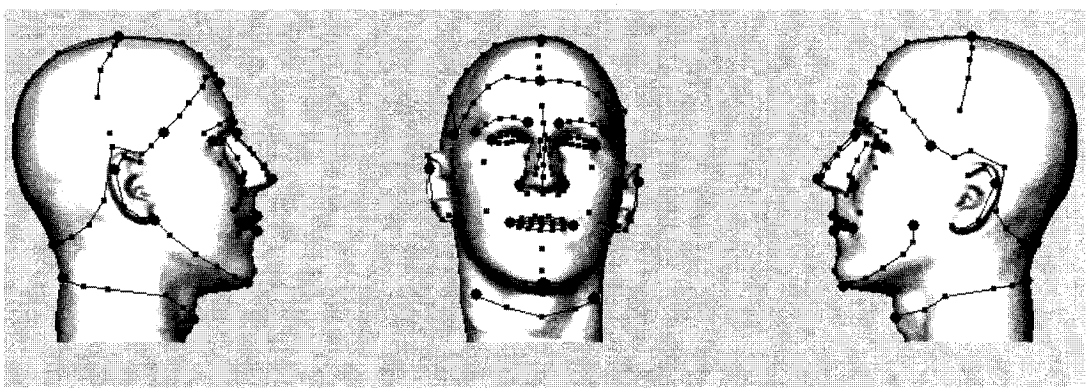


Figure 4.3 : Feature point detection for the laser-scanned data.

In this section, the goal is to capture the distinguished features of the raw laser-scanned data so that 3D feature point correspondence between the generic model and the scanned data can be established. This goal is achieved by detecting the same 172 feature points on the laser scanned data. Of those 172 feature points, twenty-eight of them are major feature points and the remaining 144 ones are minor ones. Figure 4.3 shows the major feature points in the bigger solid dots and the minor feature points in the smaller dots.

The feature point detection step requires a user to place feature point markers on the front and side images of the raw scanned data interactively. Here the semi-automatic feature point detection method is used. Semi-automatic method is named because the positions of minor feature points can be automatically calculated (by affine transformations) based on the major feature point marker positions. The positions of minor feature point markers in the side images are also automatically calculated from the front view image based on the following algorithm.

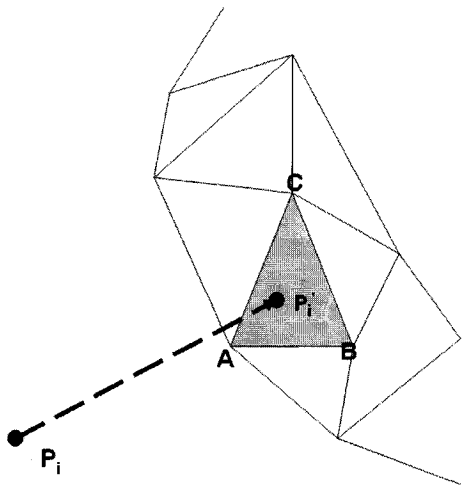


Figure 4.4 : Example of calculating depth value for the feature point automatically.

When the user places a feature point marker P_i in the front view image of the laser-scanned data, the x and y values are known, the next question is how to calculate its depth z value so that this feature point can be displayed in the side view images automatically. From P_i, a ray is cast along its orthogonal directions towards the dense laser-scanned triangle mesh. The intersection point P_i' of that ray and the laser-scanned triangle mesh can then be calculated. Figure 4.4 shows an example. P_i' should be within one specific triangle of the triangle mesh of the laser-scanned data if the intersection exists. P_i' lies within the triangle ΔABC if and only if barycentric coordinates of P_i' with respect to the triangle ΔABC are in the

range [0, 1]. Suppose z_a, z_b, z_c are then depth z values for vertices of ΔABC respectively. The z value for the point P_i is then calculated by the linear combination of z_a, z_b, z_c through barycentric coordinates.

Once all feature points are detected in the 2D images, the 2D marker positions are then used to calculate 3D feature point positions with predefined relations among points on front view and side view images. Notice that since 3D laser-scanned range data is adopted here rather than 2D photographs in the previous system for [Lee WS 2000]. Compared with the photo-based based system for [Lee WS 2000], the method here is faster and more accurate in detecting feature points.

4.3 RBF deformation

After the feature points for the laser-scanned data are detected, the rough relationship between the generic model and the raw laser-scanned data is established since the same feature points are predefined in the generic model. The next goal is to deform the generic model so that its shape matches the laser-scanned data roughly.

Section 2.3.3 introduces the concept of generating facial expressions by deforming surfaces locally using the radial basis functions (RBF) technique. In this section, the RBF technique is used to deform the face surface globally for face modeling. RBFs, known as a scattered data interpolation technique, are capable of closely approximate or interpolate smooth hyper-surfaces ([Powell 1987]). Ulgen ([Ulgen 1997]) morphs the generic model to resemble the target face based on RBFs. More detailed explanation for RBF technique can be found in [Fidaleo 2000]. In this section, the similar technique is presented to deform the generic model based on RBFs.

The radial basis functions $F(\bar{x})$ are in the following form:

$$F(\bar{x}) = \sum_{i=1}^{M} W_i H(\bar{x}, \overline{xi}) \tag{4.1}$$

where M is the number of feature points (In this case, M=172) and W_i is the weight for the corresponding feature point. $\bar{x}$ represents the input point and $\overline{xi}$ is the i[th] feature point. H is the *Hardy multi-quadric function* ([Hardy 1971]). Here the Hardy multi-quadric function is used rather than other functions because Hardy operates very stable and gives smoother result ([Fang 1996]). H is expressed as:

$$H(\bar{x}, \overline{xi}) = \sqrt{\| \bar{x} - \overline{x_i} \|^2 + s_i^2} \tag{4.2}$$

where $\| \bar{x} - \overline{x_i} \|$ is the Euclidean distance between the input point $\bar{x}$ and the feature point $\overline{x_i}$. s is the stiffness constant that reflects the separation between the feature points of the generic model. The stiffness constant is suggested by Eck ([Eck 1991]) for softer deformation where feature points are widely scattered and stronger deformation for closer separation. s_i value is chosen during the training stage and it is expressed as:

$$s_i = \min_{j \neq i} \left\| \overline{\mathbf{X}}_j - \bar{x}_i \right\| \tag{4.3}$$

where $\overline{\mathbf{X}}_j$ is the j[th] feature point in the generic model and $1 \leq j \leq M$. With initialized s, the RBF networks are trained using the generic model's feature point set $\overline{\mathbf{X}}$. During the training process, equation (4.1) actually becomes:

$$\overline{X_j'} = F(\overline{\mathbf{X}}_j) = \sum_{i=1}^{M} W_i \sqrt{\| \overline{\mathbf{X}}_j - \overline{x_i} \|^2 + s_i^2} \qquad \textbf{(4.4)}$$

where $\overline{\mathbf{X}}_j'$ is the j^{th} feature point in the laser-scanned data. Since $\overline{X'}$ are detected in the laser-scanned data in the previous section, RBF weight W can then be calculated. The system is solved by LU decomposition to obtain W. The decomposition is done only once at initialization as suggested in [Fidaleo 2000].

In the training stage, we compute the weights of the RBF representing the mapping between source and target feature points. In our case, source is the generic model and the target is the raw laser-scanned data. After training, each point of the generic model is used as the input point $\overline{x}$ in equation(4.1). Therefore, non-feature points are actually interpolated using the weights computed from the feature points. Consequently, the generic model is deformed to approximate the raw laser-scanned data.

Figure 4.5(a) is the raw laser-scanned face data and Figure 4.5(b) is the generic model. Figure 4.5(c) shows the deformed generic model using the RBF technique. How to derive models in Figure 4.5(d) and Figure 4.5(e) will be discussed in the following sections.

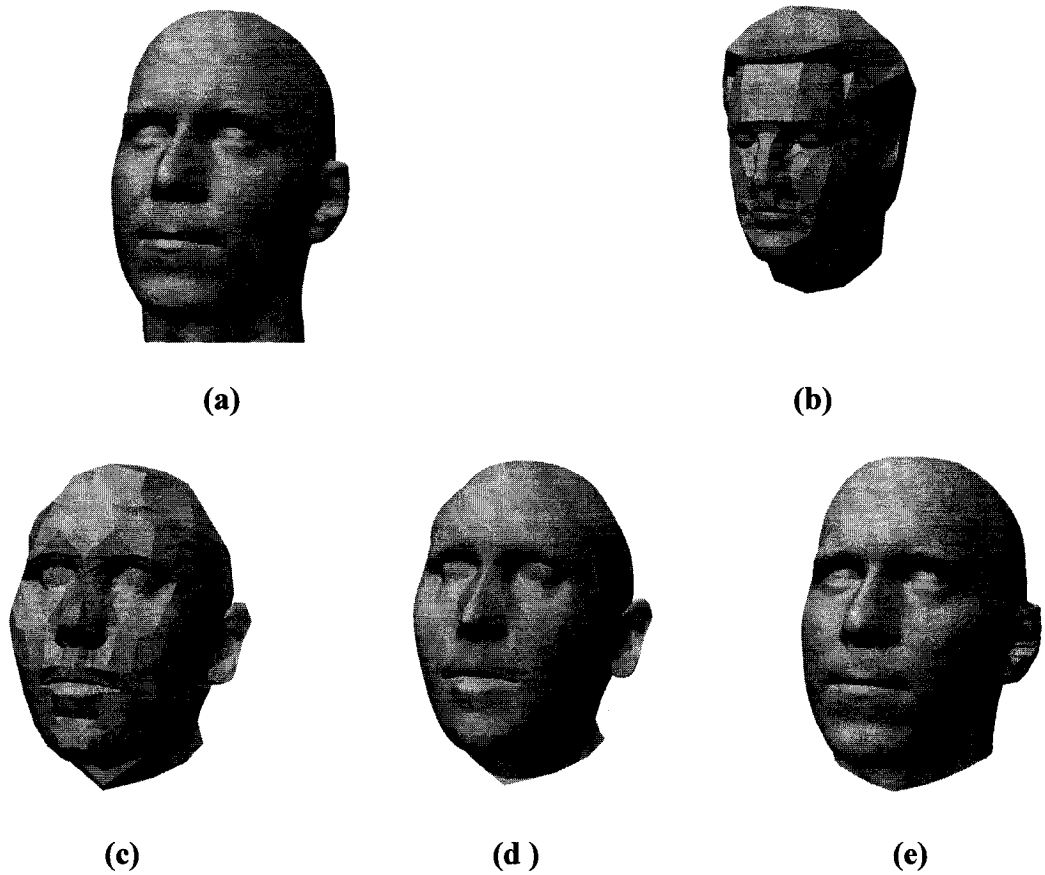


Figure 4.5 : Construction of consistent mesh from raw laser-scanned face data (a) Original laser-scanned face data (699,392 triangles); (b) Generic model (1,485 triangles); (c) Deformed generic model (1,485 triangles); (d) Preliminary consistent mesh (23,760 triangles); (e) Consistent mesh obtained after mesh refinement (23,760 triangles).

4.4 Subdivision

In the previous section, we approximate the raw laser-scanned face data by deforming the generic face model. As the derivative of the generic model, the deformed generic model also contains 1,485 triangles. However, it is not enough for our face modeling purpose. This triangle count is not big enough to model the details on the facial skin. We need to increase the number of triangles of the deformed generic model. A simple way to accomplish this is to repeatedly apply a subdivision scheme on the triangle mesh until the desired resolution is

achieved. The advantage of subdivision is that it is capable of increasing local complexity without adding global complexity.

In this section, the deformed generic model from previous section is subdivided twice using Loop's subdivision scheme ([Loop 1987]). In other words, the triangle count increases by a factor of eight. The resulting model shown as Figure 4.5(d) is named *preliminary consistent mesh* since one more refinement step is needed. The *preliminary consistent mesh* consists of 23,760 triangles.

Notice that there are some other schemes: e.g., progressive mesh by Hoppe ([Hoppe 1996])m, normales meshes by Guskov *et al.* ([Guskov 2000]) and displaced subdivision surfaces by Lee *et al.* ([Lee A 2003]). Since the triangulation of the original generic face mesh is already optimized in the preparation step, here we just look for a simple scheme which is capable of increasing the number of triangles globally. Therefore we adopt Loop's subdivision scheme instead of the others.

4.5 Mesh refinement

Although the subdivision scheme increases the triangle count, the accuracy of the resulting *preliminary consistent mesh* is still unsatisfactory since subdivision only updates the local geometry without changing the global geometry a lot. One more step is needed to refine the mesh and improve the accuracy. We can project a ray from each point of the *preliminary consistent mesh* onto the raw laser-scanned data. The intersection of the ray and the laser-scanned data lies on the actual surface being represented. Each point of the *preliminary consistent mesh* is then displaced to the corresponding intersection point. By doing so, the accuracy is improved.

The following two sub-sections will discuss and compare two ray projection schemes for mesh refinement. Section 4.5.1 presents normal projection and section 4.5.2 discusses cylindrical projection.

4.5.1 Normal projection

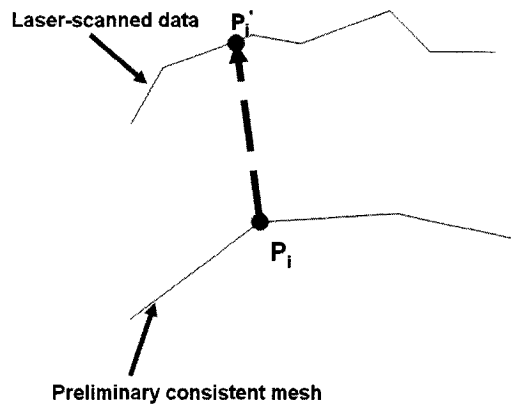


Figure 4.6 : Illustration for explaining normal projection.

For the normal projection scheme, from each point P_i in the source model (i.e. *preliminary consistent mesh*), a ray is cast along its vertex normal direction. The intersection point P_i' of that ray and the target model (i.e. laser-scanned data) is then used to update the original point P_i in the source model.

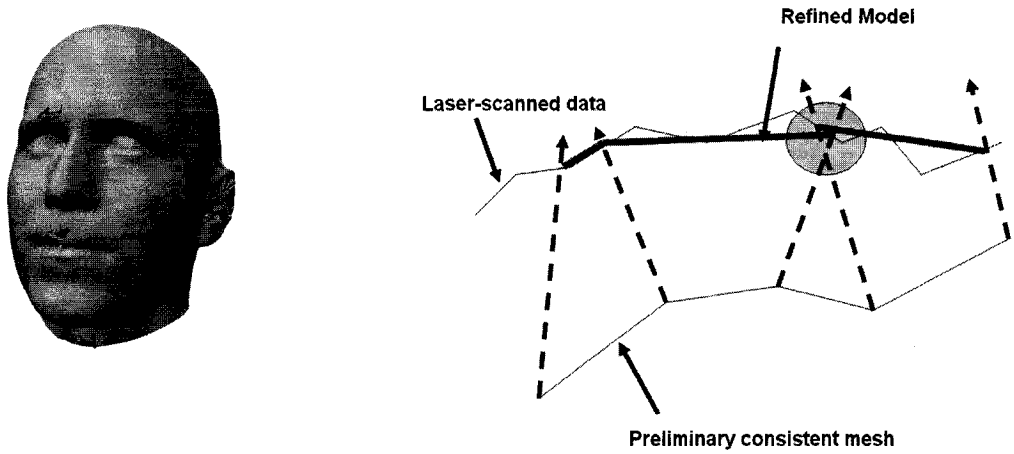


Figure 4.7 : Normal projection scheme. (a) Mesh refinement result using the normal projection scheme; (b) Illustration of the self-folding example.

Figure 4.7(a) shows the mesh refinement result using the normal projection scheme. Some holes and artifacts appear near the eye and mouth regions. Because the *preliminary consistent mesh* is just the coarsely approximated version of the original laser-scanned data, the *preliminary consistent mesh* may be quite far from the scan data. The self-folding problem might happen especially when local change is sharp in the areas like eye and mouth. Figure 4.7(b) illustrates one self-folding example which may lead to the problem. A more advanced projection scheme is discussed in the following sub-section.

4.5.2 Cylindrical projection

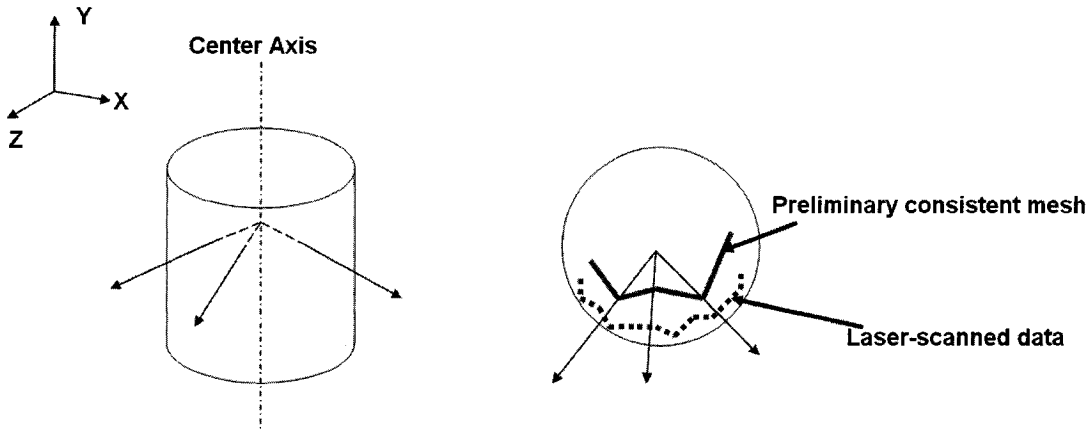


Figure 4.8 : Illustration for cylindrical projection.

Cylindrical projection scheme is used in some recent works such as [Noh 2001]. For the cylindrical projection scheme, the center axis is defined as a line passes through the center of the head model and is parallel to Y-axis. A ray perpendicular to the center line passes through each point of the source (i.e. *Preliminary consistent mesh*) and intersects with the target (i.e. Laser-scanned data). The intersection point is then used to update the original point in the *preliminary consistent mesh*. A user-defined threshold is set to avoid large displacement. The resulting model after mesh refinement using the cylindrical projection scheme is termed *consistent mesh* in this document (Figure 4.5(e)). As we can observe, the resulting *consistent mesh* bears closer resemblances to the raw laser-scanned data than the *preliminary consistent mesh* does.

Cylindrical projection scheme leads to better result than normal projection scheme in this case. This is because the human head is in somewhat cylindrical shape. The problem in the previous sub-section does not appear here because the cylindrical projection rays are defined relative to the same center axis. The rays do not intersect one another unless they are

parallel. Based on this, in this document, the cylindrical projection scheme is chosen instead of the normal projection scheme.

4.6 Experimental results

The resulting *consistent mesh* after mesh refinement is the derivative of the predefined generic model. As shown in Figure 4.5(e), it bears very close resemblance to the raw laser-scanned data. Compared with the raw scan data (Figure 4.5(a)), the resulting *consistent mesh* has far less triangles while still conveying the distinguished features in the original laser-scanned data.

Four scanned faces are used in the experiments. The faces were scanned with the *Cyberware*^{TM} *3030* laser scanner. Figure 4.9 shows them in both flat shading mode and wire-frame mode. These scanned faces contain from six to seven million triangles. They have different topologies, polygon counts and shapes.

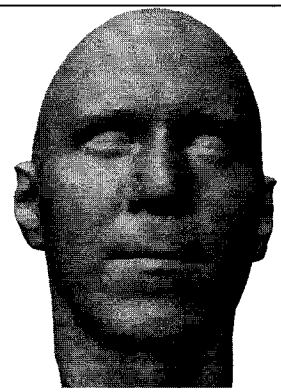
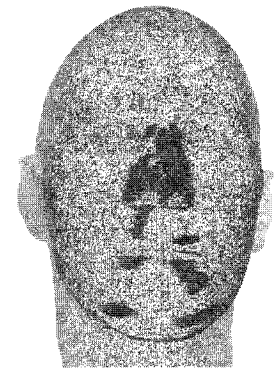
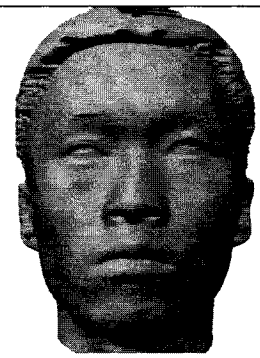
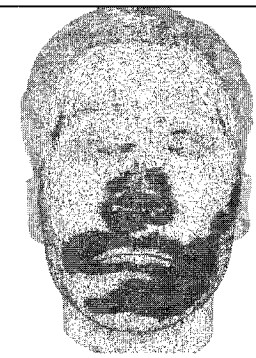
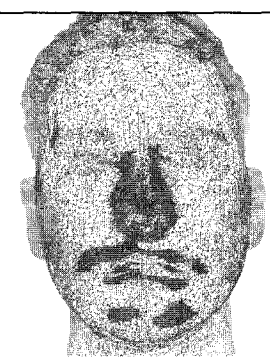
Person A (699,392 triangles)		
Person B (660,540 triangles)		
Person C (677,322 triangles)		
Person D (711, 350 triangles)		

Figure 4.9 : The laser-scanned face data used for experiments.

Figure 4.10 shows consistent parameterization results using the technique presented from section 4.2 to section 4.5. Note that the hair and back head structures are removed in the final results. This is due to the following reasons. Firstly, different people have different hair styles. And the laser-scanner is difficult to capture tiny hair structures. It is hard to maintain the consistency in the hair region from person to person. Secondly, the laser-scanner can not capture the structure of ears accurately especially when the ear is occluded by the side hairs. Thirdly, the laser-scanner can not capture the top head region correctly since the laser light is reflected away for that region. Lastly, main facial region is the focus of this document. Therefore we decide to remove those structures. Since the models are consistently parameterized now, the unwanted structures can be easily removed in the consistent way. Each of the resulting consistent meshes now contains 13,733 triangles (Figure 4.10).

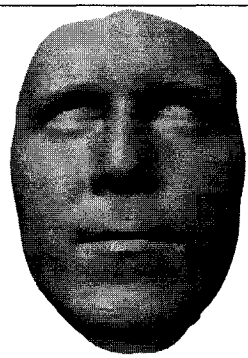
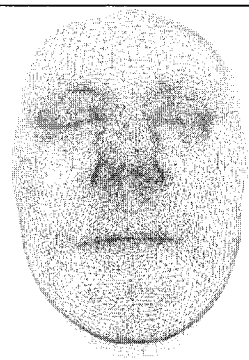
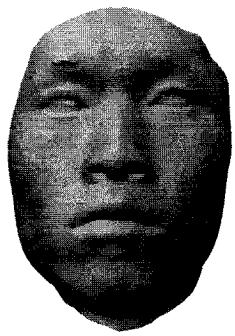
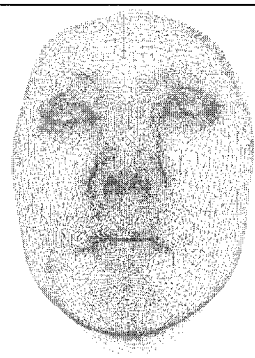
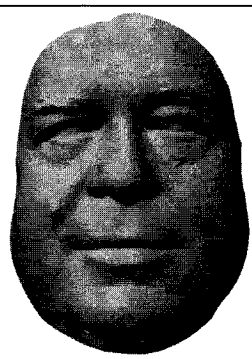
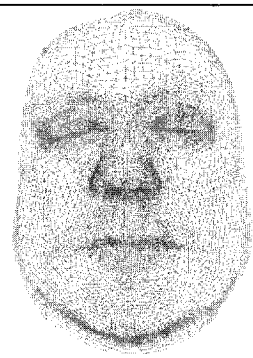
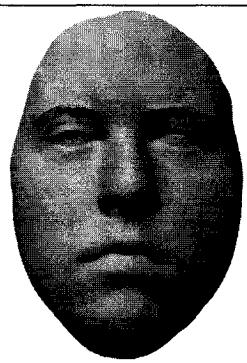
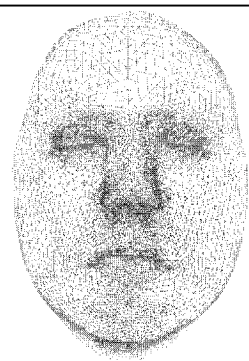
Person A (13,733 triangles)		
Person B (13,733 triangles)		
Person C (13,733 triangles)		
Person D (13,733 triangles)		

Figure 4.10 : Consistent parameterization results.

4.7 Constructing facial motion databank

In the previous section, neutral scanned faces are consistently parameterized. This section is to scan a person when he performs various expressions and visemes statically (here Person A in Figure 4.9 is used as the model). Totally, eleven expressions and fourteen visemes of this person are scanned. (The scanned visemes respect the MPEG-4 viseme table shown in Table 2.1.) Examples of scanned expression models are depicted in Figure 4.11. Examples of scanned viseme models are shown in Figure 4.12.

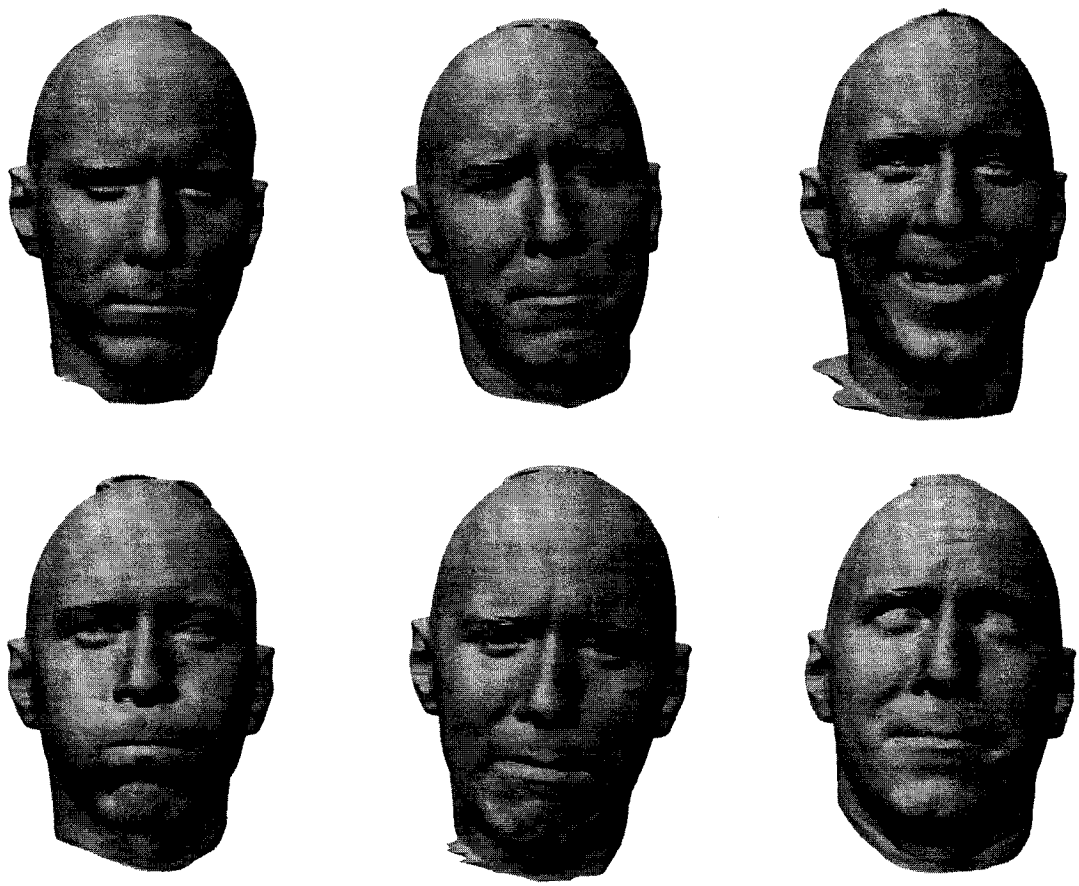


Figure 4.11 : Example laser-scanned data for Person A when he performs expressions.

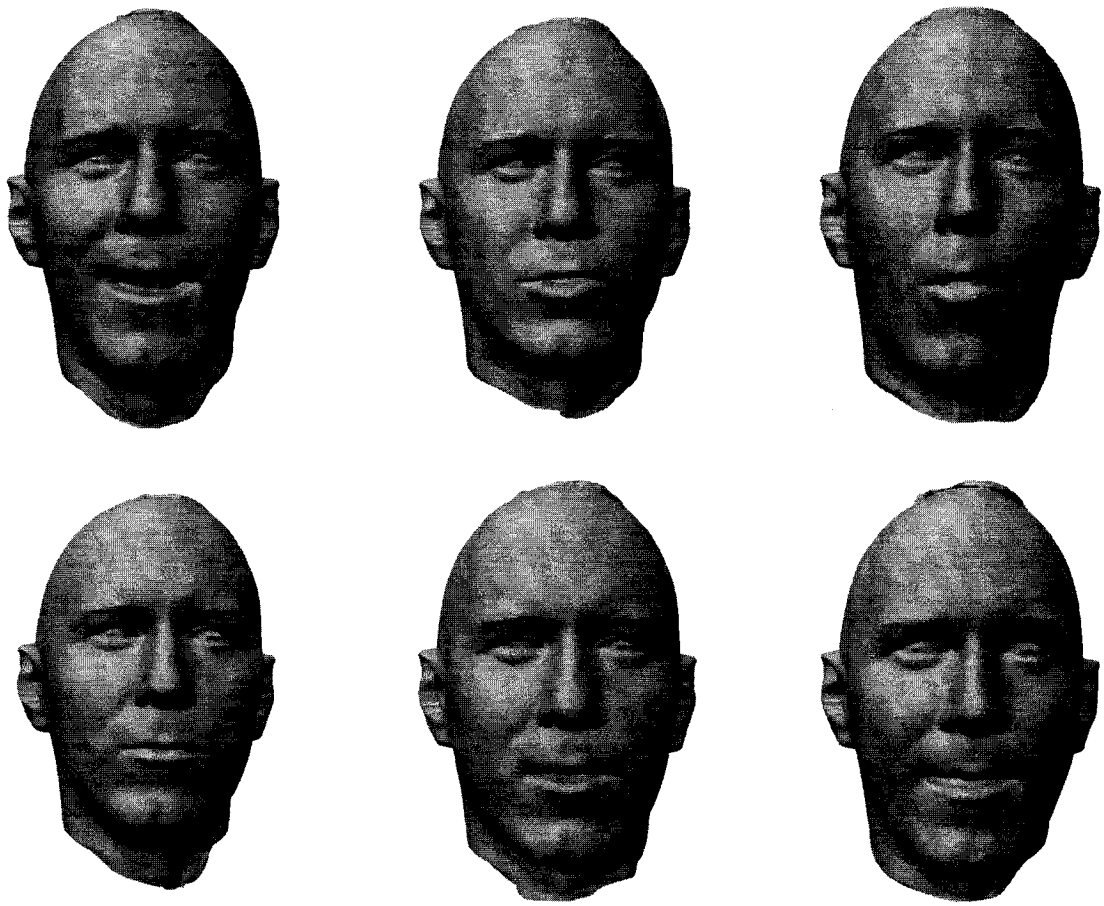


Figure 4.12 : Example laser-scanned data for Person A when he performs visemes.

After laser scanning, the consistent parameterization technique presented previously is used to process those raw laser-scanned expressions and visemes. Figure 4.13 shows the examples of the resulting consistent meshes of various expressions. Figure 4.14 depicts the examples of the resulting consistent meshes of various visemes. Notice that in Figure 4.11 and Figure 4.12, raw laser-scanned faces have different orientations. However in Figure 4.13 and Figure 4.14, the resulting consistent facial meshes are in the same orientation. The detail on how to correct the face orientation is explained as follows.

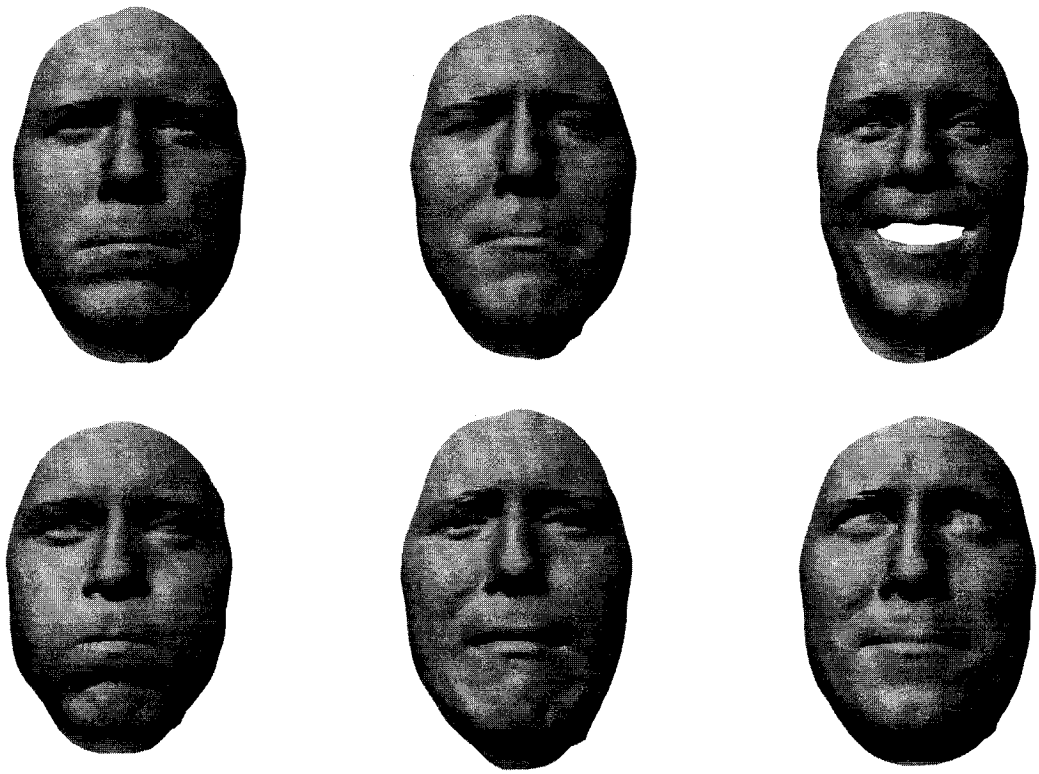


Figure 4.13 : Example expressions (consistent parameterization results).

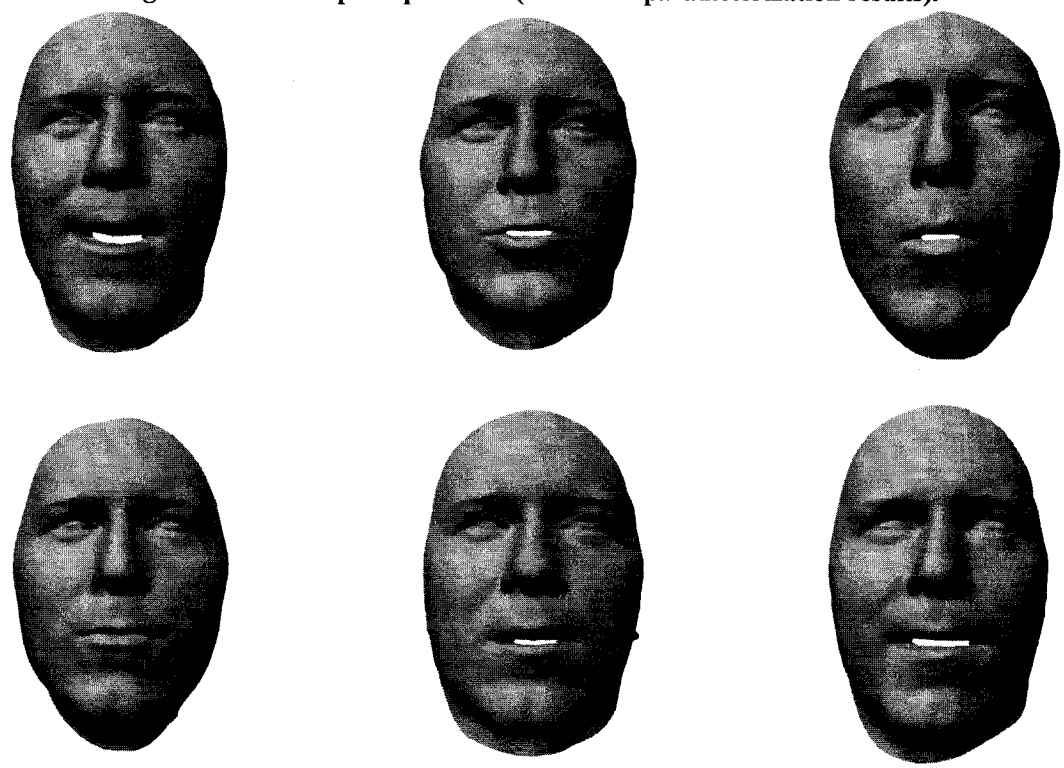


Figure 4.14 : Example visemes (consistent parameterization results).

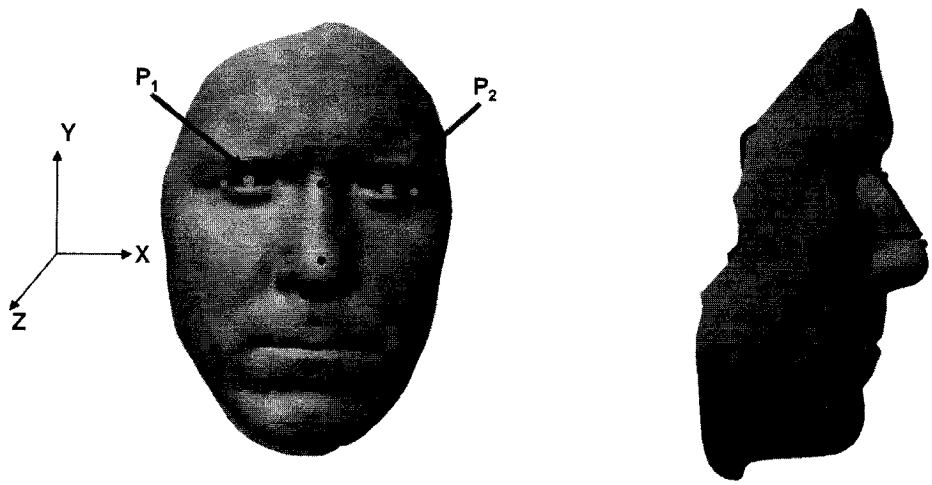


Figure 4.15 : Illustration on how to correct the face orientation.

After the consistent parameterization step, the face models are consistently constructed. Therefore the feature point indexes are easily known in the consistent way. From the eye corner feature points, the middle of the eye points P_1 and P_2 in Figure 4.15 can be calculated. The vector from P_1 to P_2 is defined as $\overrightarrow{P_1P_2}$. From the two nose ridge points, the angle α shown in the side view image of Figure 4.15 can be calculated as well. Then the algorithm to correct the face orientation consistently is defined as follows:

1. Rotate around X-axis: the angle α shown in the side image of Figure 4.15 is used as the reference angle for rotation. For all the expression and viseme models of the same person, this angle α is adjusted so that it remains constant.
2. Rotate around Y-axis: the angle for rotation is the angle between *the orthographic projection of* $\overrightarrow{P_1P_2}$ *on the x-z plane* and *the X axis*;
3. Rotate around Z-axis: the angle for rotation is the angle between $\overrightarrow{P_1P_2}$ and *the orthographic projection of* $\overrightarrow{P_1P_2}$ *on the x-z plane.*

After performing the aforementioned algorithm, the consistent meshes all have the same orientation as shown in Figure 4.13 and Figure 4.14. We can then construct the facial motion databank consisting of these consistent meshes. Totally 25 consistent meshes for Person A (eleven expressions and fourteen visemes) are used. Traditionally, animators manually craft key models and generate facial animation by interpolating those models. Significant manual work is required even using the available commercial softwares like *3Ds MAX* and *MAYA*. Here the laser-scanner is utilized to acquire the 3D range data and the facial motion databank consisting of consistent meshes is easily constructed. This databank is useful for later chapters.

Chapter 5 Facial Motion Retargeting

It is shown in Chapter 3 that guided by feature point motions, facial expressions can be generated by blending examples in the databank. Later in Chapter 4, the technique for modeling the novel face model has been shown by processing the raw laser-scanned data consistently. Our facial motion databank is also constructed by scanning and then consistently parameterizing various expressions and visemes performed by one subject. Because for each scan, the Cyberware scanner requires the subject to hold the intended pose for almost thirty seconds, it is time-consuming to scan various expressions for every novel subject. Also according to our scanning experience, not every person can perform every key expression successfully. On the other hand, when an expressive animation database is available, it is beneficial to clone that expressive data to our neutral face model. In short, it is very necessary to develop the approach to clone facial motions from one person to another. This is the goal of this chapter.

This chapter covers an efficient technique for retargeting facial motions. It shares the similar underlying expression cloning technique presented in [Noh 2001]. Firstly, a trivial case of transferring facial motions between consistent meshes is discussed in section 5.1. Then a technique is shown to transfer facial motions from non-consistent meshes in section 5.2. Here *non-consistent meshes* mean that they have the different structure as ours.

5.1 From consistent meshes

In Chapter 4, the consistent parameterization technique is shown to parameterize the raw laser-scanned data. The advantage of it is that all resulting facial meshes have the same structure, making it easier to transfer facial motions between consistent meshes.

Our goal is to transfer the facial motions between different person's face models. First question is how to obtain the facial motions in the source models. For any animated source model *SA* and suppose model *SN* is the source model in the neutral status, since they are consistently parameterized, the face motion vector V_i for each point between these two models can be expressed as:

$$V_i = SA_i - SN_i \qquad \textbf{(5.1)}$$

where SN_i and SA_i are the corresponding points in the source neutral model *SN* and animated source model *SA* respectively.

After facial motions for all facial surface points are obtained, they can be added to a novel face model directly if that novel face model is consistently parameterized in the same scheme. We name the novel person's neutral face model as the target neutral model *TN*. This is the benefit of consistent parameterization. The facial motion vectors can be transferred between consistent meshes (*i.e.* between the source neutral model *SN* and the target neutral model *TN*) easily. However, a scale factor has to be applied during retargeting since the source neutral model and the target neutral model are in the different sizes.

The following part explains how we decide the scale factor between them. Since the source neutral model and the target neutral model are both consistent meshes, the feature point indexes are easily known consistently. Figure 5.1 shows the feature points used for calculating the scale factor. The eye corner feature points are used for calculating the middle points of the eyes. Those feature distances are then used to decide the final scale factor. The final scale factor *S* is given by:

$$S = \frac{1}{4} \times \frac{x1}{x1'} + \frac{1}{4} \times \frac{x2}{x2'} + \frac{1}{4} \times \frac{y1}{y1'} + \frac{1}{4} \times \frac{y2}{y2'} \quad (5.2)$$

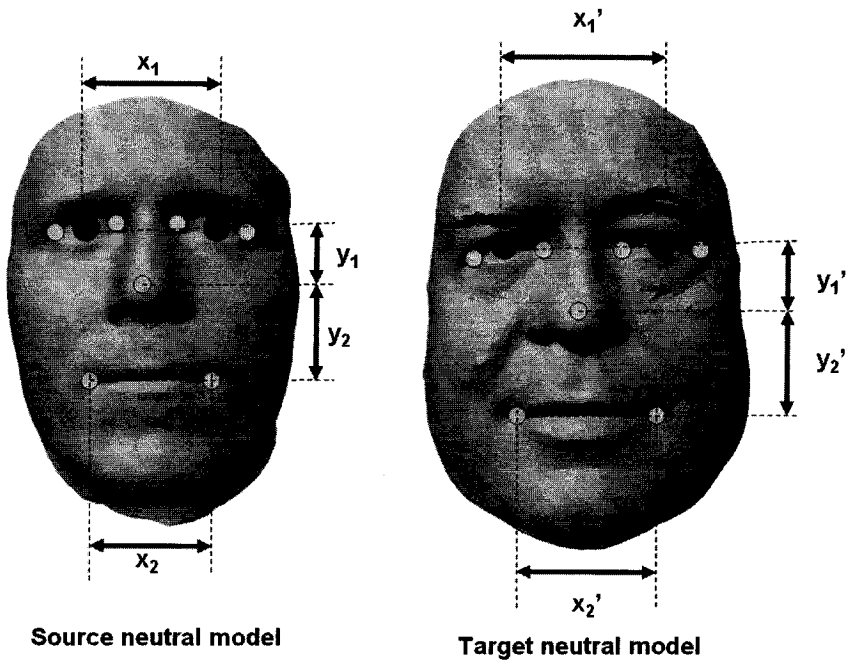


Figure 5.1 : Illustration of calculating the scale factor to retarget motion vector from a source to a target.

Now let us further discuss why the average of the four sub-scales is used here. In section 4.2, the feature point positions are defined manually. In other words, each of these sub-scales is calculated on the basis of this manual step. Therefore if the motion vectors are adjusted based on either sub-scale, the result may not be so accurate. The averaging equation (5.2) serves to decrease the amount of uncertainty. The four distance scales contribute equally in deciding the final scale factor *S*. We perform experiments on Person B, C and D in Figure 4.10. Experimental results are shown as Table 5.1. The results show that the four sub-scale values of a given person do not vary too much. The purpose of adopting the averaging scale *S* is for decreasing the amount of uncertainty.

	Person B	Person C	Person D
$x_1/x_{1'}$	1.082661	1.069508	1.017232
$x_2/x_{2'}$	0.964864	1.123640	0.947810
$y_1/y_{1'}$	1.293260	1.138791	1.161792
$y_2/y_{2'}$	0.973252	0.931445	0.960958
S	1.078509	1.065846	1.021948

Table 5.1 : Experiments for finding the scale for facial motion retargeting.

Figure 5.2 shows the results of transferring facial motions from models in the facial motions databank (Figure 5.2(a)) to consistent meshes. Figure 5.2(a) depicts example models from previously constructed databank in section 4.7. The first column of Figure 5.2 is the neutral models which are obtained by performing the consistent parameterization process on the raw laser-scanned data. Each of the resulting facial meshes contains 13.7K triangles. As shown in Figure 5.2, facial motions can be transferred into other neutral consistent meshes successfully.

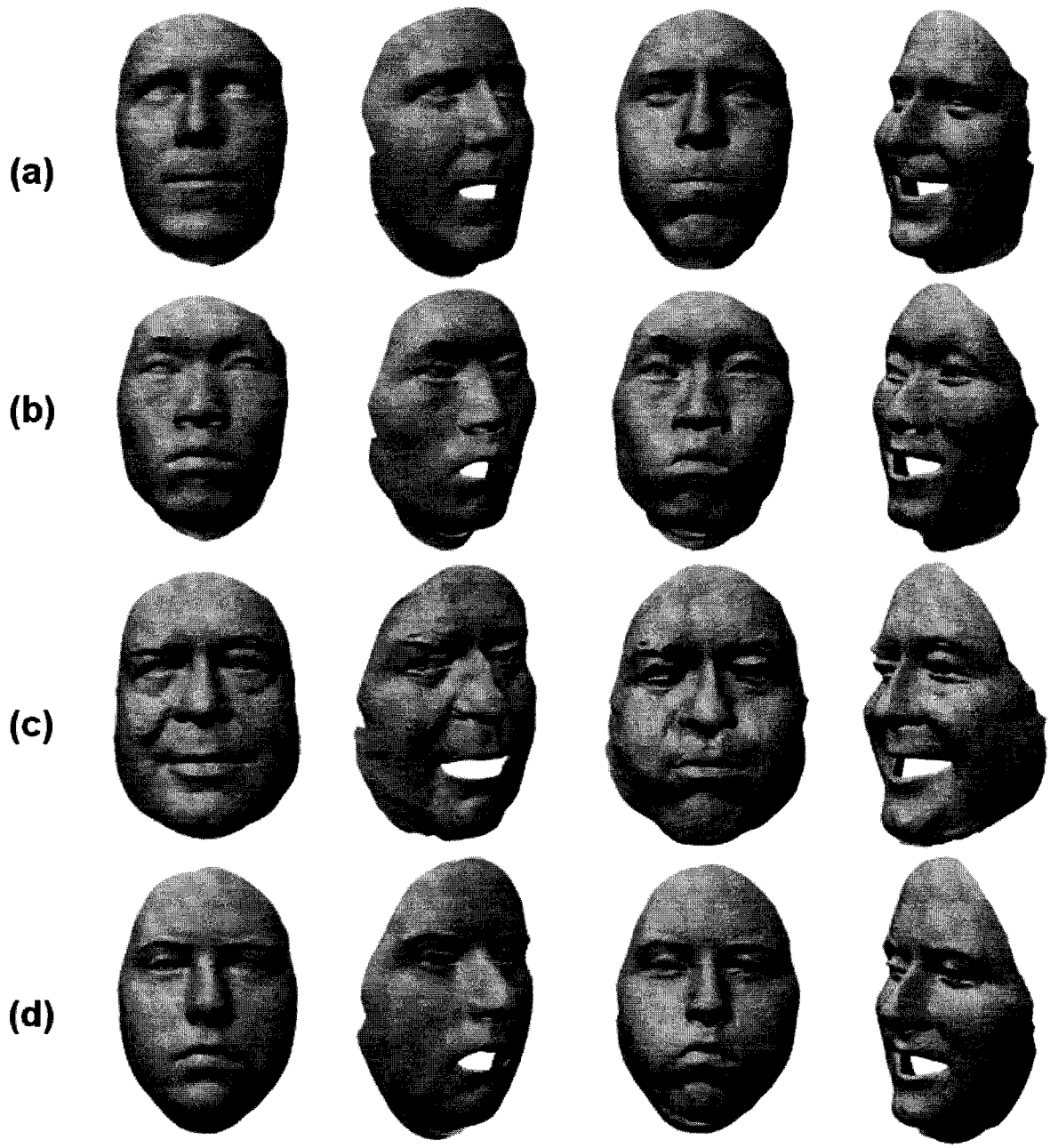


Figure 5.2 : (a) Source expressions from the facial motion databank; (b)-(d) : Facial motions in (a) are transferred to the neutral target models.

5.2 From non-consistent meshes

In section 5.1, facial motions are transferable between consistent meshes. However, it is just the ideal case. If we have another expressive animation data available and we want to make use of it for our neutral consistent meshes, but the facial mesh in the other available data usually does not have the same structure as our consistent meshes. This section will outline the idea to tackle this issue of retargeting facial motions from non-consistent meshes. Here the *facial mesh A* is called a non-consistent mesh if it satisfies both of the following two conditions:

1. If the *facial mesh A* has the different structure from our consistent meshes.
2. For any one *facial mesh A'* from the same database of the *facial mesh A, the facial mesh A* and the *facial mesh A'* should have the same structure;

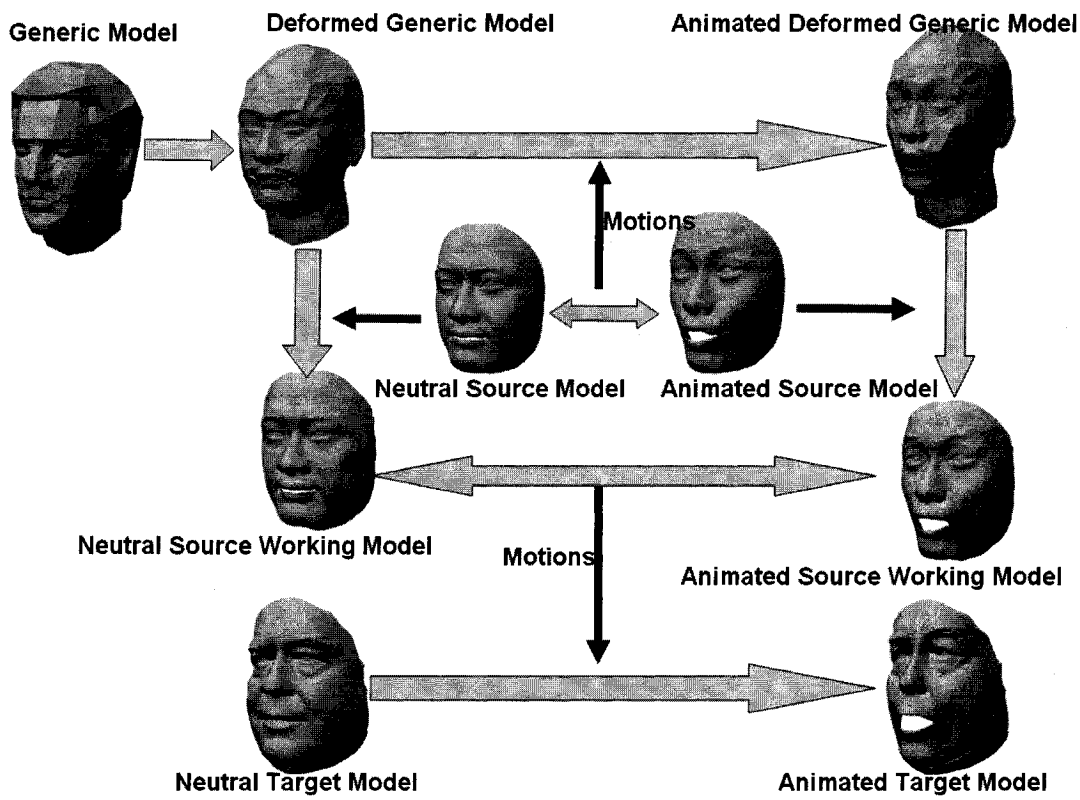


Figure 5.3 : The framework of retargeting facial motions from non-consistent meshes.

In this section, the existing animation data from Zhang *et al.*'s website[6] is utilized. Figure 5.3 illustrates the solution for retargeting facial motions from non-consistent meshes. Given the n*eutral source model* and the *animated source model* in the animation data, the goal is to animate the *neutral target model* and get the corresponding *animated target model.* The motion vectors between the *neutral source model* and the *animated source model* can be obtained by equation **(5.1)**. The *neutral target model* is produced by consistently parameterize the corresponding raw laser-scanned data. Using the same consistent parameterization technique, the n*eutral source working model* can be produced which has the same structure as the *neutral target model* (i.e. they are consistent meshes now).

The next task is to re-sample the motion vectors obtained from the animation data onto our consistent meshes. When the *neutral source working model* is generated, the *deformed generic model* is produced right after deforming the *generic model* using RBF technique (section 4.3). For each point P_i in the *deformed generic model*, we need to find the corresponding point P_i' in the *neutral source model.*

[6] http://grail.cs.washington.edu/projects/stfaces/

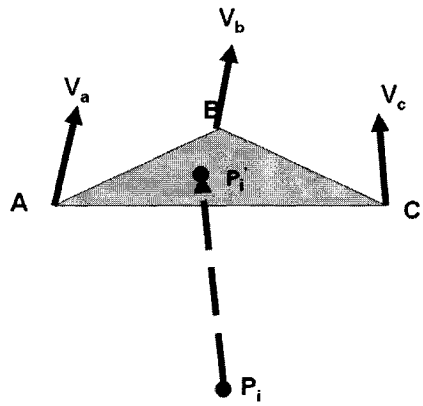


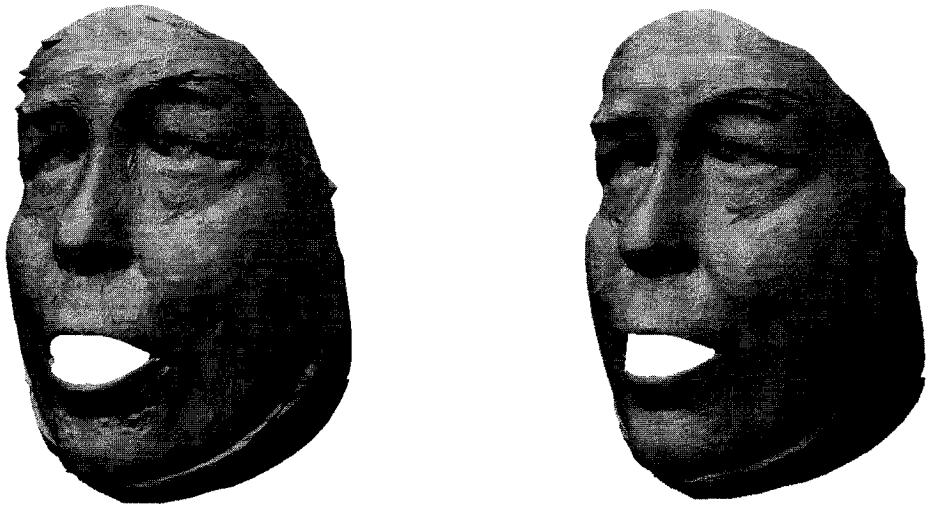
Figure 5.4 : Example of getting motion vector for a specific point P_i in the *deformed generic model.*

The *deformed generic model* is the approximation of the *neutral source model* in the low-resolution level. Using the nose tip as the reference point, the *deformed generic model* can be roughly aligned to the *neutral source model*. Then from each point P_i in the *deformed generic model*, a ray is cast along its vertex normal direction. The intersection point P_i' of that ray and the *neutral source model* can be calculated. Of course, not every point P_i in the *deformed generic model* has the corresponding point P_i' in the *neutral source model* (e.g. the point in the back head or ear area). Figure 5.4 shows an example point P_i where the corresponding intersection point P_i' could be found. P_i' should be within one triangle of the triangle mesh of the *neutral source model*. P_i' lies within the triangle ΔABC if and only if barycentric coordinates of P_i' with respect to the triangle ΔABC are in the range [0, 1]. Suppose V_a, V_b, V_c are motion vectors for vertices of ΔABC. The motion vector for the point P_i in the *deformed generic model* is then calculated by the linear combination of the motion vectors V_a, V_b, V_c through barycentric coordinates.

Once the motion vectors for the points in the *deformed generic model* are calculated, the *animated deformed generic model* (see Figure 5.3) can be produced. The consistent parameterization technique is adopted to process the *animated source model*. As a result, the

animated source working model is produced (see Figure 5.3). Notice that different from Chapter 4 where feature point detection step is necessary for the raw laser-scanned data, here the motion vectors from the animation data are used in the consistent parameterization process. The assumption is that the facial meshes in the animation data have the same structure. In other words, the *neutral source model* and the *animated source model* have the same structure.

The motion vectors between the *neutral source working model* and the *animated source working model* can then be calculated easily. Then since they share the same structure as our consistent meshes, motion vectors can be transferred to the *neutral target model* using the technique described in section 5.1. However experimentally one more step is needed before generating the *animated target model.* Figure 5.5(a) shows the resulting animated model where some noises appear. Such a problem did not appear in section 5.1 but appear now possibly because of the motion vector re-sampling process. Therefore an extra step is performed here where a smoothing filter is applied. For each point P_i in the n*eutral target model*, the motion vectors for it and its adjacent points are collected and then averaged. Here an adjacent point means any point that shares the same edge as the point P_i. In other words, they are neighboring points. The averaged motion vector for each point P_i is then applied instead of the original re-sampled motion vector. Figure 5.5(b) shows the effect after applying the smoothing filter. As we can see, the annoying noises disappear.



(a) (b)

Figure 5.5 : Example of using the smoothing filter. (a) No smoothing filter is applied (b) The smoothing filter is applied.

Figure 5.6 illustrates results obtained by transferring expressions from the existing animation data. Figure 5.6(a) shows example models in the available animation data. Each of these models is made of about 46.9K triangles. The first column of Figure 5.6 is the neutral models where the bottom three models are obtained by performing the consistent parameterization process on the raw laser-scanned face data. Each of our models has 13.7K triangles. Figure 5.6 shows that the facial motions in the existing animation data can be transferred into our neutral consistent meshes successfully, although they have different structures. This is crucial since we can make use of any existing expressive animation data from now on, regardless of the structures.

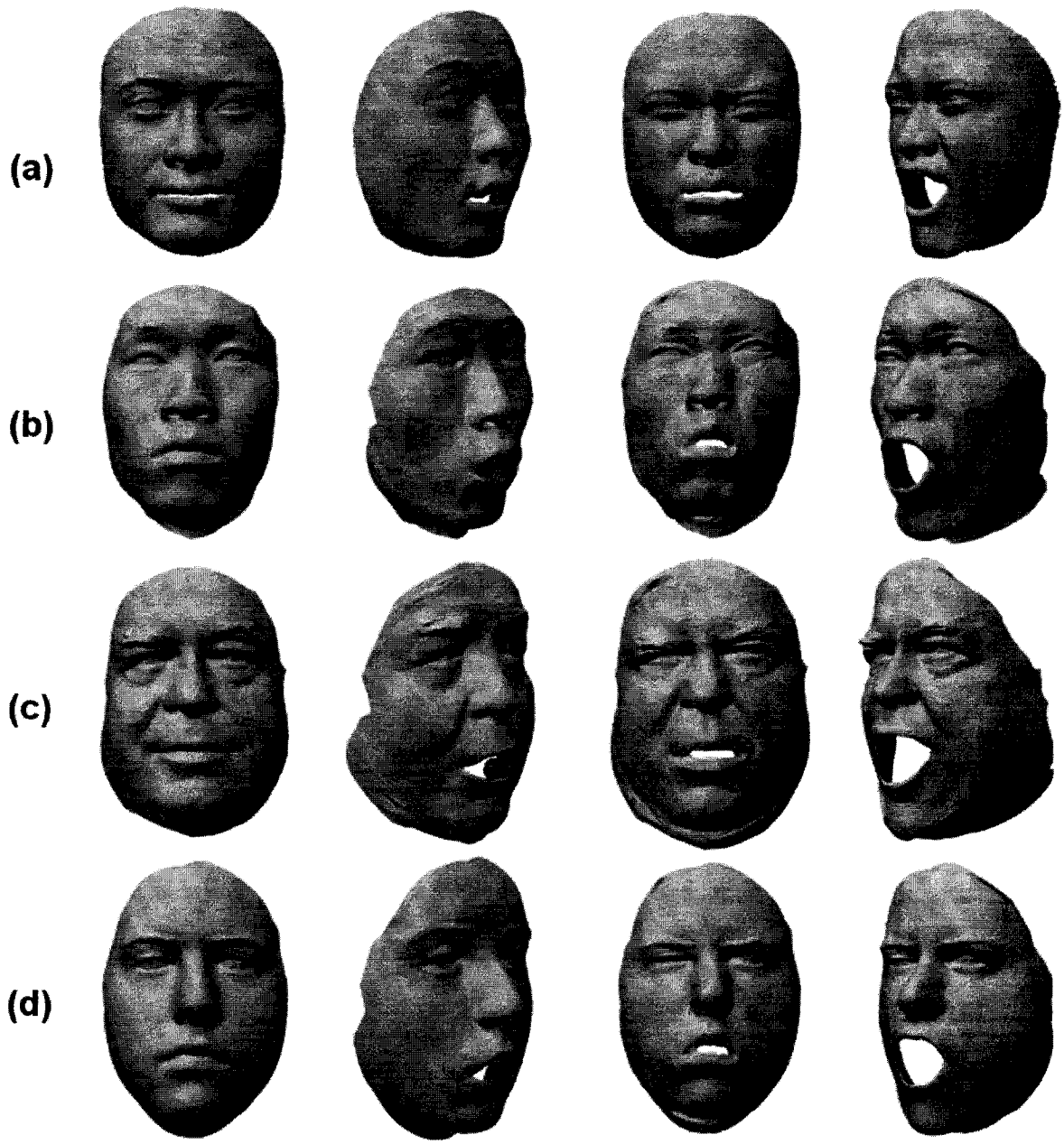


Figure 5.6 : (a) Source expressions from the available animation data ([Zhang L 2004]); (b)-(d) : Facial motions in (a) are transferred to neutral target models.

Chapter 6 Facial Motion Capture

As introduced in section 2.3.4, motion capture has proven to be a feasible and robust approach to acquire high-quality facial motion data. It has been popular in film and entertainment industry ([Scott 2003]). In this chapter, an application of facial motion capture is shown by combining several previous presented techniques. The sections are organized as follows. Firstly, a motion capture device is used to capture facial motions in section 6.1. In section 6.2, the approach of reconstructing facial motions from sparse motion capture data is presented. More experimental results are demonstrated in section 6.3.

6.1 Facial motion capture using VICON

(a) (b)

Figure 6.1 : Facial motion capture using a VICON system. (a) Sparse markers are attached to the subject's face; (b) 3D virtual markers reconstructed by the VICON system.

Section 2.3.4 introduces the VICON system. In this section, we use a VICON system to record facial motions of the performer. The subject is surrounded by eight high-resolution cameras. As shown in Figure 6.1(a), a set of reflective markers is attached to the subject's

face. A set of 32 markers is used here which is the subset of the feature point set in Figure 3.2(b). Not all feature points defined in Figure 3.2(b) are put reflective markers here since it is difficult to track the markers when they are too close to each other. We do not place markers on the eyelids since the markers may interfere with the subject when performing facial motions. Also we do not put markers in the inner lip region since the subject has trouble in performing the mouth actions with those markers. Note that three additional markers on the head mounted jig are used in our motion capture session (Figure 6.1(a)). Although they do not contribute to the facial motions, they are important for removing global movements when analyzing facial motions. With the reflective markers on the subject's face, the VICON system is able to track those markers when the subject performs facial motions. The VICON software suite is then used to process the 2D data captured from eight high-resolution cameras. The VICON software suite is able to clean and reconstruct the 3D data from those 2D motion data. Figure 6.1(b) shows the reconstructed 3D marker data in a frame.

The reconstructed 3D data needs to be normalized among frames. The following presented normalization technique is similar to the one by Busso *et al.* ([Busso 2004]). In the normalization step, the first neutral frame is used as the reference frame. Firstly, the three head jig points are used for removing the global head motions. In each frame, these three rigid points define a local coordinate system. Each frame is then rotated to align it with the reference frame. Secondly, since the nose tip marker is stable when performing facial motions and that point is near the center of the face, the translation is calculated based on that point so that the nose tip in each frame aligns with that in the first neutral frame.

After the normalization step, the facial motion for each marker in frame N can be calculated by the difference between *that marker position in frame N* and *that in the first*

neutral frame. We have to mention that the raw motion capture data may contain noises in the specific frames. Our solution for smoothing out these noises is to average the motion data in the surrounding frames. For example, if the noise appears in frame *x*, then we sum up the motion vectors in frames *x-1*, *x* and *x+1*. Then we use the averaged motions of them for frame x instead of the original one.

6.2 Reconstructing facial motions

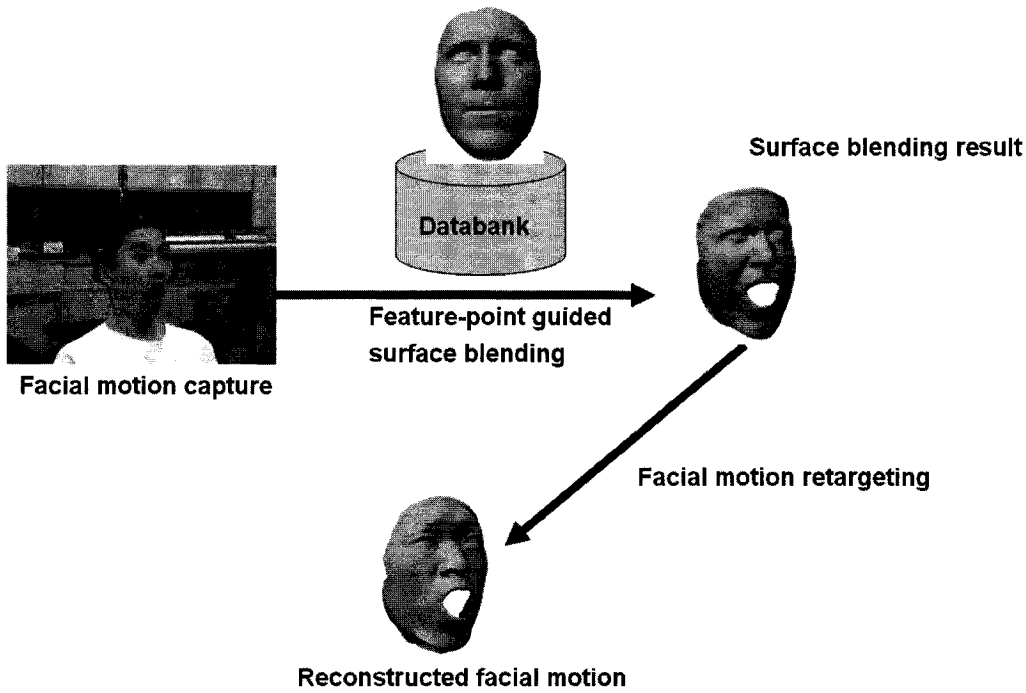


Figure 6.2 : Illustration on how to reconstruct facial motion from sparse motion capture data.

Given captured sparse marker motions (i.e. specified feature point motions), the goal of this section is to reconstruct motions for all surface points of the subject's face. Mapping the sparse marker motions into an available databank would be a suitable solution. In section 4.7, a facial motion databank is constructed. We utilize that databank in this section. Guided by the captured feature point motions, the motions for facial surface points are obtained by

blending the examples in the facial motion databank using the technique presented in Chapter 3.

Firstly, the sparse motion capture data has to be converted into the space of the databank. Chai *et al.* ([Chai 2003]) scale the extracted animation parameters to ensure that the motions from the tracked data have approximately the same magnitude as those in the database. They compute this scale by the ratio of the mouth width between the tracked data and the model in the database.

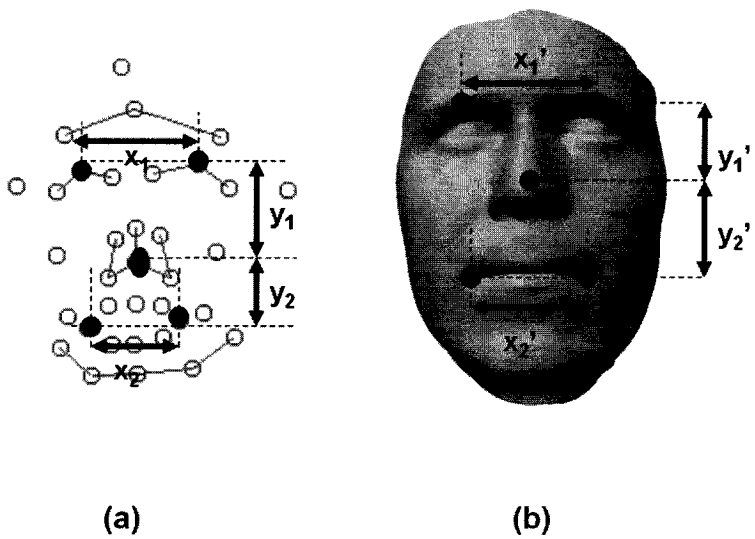


Figure 6.3 : Illustration of calculating the scale factor to convert motion capture data into the space of the databank. (a) Sparse motion capture data; (b) The neutral model in the databank.

The following explains how we decide the scale factor. The first neutral frame in the motion capture data is used as the reference frame. And the solid markers shown in Figure 6.3(a) are used as the reference points. In Figure 6.3(b), the same points are identified in the neutral model of the facial motion databank. Then the distances between those reference points are used to decide the final scale factor. The final scale factor S is given by:

$$S = \frac{1}{4} \times \frac{x1}{x1'} + \frac{1}{4} \times \frac{x2}{x2'} + \frac{1}{4} \times \frac{y1}{y1'} + \frac{1}{4} \times \frac{y2}{y2'} \quad \textbf{(6.1)}$$

The four distance scales contribute equally in deciding the final scale factor *S*. The averaging equation (6.1) decreases the amount of uncertainty.

This scale factor *S* is then applied to the sparse marker motions. After that, the captured facial motions are in the same space as the facial motion databank. Given the scaled sparse facial motions as input (i.e. specified feature point motions), the feature-point guided surface blending approach presented in Chapter 3 is then used to blend the examples in the facial motion databank. The surface blending result can be seen in Figure 6.2.

Since the person in the facial motion databank is different from the performer in the motion capture session. The surface blending result has to be retargeted to the facial mesh of the performer. Before retargeting, we have to get the shape of that performer's face. We utilize the Cyberware laser-scanner to get its shape. It is similar as the technique presented by Breidt et al. ([Breidt 2003]) where 3D scans and 3D motion capture combine together to generate facial animation. Different from their work, here only the neutral pose of the performer is required to be laser-scanned. The raw laser-scanned data is processed by the consistent parameterization technique presented in Chapter 4. Actually the result is already shown as *Person B* in Figure 4.10. The resulting performer's neutral face model shares the same structure as the models in our databank. Then the facial motion retargeting technique presented in section 5.1 is used to transfer the surface blending result onto the performer's neutral face. As shown in Figure 6.2, the final reconstructed facial motion is quite similar to the one in the real world. In the next section, more experimental results are shown.

6.3 Experimental results

As described in section 6.1, a VICON motion capture system is used to capture sparse facial motions. In our experiments, the facial motions are captured when the subject is performing facial expressions and speaking. At the same time, a digital video camera is used to record the facial motions of the subject.

Notice that for reconstructing lip motions when the subject speaks, the viseme models in the facial motion databank are actually blended. The assumption is that the lip motions related to speech can be approximated by a linear combination of a set of viseme models. This assumption has been proven to be acceptable in most popular commercial animation tools such as *3Ds MAX* and *MAYA* where shape blending is used for synthesize visible speech. However for those tools, the key shapes have to be prepared by the animators and the weights for interpolation are tuned manually by animators. In our approach, the key shapes are consistently constructed with the laser-scanned data. And the optimized weights for blending are obtained implicitly via GA.

Figure 6.4 shows reconstructed facial motions using the approach presented in section 6.2. The first, third and fifth rows are the snapshots recorded with a digital video camera. The rest rows show the corresponding reconstructed facial motions.

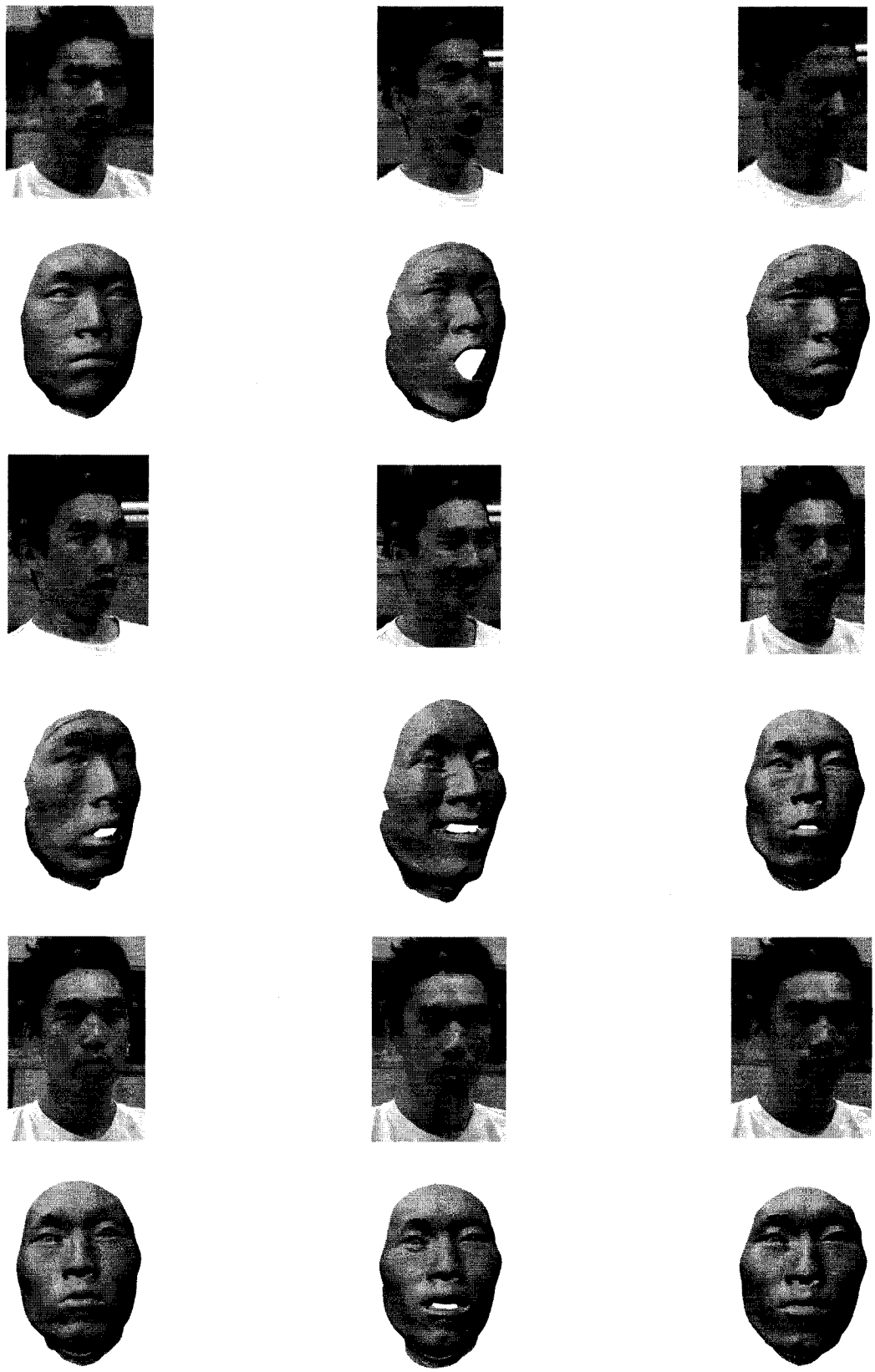


Figure 6.4 : Results of reconstructed facial motions.

Chapter 7 Conclusions

In this document, the examples in the databank are blended to generate natural facial motions for the virtual human face. The optimized blending weights are found implicitly by GA. Secondly, since it is not efficient to utilize the raw laser-scanned face data directly, the consistent parameterization technique is shown to model novel faces based on the laser-scanned data. The consistent parameterization technique establishes correspondences among different laser-scanned data and allows geometry operations to be defined in terms of the homogeneous structure rather than the individual unrelated facial mesh. Thirdly, in order to re-use the animation data, the facial motion retargeting technique is presented to transfer facial motions from existing animation data to a novel neutral face. Finally, an example application of facial motion capture demonstrates previously proposed techniques.

This chapter concludes the research documented in the preceding chapters. Section 7.1 lists the contributions made by this thesis. Section 7.2 discusses the limitations of the proposed methodology. Section 7.3 suggests future research.

7.1 Contributions

The following contributions were made in this thesis:

1. Showed the scheme which is capable of generating natural-looking facial motions while simultaneously preserving the easiness of controlling facial animation with the facial feature points.
2. Developed the novel approach to find optimized blending weights for surface blending by Genetic Algorithms (GA).
3. Proposed a technique to consistently parameterize the raw laser-scanned face data.

4. Showed an efficient approach to animate the laser-scanned data by transferring the facial motions from available animation data.
5. Developed the technique which is capable of re-using any existing facial animation data.
6. Demonstrated an example application of facial motion capture.

7.2 Discussion

The proposed scheme has many benefits, including the ability to derive realistic facial motions guided by motions of the feature points, the ability to generate facial motions by blending the examples in the databank, the ability to find optimized blending weights implicitly via GA, the ability to model and animate the laser-scanned face data efficiently, the ability to re-use any existing facial animation data, and the ability to reconstruct facial motions for the surface points based on the sparse motion capture data.

However, it has some limitations. Firstly, the proposed method for modeling the laser-scanned data consistently is suitable for the face region, but not necessarily for the full region of the head. These could be several reasons. Cylindrical projection for refinement is not appropriate for producing proper results for ears. In addition, part of the ears may be occluded by the side hairs, making it more complicated to refine the ear structures. Secondly, facial motion retargeting technique presented is efficient in mapping facial motions from one subject to another. However, different people have different styles of facial motions. When the facial motion pattern of a performer is very different from that in the databank, reconstructed facial motions may not be so similar to the reality.

7.3 Future research

The research documented in this thesis lays the groundwork for many areas of future research. Many topics may be pursued to extend this work.

- Future work can be done to include texture information for face models, making resulting facial animation more realistic. In addition, with added texture, feature points can be detected automatically as the method proposed by Goto *et al.* ([Goto 2002]).
- Future research can investigate the construction of full heads. First, more refinement algorithms should be explored to tackle the limitation on ears mentioned earlier. Future work can also be done on modeling and rendering realistic hair which could improve the quality of the facial animation.
- Future work could explore the following facial animation approaches: We could place pseudo-muscle structures in the face model. The muscle parameters can be estimated by learning the examples in the databank; we can also develop local RBF deformation scheme for generating facial expressions. The weights for RBF can be trained from the existing examples.
- We presented the facial motion retargeting technique in this document. In the future, we would like to explore the techniques for individualized facial motion retargeting which may require further facial feature analysis.

References

[Blanz 1999] V. Blanz, T. Vetter. A Morphable Model for the Synthesis of 3D Faces. *SIGGRAPH'99 Conf. Proc.*, Los Angeles, USA, 1999.

[Blanz 2003] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating Faces in Images and Video. *In Eurographics*, Granada, Spain, September 2003.

[Breidt 2003] M. Breidt, C. Wallraven, D. W. Cunningham and H. H. Bülthoff, Facial animation based on 3D scans and motion capture. *SIGGRAPH '03 Sketches & Applications*, Campbell, N. ACM Press, New York, NY, USA, July 2003.

[Busso 2004] C. Busso, Z. Deng, S. Yildirim, M. Bulut, C.M. Lee, A. Kazemzadeh, S. Lee, U. Neumann, and S. Narayanan, Analysis of emotion recognition using facial expressions, speech and multimodal information, *In Proc. of ACM 6th International Conference on Multimodal Interfaces (ICMI 2004)*, State College, PA, Oct. 2004, pp.205-211.

[Chai 2003] J. Chai , J. Xiao and J. Hodgins. Vision-based control of 3D facial animation. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* San Diego, California. July 2003.

[Chuang 2002] E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. *Standford University Computer Science Technical Report.* CSTR-2002-02, Apr. 2002.

[D'Apuzzo 2006] N. D'Apuzzo. State of the Art of the Methods for Static 3D Scanning of Partial or Full Human Body. *3D Modelling, Proceedings of Conference*, June 2006, Paris, France.

[Eck 1991] M. Eck. Interpolation methods for reconstruction of 3D surfaces from sequences of planar slices. *CAD und Computergraphik.* 13(5), pp. 109 – 120. 1991.

[Ekman 1975] P. Ekman and W.V. Friesen (1975). *Unmasking the Face. A Guideline to Recognising Emotions from Clues*. Prentice Hall, Englewood Cliffs, New Jersey, USA.

[Ekman 1978] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, CA: Consulting Psychologists, 1978.

[Fang 1996] S. Fang, R. Raghavan, J. T. Richtsmeier, Volume Morphing Methods for Landmark Based 3D Image Deformation, *SPIE Int. Symp. on Medical Imaging*, CA, 1996

[Fidaleo 2000] D. Fidaleo, J-Y. Noh, T. Kim, R. Enciso and U. Neumann. Classification and volume morphing for performance-driven facial animation. *International Workshop on Digital and Computational Video*. 2000.

[Fidaleo 2004] D. Fidaleo and U. Neumann. Analysis of co-articulation regions for performance driven facial animation. *Journal of Computer Animation and Virtual Worlds*, 2004,15: pp. 15-26.

[Garchery 2005] S. Garchery, A. Egges and N. Magnenat-Thalmann. Fast facial animation design for emotional virtual humans. *Measuring Behaviour*, Wageningen, NL, September 2005.

[Gleicher 1998] M. Gleicher. Retargeting motion to new characters. *ACM SIGGRAPH 1998*, pp. 33—42. July 1998.

[Goldberg 1989] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley. MA, 1989.

[Goto 2002] T. Goto, W.-S. Lee, and N. Magnenat-Thalmann. Facial feature extraction for quick 3D face modeling. *Signal Processing: Image Communication.* 17(3), pp. 243-259. 2002.

[Guenter 1998] B. Guenter, C. Grimm, D. Wood, H. Malvar and F. Pighin, Making Faces, *in Proc. SIGGRAPH*, pp. 55--66, 1998.

[Guskov 2000] I. Guskov, K. Vidimce, W. Sweldens, and P. Peter Schroder, Normal meshes. In *SIGGRAPH 2000 Conference Proceedings*, pages 95--102, 2000.

[Hardy 1971] R. L. Hardy, Multiquadric Equations of Topography and Other Irregular Surfaces. *Journal of Geophysical Research* 76 (8), pp.1905—1915. 1971.

[Havaldar 2006] P. Havaldar, Course Notes: Performance Driven Facial Animation. *SIGGRAPH 2006*, Boston, USA, Aug 2006.

[Herrera 1998] F. Herrera, M. Lozano and J. Verdegay. Tackling real-coded genetic algorithms: operators and tools for the behaviour analysis. *Artificial Intelligence Review* 12 (1998) pp. 265 - 319.

[Hoppe 1996] H. Hoppe, Progressive mesh. In *Proc. ACM SIGGRAPH*, 1996, pp. 99-108.

[Hsu 1992] W. Hsu, J. Hughes, H. Kaufman, Direct Manipulation of Free-form deformations, *Computer Graphics (proc. SIGGRAPH)*, Vol 26, 2, pp. 177-184. 1992.

[Ip 1996] H.H.S.Ip and L. Yin. Constructing a 3D individual head model from two orthogonal views. *The Visual Computer*. 12(5), pp. 254-266. 1996.

[Jeong 2002] W.-K. Jeong, K. Kähler, J. Haber and H.P. Seidel, Automatic generation of subdivision surface head models from point cloud data. *Proc. Graphics Interface*, pp. 181-188. 2002

[Kalra 1992] P. Kalra, A. Mangili, N. Magnenat-Thalmann and D. Thalmann. Simulation of facial muscle actions based on rational free form deformations, *Proc. EUROGRAPHICS'92*, pp. 59-69. 1992.

[Kouadio 1998] C. Kouadio , P. Poulin and P. Lachapelle. Real-time facial animation based upon a bank of 3D facial expressions. *Proceedings of the Computer Animation*. pp. 128-136. June 1998.

[Kshirsagar 2001] S. Kshirsagar, S. Garchery, and N. Magnenat-Thalmann. Feature point based mesh deformation applied to MPEG-4 facial animation. *In Deformable Avatars. Norwell, MA: Kluwer*, 2001, pp. 24–30.

[Kurihara 1991] T. Kurihara and K. Arai. A transformation method for modeling and animation of the human face from photographs. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation 91*, pp. 45--58. SpringerVerlag, Tokyo, 1991.

[LeBlanc 1991] A. LeBlanc, P. Kalra, N. Magnenat-Thalmann, D. Thalmann, Sculpting with the "Ball & Mouse" Metaphor, *Proc. Graphics Interface '91*, Calgary, Canada, pp. 152-159. 1991.

[Lee A 2003] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proc. ACM SIGGRAPH*, pages 85---94, 2000.

[Lee J 2003] J. Lee, B. Moghaddam, H. Pfister, and R. Machiraju. Silhouette-based 3D face shape recovery. *In Proc. of Graphics Interface*, pp. 21--30, 2003.

[Lee WS 1998] W.-S. Lee and N. Magnenat-Thalmann. Head Modeling from Pictures and Morphing in 3D with Image Metamorphosis Based on Triangulation. *In Modelling and Motion Capture Techniques for Virtual Environments* - MagnenatThalmann, N. and Thalmann,D. (Eds.), pp. 254---268, 1998.

[Lee WS 1999] W. –S. Lee, P. Beylot, D. Sankoff, and N. Magnenat-Thalmann. Generating 3D virtual populations from pictures of a few individuals. In *Proceedings of the 6th international Workshop on Algorithms and Data Structures* (August 1999). F. K. Dehne, A. Gupta, J. Sack, and R. Tamassia, Eds. Lecture Notes In Computer Science, vol. 1663, Springer-Verlag, London, pp. 134-144.

[Lee WS 2000] W. –S. Lee, and N. Magnenat-Thalmann. Fast head modeling for animation. *Journal Image and Vision Computing* (Elsevier, March 1, 2000). 18(4), pp. 355-364.

[Lee Y 1995] Y. Lee, D. Terzopoulos, and K. Waters. Realistic face modeling for animation. In *Proceedings of SIGGRAPH 95*, pp. 55-62. 1995.

[Loop 1987] C. Loop. 1987. *Smooth subdivision surfaces based on triangles.* Master's thesis, University of Utah, Department of Mathematics.

[Magnenat-Thalmann 1987] N. Magnenat-Thalmann, D. Thalmann. The direction of synthetic actors in the film rendez-vous a montreal. *IEEE Computer Graphics and Applications*, pp. 9—19. 1987

[Mani 2001] M. V. Mani and J. Ostermann. Cloning of MPEG-4 Face Models. *International Workshop on Very Low Bitrate Video Coding (VLBV 01)*, Athen, Greece 2001.

[Na 2004] K. Na and M. Jung. Hierarchical Retargeting of Fine Facial Motions. *Proceedings of EUROGRAPHICS 04*, pp. 687-695. 2004.

[Nagel 1998] B. Nagel, J. Wingbermuhle, S. Weik and C. -E. Liedtke. Automated Modelling of Real Human Faces for 3D Animation, *Proceedings of the 14th International Conference on Pattern Recognition*, Volume 1, p 693. 1998

[Noh 2000] 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 (VRST '00)*. ACM Press, New York, NY, pp. 166-174. 2000.

[Noh 2001] J. Noh and U. Neumann. Expression cloning. In *Proceedings of the 28th Annual Conference on Computer Graphics and interactive Techniques (SIGGRAPH 2001)*. ACM Press, New York, NY, pp. 277-288. 2001.

[Ostermann 1998] J. Ostermann. Animation of synthetic faces in MPEG-4, *IEEE Computer Animation*, pp. 49-55, 1998.

[Pandzic 2002] I. S. Pandzic and R. Forchheimer. *MPEG-4 facial animation: the standard, implementation and applications*. Wiley, 2002.

[Parent 2002] R. Parent. (2002) *Computer animation algorithms and techniques*. Morgan Kauffmann. Section 6.3. pp. 339-353.

[Parke 1972] F. I. Parke. Computer generated animation of faces. *Proceedings of the ACM annual conference*. Boston, Massachusetts, United States, pp. 451 – 457. 1972

[Parke 1974] F. I. Parke. *A parametric model for human faces*. PhD Thesis, University of Utah, Department of Computer Science, 1974.

[Parke 1996] F.I. Parke, K. Waters, *Computer Facial animation*, A.K.Peters Ltd, 1996, ISBN 1-56881-014-8

[Pighin 1998] F. Pighin , J. Hecker, D. Lischinski, R. Szeliski, D. Salesin. Synthesizing realistic facial expressions from photographs. *In Proc. of the 25th annual conference on Computer Graphics (SIGGRAPH 1998)*, pp. 75-84. July 1998.

[Powell 1987] Powell, M. J. D. 1987. Radial basis functions for multivariate interpolation: a review. In *Algorithms for Approximation*, Clarendon Press, Oxford. pp. 143-167.

[Proesmans 1997] M. Proesmans and L. Van Gool. Reading between the lines—a method for extracting dynamic 3D with texture. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (Lausanne, Switzerland). VRST '97. ACM Press, New York, NY, pp. 95-102. 1997.

[Pyun 2003] H. Pyun , Y. Kim , W. Chae , H. W. Kang and S. Y. Shin. An example-based approach for facial expression cloning. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, California, pp. 167 - 176. July 2003.

[Sannier 1997] G. Sannier, and N. Magnenat-Thalmann. A user-friendly texture-fitting methodology for virtual humans. *Proceedings of Computer Graphics International '97* (Los Angeles, California, USA, 1997). pp. 167-176. 1997

[Scott 2003] R. Scott. Sparking life: notes on the performance capture sessions for the lord of the rings: the two towers. *ACM SIGGRAPH Computer Graphics*. Volume 37 , pp. 17 – 21. 2003.

[Sederberg 1986] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *In D. C. Evans and R. J. Athay, Eds. SIGGRAPH 1986.* ACM Press, New York, NY, pp. 151-160. 1986.

[Terzopoulos 1990] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis, and animation, *Journal of Visualization and Computer Animation*, Vol. 1, pp. 73-80. 1990.

[Terzopoulos 1993] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):569--579, June 1993.

[Ulgen 1997] F. Ulgen. A step toward universal facial animation via volume morphing. *In Proceedings 6th IEEE International Workshop on Robot and Human communication*, pp. 358—363. 1997.

[Waters 1987] K. Waters. A Muscle model for animating three-dimensional facial expression. *Computer Graphics*, Vol. 21, 4, July, pp. 17-24. 1987.

[Williams 1990] L. Williams. Performance-driven facial animation. *In Computer Graphics (Proceedings of SIGGRAPH 90)*, pp. 235--242, August 1990.

[Zhang L 2004] L. Zhang, N. Snavely, B. Curless, and S.M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.* 23, 3 (Aug. 2004). pp. 548-558.

[Zhang Y 2004] Y. Zhang, T. Sim, and C. L. Tan. Adaptation-based individualized face modeling for animation using displacement map. In *Proceedings of Computer Graphics International 2004* (Crete, Greece). pp. 518-521. 2004

[Zheng 1994] J. Y. Zheng. Acquiring 3-D models from sequences of contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 163-178. 1994.

Related publications by the author

1. Lijia Zhu and Won-Sook Lee, "Modeling and Animating for the Dense Laser-scanned Face in the Low Resolution Level", *The 17th IASTED International Conference on MODELLING AND SIMULATION*, Montreal, Canada, May 2006.
2. Lijia Zhu and Won-Sook Lee, "Facial Expression via Genetic Algorithms", *the COMPUTER ANIMATION and SOCIAL AGENTS Conference 2006 (CASA 2006)*, Geneva, Switzerland, July 2006.
3. Lijia Zhu and Won-Sook Lee, "Natural MPEG-4 Facial Expression using Genetic Algorithms", *Proceedings of the 4th International Symposium on Ubiquitous VR (ISUVR2006)*, pp. 97-98, *CEUR online workshop proceeding of Sun SITE Central Europe*, July 2006.