3D FACE MODEL GENERATION

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF

MIDDLE EAST TECHNICAL UNIVERSITY

BY

SONER BÜYÜKATALAY

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet Erkmen
Head of the Department

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

_____

Prof. Dr. Uğur Halıcı
Supervisor

Examining Committee in Charge:

Prof. Dr. Kemal Leblebicioğlu	(METU, EE)	_____
Prof. Dr. Uğur Halıcı	(METU, EE)	_____
Prof. Dr. Turgut Tümer	(METU, ME)	_____
Dr. İlkay Ulusoy	(METU, EE)	_____
Dr. Uğur Leloğlu	(TÜBİTAK-BİLTEN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Soner Büyükatalay

Signature :

ABSTRACT

3D FACE MODEL GENERATION

Büyükatalay, Soner

M.S., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Uğur Halıcı

November 2004, 76 pages

Generation of photo-realistic 3D human face models is a hot topic in the area joining computer graphics and computer vision. Many different techniques are used for this purpose, but most of them are not feasible for home users. These techniques may use advanced hardware such as laser scanners, calibrated stereo cameras, or very sophisticated software that can be as expensive as advanced hardware. Face model generation by morphing an initial 3D model with uncalibrated camera photographs is studied in this thesis. Manually marked feature points on photographs are used to deform initial 3D face model. Initial photographs also are processed to form a single texture image covering deformed 3D face model.

Keywords : Human face, 3D model generation, uncalibrated camera,

ÖZ

3 BOYUTLU YÜZ MODELİ OLUŞTURMA

Büyükatalay, Soner

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Kasım 2004, 76 sayfa

Fotograf gerçekçiliğinde 3 boyutlu (3B) yüz modelleri oluşturma, bigisayar grafikleri ve bilgisayarla görme alanlarının her ikisini de kullanan en sıcak konulardan biridir. Bu konuda birçok uygulama geliştirilmiş olsa da ev kullanıcılarına uygun çözümler hala eksik kalmaktadır. Kullanılan tekniklerin çoğu lazer tarayıcılar, kalibre edilmiş stereo kameralar gibi pahalı ve ev kullanıcıların sahip olamayacağı donamınlar veya oldukça karmaşık yazılımlara ihtiyaç duymaktadır. Bu tezde kalibre edilmemiş kamera fotografları kullanılmsı suretiyle üç boyutlu başlangıç yüz modeli değiştirilerek yeni bir yüz modeli oluşturulmaya çalışılmıştır. 3B Başlangıç yüz modelini değiştirmek için manuel olarak işaretlenmiş özel noktalar kullanılmaktadır. Ayrıca başlangıç fotografları, değiştirilmiş 3B başlangıç modelini saracak tek bir doku resmi oluşturmak için işlenmektedir.

Anahtar Kelimeler: İnsan yüzü, 3B model oluşturma, kalibrasyonsuz kamera,

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and appreciation to my supervisor Prof. Dr. Uğur Halıcı who inspired, encouraged and supported me at all levels of this study.

I would like to thank Elçin Özgür, Erdem Akagündüz, Varlık Kılıç and Can Aydın whose friendship, support and suggestions made great contributions to this work.

The greatest thanks go to my family members for their infinite support.

TABLE OF CONTENTS

PLAGIARISM ..... *iii*

ABSTRACT ..... *iv*

ÖZ ..... *v*

ACKNOWLEDGEMENT ..... *vi*

TABLE OF CONTENTS ..... *vii*

LIST OF FIGURES ..... *x*

LIST OF ABBREVIATIONS ..... *xi*

CHAPTER 1 INTRODUCTION *1*

1.1 A Survey on Face Modeling **3**

1.2 Face Modeling Techniques **4**

1.3 Organization **10**

CHAPTER 2 FEATURE POINT CALCULATION *11*

2.1 Introduction **11**

2.2 Feature Point Normalization **12**

2.3 Camera Calibration **15**

2.3.1 Computation of Camera Projection Matrix (P) 16

2.3.2 Camera Center Extraction from Camera Projection Matrix (P) 19

2.3.3 Principal Plane Extraction from Camera Projection Matrix (P) 20

2.3.4 Camera Projection Matrix (P) Decomposition 20

2.4 3D Feature Point Calculation 23

CHAPTER 3 FACE MESH DEFORMATION 24

3.1 Introduction 24

3.2 Addition of Refining Feature Points 27

CHAPTER 4 TEXTURE EXTRACTION 29

4.1 Introduction 29

4.2 Texture Image Formation 30

CHAPTER 5 FACEGENERATOR 34

5.1 Introduction 34

5.2 Java 35

5.3 Java2D 36

5.4 JAXP 37

5.5 Java3D 38

5.6 FaceGenerator 41

5.6.1 Image Calibration 41

5.6.2 Model Deformation 48

5.6.3 Texture Generation 51

CHAPTER 6 EXPERIMENTAL RESULTS 55

6.1 Introduction 55

6.2 Models with Different Number Photographs 60

6.3 Models with Different Direction of Photography 62

6.4 Models with Different RBF 65

6.5 Models with Occluded Faces 67

CHAPTER 7 CONCLUSION 69

7.1 Future Works 70

REFERENCES 72

LIST OF FIGURES

Figure 1.1 : System overview *7*

Figure 5.1 : Java3D scene graph *39*

Figure 5.2 : Image Calibration *42*

Figure 5.3 : Feature Points in 2D and 3D *45*

Figure 5.4 : Model Deformation *48*

Figure 5.5 : Radial Basis Function (RBF) *50*

Figure 5.6 : Texture Generation *51*

Figure 5.7: Texture Map *52*

Figure 6.1 : Face images with feature points *56*

Figure 6.2 : 3D deformation of initial face mesh *57*

Figure 6.3 : Refinement on nose and cheek *57*

Figure 6.4 : 3D face model and texture image *58*

Figure 6.5 : Face images with projected final 3D mesh *59*

Figure 6.6 : 3D feature points found in each trial *61*

Figure 6.7 : 3D face models found in each trial *61*

Figure 6.8 : Images taken with horizontal alignment *63*

Figure 6.9 : Images taken from downwards direction *64*

Figure 6.10 : Images taken with horizontal rotation *64*

Figure 6.11 : 3D face models with different direction of photography *65*

Figure 6.12 : Variation of RBF for the same model *66*

Figure 6.13 : 3D model of a man with eye glasses *68*

LIST OF ABBREVIATIONS

2D : Two dimensional

3D : Three dimensional

RBF : Radial Basis Function

SDK : Standard Development Kid

JRE : Java Runtime Environment

XML : Extensible Markup Language

JAXP : Java API for XML Processing

CHAPTER 1

INTRODUCTION

Human face model generation process can be explained as a translation from two dimensional representations (2D) of a face that are photographs, to three dimensional representation (3D) that is 3D space coordinates of a mesh representing the shape of face and a texture image to wrap this mesh.

Face modeling can be used in many areas. It is the first step in character animation, which is widely used in films, advertising and computer games. Some newly developing fields are user-interface agents and avatars, facial surgery planning and video teleconferencing [12] [19]. In video teleconferencing, forming face models of callers, and transmitting only the speech and gesture data will reduce bandwidth. Many application examples will follow these if home users with a simple web cam and a conventional PC can generate face models.

In recent years, there has been considerable interest in computer-based 3D face modeling and animation. However this is not a very new subject, hence twenty years ago, researchers used computer to represent and animate human faces [6]. But, realistic facial animation has become recently possible partially due to the incredible growth of the computation power during the last 20 years. Since,

computers and internet have become essential parts of our daily life, computerized face animation has more importance than before.

The human face is interesting and challenging because of its familiarity. We distinguish individuals mostly from their face. Also we are able to detect even small changes in facial expressions in social life. Humans learn these recognition skills in childhood and develop rapidly since many different faces are seen every day.

The ability to model a realistic human face still remains a significant challenge in computer graphics [2] [4] [14] [22]. Despite of the traditional computer graphics algorithms such as modeling and rendering, realistic face model generation of human face is still one of the most difficult goals in computer animation. Because of familiarity of human face, human beings try to find tiny details on generated face model, and this causes the main problem in face modeling. In addition to familiarity, the geometric form of the human face is extremely complex, which is another drawback of face modeling. Indeed, there are a few impressive facial animations such as the ones created by Pixar Co., but these results need many years of highly skilled animators.

1.1 A Survey on Face Modeling

This section is a survey of computer face modeling [8]. The most significant results in this area have been published at ACM SIGGRAPH conferences as well as in other related computer graphics journals and conference proceedings.

The first computer-generated images of faces were generated by F. I. Parke in the early 1970s. The head was represented by polygons, which were used to open and close eyes and mouth [6]. Henri Gouraud was also working on his smooth polygon shading algorithm (Gouraud shading) during this period and he demonstrated it by applying the new technique to a digitized face model. Parke [7] had completed the first parameterized facial model by 1974. In 1971, Chernoff [11] published for the first time; his work of using computer generated 2D face drawings to represent a k-dimensional space.

In 1980, Platt published his master thesis on a physically based muscle-controlled facial expression model [20]. Weil's [16] work which uses video system to interactively select and composite facial features was reported in 1982.

A new wealth of data for facial modeling was provided through the development of optical range scanners, such as the Cyberware optical laser scanner [3]. In 1998, new techniques for creating a photorealistic textured 3D facial model from

photographs and for creating smooth transitions between different facial expressions to make realistic facial animation were presented by Pighin et al. [9].

A system to capture three-dimensional geometry, color, and shading information for human facial expressions was created by Guenter et al, [2] these data were used to reconstruct photo-realistic 3D facial animation.

1.2 Face Modeling Techniques

Generation of face model involves determining geometric descriptions like 3D coordinates, and additional surface color attributes named as texture. Face model generation techniques can be divided into two main categories; photogrammetric techniques and laser scanner based approach. Photogrammetric techniques are acquisition of geometry and texture from photographs or videos. In laser scanner based approach, scanning devices, such as those developed by Cyberware [3] are employed to digitize face geometry and texture. These systems can acquire more precise 3D surface shape data, as well as surface color, of static sculptures, but in case of real people, subject tends to move during scanning since it takes a long time.

Face modeling algorithms, and also modeling algorithms in general, require a decision concerning the trade-off between model specificity and generality. With

this respect, model generation can be categorized in two based on the object restrictions:

- **Unconstrained modeling:** There is no constraint on the object to be modeled. Free-form meshes are used in this category. In case of face modeling, this type may not show detailed information about the face.
- **Constrained modeling:** In this category modeling algorithm has a bias towards representing the desired class of objects. Object to be modeled must belong to this class. Two different face modeling algorithm are developed up to now in this category. The first one employs a 3D scanned face model database, which generates new face models as a linear combination of the models from database, based on some statistical parameters derived from a single face photograph [22]. The second one, which is also used in this thesis, is to deform an initial face model with some 3D coordinates derived from multiple photographs.

In this study, an easy-to-use technique is proposed. The process consists of several basic steps similar to those explained in [9]:

1. Multiple photographs of the human face to be modeled are captured from arbitrary directions where the individual stands still with a firm face expression.
2. Some corresponding points (feature points of face, such as corners of eyes, and mouth, tip of nose etc.) are marked manually on each

photograph and 3D generic face model separately. These points will be used to find camera position relative to feature points of face.

3. 3D space coordinates of these feature points and relative camera positions are calculated using an iterative algorithm.
4. More refining feature points other than the ones used for camera calibration can be defined after camera calibration. These refinement points will not be used for camera calibration, but only for a better deformation of 3D face mesh.
5. 3D coordinates of feature points are used to deform the generic initial 3D face mesh [24].
6. Texture map and texture image are formed from the photos and the 3D face mesh. Then texture image is fitted on to 3D face mesh.

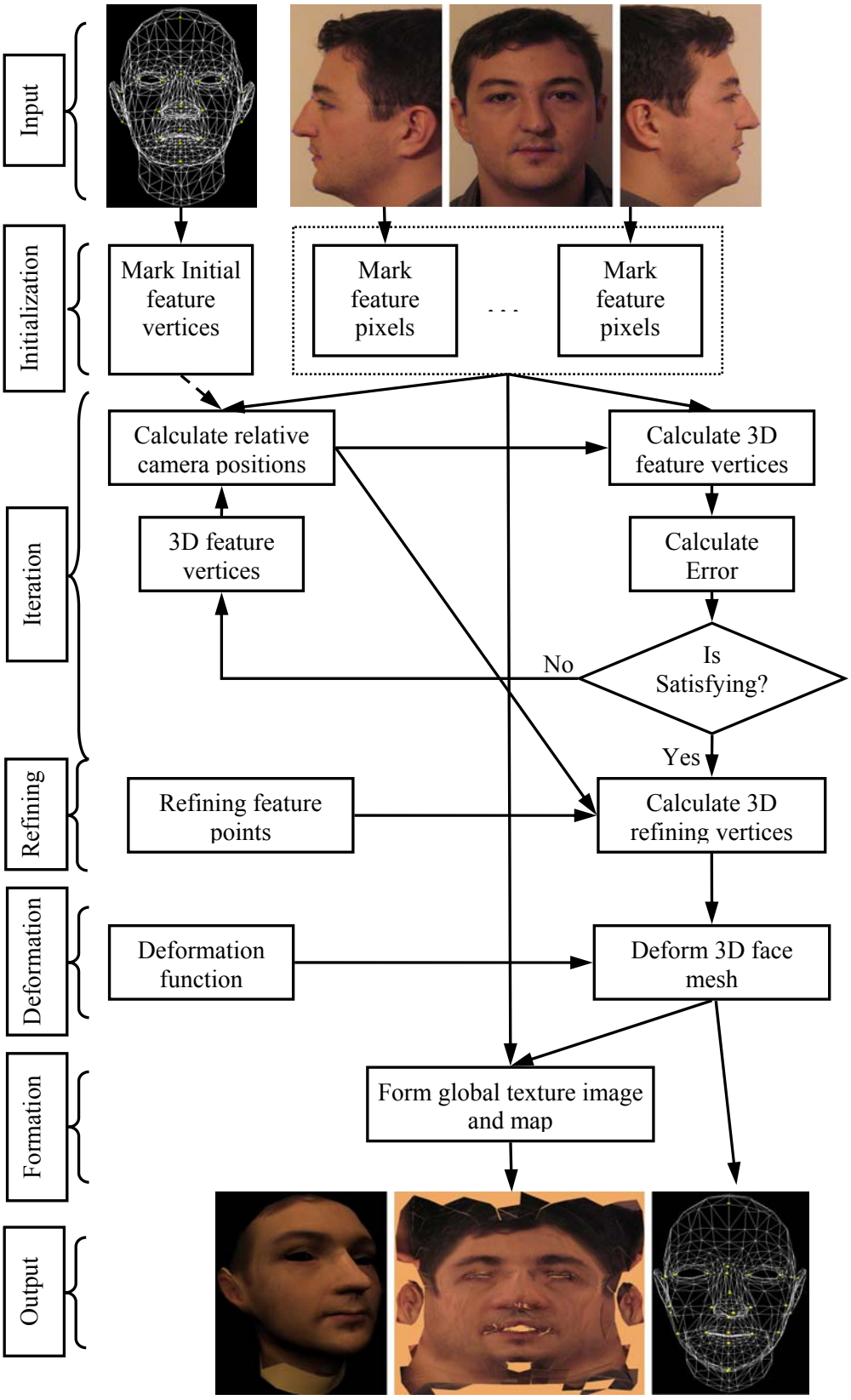


Figure 1.1 : System overview

An overview of the system that we developed in this thesis is given in Figure 1.1. In our system, the images that are obtained by arbitrarily placed cameras are used. Beside arbitrary camera placement, algorithm does not have any constraints on camera lenses. In the following text, the term “uncalibrated images” means that these images are taken without any calibration constraint. In this thesis predefined 18 feature points are used for model generation. There are two constraints on the number of these feature points, thus the images must be taken considering these constraints which are listed as follows:

- A feature point must be specified in at least 2 different images to calculate its 3D coordinates.
- At least 6 feature points must be specified at each image so that the camera position related to that image can be calculated.

After the feature points are specified at the initialization step, the 3D coordinates of these points is calculated iteratively in the next step. Each iteration is composed of two linear calculations, which calculate relative camera positions and 3D coordinates of the features. The iteration loop is initialized by 3D initial feature points. After initialization, at each loop of iteration, camera positions and new 3D feature points are calculated sequentially. In this thesis, another initialization method which initializes the iteration with camera positions is also applied.

The next step is deforming a generic face mesh considering the coordinates of the 3D feature points found in the iteration phase. This small number of feature points will shape the rest of the face employing Radial Basis Functions. After deformation, if results are not satisfactory, then some more new refining feature points can be added to the initial images. This sub-step of deformation is called refining in this work. The aim of the refining step is to specify more feature points which are specific to the current subject's face. There are two reasons why we do not include these refining feature points in camera calibration. First, these points may not be easily marked on images and can affect calibration of camera. Second, these 3D feature points may be specific to the current subject only and may not be marked on the initial generic 3D face model.

The last step is forming of a texture image to wrap the final 3D face mesh. An initial texture map that defines the wrapping of texture image to the 3D face mesh is used. Whole deformed 3D face is projected to each image considering camera position of that image and a new texture image is formed by combining pixels of images with respect to the projected 3D coordinates. Brightness equalization can be applied to reduce differences from image to image and a blur function can be employed to reduce contours emerged by image combination in order to obtain the texture image.

1.3 Organization

The rest of this thesis is organized as follows: The next chapter describes camera calibration and calculation of 3D coordinates of feature points altogether. The third chapter describes the generic face mesh deformation using the feature points. The fourth chapter describes the texture extraction using initial images and the feature point coordinates. The fifth chapter describes our FaceGenerator application, in which programming language Java and Java3D library employed. In the sixth chapter experimental results are presented. Finally in the last chapter, conclusion and possible future works are explained.

CHAPTER 2

FEATURE POINT CALCULATION

2.1 Introduction

Feature point calculation is composed of four parts, which are feature point normalization [19], camera calibration [18], 3D feature point calculation and data renormalization. The flow of these calculations is listed below:

1. First, feature points from the images and the 3D face mesh are normalized separately for each image in order to the 3D face mesh to have a scale invariant space. After normalization, 3D coordinates of feature points do not change place with scaling or translation of the face in the image.
2. After normalization, 3D positions of feature points are used to calculate the camera parameters.
3. Recovered camera parameters and 2D coordinates of feature points are used to find the new 3D positions of the feature points. If the total amount of error between the initial and new 3D coordinates of feature points is greater than the desired error value, process goes back to step 2 with new 3D positions. This iteration process finishes when error is reduced to an acceptable level.
4. Finally normalized 3D and 2D coordinates of feature points are renormalized to real world space. This step is simply composed of

application of the inverse normalizations to feature points of 3D face mesh and this process will not be mentioned again in the rest of the thesis.

In this process, beside the calculation of 3D coordinates of feature points, camera positions, which will be employed later in texture generation, are computed.

2.2 Feature Point Normalization

Feature Point Calculation uses both 2D and 3D coordinates of the marked feature points. The performance of the 3D calculation process should not be affected by the change of the origin and scaling of images or 3D face mesh. While relative 3D coordinates of feature points will be calculated, algorithm should depend only to relative displacements of feature points. But this is not appropriate when camera calibration is solved linearly [19]. Normalization step defined here provides origin and scale invariant solution for camera calibration. This is essential for minimization of error for each image equally so that each image has the same contribution to the result.

Also normalized data produces normalized error values that can be used as performance criteria. With the help of this performance value, we can compare the algorithm performance on different faces.

Before normalization, in 2D images, origin of coordinates are at the top left corner, with increasing pixel units from left to right and up to down. In 3D face mesh model, origin of coordinates is at the center of head model, with meter units. Tip of the nose is on the positive Z axis, and X axis passes through the center of ears of 3D face mesh.

As the first step of normalization, the coordinates in each image and 3D object are translated separately (by a different translation for each image) so that centroid of all feature points in each image is at the center of image. Secondly, the coordinates in each image are also scaled so that the average magnitudes of all points in each image are the same. Rather than choosing different scale factors for each coordinate direction, an isotropic scaling factor is chosen so that x, and y coordinates (also z coordinate for 3D case) are scaled equally. We scale the coordinates so that the average distance to origin is $\sqrt{2}$ for 2D, $\sqrt{3}$ for 3D.

Normalization method is defined as follows;

- Center of mass of the feature points will be at origin. (0, 0) for 2D images, (0, 0, 0) for 3D face model.
- Average distance of the feature points to the center of mass is $\sqrt{2}$ for 2D images, $\sqrt{3}$ for the 3D face model.

The above rules can be applied with a normalization transformation **N,** defined as follows. For a sample image having n 2D feature points, $\mathbf{p}_i = (x_i, y_i) \quad i = 1 \cdots n$,

translation parameters o_x, o_y, o_z and scaling factors s_2, s_3 for 2D and 3D are defined respectively as follows:

$$o_x = -\frac{\sum_{i=1}^{n} x_i}{n} \quad , \quad o_y = -\frac{\sum_{i=1}^{n} y_i}{n} \quad , \quad o_z = -\frac{\sum_{i=1}^{n} z_i}{n}$$

$$s_2 = \frac{\sqrt{2}\, n}{\sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2}} \quad , \quad s_3 = \frac{\sqrt{3}\, n}{\sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2 + z_i^2}} \qquad \text{Eq. 2.1}$$

Then normalization matrix for 2D is:

$$\mathbf{N} = \begin{bmatrix} s_2 & 0 & s_2 o_x \\ 0 & s_2 & s_2 o_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq. 2.2}$$

Similarly normalization matrix for 3D is;

$$\mathbf{N} = \begin{bmatrix} s_3 & 0 & 0 & s_3 o_x \\ 0 & s_3 & 0 & s_3 o_y \\ 0 & 0 & s_3 & s_3 o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq. 2.3}$$

These normalization matrices are preserved to be used later for inverse normalization.

2.3 Camera Calibration

Camera calibration means the calculation of the perspective projection matrix of the camera which transforms the 3D real world to a 2D photograph and it can be expressed as follows [19]:

$$s\begin{bmatrix}\mathbf{x}\\1\end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix} \qquad \text{Eq. 2.4}$$

Where $\mathbf{x} = (x, y)$ is 2D image coordinates and $\mathbf{X} = (X, Y, Z)$ is 3D coordinates. In Eq. 2.4, **P**, which expressed in homogeneous coordinates, is a **3x4** camera projection matrix and s is the arbitrary scale factor. **P** can be decomposed as;

$$\mathbf{P} = \mathbf{C}\left[\mathbf{R} \mid \mathbf{T}\right] \qquad \text{Eq. 2.5}$$

In this decomposition **C** is an upper triangular matrix, representing the intrinsic parameters of the camera such as:

$$\mathbf{C} = \begin{bmatrix}\alpha_u & k & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{bmatrix} \qquad \text{Eq. 2.6}$$

$\alpha_u = \frac{focal\ length}{pixel\ width}$, $\alpha_v = \frac{focal\ length}{pixel\ height}$

(u_0 , u_0) : Center of projection plane with respect to top left corner

k : skew factor (non-zero when coordinates are not orthogonal)

R is a **3x3** orthogonal rotation matrix and **T** is the **1x3** translation vector of the camera.

The linear camera calibration algorithm used in this thesis is composed of 2 parts:

1. Computation of **P** using a set of feature points whose image positions and 3D positions are known;
2. Computation of **R** and **T** using **QR** decomposition.

The first part is sufficient to re-calculate the 3D coordinates of the feature points. Although the second part is not necessary for 3D feature point calculation process, it needs to be calculated later for texture generation where the direction of the camera is necessary [1]. However, the whole calibration process will be described in the followings.

2.3.1 Computation of Camera Projection Matrix (P)

Eq. 2.4 can be written in open form as follows for the *i*`th feature point:

$$s\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \qquad \text{Eq. 2.7}$$

While the equation has an arbitrary scale factor *s* and **P** has 11 level of freedom (3 from rotation, 3 from translation, 5 from intrinsic parameters), we can set $p_{34} = 1$.

For each 2D feature point we have 2 equation:

$$u_i = \frac{X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + 1}, \; v_i = \frac{X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + 1} \qquad \text{Eq. 2.8}$$

To find **P** we rearrange these equations as follows;

$$\begin{aligned} X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14} - u_i X_i p_{31} - u_i Y_i p_{32} - u_i Z_i p_{33} &= u_i \\ X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24} - v_i X_i p_{31} - v_i Y_i p_{32} - v_i Z_i p_{33} &= v_i \end{aligned} \qquad \text{Eq. 2.9}$$

We can write these equations for all of the feature points in matrix form:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -u_NX_N & -u_NY_N & -u_NZ_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -v_NX_N & -v_NY_N & -v_NZ_N \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ \vdots \\ u_i \\ v_i \\ \vdots \\ \vdots \\ \vdots \\ u_N \\ v_N \end{bmatrix} \quad \text{Eq. 2.10}$$

We can rewrite Eq. 2.10 as follows:

$$\mathbf{B}[\mathbf{p}] = \mathbf{a} \qquad \text{Eq. 2.11}$$

Where **B** is the **2N x 11** matrix of known values. With 11 unknowns, the linear equation requires at least 6 feature points (**N >= 6**) to have a unique solution.

In Eq. 2.11 **B** is a non-square matrix and can be solved using various linear algebra methods, like **LU Decomposion, QR Decomposion,** and **Pseudo-Inverse.** In this work **Pseudo-Inverse** method is employed which can be formulated as follows;

$$[\mathbf{p}] = \mathbf{B}^{+}\mathbf{a}$$

$$\mathbf{B}^{+} = \left[\mathbf{B}^{T}\mathbf{B}\right]^{-1}\mathbf{B}^{T} \qquad \text{Eq. 2.12}$$

Where $\mathbf{B}^{+}$ is the pseudo-inverse of **B.** This method will work if we have sufficient number of feature points, and if they are not planar in 3D space.

2.3.2 Camera Center Extraction from Camera Projection Matrix (P)

Camera Center is the point for which $\mathbf{Pc} = \mathbf{0}$. This point can be used as camera position in 3D world to calculate imaging angle.

In homogenous coordinates **c** is

$$\mathbf{c} = (x, y, z, t)$$

and can be calculated from **P** as follows:

$$x = \begin{vmatrix} p_{12} & p_{13} & p_{14} \\ p_{22} & p_{23} & p_{24} \\ p_{32} & p_{33} & p_{34} \end{vmatrix} \quad , \quad y = -\begin{vmatrix} p_{11} & p_{13} & p_{14} \\ p_{21} & p_{23} & p_{24} \\ p_{31} & p_{33} & p_{34} \end{vmatrix} \quad , \quad z = \begin{vmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{vmatrix} \quad , \quad t = -\begin{vmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{vmatrix} \qquad \text{Eq. 2.13}$$

Where $|.|$ denotes the determinant of the covered matrix.

2.3.3 Principal Plane Extraction from Camera Projection Matrix (P)

The principal plane is defined as the plane passing through the camera center and parallel to the image plane. It consists of the set of 3D points **X,** which are imaged as a line at infinity of the image.

$$\mathbf{P}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix}=\begin{bmatrix}x\\y\\0\end{bmatrix} \qquad \text{Eq. 2.14}$$

So equation of principal plane is;

$$p_{31}X + p_{32}Y + p_{33}Z + p_{34} = 0 \qquad \text{Eq. 2.15}$$

The normal of this plane, that is normalized (p_{31}, p_{32}, p_{33}) vector, defines the direction of camera that the photograph is taken.

2.3.4 Camera Projection Matrix (P) Decomposition

As expressed in Eq. 2.5, **P** can be decomposed into **C, R** matrices and T vector, which are calibration, rotation matrices and translation vector respectively. This can be done in two different ways; either by using QR Decomposition or by using orthogonal properties of rotation matrix.

QR Decomposition is simple because C is already an upper triangular matrix, and R is a rotation matrix that is orthogonal where

$$R^{-1} = R^T$$

On the other hand if we write Eq. 2.5 in a more open form

$$\mathbf{P} = p_{34}\begin{bmatrix}\mathbf{p}_1 & p_{14}\\ \mathbf{p}_2 & p_{24}\\ \mathbf{p}_3 & 1\end{bmatrix} = \begin{bmatrix}\alpha_u & k & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{r}_1 & t_x\\ \mathbf{r}_2 & t_y\\ \mathbf{r}_3 & t_z\end{bmatrix} \qquad \text{Eq. 2.16}$$

Where, $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ are the 1x3 row vectors of **P** without the last coordinate and similarly $\mathbf{r}_1, \mathbf{r}_2$ and $\mathbf{r}_3$ are the row vectors of **R**. When calculating **P**, the equation has an arbitrary scale factor and p_{34} is set to 1. But when we decompose **P** this scale factor must be considered as well. The following equations are derived by employing orthogonallity property of **r** vectors [17].

$$t_z = p_{34} = \frac{1}{\|\mathbf{p}_3\|} \qquad \text{Eq. 2.17}$$

$$\mathbf{r}_3 = \frac{\mathbf{p}_3}{p_{34}}$$

$$u_0 = p_{34}^2\left(\mathbf{p}_1 \bullet \mathbf{p}_3\right)$$

$$v_0 = p_{34}^2 \left(\mathbf{p}_2 \bullet \mathbf{p}_3\right)$$

$$\alpha_v = p_{34}^2 \left\|\mathbf{p}_2 * \mathbf{p}_3\right\|$$

$$\mathbf{r}_2 = \frac{p_{34}}{\alpha_v}\left(\mathbf{p}_2 - v_0 \mathbf{p}_3\right)$$

$$\alpha_u = \sqrt{p_{34}^2 \left\|\mathbf{p}_1 * \mathbf{r}_2\right\|^2 - u_0^2}$$

$$k = \sqrt{p_{34}^2 \left\|\mathbf{p}_1 * \mathbf{r}_3\right\|^2 - \alpha_u^2}$$

$$t_y = \frac{p_{34}}{\alpha_v}\left(p_{24} - v_0\right)$$

$$\mathbf{r}_1 = \frac{1}{\alpha_u}\left(p_{34}\mathbf{p}_1 - k\mathbf{r}_2 - u_0\mathbf{r}_3\right)$$

$$t_x = \frac{1}{\alpha_u}\left(p_{34}p_{14} - kt_y - u_0 t_z\right)$$

Where $\|.\|, (*), (\bullet)$ denote magnitude of vector, cross product of vectors and dot product of vector respectively.

All of these parameters can be determined if we know the sign of k. This is equivalent to knowing whether the origin of the 3D world coordinate system is in front of the camera or not ($t_z < 0 \quad or \quad t_z > 0$).

2.4 3D Feature Point Calculation

To find the 3D coordinates of feature points, we bring together the 2D coordinated of feature points and calibration matrices from all photographs [10]. Then Eq. 2.7 can be rearranged as follows:

$$X\,(p_{11}^{i} - u_i p_{31}^{i}) + Y\,(p_{12}^{i} - u_i p_{32}^{i}) + Z\,(p_{13}^{i} - u_i p_{33}^{i}) = u_i - p_{14}^{i}$$
$$X\,(p_{21}^{i} - v_i p_{31}^{i}) + Y\,(p_{22}^{i} - v_i p_{32}^{i}) + Z\,(p_{23}^{i} - v_i p_{33}^{i}) = v_i - p_{24}^{i} \qquad \text{Eq. 2.18}$$

Where u_i and v_i are 2D coordinates of the feature points on the *i*`th image and *p*`s are the entries of the perspective projection matrix **(P)** of the same image. For this equation system to have a valid solution, each feature point must be specified in at least two images. These equations can be written in a matrix form;

$$\begin{bmatrix} (p_{11}^{1} - u_1 p_{31}^{1}) & (p_{12}^{1} - u_1 p_{32}^{1}) & (p_{13}^{1} - u_1 p_{33}^{1}) \\ (p_{21}^{1} - v_1 p_{31}^{1}) & (p_{22}^{1} - v_1 p_{32}^{1}) & (p_{23}^{1} - v_1 p_{33}^{1}) \\ \vdots & \vdots & \vdots \\ (p_{11}^{N} - u_N p_{31}^{N}) & (p_{12}^{N} - u_N p_{32}^{N}) & (p_{13}^{N} - u_N p_{33}^{N}) \\ (p_{21}^{N} - v_N p_{31}^{N}) & (p_{22}^{N} - v_N p_{32}^{N}) & (p_{23}^{N} - v_N p_{33}^{N}) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u_1 - p_{14}^{1} \\ v_1 - p_{24}^{1} \\ \vdots \\ u_N - p_{14}^{N} \\ v_N - p_{24}^{N} \end{bmatrix} \qquad \text{Eq. 2.19}$$

This linear equation system is solved by pseudo-inverse method which was used for perspective projection matrix calculation in Eq. 2.12.

CHAPTER 3

FACE MESH DEFORMATION

3.1 Introduction

In the previous chapter we found the 3D coordinates of the feature points. Now, feature points will be used to deform all the vertices in the generic face mesh [9]. A smooth interpolation function will be constructed for the 3D displacements of the mesh vertices. Given *n* feature points, displacement for a feature point *i* is defined as follows:

$$\mathbf{d}_i = \mathbf{X}_i - \mathbf{X}_i^0 \qquad \text{Eq. 3.1}$$

Where $\mathbf{X}_i$, $\mathbf{X}_i^0$ are final and initial 3D coordinates of *i*'th feature point.

We want to construct a function, producing displacement vectors for unconstrained vertices. So we attempt to find a smooth vector-valued function **f** satisfying;

$$\mathbf{d}_i = \mathbf{f}(\mathbf{X}_i) \quad i = 1...n \qquad \text{Eq. 3.2}$$

Then function **f** can be applied to the vertices of face mesh. There are several ways to construct such an interpolation function **f**. We use a weighted linear combination of radial basis functions that give reasonable results for face mesh deformation [13].

Function **f** is in the following form:

$$\mathbf{f}(\mathbf{X}) = \sum_i \mathbf{c}_i \mathbf{r}(\left\|\mathbf{X} - \mathbf{X}_i^0\right\|) \qquad \text{Eq. 3.3}$$

Where **r** stands for radial basis function (RBF), and **c** is the displacement coefficient for each feature point. First of all RBF should be selected as a smooth, continuous, and decreasing function. Possible RBF functions are:

Polynomial Functions: $\mathbf{r}(\mathbf{x}) = a + bx + cx^2 + \cdots + kx^n$

Exponential Functions: $\mathbf{r}(\mathbf{x}) = \mathbf{e}^{-(a + bx + cx^2 + \cdots + kx^n)}$ Eq. 3.4

In this work, both of these functions were tried and the best result has been obtained with the Gaussian Function which is the Exponential Function with only a single non-zero coefficient of x^2. In addition to these RBF's, we also applied a bounded RBF function that is zero outside the given boundary.

Also a variable parameter RBF method, in which **x** is scaled such that RBF function output at nearest neighboring feature point will be close to zero, is applied. Hence RBF of dense feature points have smaller boundaries than sparse feature points. As a result, we minimize the number of vertices which remain unchanged by widening the boundaries of sparse feature points, and minimize the cross effects of feature points to each other.

After deciding on an RBF type, the construction of the **f** function needs the calculation of **c** vectors that are the coefficients of RBF. Values of **c** vectors will be arranged so that function **f** moves initial feature points to their known destination coordinates. For this aim, the following matrix is formed which is the combined form of the **f** function for all known feature points.

$$\begin{bmatrix} \mathbf{r}(\|\mathbf{X}_0^0-\mathbf{X}_0^0\|) & \cdots & \mathbf{r}(\|\mathbf{X}_j^0-\mathbf{X}_0^0\|) & \cdots & \mathbf{r}(\|\mathbf{X}_N^0-\mathbf{X}_0^0\|) \\ \vdots & & \vdots & & \vdots \\ \mathbf{r}(\|\mathbf{X}_0^0-\mathbf{X}_i^0\|) & \cdots & \mathbf{r}(\|\mathbf{X}_j^0-\mathbf{X}_i^0\|) & \cdots & \mathbf{r}(\|\mathbf{X}_N^0-\mathbf{X}_i^0\|) \\ \vdots & & \vdots & & \vdots \\ \mathbf{r}(\|\mathbf{X}_0^0-\mathbf{X}_N^0\|) & \cdots & \mathbf{r}(\|\mathbf{X}_j^0-\mathbf{X}_N^0\|) & \cdots & \mathbf{r}(\|\mathbf{X}_N^0-\mathbf{X}_N^0\|) \end{bmatrix} \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_i \\ \vdots \\ \mathbf{c}_N \end{bmatrix} = \begin{bmatrix} \mathbf{d}_0 \\ \vdots \\ \mathbf{d}_i \\ \vdots \\ \mathbf{d}_N \end{bmatrix} \qquad \text{Eq. 3.5}$$

Simply

$$\mathbf{R} * \mathbf{C} = \mathbf{D} \qquad \text{Eq. 3.6}$$

In the linear equation system Eq. 3.6, **R** is the RBF results of known feature point displacements. **C** is the unknown coefficient vectors written as a vector and **D** is the displacement vectors of feature points from their initial to final positions.

Since matrix **R** is an NxN square matrix and has an inverse, **C** can be calculated as follows:

$$\mathbf{C} = \mathbf{R}^{-1} * \mathbf{D} \qquad \text{Eq. 3.7}$$

Now we can apply this **f** function to all vertices of the generic face mesh, and deform it to fit to our subject's face features.

3.2 Addition of Refining Feature Points

Refining points are only used for 3D face model deformation, but they are not included in camera calibration. Their 3D coordinates are calculated in the same way as the feature points are. In deformation step, they are used together with feature points. There are no constraints on the number of refining points to be used. However if places of refining points are too close to the feature points, and variable parameter RBF's are used, the resulting face mesh deformation may not be very successful.

Feature points are assumed to be independent from each other, which means that a change in the coordinates of one, should not affect the other. Actually this is not an obligation and that can be eliminated by wider RBF. But experimental results show that the best results are achieved when the boundaries are arranged in a way that feature points have minimum affect on the others as in variable parameter RBF case. When a refining point is defined close to a feature point it reduces the boundary of this feature point and may cause undesired results in variable parameter RBF case. Instead bounded RBF's should be employed in such a case.

CHAPTER 4

TEXTURE EXTRACTION

4.1 Introduction

Up to now we have captured photographs of the subject's face, and manually marked some feature points on these photographs. Feature points on the photographs are employed to calibrate these photographs. Then, 3D coordinates of feature points are calculated and these coordinates are used to deform the 3D generic face mesh. It is now time to form the texture of our deformed face mesh.

The texture extraction problem can be defined as follows: Given a set of photographs, their viewing parameters and fitted face mesh, compute texture color **T(p)** for each point **p** on the face mesh.

Texture extraction is composed of two parts that are texture image formation and texture map formation, both of these are solved together in this thesis. We have multiple photographs of the subject which need to be combined to form a single texture image. Combined texture image is transformed with respect to a constant texture map. We also consider the reverse case, keeping texture image constant and transforming the texture map. But results obtained for this case are

unsatisfactory due to the lack of control on texture mapping in the 3D library used.

4.2 Texture Image Formation

Texture image function **T(p)** is a function of 3D coordinates of face mesh and gives the color values for each mesh point **p**. **T** can be expressed as the weighted combination of all images that have texture color information. Then

$$\mathbf{T}(\mathbf{p}) = \frac{\sum_i m_i(\mathbf{p}) I_i(x_i, y_i)}{\sum_i m_i(\mathbf{p})} \qquad \text{Eq. 4.1}$$

In this equation, function I_i stands for the *i*'th image that returns color value of input point (x_i, y_i). $m_i(\mathbf{p})$ is the weighting function of *i*'th image to texture image. These weights should be calculated by considering the following facts:

- Self-Occlusion: $m_i(\mathbf{p})$ should be zero if point **p** is occluded in the *i*'th photograph.
- Positional Certainty: $m_i(\mathbf{p})$ should depend on the "Positional Certainty" of point **p** that is defined as the dot product of surface normal and direction of perspective projection [21]. So $m_i(\mathbf{p})$ has the maximum value with the surfaces perpendicular to photograph direction, and 0 if they are parallel.

Self-Occlusion part can be divided in two parts. Surfaces whose front side are not visible from the point of view and surfaces which are occluded by other parts of the face. First is the definition of back face culling which can be calculated easily as follows:

If normalized direction of *i*'th camera is labeled as $\mathbf{d}_i$ and surface normal at point **p** is $\mathbf{n_p}$**,** then:

$$\cos\alpha_i^{\mathbf{p}} = (\mathbf{d}_i \bullet \mathbf{n_p}) \qquad \text{Eq. 4.2}$$

Where $(\bullet)$ is dot product and $\alpha_i^{\mathbf{p}}$ is the angle between the surface normal and the camera direction. If surface can be seen from the camera direction then the angle between them must be greater than 90 degrees which is $\cos\alpha_i^{\mathbf{p}} < 0$.

Second type of Self-Occlusion is not easy to calculate and it can be neglected after back face culling for performance reasons.

Positional certainty is calculated similar to back face culling, where $\alpha_i^{\mathbf{p}}$ is the angle between *i*'th camera direction and the normal of surface containing point **p.** Weighting function of images in combining process is defined as:

$$m_i(\mathbf{p}) = -\cos\alpha_i^{\mathbf{p}}$$ Eq. 4.3

Self-Occlusion occurs at concave surfaces such as sides of the nose. Depending on the direction of camera, the nose can occlude a part of the cheek. While the algorithm used here does not eliminate Self-Occlusion, texture from the nose can also be used for the occluded cheek part. A lower limit to weighting function can help to reduce occlusion errors while occluded parts usually have fewer weights. This limit also eliminates other small errors which may be mapped to large surfaces.

After texture image blending using Eq. 4.1**,** blended texture will be transformed to fit the surfaces as defined at generic face texture map. For each surface (in our case triangular patch) a transformation function is defined by colliding corners of the surface with the coordinates from the texture map. So we reduce the transformation of texture image problem to calculation of a 2D affined transformation matrix **T,** capable of transforming initial triangle to final one. Corners of initial and final triangles are i_1 , i_2 , i_3 and f_1 , f_2 , f_3 respectively, where:

$$i_1 = \begin{bmatrix} x_1^i \\ y_1^i \\ 1 \end{bmatrix}, i_2 = \begin{bmatrix} x_2^i \\ y_2^i \\ 1 \end{bmatrix}, i_3 = \begin{bmatrix} x_3^i \\ y_3^i \\ 1 \end{bmatrix}$$ Eq. 4.4

$$f_1 = \begin{bmatrix} x_1^f \\ y_1^f \\ 1 \end{bmatrix}, i_2 = \begin{bmatrix} x_2^f \\ y_2^f \\ 1 \end{bmatrix}, i_3 = \begin{bmatrix} x_3^f \\ y_3^f \\ 1 \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}\, i_1 = f_1 \,, \mathbf{T}\, i_2 = f_2 \,, \mathbf{T}\, i_3 = f_3$$

Eq. 4.4 can be rewritten in a combined form:

$$\begin{bmatrix} x_1^i & y_1^i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1^i & y_1^i & 1 \\ x_2^i & y_2^i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2^i & y_2^i & 1 \\ x_3^i & y_3^i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3^i & y_3^i & 1 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{21} \\ t_{12} \\ t_{22} \\ t_{13} \\ t_{23} \end{bmatrix} = \begin{bmatrix} x_1^f \\ y_1^f \\ x_2^f \\ y_2^f \\ x_3^f \\ y_3^f \end{bmatrix} \qquad \text{Eq. 4.5}$$

This linear equation system has a solution if rows of the left most matrix are linearly independent, in other words, corners of triangles do not lie on a line.

CHAPTER 5

FACEGENERATOR

5.1 Introduction

Previous chapters describe the theoretical aspects of the study. This chapter explains the FaceGenerator application that is capable of forming 3D face models from photographs.

The FaceGenerator application is compiled with Standard Development Kit (SDK) version 1.4.2. In addition to this, standard Java core application employs Java3D and JFreeChart. Standard JDK includes Java2D that has been employed in this thesis for image processing and viewing purposes. Standard JDK also includes “Java API for XML Processing” (JAXP) library that has been employed for Extensible Markup Language (XML) processing. Java3D is a 3 dimensional, high-level graphics extension of Java that is used to form 3D part of FaceGenerator. JFreeChart is a free library capable of drawing vast variety of plots on to the screen, and is used for process analysis and debugging.

This chapter explains the programming language Java and Java3D and FaceGenerator application.

5.2 Java

First of all, Java is a platform independent programming language that needs an interpreter Java Runtime Environment (JRE) to run java programs. Platform independency makes java very feasible for internet applications. Code is written and deployed once and runs on all operating systems.

Second, Java has a huge developer community that publishes free extensions like JFreeChart which is also used in this study for plotting purposes in explaining program execution. A lot of open source examples and free books are also available at internet on nearly all Java related subjects.

Finally, java is a pure object-oriented language that makes it invaluable. The programs that are written in this manner are extensible and easy-to-understand. It also has a default exception handling mechanism that is easy-to-debug and shortens the development time.

However, Java has some drawbacks such as the lack of simple binary types like unsigned char and the interpreter that could cause a performance decrease. Since excluding some simple data types is the choice of SUN Inc. developers, there is nothing to be done about it. But on the other hand, new versions of JRE use Just-In-Time Compilers that convert simple binary operations to native platform

machine code, and nearly no performance difference can be sensed compared to native codes.

5.3 Java2D

Java2D is a 2D graphics framework and is included in standard JRE. This library is capable of creating arbitrary shapes, texts and images. You can manipulate generated graphics with the help of Java2D API. Affined Transformations that are rotation and translation, Composition that generates transparent graphics, Color Transformation that changes any color parameter, Filtering that is applies any kind of convolution to graphics for blurring or sharpening purposes, Data Selection that chooses any arbitrary region of interest, are widely used tools in FaceGenerator.

In FaceGenerator, points, lines, bezier curves are the main tools for feature point marking. It is also very flexible so that user can modify the rendered lines by mouse movements. Also lines can define regions that can be used for any purposes such as affine transformation, clipping, etc.

Images can be loaded from and saved to JPEG, PNG and GIF formats which are most common ones. Images are easily resized by the Java2D API. User can define rendering quality and performance by setting rendering parameters.

All of these graphics can be manipulated easily with vast variety of transformations. FaceGenerator creates texture image by employing affine transforms, composition, color transformation, filtering by the help of feature lines and points defining the region of interest. Affined transforms are applied in homogeneous coordinates, so that user can apply rotation and translation at the same time. Color transformation is used to equalize brightness of the face images. Composition can be applied to any image loaded to the program with an alpha parameter that is weight of newly rendered image. New graphics and background are blended with alpha parameter providing a transparent look. At the last stage of texture generation blur filtering is applied on to the texture to reduce the contours of texture image by Java2D API.

5.4 JAXP

Java API for XML Processing (JAXP) is a standard API in Java, enables parsing and transforming XML documents. XML is a simple, widely used standard which enables the user to define complex data types. In FaceGenerator, all the application data specific to FaceGenerator is stored in XML format. The main advantage of XML is its capability of viewing and editing them by external programs. This is very helpful for debugging. The future advantage of XML usage is the capability of sharing XML documents over internet for teleconferencing.

5.5 Java3D

Java3D is a Java extension developed by Sun Microsystems, Inc. as a joint collaboration with Silicon Graphics, Inc., Intel Corporation, Apple Computer, Inc. and it is free to use. In this study Java3D version 1.3 is employed. Actually this extension is not implemented in pure Java; it is a wrapper of two commonly used graphics libraries, openGL and directX. This study uses openGL core Java3D that is used on platforms like UNIX.

Java3D extension is designed as a high-level library, which is the main advantage of using it. Developers do not need to optimize its performance for different hardware platforms.

The scene graph programming model is a simple and flexible way for both representing and rendering complex 3D worlds. All information of the virtual world and the entire scene is included in this scene graph.

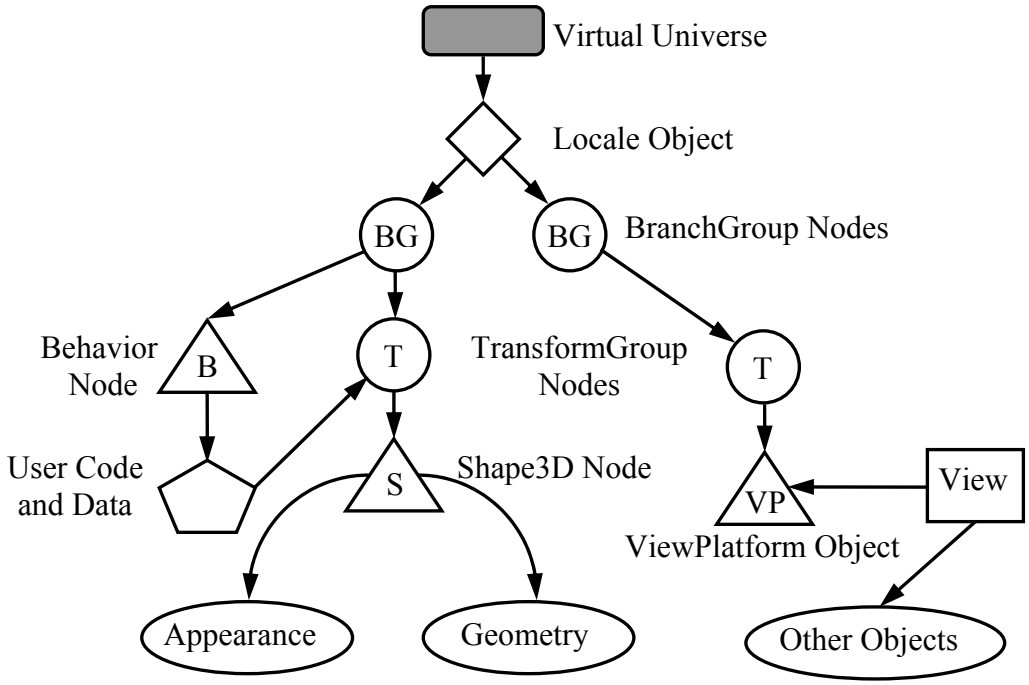


Figure 5.1 : Java3D scene graph

Figure 5.1 illustrates a simple application's scene graph. A simple scene graph consists of superstructure components which are a VirtualUniverse, Locale objects, and scene graph branches, or subgraphs. Each subgraph is rooted by a BranchGroup node that is attached to the superstructure. Subgraphs hold data of 3D shapes and viewing properties.

A VirtualUniverse object is the root of a scene graph. User can create more than one VirtualUniverse in Java 3D which is not the case vast majority of applications. All Java 3D scene graphs must be connected to a Virtual Universe object to be displayed.

Below the VirtualUniverse object is a Locale object. The Locale object defines the origin, in high-resolution coordinates, of its attached subgraphs. A Virtual Universe may contain as many Locales as needed.

The scene graph branch itself starts with the BranchGroup nodes. Only BranchGroup objects can attach to Locales. A BranchGroup roots a subgraph, or branch graph, of the scene graph. In Figure 5.1, there are two subgraphs and, thus, two BranchGroup nodes. Attached to the left BranchGroup are two subtrees. One subtree consists of a user-extended Behavior leaf node which contains Java programming language code to manipulate the transform matrix associated with the object's geometry. The other subtree in this BranchGroup consists of a TransformGroup node that specifies the position (relative to the Locale), the orientation, and the scale of the geometric object in the virtual universe. A single child, a Shape3D node, refers to two component objects: a Geometry object and an Appearance object. The Geometry object describes the geometric shape of a 3D object and the appearance object describes the appearance of the geometry.
The right BranchGroup has a single subtree that consists of a TransformGroup node and a ViewPlatform leaf node. The TransformGroup specifies the position (relative to the Locale), the orientation, and the scale of the ViewPlatform. This transformed ViewPlatform object defines the end user's view within the virtual universe. Finally, the ViewPlatform is referenced by a View object that specifies all of the parameters needed to render the scene from the point of view of the ViewPlatform.

5.6 FaceGenerator

In this section, our face modeling process is explained in conjunction with our application named FaceGenerator. Face modeling can be divided into three sub-roblems which are:

1. Camera calibration and feature point calculation handled in Image Calibration module,
2. 3D model deformation handled in Model Deformation module and
3. Texture generation handled in Texture Generation module.

A proper process must be performed in the same order that is calibration, deformation and texture generation.

5.6.1 Image Calibration

Face Modeling starts with image calibration in which captured digital photographs are loaded to FaceGenerator. FaceGenerator can use common image formats JPEG, GIF and PNG, which are standard Java2D file formats.

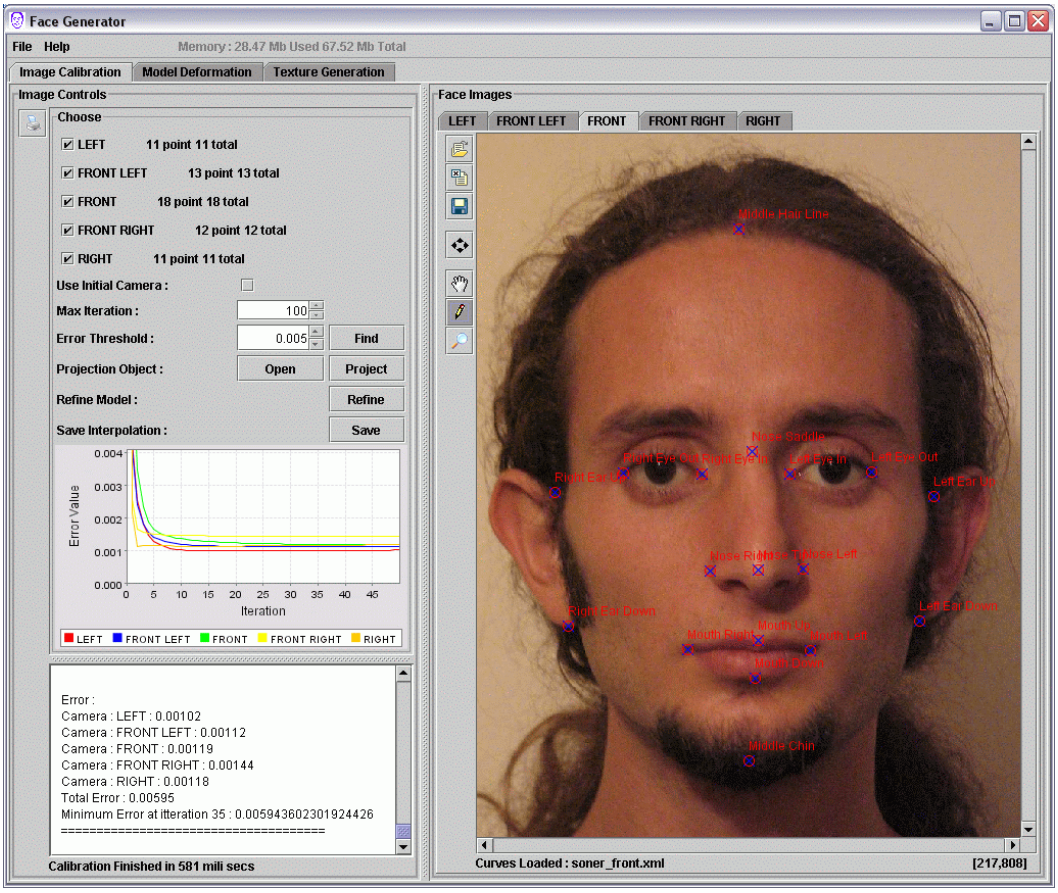


Figure 5.2 : Image Calibration

Quality of face images directly effects the results of the modeling process. Better resolution images result better modeling which is bounded by the capability of computer used in modeling. Images occupy a lot of memory when they are viewed. Since multiple images are viewed - typically 5 face images in our application - an optimum size should be selected. Experimentally we decided on image size of 800x600 pixels which requires 60 kilobytes on average in JPEG format. A sample FaceGenerator application operating on 5 such images uses up to 70 megabytes of memory.

Besides resolution, sharpness is another important factor that affects feature point marking process and calibration in turn. Feature points can be recognized and marked more precisely on shaper images.

These loaded images will be used in texture generation which has similar image prerequisites with calibration phase. In addition to these constraints, images of face should have equivalent luminance. Flash should not be used because shining effects texture images. It is also better not to have shadow on face for the same reason. Some of these imperfections can be eliminated by the help of histogram equalization of face images. But histogram equalization is left as future work and only simple brightness equalization is performed over feature points. So the photographs used in this thesis should be taken in equivalent lighting conditions.

All of these practical rules are summarized as follows:

- Photographs should be taken roughly at the same vertical level.
- Landscape orientation of cameras should be preferred to increase face details.
- Resolution of images should be larger than 800x600.
- Similar lighting conditions should be supplied. Photographs should be taken with plenty of light, which should not cause shining and shadows on the face.

- Subject should have neither any hair on face, like bread or mustache, nor any instruments that occlude face parts, like glasses.
- Direction of the camera should be such that maximum number of feature points appears on photographs.

FaceGenerator application can load up to 5 images of the subject, which are left, front, right and 2 front side views. To calculate 3D coordinates of a feature point it must be marked on at least 2 images. As a result we need at least 2 images of subject theoretically, but because of occlusion of feature points – only front image can see all of them – practically 3 images are needed for a successful solution.

Next step in the image calibration part is feature point marking on photographs. In this work 18 predefined feature points are used for calibration. They are inner and outer corners of eyes, upper and lower connections of ears to the face, left, right up and down corners of mouth and left, right, and tip of nose. These points are selected for their ease of distinguish ability

[5] from all point of views.

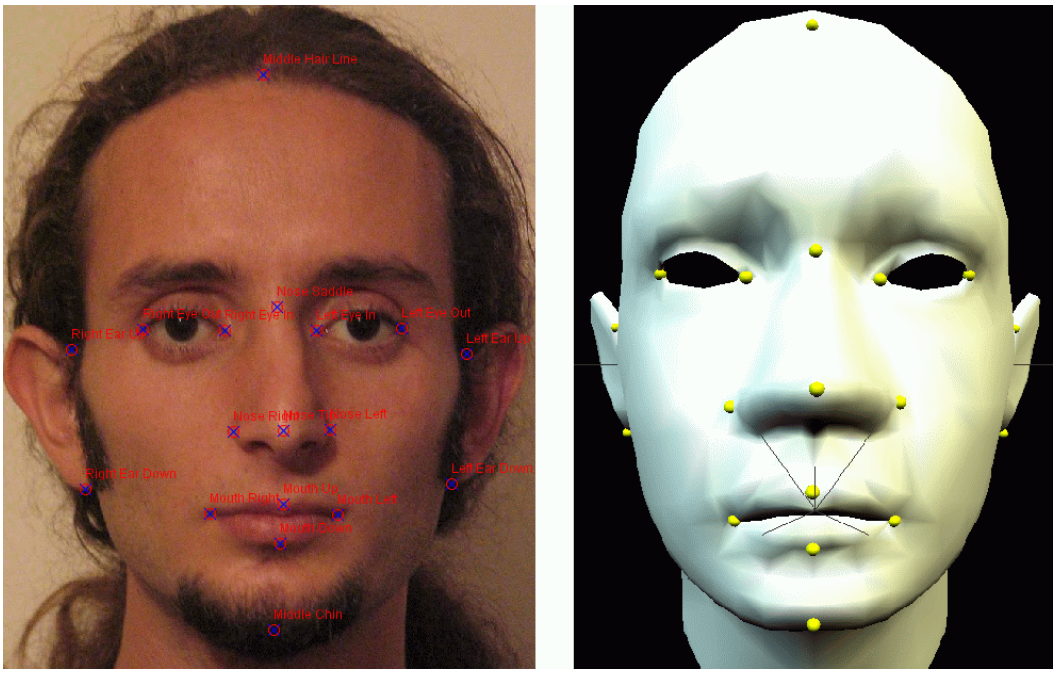


Figure 5.3 : Feature Points in 2D and 3D

There are same other parameters used in calibration process which will be mentioned later as they are used.

Calibration process starts with normalization phase. Feature points in each image and face model will be normalized before calibration. Normalization is performed separately for each image and face model.

After image calibration is completed, inverse of these normalization transforms will be applied to the calibration results to convert the normalized results to real world coordinates.

Feature point normalization is followed by the camera calibration and 3D feature point calculation. This is an iterative process composed of three subtasks that are camera calibration, error calculation and 3D feature coordinate calculation. Convergence of this iterative process depends on the error of initial state that you start. If inputs differ from the generic face hugely, then iteration may not converge or it converges to a local minimum. Since this iteration has two sets of unknowns, 3D feature coordinates and projection matrices of cameras, the iteration can be started in two different ways. First one is guessing the initial 3D feature points, and the trying to find the projection matrices. The second one is guessing the initial projection matrices of cameras, and then trying to find coordinates of feature points. They both have some advantages compared to each other.

In the first case, initial 3D feature coordinates are feature points of the generic face. Also algorithm becomes capable of using arbitrary face images while there is not any assumption on initial camera positions.

Second case demands a guess of initial projection matrices which hold the information of internal camera parameters and geometric positions of the camera. In case of normalized data, scaling and translation are eliminated but it is still a difficult task to guess other parameters like focal length. This type of iteration is useful when projection matrix of an image has inverse rotation (determinant of rotation matrix is -1) or symmetric projection (center of projection is symmetric with respect to origin in normalized case).

Mathematical details of iterative camera calibration and 3D feature coordinate calculation were explained in Chapter 2 previously. There are two types of pseudo-inverse method used in these calculations in over-defined problem case. First one is direct implementation of pseudo-inverse method that is using all the data at once. The second type is using a combination of input data and then averaging the results of each combination to find the final result. An example to the second method is the calculation of the projection matrix for the front image that has typically 18 feature points where 6 of them could be enough. Projection matrix will be calculated for each of the 6 combinations of 15 feature points and the results will be averaged. No improvement has been achieved with this second method while the calculation costs much more CPU time compared to using all data at once.

Error value is calculated as two dimensional displacement between the marked feature point and the iterated one which was projected to the image plane. The program draws a graph of error versus iteration count so that user can see if the error of each image decreases to an acceptable level. Feature points with minimum error values are displayed on images and saved for 3D face model deformation.

5.6.2 Model Deformation

FaceGenerator has a model deformation module similar to Image Calibration one.

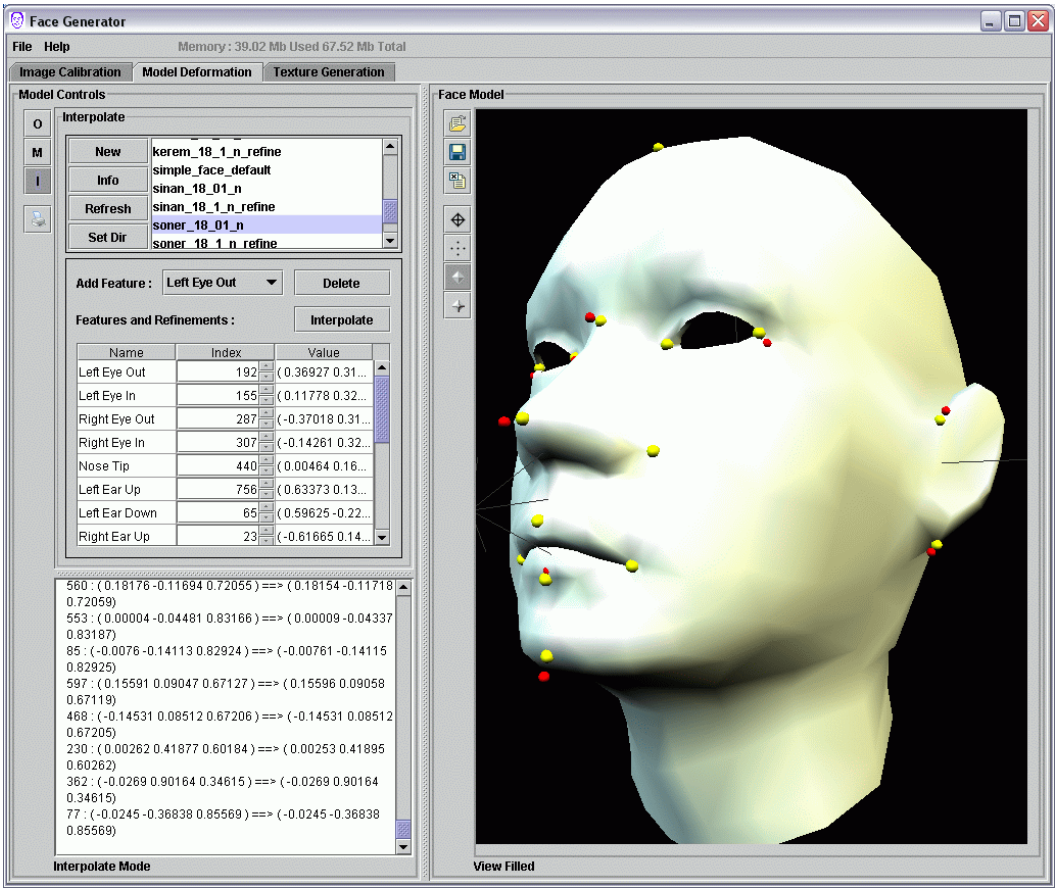


Figure 5.4 : Model Deformation

Model deformation panel is composed of a 3D scene viewer and feature point edition property.

Initial face mesh whose feature points were used in calibration process must be loaded to the scene viewer. FaceGenerator application can load OBJ and VRML 3D file formats. In addition to 3D geometry definition, these formats also support texture loading. A default face mesh and a texture map with 827 vertices are mainly used in this thesis.

Following the face model loading, results of image calibration module, 3D feature and refinement points must also be loaded. This module is capable of editing all attributes including deletion of 3D feature points. The program displays initial and final coordinates of features and refinements in 3D scene viewer.

The next step is to select a radial basis function (RBF) which will be used to deform the generic face model. The user can define the type of RBF which are polynomial and exponential RBF's. In boundary definition we propose two solutions, fixed boundary and variable boundary.

In fixed boundary definition, the RBF has zero value outside the given boundary.

In variable boundary definition, the parameters of RBF are calculated for each feature point so that RBF has the desired value at the nearest neighboring feature point.

Given a 3D feature point set, a RBF and a generic 3D face mesh, FaceGenerator can deform the face mesh and show the resultant mesh in this module. Feature points can be applied more than once to have less error.

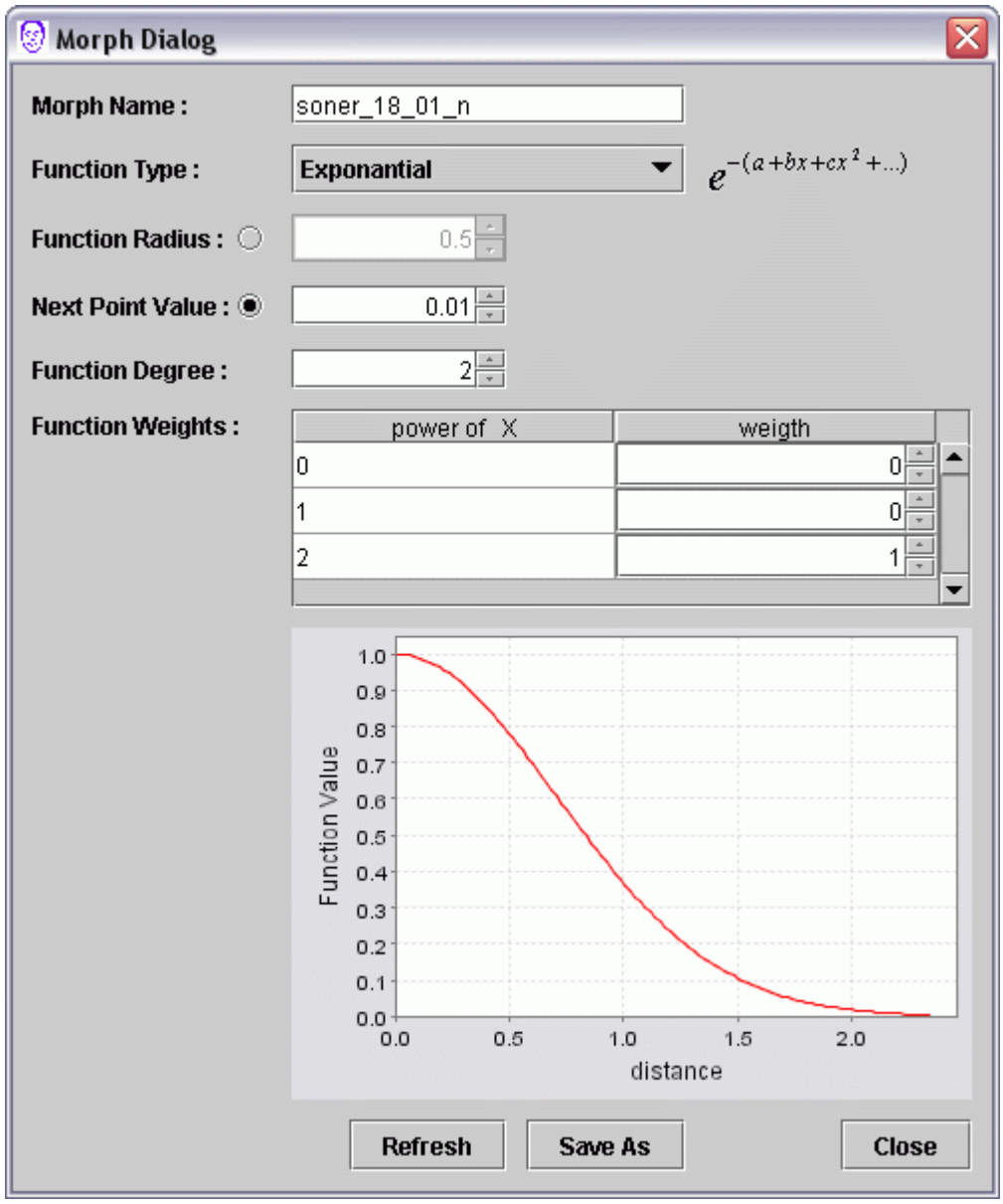


Figure 5.5 : Radial Basis Function (RBF)

5.6.3 Texture Generation

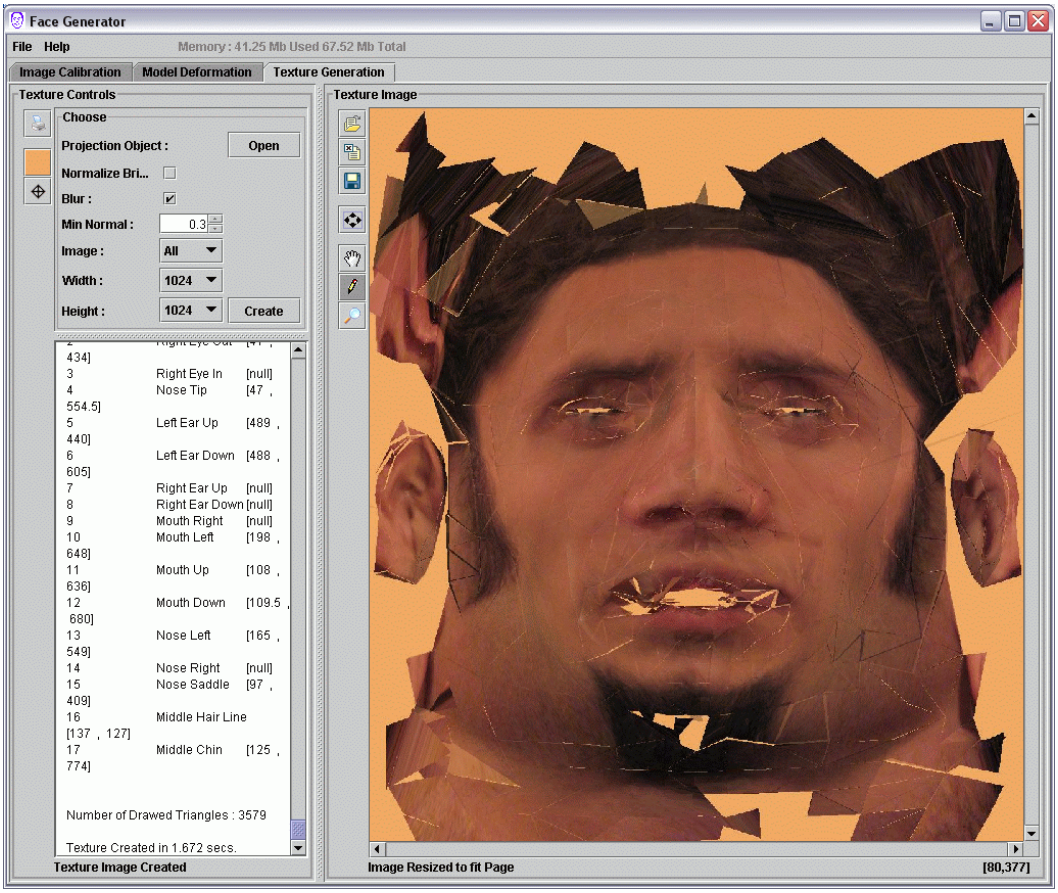


Figure 5.6 : Texture Generation

Texture generation module is developed to collect all the necessary information and then generate texture image. Data used for texture generation is listed below:

- Deformed face mesh
- Camera directions for each image
- Image files

- Texture map

This module automatically collects the most recent versions of listed information.

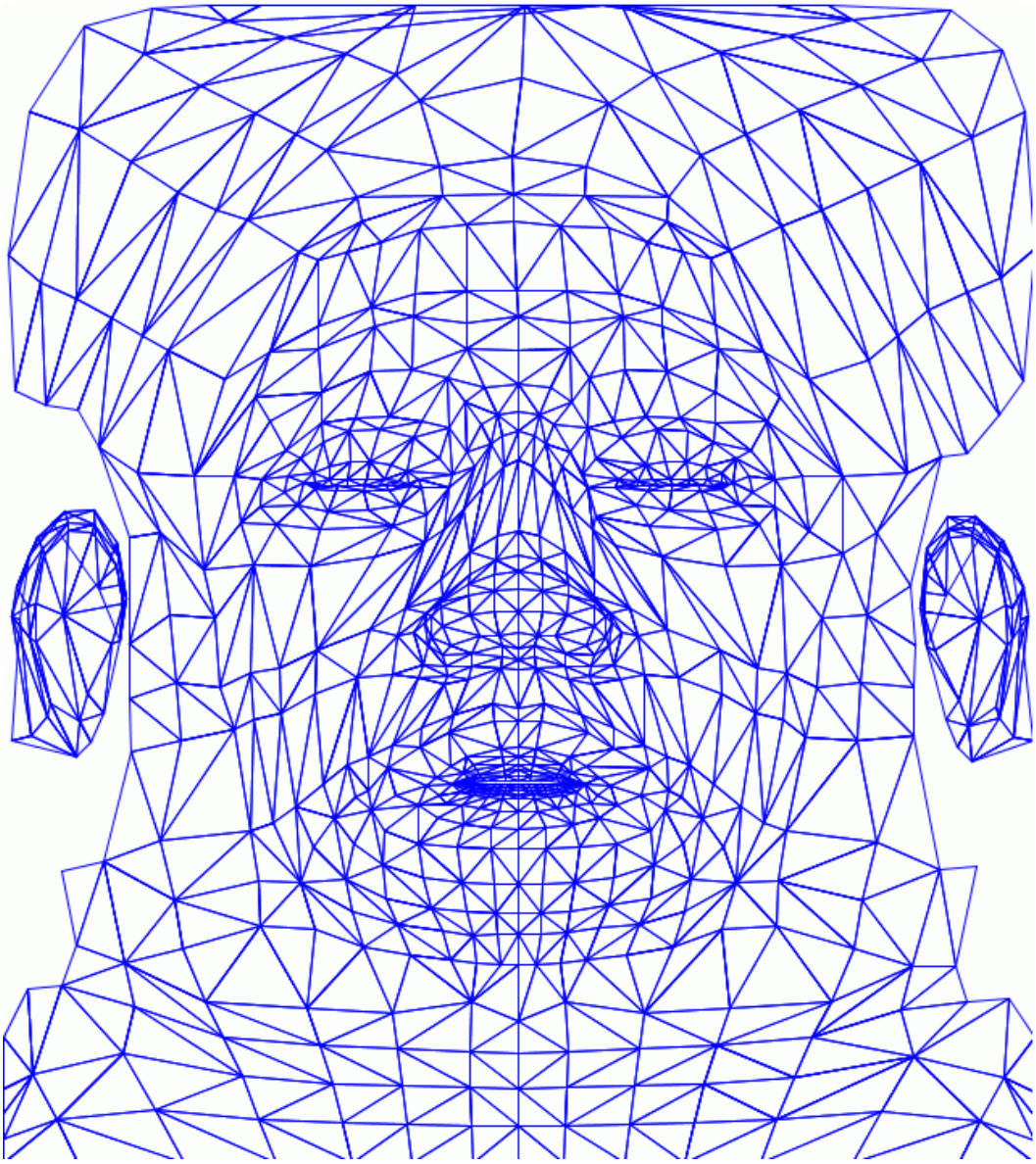


Figure 5.7: Texture Map

In this thesis a fixed texture map as shown in Figure 5.7 is employed, and texture image is transformed to fit this texture map.

In addition to texture generation data, some adjustable performance criterias may be involved to achieve a better texture image. Now these criterias will be explained.

The most important parameter is the threshold that is applied to weight images in blending process. This threshold is very useful in eliminating faulty texture triangles. The higher the threshold value, the more triangles discarded, which may be both faulty or not.

The next property of texture generation module is brightness normalization. This is a simple function that calculates brightness of feature points. Then average brightness's of all feature points are equalized. Reference brightness is obtained from frontal image. This is not working perfectly because of nature of the feature points. Feature points are corners of face organs, which do not have a smooth surface. As an alternative method, 25 pixels (5x5 window) neighborhood of each feature point is averaged to reduce the brightness variation around that feature point. With this method, satisfactory results are obtained.

Another property of this module is blurring. While triangles are transformed to their mapped positions, small errors may occur due to casting of double values of pixels to integers. This type of error usually creates contour lines at the

intersection of triangles. A color value close to human skin color can be selected as background and if the result is still poor, we may apply a blurring convolution to the texture map. Kernel of blurring convolution is:

$$BLUR = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} \qquad \text{Eq. 5.1}$$

Height and width of the generated texture can be changed. Performance of face rendering is directly affected by the size of the texture image.

Finally, images which will contribute to the texture image can be selected. This selection can be used to discard the mis-calibrated images.

CHAPTER 6

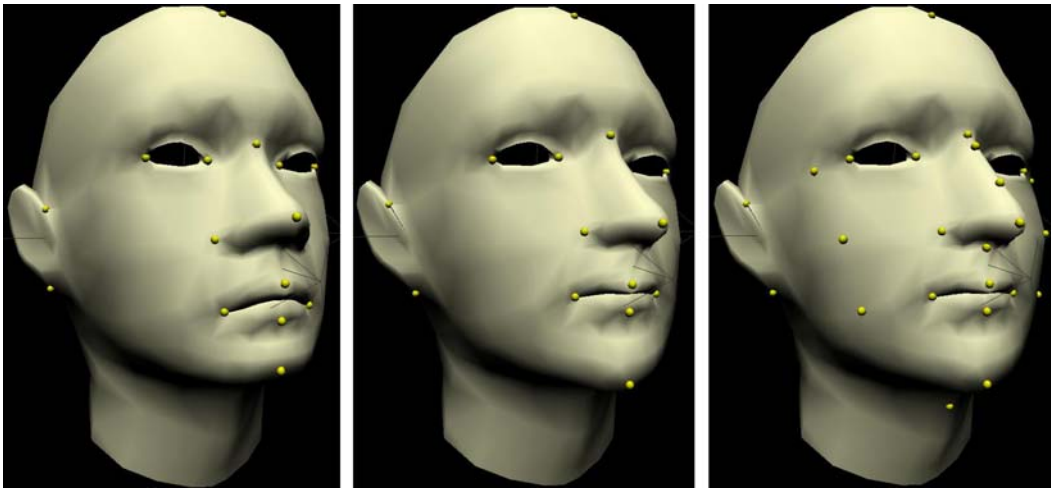
EXPERIMENTAL RESULTS

6.1 Introduction

The deliverables of face generation method and the results are explained in this chapter. There are two deliverables for each face model. One of them is the face mesh deformation file in XML format and the other is the face texture image file in JPEG format. 3D face model can be generated with these files and initial face model.

FaceGenerator application is executed with 5 images and 18 feature points as shown in Figure 6.1. Blue marks on face images indicate feature points of this sample run. 11 points on the left image, 15 points on the front left image, 18 points on the front image, 13 points on the front right image, 11 points on the right image are marked.

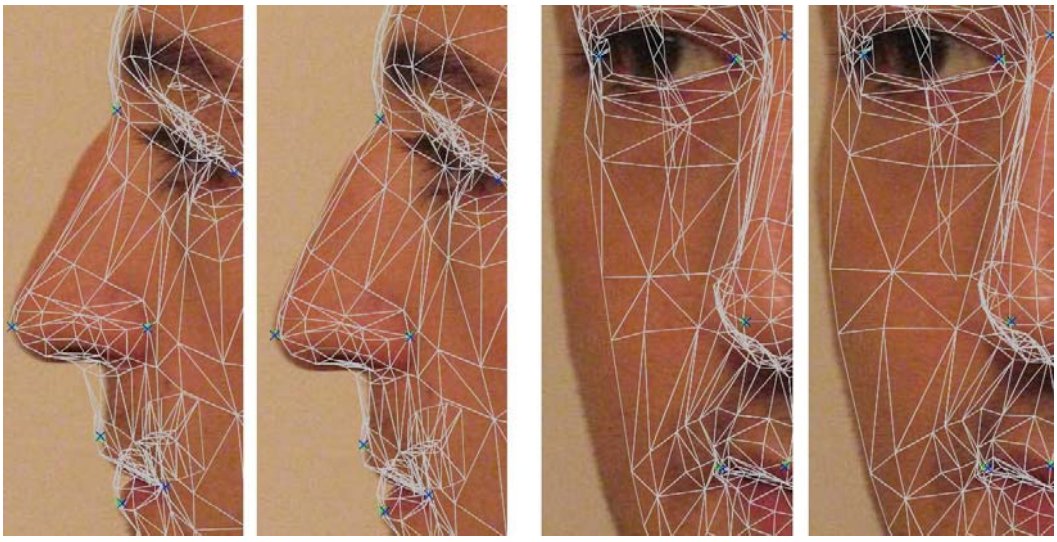
Figure 6.1 : Face images with feature points



(a) (b) (c)

Figure 6.2 : 3D deformation of initial face mesh

(a) initial face mesh with feature points, (b) deformed face mesh with feature points, (c) deformed face mesh with feature points and refinement points.



(a) (b)

Figure 6.3 : Refinement on nose and cheek

(a) mesh deformation for nose contour, (b) mesh deformation for cheek contour.

Figure 6.2 shows the sequence of 3D deformation of face mesh. Figure 6.2 (b) is the deformed 3D face model with 18 feature points and Figure 6.2 (c) is with 18 feature points and 10 additional refinement points. Figure 6.3 shows enhancement obtained at refinement step.

(a) (b)

Figure 6.4 : 3D face model and texture image
(a) Textured 3D face model, (b) generated texture image.

Figure 6.4 (a) shows the textured 3D face model using texture image given in Figure 6.4 (b). In texture image the undefined texture surfaces such as inside the mouth and below the chin and unused parts of the texture image such as areas around ears are painted in yellow background color.

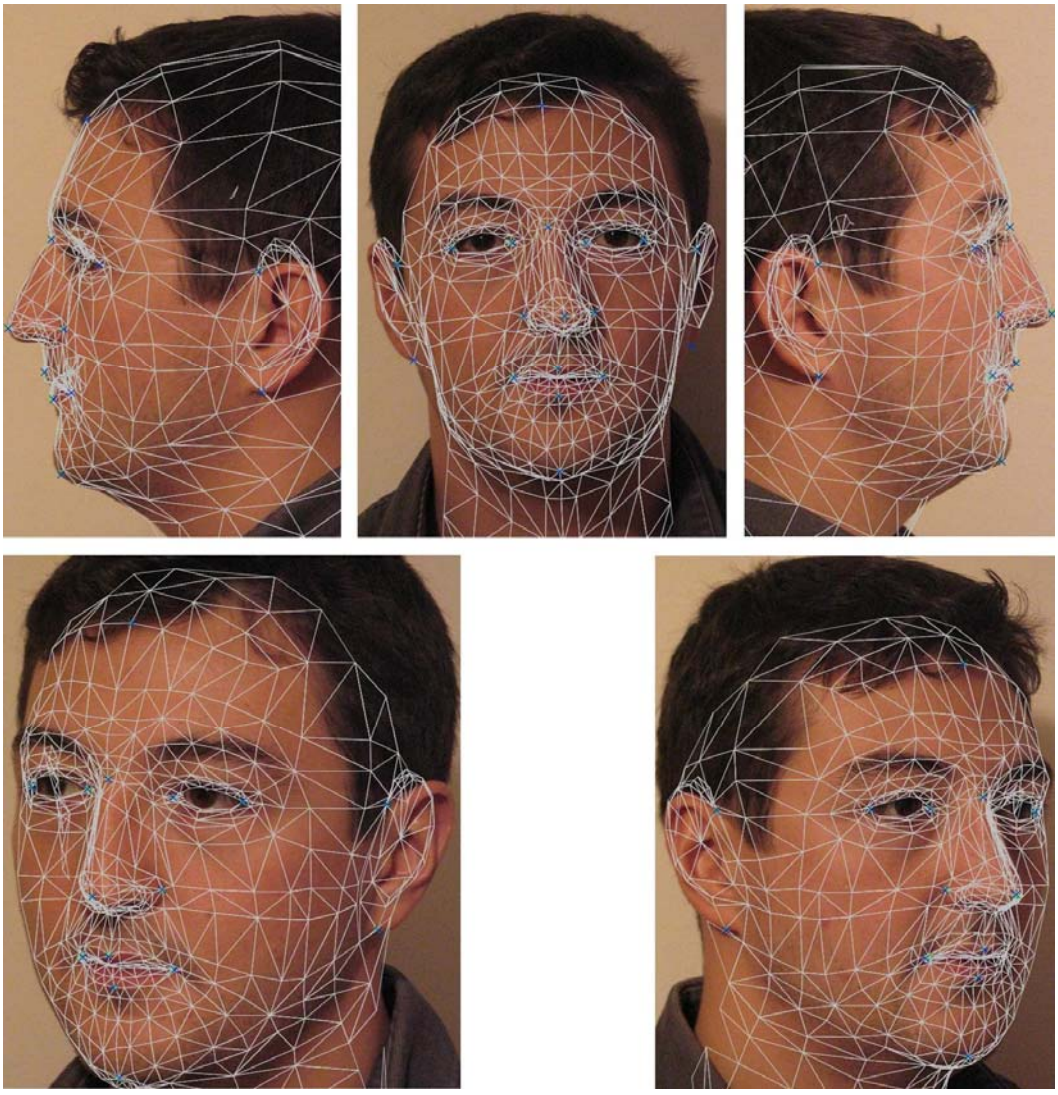


Figure 6.5 : Face images with projected final 3D mesh

Final results of the 3D face mesh deformation with feature and refinement points are shown in Figure 6.5 by projecting the mesh on to the initial images.

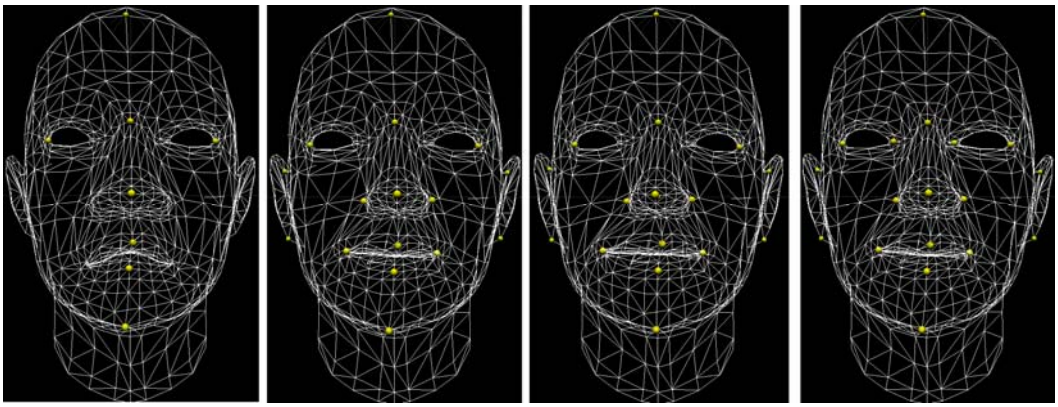
Face model generation method proposed in this thesis depends on different parameters. The following sections explain the effects of these parameters for each of them separately while keeping others constant.

6.2 Models with Different Number Photographs

Feature point calculation method seeks at least two 2D coordinates of a feature point to find its 3D coordinate. Hence the whole method needs at least two photographs of the face. Since feature points can not be viewed at each image simultaneously, we may need to increase the number of images to maintain all feature points to be seen at two or more images simultaneously. 3D coordinates of 18 feature points defined in this thesis can be found typically with 5 images.

Examples of face modeling with 2 to 5 images are presented below. The same images with the same feature points are used but only the number of images is changed. The images for each example are taken in a way that the maximum possible number of feature points is observed in them. Chosen images and the number of feature points with their 3D coordinates are as follows:

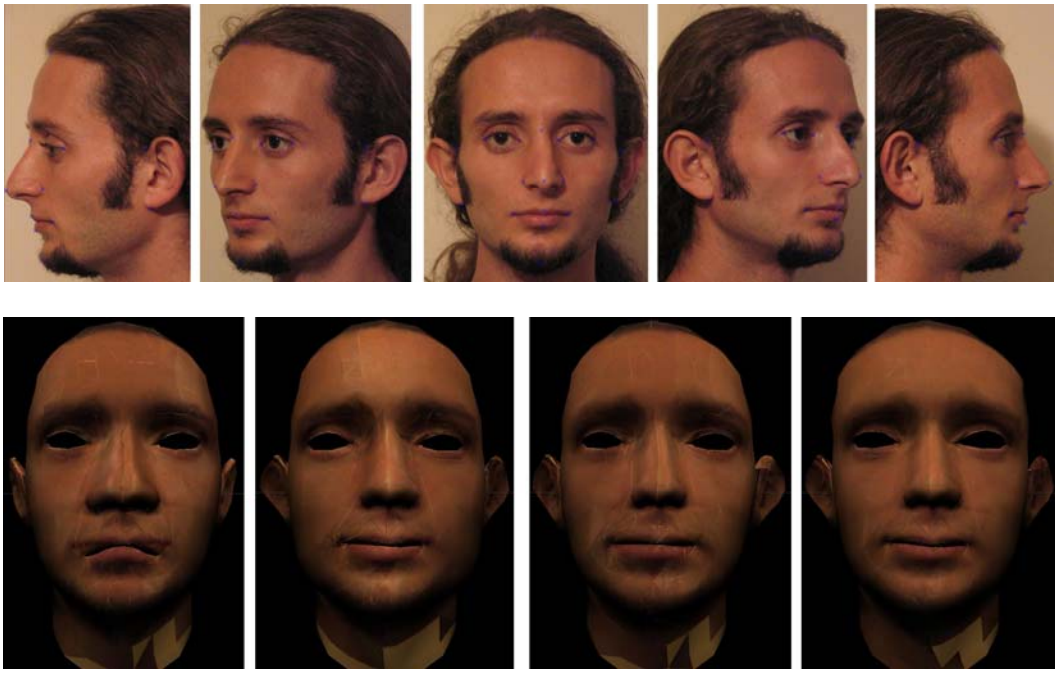
- 2 Images (front left, front right) : 8
- 3 Images (left, front, right) : 16
- 4 Images (left, front left, front right, right) : 16
- 5 Images (left, front left, front, front right, right) : 18



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

Figure 6.6 : 3D feature points found in each trial

(a) for 2 images, (b) for 3 images, (c) for 4 images, (d) for 5 images.



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

Figure 6.7 : 3D face models found in each trial

(a) for 2 images, (b) for 3 images, (c) for 4 images, (d) for 5 images.

In Figure 6.6, feature points on deformed 3D face mesh are shown. With two images, we can not deform both sides of the mouth and nose. We can deform the inner corners of the eyes only when the front image is introduced together with the front left and front right images, which is 5 images case.

In addition to the feature points, less number of images reduces our ability to refine the model. Also blending in texture is reduced while we less texture data for same point on face. The effects of change in number images on refined textured models are shown in Figure 6.7.

6.3 Models with Different Direction of Photography

Our face modeling method is independent of imaging directions. All kinds of images satisfying the constraints of having two 2D points for each feature point and six 2D feature points for each image, are enough modeling.

An exception to this rule occurs when the iteration for feature point calculation in the FaceGenerator Application is desired to be initialized with initial camera directions. In this case, since we want to employ a guess on camera directions, the direction of photograph is important. Infeasible results can be obtained when compared to the method used in this thesis that initializes the iteration using the initial 3D feature points from the initial face mesh.

Three examples of face modeling, in which images of same the face are taken from different directions, are given below. The blue points on images are the marked feature points. The first rows of the images are images used in modeling; the second rows contain the same images with deformed face meshes projected on them. These projections are performed with the projection matrices which are calculated in feature point calculation phase.

The images in Figure 6.8 are taken horizontally while images in Figure 6.9 are taken downwards with 30 degrees angle with horizontal plane, and images in Figure 6.10 are taken from horizontal direction with 30 degrees of camera rotation around direction of projection.

Figure 6.8 : Images taken with horizontal alignment

Figure 6.9 : Images taken from downwards direction

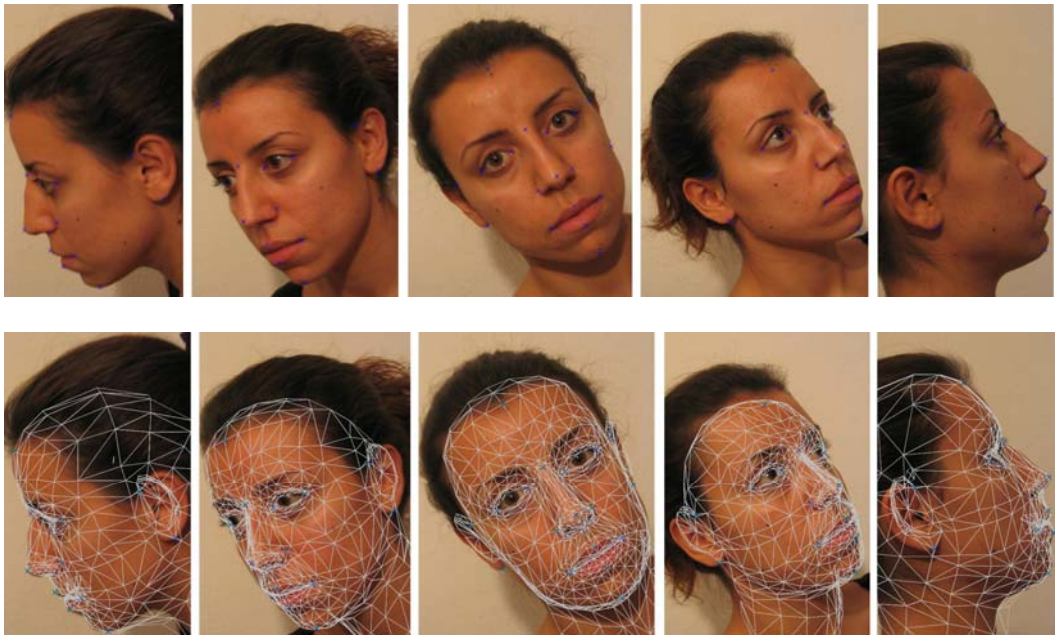


Figure 6.10 : Images taken with horizontal rotation

(a) (b) (c)

Figure 6.11 : 3D face models with different direction of photography
(a) horizontal images, (b) downwards images, (c) horizontally rotated images

Figure 6.11 shows that 3D face meshes do not change with changing direction of photography if we ignore small feature point marking errors. But same thing can not be said for the generated texture image, since the direction of photographs are used as weighting function in texture blending, and also illumination of images can change with direction.

6.4 Models with Different RBF

Radial basis functions (RBF) are used to deform the whole 3D face mesh with the guidance of feature points. FaceGenerator application involves two types of RBF's:

- RBF with constant boundary for all feature points

- RBF with variable boundary for each feature point which has constant value at neighboring feature points

Figure 6.12 : Variation of RBF for the same model

This method of face model generation usually employs the variable boundary Gaussian RBF. Figure 6.12 presents the effect of RBF boundary changes, which have constant boundaries relatively 1 at the top left image, 2.5 at the top right image, 5 at the bottom left image, and variable boundary at bottom right.

6.5 Models with Occluded Faces

One of the biggest difficulties of this thesis is to find perfect faces which do not have parts occluded with any apparatus like glasses, beard or mustache. However, a face modeling should also handle such occluded images. Human beings of recent year's society like to look different with such apparatus. While this is a way of expressing yourself, so an important feature of face, face modeling should handle these apparatus.

If the apparatus does not occlude any feature point as on the face as in Figure 6.13, 3D mesh is not affected severely but this is not true for texture. Eye glasses project to different places on the different face images and generate alias effects on the texture image. For example you can see that the same nose contact can be seen three times in texture image, each comes from front, front right and right images. In the case a beard, this alias effect causes a blurred hairy texture.

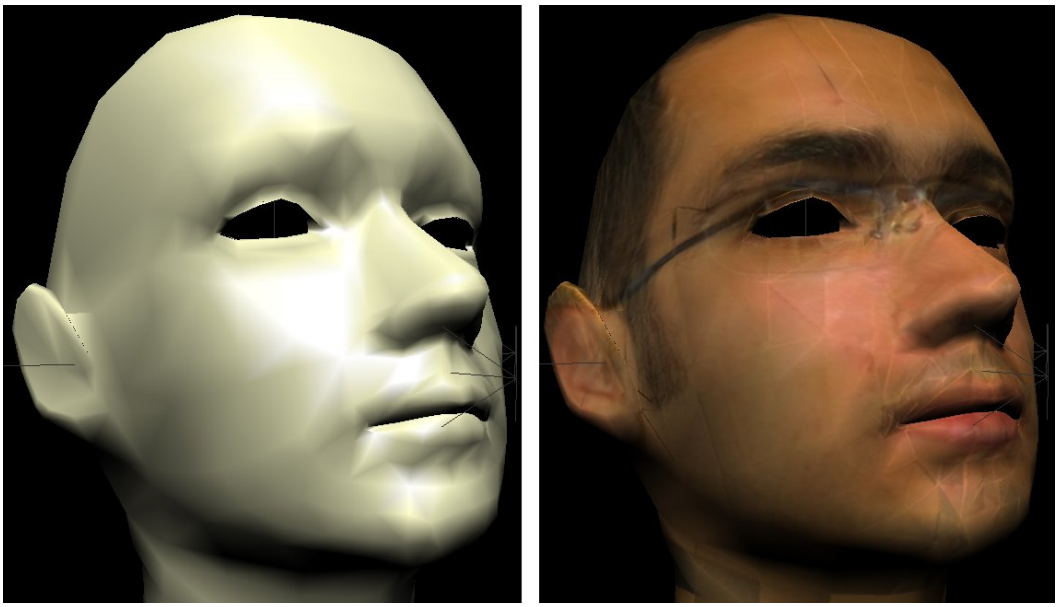


Figure 6.13 : 3D model of a man with eye glasses

CHAPTER 7

CONCLUSION

3D computer graphics is a rapidly developing area. Many complicated 3D games and 3D animation films are available in the market. Home users can play the games with their standard PC's capable of creating realistic virtual universes. Unfortunately, applications of 3D are usually trapped in games and films. Users are not able to generate their own 3D environment. This study is concentrated on a simple method for the generation of 3D models - especially human face models.

In the market, companies like Pixar Inc. generate photo-realistic face models that are indistinguishable from real ones. But their generation method is very complicated and involves extremely expensive hardware. The method presented in this thesis will make it possible for home users to generate their own face models. A conventional PC with a web cam will be sufficient.

In this thesis, face model generation by morphing an initial 3D model with uncalibrated camera photographs is performed satisfactorily with some simple problems. On initial photographs, manually marked feature points are used to deform initial 3D face model. These photographs also are processed to form a single texture image covering deformed 3D face model.

7.1 Future Works

Lots of additional work can be introduced to the proposed method of face generation. These additions will improve the performance of face model generation.

In our algorithm, there is no constraint in initial images of the subjects. Introduction of some constraints such as structured light and multiple cameras for simultaneous photography can improve calibration of cameras dramatically.

The main drawback of our method is the manual feature point marking process. The number of markings can be reduced or totally eliminated by employing a face feature extraction. Such an algorithm may place features automatically perhaps with a better precision than human users. Resulting in the development of a much more autonomous image generation process, a potential tool that carries great promise for mass application purposes.

An improvement in feature marking can also be achieved with the help of some better marking methods. Nature of the face limits the number feature points since we are limited to the corners of each face part. The feature line making concept can be introduced. Use of feature lines may be difficult in calibration but they can

easily be employed in the refinement process. Furthermore structured light sources can be used to create additional feature points on the face.

The second biggest drawback of the method explained in this thesis is the calibration step. The simple linear calibration process can be replaced with a more sophisticated calibration algorithm such as error minimization calibration. Error function can be defined in a way that calibration has a bias towards a predefined camera position.

Manual marking of feature points may also introduce some errors. This possible error has not been considered in our face modeling algorithm. In the iterative feature point calculation process, an error value can be defined for each feature point and feature point can be updated in an error boundary. Updating initial feature point can help the iteration to find the exact solution of camera calibration.

Finally, blurring operation applied to the generated texture image can be replaced by a non-linear moving average operation. Moving average may be more powerful in filtering contour lines in such texture images.

These future works have a great potential on the improvement of the face modeling. Face model generation process could become an indispensable part of our life if these improvements were achieved.

REFERENCES

[1] Alper Yilmaz, Mubarak A. Shah "Automatic Feature Detection and Pose Recovery for Faces", ACCV2002, (2002)

[2] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, Fredric Pighin, "Making Faces", in Proc. 25th annual conference on Computer graphics and interactive techniques, p.55-66, (July 1998)

[3] Cyberware Laboratory Inc. 4020/RGB 3D Scanner with Color Digitizer. Monterey, CA, (1990)

[4] Douglas M. DeCarlo, "Generation, Estimation and Tracking of Faces" University of Pennsylvania, Department of Computer and Information Science (1998)

[5] Douglas M. DeCarlo, Dimitris Metaxas and Matthew Stone. "An Anthropometric Face Model using Variational Techniques". in SIGGRAPH '98, pp. 67-74 (1998)

[6] Frederic I. Parke. “Computer Generated Animation of Faces” Master's thesis, University of Utah, Salt Lake City, UT, (1972)

[7] Frederic I. Parke. A Parameterized Model for Human Face. PhD thesis, University of Utah, Salt Lake City, UT, (1974).

[8] Frederic I. Parke, Keith Waters “Computer Facial Animation” A K Peters, Wellesley, Massachusetts, (1996).

[9] Frederic Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, David H. Salesin, “Synthesizing Realistic Facial Expressions from Photographs”, in Computer Graphics (Proc. SIGGRAPH'98), pp. 75-84, (1998)

[10] G. M. Nielson, “Scattered Data Modeling”, Computer Graphics and Applications, pp. 60-70 (January 1993)

[11] H. Chernoff “The Use of Face to Represent Points in n-dimensional Space Graphically” Technical Report NR-042-993, Office of Naval Research, Washington DC (1971)

[12] Irfan Essa, Sumit Basu, Trevor Darrell, Alex Pentland, “Modeling, Tracking and Interactive Animation of Faces and Heads using Input from Video”, in Proceedings of Computer Animation ’96 Conference, Geneva, Switzerland, (June 1996)

[13] Junyong Noh, Douglas Fidaleo, Ulrich Neumann, “Animated Deformations with Radial Basis Functions”, ACM Virtual Reality and Software Technology (VRST) pages 166-174 (2000)

[14] Lee W. S., Magenat Thalmann N, “Head Modeling from Pictures and Morphing in 3D with Image Metamorphosis based on triangulation”, in Proc. Captech’98, LNCS(Subsereires LNAI), pp. 254-267, (1998)

[15] Leung W. H., Tseng B., Shae Z.-Y., Hendriks F. and Chen T. “Realistic Video Avatar”, IEEE Intl. Conf. on Multimedia and Expo., New York, (July 2000)

[16] P. Weil. “About Face” Master's thesis, Massachusetts Institute of Technology, Architechture Group, Cambridge, MA, (1982)

[17] Ricardo Chavarriaga, Raquel Urtasun, “Camera Calibration” Graduate School in Computer Science 00/01 Ecole Polytechnique Federale de Lausanne, (2001)

[18] Richard I. Hartley, “Self-Calibration of Stationary Cameras”, in International Journal of Computer Vision, v.22 n.1, pp.5-23, (Feb./March 1997)

[19] Richard I. Hartley, Andrew Zisserman “Multiple View Geometry in Computer Vision”, ISBN 0 521 62304 9, (2000)

[20] S. M. Platt “A System for Computer Simulation of the Human Face” Master's thesis, The Moore Schook, University of Pennsylvania, Philadelphia, (1980)

[21] Tsuneya Kurihara, Kiyoshi 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*, pages 45–58. Springer-Verlag, Tokyo, (1991)

[22] Volker Blanz, Thomas Vetter, “A Morphable Model For The Synthesis Of 3D Faces” in SIGGRAPH 99 pp. 187-194, (1999)

[23] Zicheng Liu, Zhengyou Zhang, Chuck Jacobs, Michael Cohen, “Rapid Modeling of Animated Faces From Video” in Proc. The Eighth ACM International Conference on Multimedia, Marina del Rey, California, United States pp. 475-476 (2000)

[24] Zicheng Liu, Y. Shan, Zhengyou Zhang, “Model-Based Bundle Adjustment with Application to Face Modeling” in Proc. International Conference on Computer Vision (ICCV’01), Vancouver, Canada, (July 2001)