3D SYNTHETIC HUMAN FACE MODELLING TOOL

BASED ON T-SPLINE SURFACES

ALİ AYDOĞAN

DECEMBER 2007

3D SYNTHETIC HUMAN FACE MODELLING TOOL

BASED ON T-SPLINE SURFACES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ AYDOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2007

Approval of the thesis:

"3D SYNTHETIC HUMAN FACE MODELLING TOOL
BASED ON T-SPLINE SURFACES"

submitted by **ALİ AYDOĞAN** in partial fulfillment of the requirements for the degree of Master of Science in Electrical Electronics Engineering Department, Middle East Technical University by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof.Dr. İsmet ERKMEN _____
Head of Department, **Electrical Electronics Engineering**

Assist. Prof. Dr. İlkay Ulusoy _____
Supervisor, **Electrical Electronics Engineering Dept., METU**

Prof. Dr. Kemal Leblebicioğlu _____
Co-Supervisor, **Electrical Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Uğur Halıcı
Electrical Electronics Engineering Dept., METU _____

Assist. Prof. Dr. İlkay Ulusoy
Electrical Electronics Engineering Dept., METU _____

Prof. Dr. Kemal Leblebicioğlu
Electrical Electronics Engineering Dept., METU _____

Assoc. Prof. Dr. Veysi İşler
Computer Engineering Dept., METU _____

Assist. Prof. Dr. Cüneyt Bazlamaçcı
Electrical Electronics Engineering Dept., METU _____

Date: 05/12/2007

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: Ali AYDOĞAN

Signature:

ABSTRACT

3D SYNTHETIC HUMAN FACE MODELLING TOOL BASED ON
T-SPLINE SURFACES

AYDOĞAN, Ali

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Assist. Prof. Dr. İlkay ULUSOY

Co-Supervisor: Prof. Dr. Kemal LEBLEBİCİOĞLU

December 2007, 90 pages

In this thesis work, a 3D Synthetic Human Face Modelling Software is implemented using C++ and OpenGL. Bézier surfaces, B-spline surfaces, Nonuniform Rational B-spline surfaces, Hierarchical B-Spline surfaces and T-spline surfaces are evaluated as options for the surface description method. T-spline surfaces are chosen since they are found to be superior considering the requirements of the work. In the modelling process, a modular approach is followed. Firstly, high detailed facial regions (i.e. nose, eyes, mouth) are modelled, then these models are unified in a complete face model employing the merging capabilities of T-splines. Local and global features of the face model are parameterized in order to have the ability to create and edit various face models. To enhance the visual quality of the model, a region-variable rendering scheme is employed. In doing this, a new file format to define T-Spline surfaces is proposed. To reduce the computational and memory cost of the software,

a simplified version of the T-Spline surface description method is proposed and used.

Key Words: 3D Synthetic Human Face Modelling, T-spline, surface parameterization, surface merging

ÖZ

T-SPLINE YÜZEYLERİ TABANLI 3 BOYUTLU SENTETİK İNSAN YÜZÜ MODELLEME ARACI

AYDOĞAN, Ali

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. İlkay ULUSOY

Ortak Tez Yöneticisi: Prof. Dr. Kemal LEBLEBİCİOĞLU

Aralık 2007, 90 sayfa

Bu tez çalışmasında, C++ ve OpenGL kullanılarak 3 Boyutlu Sentetik İnsan Yüzü Modelleme Yazılımı gerçeklenmiştir. Yüzey tanımlama yöntemi seçenekleri olarak Bézier Yüzeyleri, B-spline Yüzeyleri, Biörnek Olmayan Rasyonel B-spline Yüzeyleri, Sıradüzensel B-spline Yüzeyleri ve T-spline Yüzeyleri değerlendirilmiştir. Çalışmanın gereksinimleri göz önünde tutularak, T-spline Yüzeylerinin daha üstün olduğu sonucuna varılmıştır. Modelleme sürecinde, modüler bir yaklaşım takip edilmiştir. İlk olarak yüksek detaylı yüz bölgeleri (burun, gözler, ağız) modellenmiş, daha sonra bu modeller T-spline yüzeylerinin birleşme özelliği kullanılarak tüm bir yüz modelinde birleştirilmiştir. Yüz modelinin yerel ve genel özellikleri, çeşitli yüz modellerini oluşturma ve düzenleme yeteneğine sahip olmak için parametrelendirilmiştir. Modelin görüntü kalitesini arttırmak için, bölge-değişken bir çizim tasarısı kullanılmıştır. T-spline yüzeylerini tanımlayan yeni bir

dosya yapısı önerilmiştir. Yazılımın hesap ve bellek maliyetini düşürmek için, T-spline yüzey tanımlama yönteminin basitleştirilmiş bir uyarlaması önerilmiş ve kullanılmıştır.

Anahtar Kelimeler: 3 Boyutlu Sentetik İnsan Yüzü Modellemesi, T-spline, yüzey parametrizasyonu, yüzey birleştirilmesi

ACKNOWLEDGEMENTS

I would like to thank Assist. Prof. Dr. İlkay ULUSOY, for her valuable guidance, supervision and tolerance throughout this thesis work.

I would also like to thank Prof. Dr. Kemal LEBLEBİCİOĞLU for his support, guidance and co-supervision.

Special thanks to Erdem Akagündüz for his suggestions and valuable ideas.

Special thanks to my colleagues in ASELSAN Inc. for their continuous support towards the realization of this thesis work.

Finally, I would like to express my greatest thanks to my family and my beloved Yeşim for their endless support during this thesis work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

3D : 3 Dimensional

CPU : Central Processing Unit

CAD : Computer Aided Design

GPU : Graphics Processing Unit

GUI : Graphical User Interface

MFC : Microsoft Foundation Class

NURBS : Non Uniform Rational B-splines

OpenGL : Open Graphics Library

OOP : Object Oriented Programming

CHAPTER 1

INTRODUCTION

1.1 Motivation

Face is the region of human body, features of which can hugely vary from one person to another, making an individual unlike the other. This distinguishing property of the face helps us to recognize an individual by just looking at his/her face. As the technology evolved and computers equipped with powerful GPUs became available to many users, representing the 3D facial information of a person in computer environment became an area of interest.

3D Synthetic Human Face Modelling has been a research area for more than 30 years. With the progress of computer graphics technology, it has been seen that several application areas for 3D Face Modelling exists. Some application areas for 3D Synthetic Human Face Modelling are:

- ***Computer games industry***

  In most of the recent computer games, the characters are generally created with facial details. Some computer games include characters from real life, such as football players and high quality 3D face models are used to enhance the realism of the game. The emphasis given to human face modelling in computer games industry can be observed in the popular soccer game *FIFA 08* of Electronic Arts.

- ***Films industry***

  The number of animation movies is continually increasing thanks to developments in computer graphics and the appreciation of the audience. High quality and realistic 3D human face models are used in some movies such as *The Final Fantasy: The Spirits Within (2001)* and caricatured characters are used in some movies such as *Toy Story (2001).*

- ***Teleconferencing***

  In teleconferencing applications, the aim is to transmit the facial image of people to the other side together with the voice data. Since transmitting the complete video data requires a high bandwidth link, another approach which employs parameterized facial models is being researched. With this technique, each frame is processed to detect the predefined parameters of the face and only these parameters are transmitted to the other side. The receiver side processes the received data to generate a corresponding facial image. Detailed information about the applications of face modelling in teleconferencing can be found in [11].

- ***Social agents and avatars***

  There are several researches for creating computer generated characters which can interact with people. These characters are planned to be used instead of traditional GUIs and in computer-aided learning especially for children. More information about social agents can be found in [12].

- ***Craniofacial surgical operations planning***

  Surgical operations can be planned and simulated on 3D face models and the desired situation of the face after operations can be visualized before the operation is actually performed. For an advanced study on facial surgery simulation, reader can refer to [15].

- ***Production of 3D Human Face recognition test/training data***

  A large set of random face models can be useful in the verification tests of 3D human face recognition software, as well as in the training phase of them. Reader can refer to [19] for further information.

Our motivation in this thesis work is, to design and implement 3D Synthetic Human Face Modelling Software. With the implementation of this software, capability of creating a wide range of realistic 3D synthetic human face models is aimed. Details about the requirements of the software are discussed in Chapter 3.

The human face has a complex structure which makes modelling process challenging. Several techniques have been researched and employed in this area. The most popular 3D human face modelling approaches are [7]:

- Anatomy based modelling
- Point based modelling
- Surface based modelling

In anatomy based modelling, face is modelled using the detailed anatomic information as the name recalls. Facial bones, muscles, skin layer and their connection information are modelled in detail. This is the most realistic and visually superior modelling technique, but it produces very complex outputs and long processing times are needed to render these models [7].

In point based modelling approach, only skin, which is the outermost layer of the face, is included in the model. Group of points on the model surface may be controlled by user defined functions which modify the coordinates of the points in the model. Since many points are needed to create a visually satisfactory model, the creation and management of this type of model becomes very difficult. A rectangular array of polygon vertices or arbitrary polygonal networks can be used in the definition of model surface. [3]

In surface based modelling; only skin layer is considered parallel to point based modelling. But this technique differentiates than the previous one in the way model surface is defined. The 3D human face model is formed as a parametric surface, where model surface is defined by a limited number of control points. These control points are then managed to modify the face surface. The association and management of control points with the final model surface is called *parameterization*. Since the number of control points are comparably less than the number of surface points in a visually equal quality point based model, creation and management of this kind of model is much easier. [7]

In our work, surface based modelling is preferred as the modelling technique since it provides a balance between visual quality and ease of management of the surface. A parametric surface description method is needed to be employed in surface based modelling. Bezier Surfaces, B-spline Surfaces, Nonuniform Rational B- Spline Surfaces, Hierarchical B-Spline Surfaces and T-spline Surfaces are the parametric surfaces which are researched and compared with each other in various aspects.

A Bézier Surface is easy to implement but its degree is fixed by the number of control points making it inflexible topologically and its local control capability is poor. A B-spline Surface is difficult to implement but it is a more powerful and more flexible method compared to a Bézier Surface. Degree of a B-spline Surface can be varied by the selection of knot vectors with certain limitations making it flexible. A stronger local control on the model surface can be obtained by B-spline Surfaces. Nonuniform Rational B- Spline Surfaces are a step forward after B-spline Surfaces. With the introduction of weight factors, even more local control on the model surface is achieved. Hierarchical B-Spline Surfaces offer very fine local control but their tight topological constraints make implementations difficult. A T-Spline Surface inherits all advantages of a Nonuniform Rational B- Spline Surface since T-spline Surfaces are a generalization of NURBS Surfaces. With the introduction of *T-junctions,* necessity of a rectangular grid of control points in B-spline Surfaces is eliminated and a more flexible topology is achieved. The number

of control points in a model can be reduced if a T-spline Surface is preferred instead of a B-spline Surface. This reduction makes the modification of a model easier, since modeller has to deal with less control points to modify the model surface. A more detailed explanation of these parametric surface description methods can be found in Chapter 2. As a result of our comparison, T-spline Surface is concluded to be the most suitable parametric surface description scheme for our purposes and is used in this work.

1.2 Outline of the Thesis

Introduction for this thesis is given in Chapter 1, in which the general information on 3D Synthetic Human Face Modelling and its application areas, our motivation in this work, previous approaches to the topic and the methods used in this work are conducted.

In Chapter 2, background information on 3D object representation and human face modelling is given. Emphasis is laid on parametric surface description methods, namely Bézier Surfaces, B-spline Surfaces, Nonuniform Rational B-spline Surfaces, Hierarchical B-spline Surfaces and T-spline Surfaces.

In Chapter 3, design and implementation of our 3D Synthetic Human Face Modelling software is explained. Firstly, reasoning for the selection of T-spline method for the implementation is presented in a comparative approach. Secondly, implementation of T-spline surfaces, which is a crucial step in the software, is discussed. A new file format to represent T-spline surfaces is proposed and explained. Chapter continues with the discussion of surface fitting with B-spline surfaces, which was employed in the creation of raw generic nose, eye, mouth and face models. Then production of generic nose, eye, mouth and face models by simplification and parameterization is explained. Sample results obtained with generic models are presented for corresponding model. As the concluding step, unification of partial models and obtaining a complete human face model is discussed with sample results.

Finally in Chapter 4, conclusions drawn from this work are discussed and future work is stated.

CHAPTER 2

BACKGROUND

In the first part of this chapter, general information on 3D object modelling is given. Emphasis is especially laid on parametric surface description methods, namely Bézier Surfaces, B-spline Surfaces, Nonuniform Rational B-spline Surfaces, Hierarchical B-spline Surfaces and T-spline Surfaces.

In the second part of the chapter, background information about 3D synthetic human face modelling is conducted. Firstly, anatomy of the human face is introduced as a summary. Then human face modelling techniques are discussed.

2.1 3D Object Representation Techniques

Many different 3D objects with typical geometric properties may be placed in a computer generated scene. Some of these objects may be as simple as a cube in terms of geometrical structure, whereas some others may be as complex as a detailed tree coated with thousands of leaves. 3D objects must be geometrically defined in the relevant computer graphics program which creates the 3D scene, in order to be rendered by the GPU and to be included in the scene. In this thesis work, we refer to *3D models* as the geometrical description of objects in computer environment. Since objects have different typical geometric properties, several application specific modelling methods can be used in the modelling process.

Representation methods for solid objects are divided into two categories which are namely Space-partitioning Representations and Boundary Representations [1]. These

two methods are also known as Volume Representations and Surface Representations [3]

2.2 Space-partitioning Representations (Volume Representations)

This approach defines the 3D object as a volume element which is formed by smaller volume elements. Object to be modelled can be partitioned into a set of small, non-overlapping, contiguous solids, such that these smaller partitions add up to whole object by the union operation. Other than union operation, intersection and difference operations are also allowed in the modelling of new objects. This method should be preferred if interior properties of the object is a matter of concern to the modeller. Constructive Solid Geometry (CSG) and octrees are two examples of Volume Representation techniques. An example application of CSG can be seen in **Figure 1**. In this example, final 3D object model is obtained by the difference of two models, which are previously defined by the intersection and union of the other objects respectively.

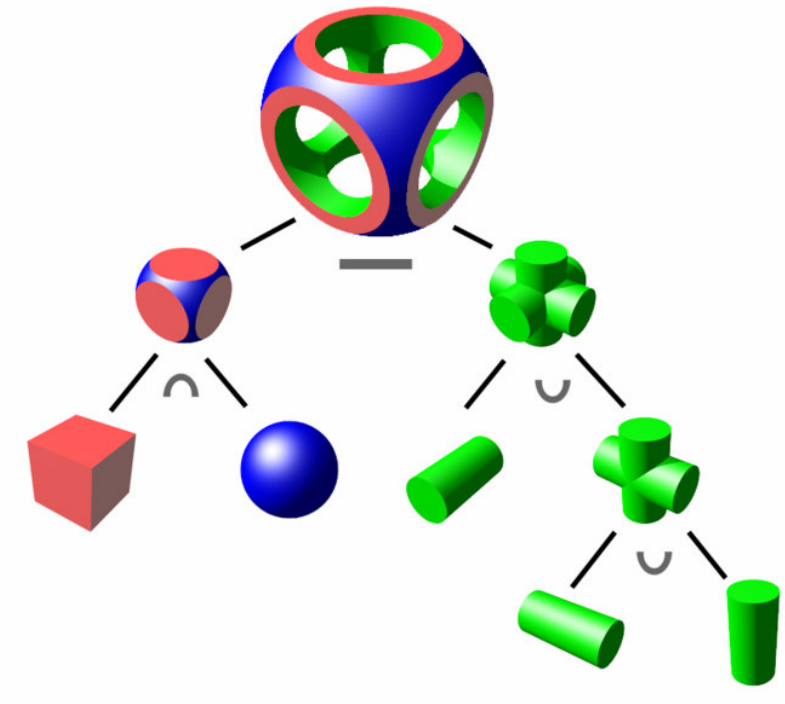

Figure 1 –Definition of an object with CSG [13]

Hierarchical tree structures, which are organized so that each node corresponds to a region of three dimensional space, are called octrees [1]. A region in three dimensional space is divided into octants, such that each subdivided region in the final representation is homogeneous in terms of physical characteristics such as color and material type. These smallest volume elements are called *voxels*, in analogy with *pixels* in two dimensional representations.

2.3 Boundary Representations (Surface Representations)

Appearance of an object is what our eyes perceive and it is usually formed by the outermost layer of the object. Using this idea, boundary representation approach defines the 3D object as a volume contained in a set of faces together with topological information which defines the relationships between the faces [18]. Main items of this topological information are faces, edges and vertices. Boundary representations are widely used in computer graphics, especially in CAD software drawings for automotive and aerospace applications. A teapot modelled using boundary representations is given in **Figure 2**. The teapot in the figure is defined as a set of surface vertices and the wireframe model on the left is obtained. Then model is rendered by defining the surface properties of the object.

Implicit surfaces, polygonal surfaces and parametric surfaces are three most common surface description techniques [3].

Figure 2- Boundary representation of a teapot [14]

2.3.1 Implicit Surfaces

An implicit surface is defined as a function *F(x,y,z)*, which maps each point in *x, y, z* space to a scalar value. A sphere of unit radius centred at *(1.0, 2.0, 3.0)* can be defined with the implicit function:

$$f(x, y, z) = (x - 1.0)^2 + (y - 2.0)^2 + (z - 3.0)^2 - 1$$

With current techniques, implicit surfaces take more time to interactively manipulate and display than parametric or polygonal surfaces [3]. Another difficulty arises when a complex object is to be modelled. Expressing the whole details of an object with implicit functions can prove to be very demanding and time consuming.

2.3.2 Polygonal Surfaces

Polygonal surfaces technique uses polygons to describe the boundary surface of the object. This technique has an advantage in terms of rendering speed, because most graphics systems store objects as a sum of polygons. Graphics packages generally have routines to display polygon surfaces effectively and this makes the implementation of polygon surfaces relatively straightforward. To define an object with polygonal surfaces, all polygons lying on the boundary surface of the object should be defined together with the corresponding vertex points which form the polygons. This requirement increases the storage cost of this scheme and also causes the modifications on the modelled objects to become more complex, since a modeller has to deal with surface points one by one.

Implicit surfaces and parametric surfaces are also converted to polygonal surfaces before being displayed on the display unit.

2.3.3 Parametric Surfaces

Parametric surfaces are defined by three bi-variant parametric functions where each function is used to calculate one of the *x, y, z* coordinates of a point lying on the boundary surface of the object.

Parametric surfaces can be expressed in the following form:

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

$S(u,v)$ is the parametric surface, $P_{i,j}$ are the control points, $B_{i,k}(u)$ and $B_{j,m}(v)$ are basis functions of polynomial order *k* and *m* respectively, in the parametric dimensions *u* and *v* respectively.

Bézier Surfaces, B-spline Surfaces, NURBS Surfaces, Hierarchical B-Spline Surfaces and T-spline Surfaces are examples of parametric surface description schemes. These schemes are explored in detail below. To give a complete insight into parametric surfaces, explanations of corresponding parametric curves are also included.

2.3.3.1 Bézier Surfaces

This method was originally developed by Pierre Bézier to be used in automobile body design. Most graphics packages and drawing packages support this method since it is simple to implement and widely used.

Parametric Bézier curve is defined as

$$P(t) = \sum_{i=0}^{n} B_i\ J_{n,i}(t) \qquad 0 \le t \le 1$$

where B_i 's are the control points (a total of *n+1* control points) and the polygon formed by the set of control points is called *defining polygon*, where $J_{n,i}(t)$ is the *i*th *n*th-order Bernstein basis function defined as:

$$J_{n,i}(t) = \binom{n}{i} t^i\ (1-t)^{n-i}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

The properties of Bézier curves are as follows [2]

- The basis functions are real and nonzero for all parameter values. So a change in the position of a control point is reflected throughout the entire curve and this makes applying local changes to the shape of the curve difficult.
- The degree of the polynomial defining the curve segment is one less than the number of defining polygon points, which means that number of defining polygon points fixes the degree of the resulting polynomial.
- The curve generally follows the shape of the defining polygon.
- The first and last points on the curve are coincident with the first and last points of the defining polygon.
- The tangent vectors at the ends of the curve have the same direction as the first and last polygon spans, respectively.
- The curve is contained within the *convex hull* of the defining polygon (Any curve position is the weighted sum of control points).
- The curve exhibits *variation diminishing property*. Basically this means that the curve does not oscillate about any straight line more often than the defining polygon.
- The curve is invariant under an affine transformation.

Extending the definition of Bézier curves to surfaces, Bézier surfaces are defined as

$$P(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,j}\ J_{n,i}(u)\, K_{m,j}(v)$$

$B_{i,j}$'s are the control points (a total of *(n+1)*(m+1)* control points),

$J_{n,i}(u)$ is the Bernstein basis function in *u* parametric direction and $K_{m,j}(v)$ is the Bernstein basis function in *v* parametric direction and defined as follows:

$$J_{n,i}(u) = \binom{n}{i} u^i\ (1-u)^{n-i}$$

$$K_{m,j}(v) = \binom{m}{j} v^i\ (1-v)^{m-j}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

$$\binom{m}{j} = \frac{m!}{j!(m-j)!}$$

The properties of Bézier surfaces are [2]

- The degree of the surface in each parametric direction is one less than the number of defining polygon vertices in that direction.
- The continuity of the surface in each parametric direction is two less than the number of defining polygon vertices in that direction.
- The surface generally follows the shape of the defining polygon net.
- Only the corner points of the defining polygon net and the surface are coincident.
- The surface is contained within the *convex hull* of the defining polygon net.
- The curve does not exhibit *variation diminishing property*. The variation diminishing property for bi-variant surfaces is both undefined and unknown.
- The surface is invariant under an affine transformation.

2.3.3.2 B-spline Surfaces

This scheme is the most popular curve definition technique, being widely used in most CAD systems and graphics packages.

A B-spline curve is defined as:

$$P(t) = \sum_{i=0}^{n} B_i \; N_{i,k}(t) \qquad t_{\min} \le t \le t_{\max},\; 2 \le k \le n+1$$

B_i 's are the control points (a total of $n+1$control points) and k is the order.

$N_{i,k}$'s are the B-spline basis functions defined by Cox-deBoor recursion formulas as:

$$N_{i,1}(t) = \begin{cases} 1 \;\; if \;\; x_i \le t < x_{i+1} \\ 0 \;\; otherwise \end{cases} \qquad \text{and}$$

$$N_{i,k}(t) = \frac{(t - x_i)\, N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)\, N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

$\frac{0}{0} = 0$ Convention is used in the calculation of basis functions. x_i's are the elements of a monotonically increasing series of real numbers, namely *knot vector,* satisfying $x_i \leq x_{i+1}$. Three types of knot vectors can be used. These are *uniform, open uniform* and *nonuniform* knot vectors.

A uniform knot vector has evenly spaced knot values. Two uniform knot vectors can be given as:

[0 2 4 6 8] and [-0.5 -0.25 0 0.25 0.5]

Although examples above are valid, uniform knot vectors are generally written beginning from zero and with increments of 1 or normalized into the [0 1] range. Then the example knot vectors are written as:

[0 1 2 3 4] and [0 0.25 0.5 0.75 1]

For a given order *k*, uniform knot vectors create periodic uniform basis functions as:

$$N_{i,k}(t) = N_{i-1,k}(t-1) = N_{i+1,k}(t+1) \quad [2]$$

In an open uniform knot vector, first and last knot values are repeated *k* times, as much as the order of the curve. Two examples can be given as:

k=2 [0 0 1 2 3 3]

k=4 [0 0 0 0 1 2 2 2 2]

In a nonuniform knot vector, knot values are possibly unequally spaced. Two examples can be given as:

[0 1 2 2 3 4] and [0 2 5 7 8]

Nonuniform knot values increase the control ability of the modeller on the resulting curve since it allows knot multiplicity and unequally spaced knot values.

The properties of B-spline curves are [2]

- Each basis function is positive or zero for all parameter values
- The sum of the B-spline basis functions for any parameter value *t* can be shown to be

$$\sum_{i=0}^{n} N_{i,k}(t) = 1$$

- Except for *k=1* each basis function has precisely one maximum value.
- The maximum order of the curve is equal to the number of defining polygon vertices, i.e. n+1.
- The continuity of the curve is C^{k-2}, two less than the order *k*.
- The curve exhibits the *variation diminishing property*. Thus the curve does not oscillate about any straight line more often than its defining polygon.
- The curve generally follows the shape of the defining polygon.
- Any affine transformation can be applied to the curve by applying it to the defining polygon vertices.
- The curve lies within the *convex hull* of its defining polygon, stronger than that of Bézier curves. A point on the B-spline curve lies within the convex hull of *k* neighbouring control points.

Extending the definition of B-spline curves, B-spline surfaces are defined as:

$$P(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}\ N_{i,k}(u) M_{j,l}(v)$$

B_i 's are the control points (a total of *(n+1)*(m+1)* control points),

$N_{i,k}$ and $M_{j,l}$ are the B-spline basis functions defined by Cox-deBoor recursion formulas as:

$$N_{i,1}(u) = \begin{cases} 1 \ \ if \ \ x_i \le u < x_{i+1} \\ 0 \ \ otherwise \end{cases} \qquad \text{and}$$

$$N_{i,k}(u) = \frac{(u - x_i)\, N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u)\, N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}}$$

$$M_{j,1}(v) = \begin{cases} 1 \ \textit{if} \ y_i \le v < y_{i+1} \\ 0 \ \textit{otherwise} \end{cases} \qquad \text{and}$$

$$M_{j,l}(v) = \frac{(v - y_j)\, M_{j,l-1}(v)}{y_{j+l-1} - y_j} + \frac{(y_{j+l} - v)\, M_{j+1,l-1}(v)}{y_{j+l} - y_{j+1}}$$

x_i and y_j are the elements of knot vectors satisfying $x_i \le x_{i+1}$ and $y_j \le y_{j+1}$.

The properties of B-spline surfaces are [2]

- The maximum order of the surface in each parametric direction is equal to the number of defining polygon vertices in that direction, i.e *n+1* and *m+1* respectively.
- The continuity of the surface in each parametric direction is C^{k-2} and C^{l-2}, two less than *k* and *l*, the order in each parametric direction.
- The surface is invariant with respect to an affine transformation. Any affine transformation can be applied to the surface by applying it to the defining polygon vertices.
- The *variation diminishing property* for B-spline surfaces is currently not known.
- The influence of a single polygon net vertex is limited to *±k/2*, *±l/2* spans in each parametric direction.
- The surface lies within the *convex hull* of the defining polygon net formed by the union of all convex hulls of *k, l* neighbouring control points.

2.3.3.3 Nonuniform Rational B-spline Surfaces (NURBS Surfaces)

The definition of a rational B-spline curve is:

$$P(t) = \frac{\sum_{i=0}^{n} B_i \ w_i \ N_{i,k}(t)}{\sum_{i=0}^{n} w_i \ N_{i,k}(t)}$$

B_i 's are the control points (a total of *n+1* control points), w_i 's are the weight factors and *k* is the order. $N_{i,k}$ are the nonrational B-spline basis functions defined by Cox-deBoor recursion formulas given in the previous section.

Because of their definition, rational B-splines can be considered as the projection of a 4D nonrational B-spline curve into 3D space. Definition can be extended in this format as:

$$P(t) = \sum_{i=0}^{n} B_i^{\,h}\ N_{i,k}(t)$$

B_i^h are the 4D control points for 4D nonrational B-spline curve.

The definition of a rational B-spline curve can be rewritten as:

$$P(t) = \sum_{i=0}^{n} B_i\ R_{i,k}(t)$$

B_i 's are 3D control points and $R_{i.k}(t)$ are the rational B-spline basis functions defined as:

$$R_{i,k}(t) = \frac{w_i\ N_{i,k}(t)}{\sum_{i=0}^{n} w_i\ N_{i,k}(t)}$$

The properties of rational B-spline curves are [2]

- Each rational basis function is positive or zero for all parameter values.
- The sum of the rational B-spline basis functions for any parameter value *t* can be shown to be:

$$\sum_{i=0}^{n} R_{i,k}(t) = 1$$

- Except for *k=1,* each rational basis function has precisely one maximum value.
- The continuity of the curve is C^{k-2}, two less than the order *k.*

- The maximum order of the rational B-spline curve is equal to the number of defining polygon vertices, i.e. $n+1$.
- The curve exhibits the variation diminishing property.
- The curve generally follows the shape of the defining polygon.
- The curve lies within the union of convex hulls formed by k successive defining polygon vertices.
- Any projective transformation can be applied to the curve by applying it to the defining polygon vertices, which is a stronger condition than that for a nonrational B-spline curve.
- Any affine transformation can be applied to the curve by applying it to the defining polygon vertices.

When we consider the case that all $w_i = 1$, then $R_{i,k}(t) = N_{i,k}(t)$ and a rational B-spline curve reduces to a nonrational B-spline curve. So we can say that nonuniform rational B-spline curves are special cases of ordinary B-spline curves.

Weight factors used in rational B-spline curves provide a better control on the shape of the curve. By adjusting the w_i values, the effect of the control points on the curve can be altered. As w_i is increased, the curve is pulled stronger to the corresponding control point B_i.

Rational B-spline curves have the ability to represent quadratic curves like circles and ellipses, whereas their nonrational counterpart can approximate these curves.

Similarly to nonrational B-spline curves, *uniform*, *open uniform* and *nonuniform knot vectors* can be used in the definition of rational B-spline curves. Since nonrational B-splines are a special case of rational B-splines and uniform knot vectors are a special case of nonuniform knot vectors, graphics packages generally represent all B-spline curves with nonuniform knot vectors, in rational B-spline form, which has become a standard in 1983 with the name NURBS (Nonuniform Rational B-splines).

Extending the definition of rational B-spline curves to surfaces, rational B-spline surfaces are defined as:

$$P(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\ B_{i,j}\ N_{i,k}(u)M_{j,l}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\ N_{i,k}(u)M_{j,l}(v)}$$

$B_{i,j}$'s are the control points, $w_{i,j}$ are the weight factors, k and l are the order in each parametric direction. $N_{i,k}$ and $M_{j,l}$ are the nonrational B-spline basis functions defined by Cox-deBoor recursion formulas.

Definition can be done in the 4D homogenous coordinate system as

$$P(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B^{h}{}_{i,j}\ N_{i,k}(u)M_{j,l}(v)$$

$B^{h}{}_{i,j}$'s are the 4D control points for 4D nonrational B-spline curve.

The definition of the rational B-spline surfaces can be rewritten as

$$P(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}\ S_{i,j}(u,v)$$

$B_{i,j}$ are 3D control points and $S_{i,j}(u,v)$ are the rational B-spline surface basis functions defined as:

$$S_{i,j}(u,v)=\frac{w_{i,j}\ N_{i,k}(u)M_{j,l}(v)}{\sum_{p=0}^{n}\sum_{r=0}^{m} w_{p,r}\ N_{p,k}(u)M_{r,l}(v)}$$

The properties of rational B-spline surfaces are [2]

- The sum of the rational B-spline surface basis functions for any parameter value u,v can be shown to be

$$\sum_{i=0}^{n}\sum_{j=0}^{m} S_{i,j}(u,v)=1$$

- Each rational B-spline surface basis function is positive or zero for all parameter values.
- Except for $k=1$ or $l=1$, each rational B-spline surface basis function has precisely one maximum value.
- The maximum order of the rational B-spline surface in each parametric direction is equal to the number of defining polygon vertices in that direction.
- The continuity of the curve is C^{k-2}, C^{l-2} continuous, two less than the orders k and l.
- Rational B-spline surfaces are invariant under projective transformation.
- The surface lies within the union of convex hulls formed by k, l neighbouring defining polygon vertices.
- Variation diminishing property for rational B-spline surfaces is not known.

2.3.3.4 Hierarchical B-spline Surfaces

Hierarchical B-splines are proposed as an extension to B-splines, to provide more local control on B-spline curves and especially surfaces. This scheme was firstly introduced by Forsey and Bartels in 1988 [6].

In 3D graphical modelling applications, modellers face difficulties in high detailed regions of the object to be modelled. When we speak for B-splines, modeller has to insert additional control points if he wants to have more precise control on the B-spline model surface. But just to meet the topological constraints of B-splines (i.e. necessity of a rectangular grid of control points), one has to add several more control points (i.e. extend the additional control points along a row or column of the defining polygon net) than the necessary ones for local control.

As a solution to local refinement problem of B-splines, hierarchical B-splines were proposed as follows:

$$P(u,v) = \sum_i \sum_j B_{i,j} N_{i,k}(u) M_{j,l}(v)$$

In an ordinary B-spline surface definition, basis functions $N_{i,k}$ and $M_{j,l}$ can be rewritten as linear combinations of smaller basis functions as:

$$N_{i,k}(u) = \sum_{r} \alpha_{i,k}(r) R_{r,k}(u)$$

$$M_{j,l}(v) = \sum_{s} \alpha_{j,l}(s) R_{s,l}(v)$$

After this step, the definition of the surface can be written as:

$$P(u,v) = \sum_{r} \sum_{s} W_{r,s} R_{r,k}(u) R_{s,l}(v)$$

$$W_{r,s} = \sum_{i} \sum_{j} \alpha_{i,k}(r) \alpha_{j,l}(s) B_{i,j}$$

These additional definitions tell us that, new control points can be added locally and these control points affect a smaller parametric range due to the redefinition of basis functions. This is how local refinement is achieved. The region with denser control points is called an *overlay.*

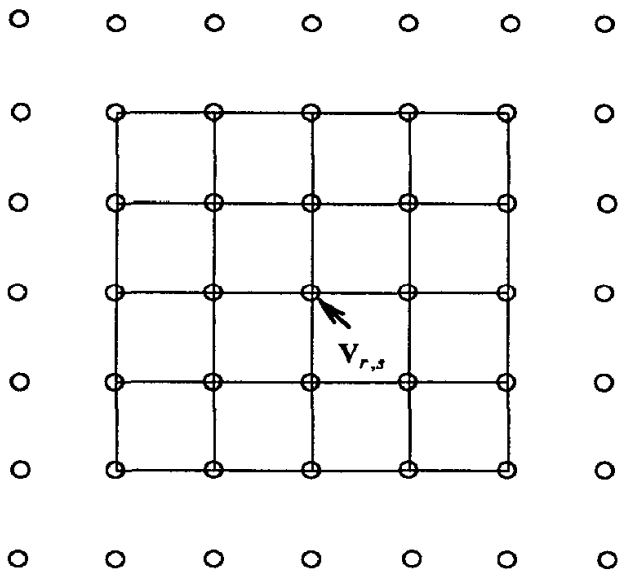

Figure 3- 7x7 B-spline control grid [6]

A 7x7 grid of control points is given in **Figure 3**. The movement of the central point $V_{r,s}$ effects the inner 16 square region. If we decide that a local refinement should be carried in this region, we should insert new control points to gain more local control. This is shown in **Figure 4**.

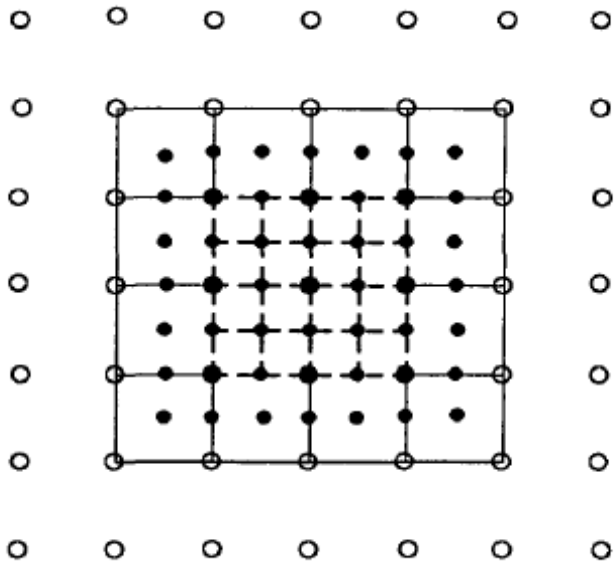

Figure 4- Hierarchical B-spline grid [6]

New control points are added in the middle way between the existing control points, in terms of 3D spatial coordinates and *u,v* parametric range. In this new configuration, the movement of the central point only effects the 16 dashed lined squares. We see that this process decreases the area to be effected by the movement of the central point to a fraction of four.

Another ability hierarchical B-splines offer is *offset referencing*. The 3D spatial coordinates of control vertices in an overlay are stored by taking some control points out of the overlay region as reference. With this technique, even if the lower detailed region has some changes, the high detailed region will move accordingly because its control points are defined in terms of the low detailed region. This supplies a dynamical relationship between the overlay and the main surface.

2.3.3.5 T-splines

T-splines have been proposed as the generalization of NURBS. It has been developed to overcome the difficulties experienced with the use of NURBS in local refinement and merging of surfaces.

T-splines are a special case of PB-splines. The name PB-spline stands for *point based spline* since PB-splines are point based instead of grid based. The control points defining the PB-spline do not have a topological relationship. PB-spline is defined as:

$$P(s,t) = \frac{\sum_{i=1}^{n} P_i \, B_i(s,t)}{\sum_{i=1}^{n} B_i(s,t)}$$

P_i's are control points and $B_i(s,t)$ are basis functions defined as:

$$B_i(s,t) = N_i^{\,3}(s)\, N_i^{\,3}(t)$$

for cubic case, where $N_i^3(s)$ and $N_i^3(t)$ are basis functions associated with the knot vectors

$$\bar{s}_i = [\, s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$$

$$\bar{t}_i = [\, t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$$

for $(s,t)=(s_{i2}, t_{i2})$ in **Figure 5**. These knot vectors are formed using the control points of the defining polygon network.

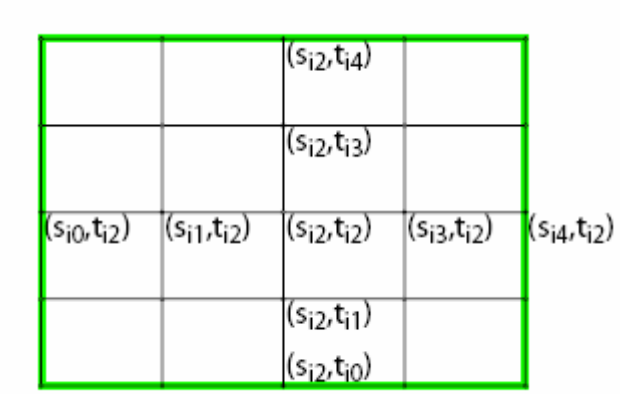

Figure 5- Control Points in a PB-Spline [4]

A T-spline is a special form of PB-spline defined by a mesh structure called *T-mesh*. The name T-spline is given to the topology since T-meshes support a new type of control point named *T-junction.* A surface defined by a T-mesh without any T-junctions reduces to a NURBS surface.

A sample T-mesh is shown in **Figure 6**. If a vertex is on the boundary edge of the T-mesh, it is called a *boundary vertex*. If a vertex is not a *boundary vertex,* it is an *interior vertex*. F is a boundary vertex, whereas A,B,C,D and E are interior vertices.

Points A, B and C are T-junctions (also called *T-vertices*). The name T-junction is given to these vertices since their appearance in the T-mesh resembles the letter T. The vertices which are connected to four edges are called *crossing vertices*. D and E are examples for crossing vertices.

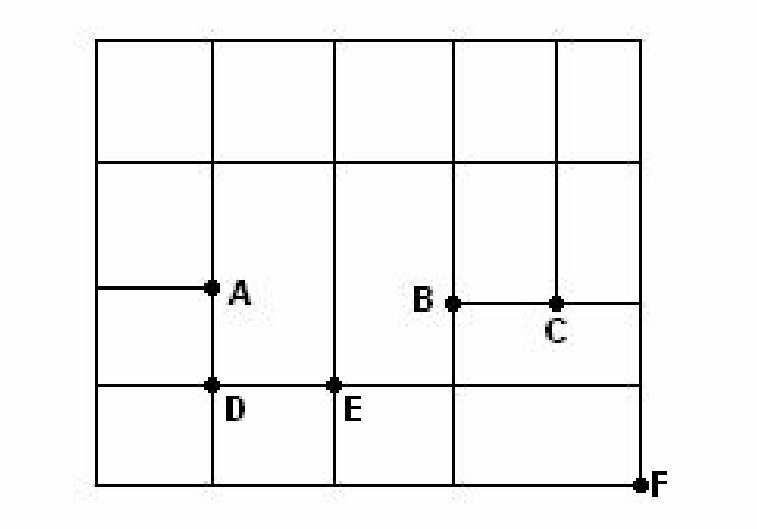

Figure 6- Sample T-mesh

T-spline surfaces can be formulated as

$$P^4(s,t) = (x(s,t), y(s,t), z(s,t), w(s,t)) = \sum_{i=1}^{n} P_i^4 \; B_i(s,t)$$

P_i^4's are control points in 4D coordinate system.

When we rewrite the expression for Cartesian coordinates:

$$P^3(s,t) = (x(s,t), y(s,t), z(s,t)) = \frac{\sum_{i=1}^{n} P_i^3 \; w_i \; B_i(s,t)}{\sum_{i=1}^{n} w_i \; B_i(s,t)}$$

P_i^3 's are control points in Cartesian coordinate system. $B_i(s,t)$ is the blending function defined by:

$$B_i(s,t) = N[\bar{s}_i](s)\, N[\bar{t}_i](t)$$

$\bar{s}_i$ and $\bar{t}_i$ are knot vectors deduced from the T-mesh in the form:

$$\bar{s}_i = [\; s_{i0}, s_{i1}, .., .., s_{i(k-2)}, s_{i(k-1)} \;] \text{ and } \bar{t}_i = [\; t_{i0}, t_{i1}, .., .., t_{i(k-2)}, t_{i(k-1)} \;]$$

For rational and nonrational B-splines, knot vectors are fixed and same throughout the whole rectangular grid of control points. In the case of T-splines, knot vectors may vary for every control point and knot vector information is obtained from the T-mesh. Each edge in the T-mesh corresponds to a *knot interval*. A *knot interval* is a nonnegative number which represents the difference between two knot values. To clarify the concepts in T-splines, a sample T-mesh is given in **Figure 7**.

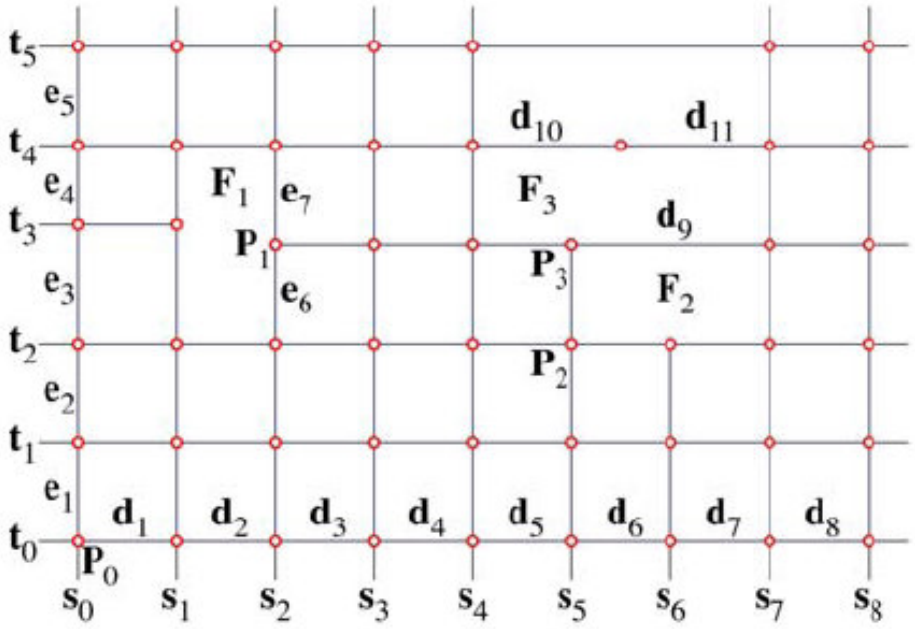

Figure 7- T-mesh with knot coordinates [5]

The T-mesh in **Figure 7** is in *s,t* parameter space. The Cartesian coordinates of these points are not expressed at this point to keep clarity. d_1, d_2, d_3, d_4, d_5, d_6, d_7 and d_8 are knot intervals in *s* direction and e_1, e_2, e_3, e_4 *and* e_5 are knot intervals in *t* direction.

As well as Cartesian coordinates, every control point has its parameter coordinates in the form of *(s,t)* and these coordinates are called *knot coordinates*. For example knot coordinates of P_2 is *(s5,t2)* if we decide that P_0 is the knot origin with knot coordinates *(0,0).* These knot coordinates are used to deduce knot vectors from the T-mesh, which is crucial in the calculation of surface points.

Face is the smallest closed polygon surrounding a point in knot coordinates. The sum of knot intervals on one side of a face must be equal to the sum of knot intervals on the opposite side of the face. Namely for face F1 in **Figure 7**, *e3+e4=e6+e7* should be satisfied.

If a T-junction on one edge of a face can be connected to another T-junction on the opposite side of the face, satisfying the previous rule about knot intervals sum, those T-junctions must be connected.

The knot vectors are deduced from the T-mesh, as previously expressed. This procedure will be explained for a cubic case, for which knot vectors will be in the form:

$$\bar{s}_i = [\ s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$$

$$\bar{t}_i = [\ t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$$

For a control point P_i, s_{i2} and t_{i2} are directly copied from the knot coordinates of P_i. To find s_{i3} and s_{i4}, a ray parameter space is written as

$$R(\alpha) = (s_{i2} + \alpha, t_{i2})$$

Then s_{i3} and s_{i4} are found to be the s coordinates of the first two *s-edges* (vertical line segment of constant s) intersected by the $R(\alpha)$. s_{i1} and s_{i0} are found in a similar manner in the negative s direction.

When we consider the T-mesh in **Figure 8**, the knot vectors for P_1 can be written as

$$\bar{s}_i = [\ s_1, s_2, s_3, s_4, s_5 - d_8]$$

$$\bar{t}_i = [\ t_1 - e_0, t_1, t_2, t_3, t_4 + e_9]$$

The knot vectors for P_2 can be written as

$$\bar{s}_i = [\ s_3, s_3 + d_6, s_5 - d_8, s_5, s_5 + d_5]$$

$$\bar{t}_i = [\ t_1, t_2, t_3, t_4, t_5]$$

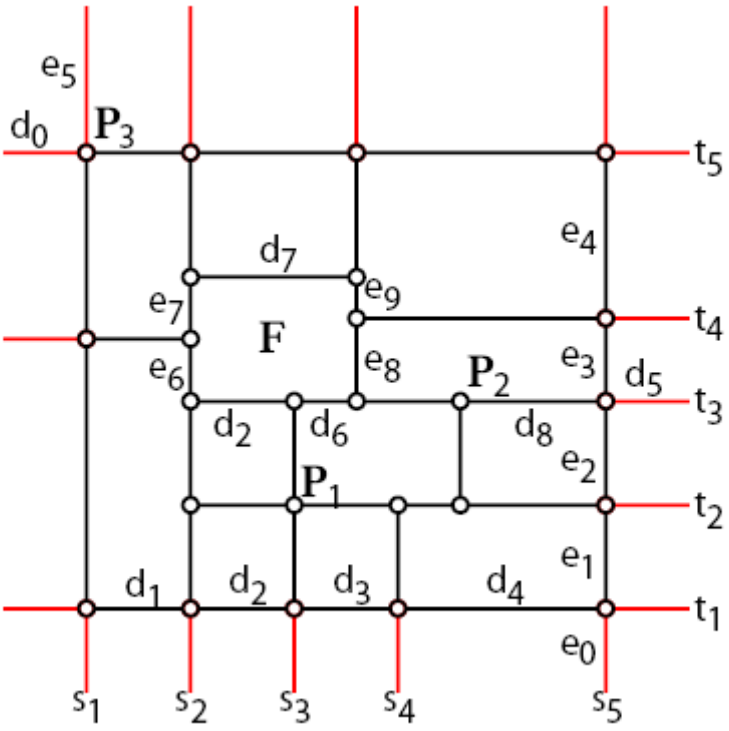

Figure 8 – Knot vectors in T-mesh [4]

2.4 3D Synthetic Human Face Modelling

In this part of the chapter, a brief introduction on the anatomy of the human face is conducted in order to inform the reader about the object to be modelled in this thesis work. A review of the techniques used in 3D human face modelling with a comparative approach concludes the chapter.

2.4.1 Anatomy of the Human Face

Human anatomy has been an area of interest for both science and art for centuries. Human body has been studied extensively by scientists to have a complete understanding of its physical structure and functions. It has also been studied by artists focusing on its appearance. Since our aim is to produce 3D synthetic human face models, it would be beneficial to give some information on anatomy of human

face in an introductory level. Reader may refer to *Gray's Anatomy* [20] for a more detailed and complete description of the facial anatomy.

2.4.1.1 Skull

Skull is the name given to the bone structure of the head. Skull can be divided into two parts, namely cranium and skeleton of the face. Cranium is responsible for protecting the brain and it is formed by two subparts calvaria and cranial base. Bones of the calvaria can be listed as frontal bone, occiptical bone and parietal bones. Bones of the cranial base are temporal bone and sphenoid bone. The layout of the bones mentioned above can be seen in **Figure 9**.

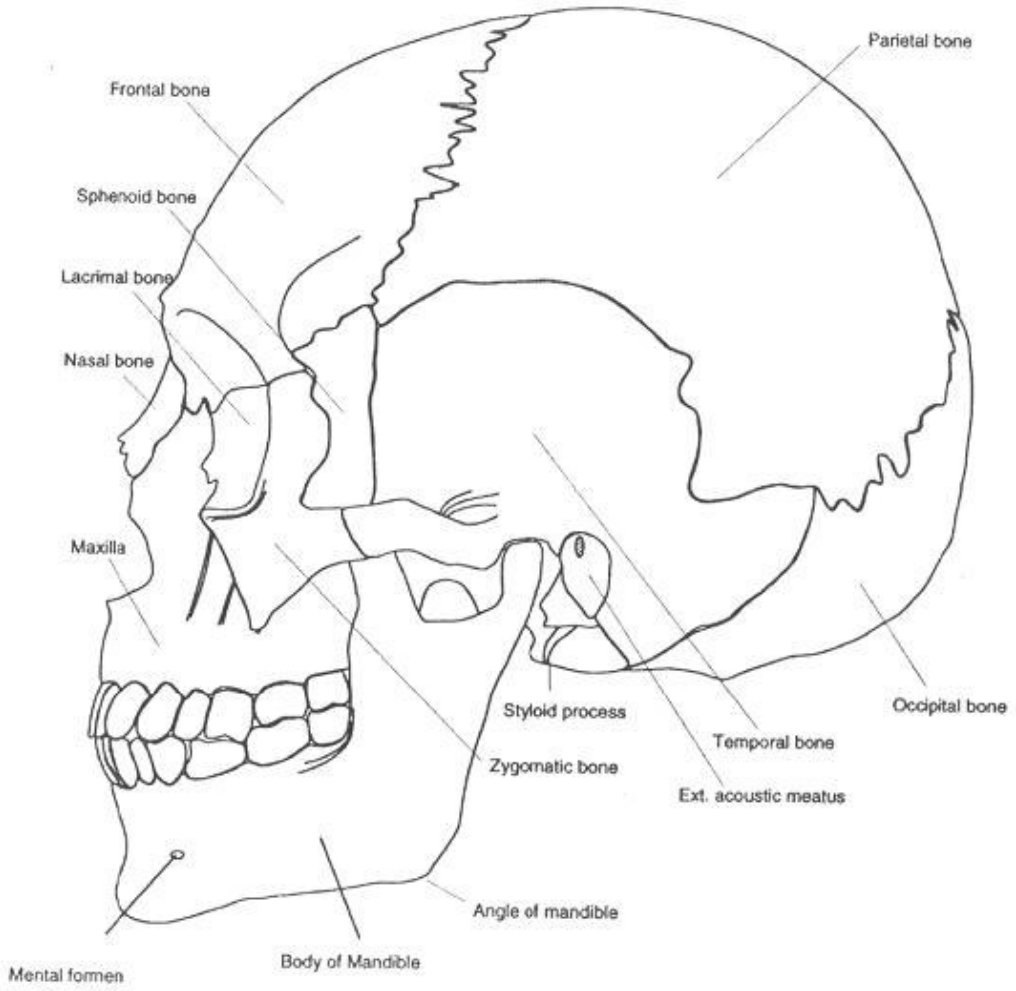

Figure 9- Human skull [3]

Facial skeleton, which rests in the frontal region of the head and serves as a base for the face, is formed by ethmoid bone, palatine bone, maxillae, inferior nasal concha, zygomatic bones,nasal bones, lacrimal bone, mandible, hyoid bone and vomer. The layout of facial skeleton can be seen in **Figure 10**.

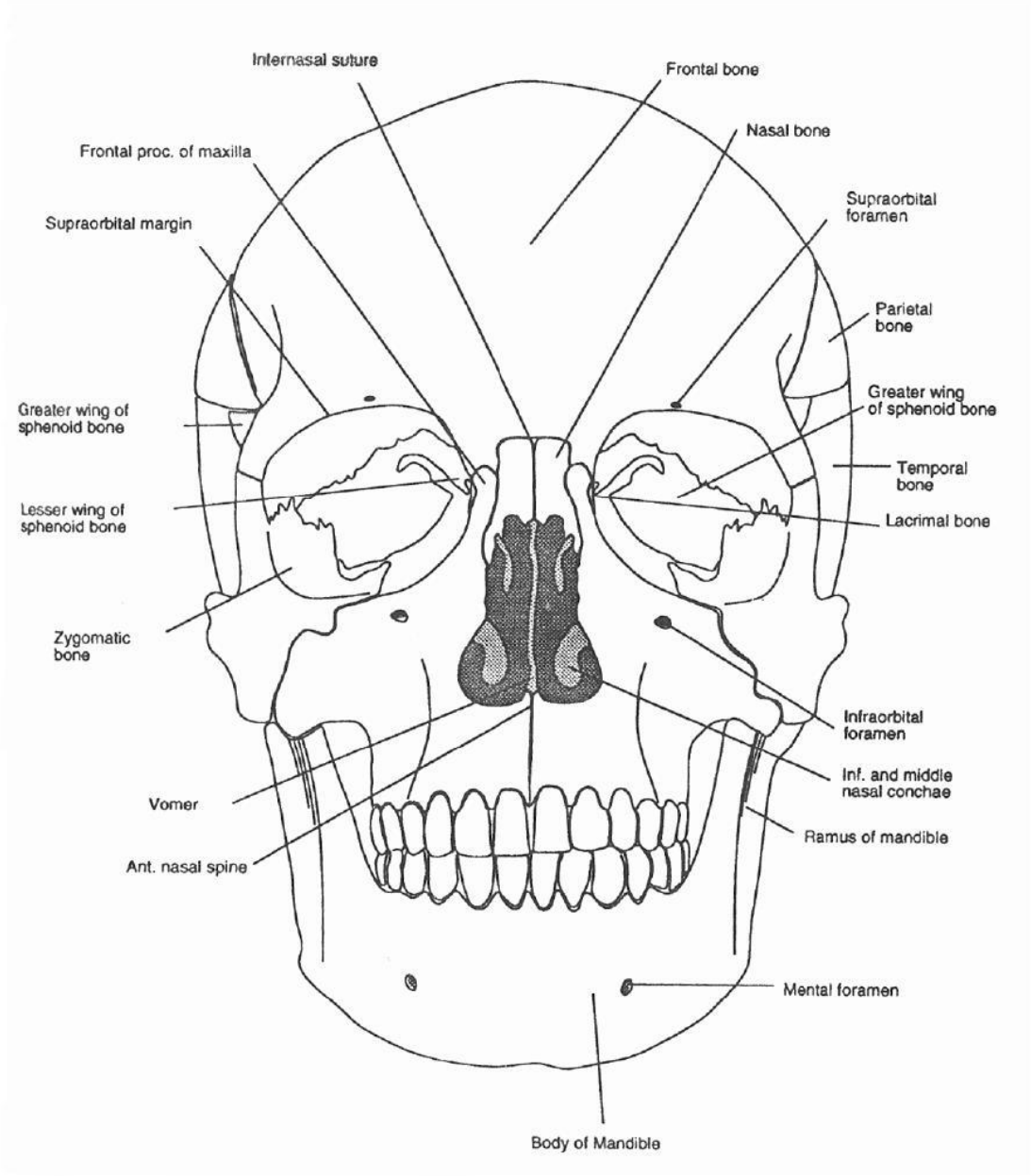

Figure 10- Facial skeleton [3]

2.4.1.2 Muscles

Muscles are the soft tissue, which are responsible for the motion of the human body. Motion is supplied by the contractions and the relaxations of the muscles. Muscles are generally attached to two bones, to a bone and skin, to two different areas of skin or to two organs [3]. Since they cover various areas of the skull, muscles have an important contribution in the shape of the human face. The frontal and lateral settlement of the muscles of human face can be seen in **Figure 11** and **Figure 12**.

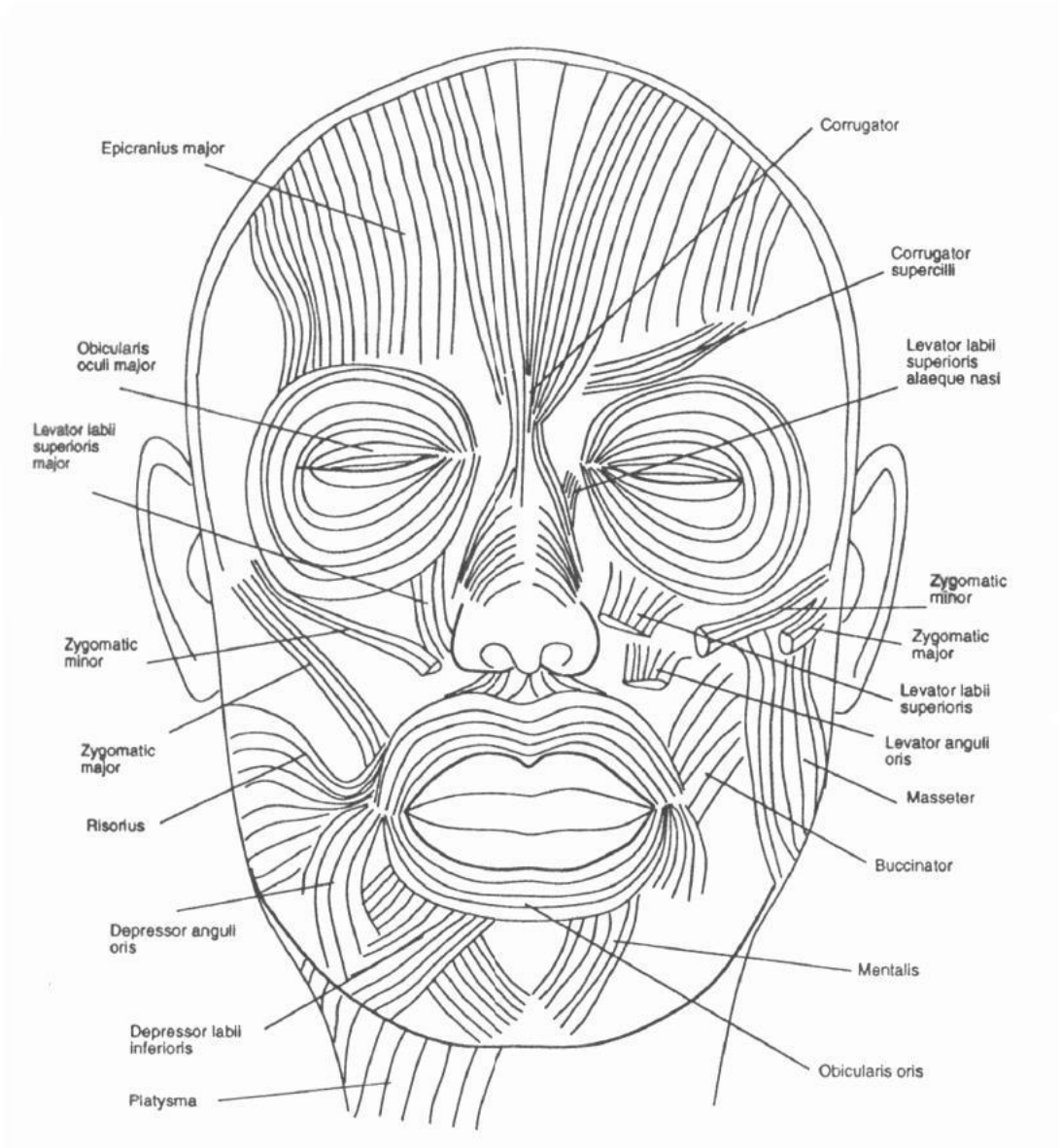

Figure 11- Frontal view of human face muscles [3]

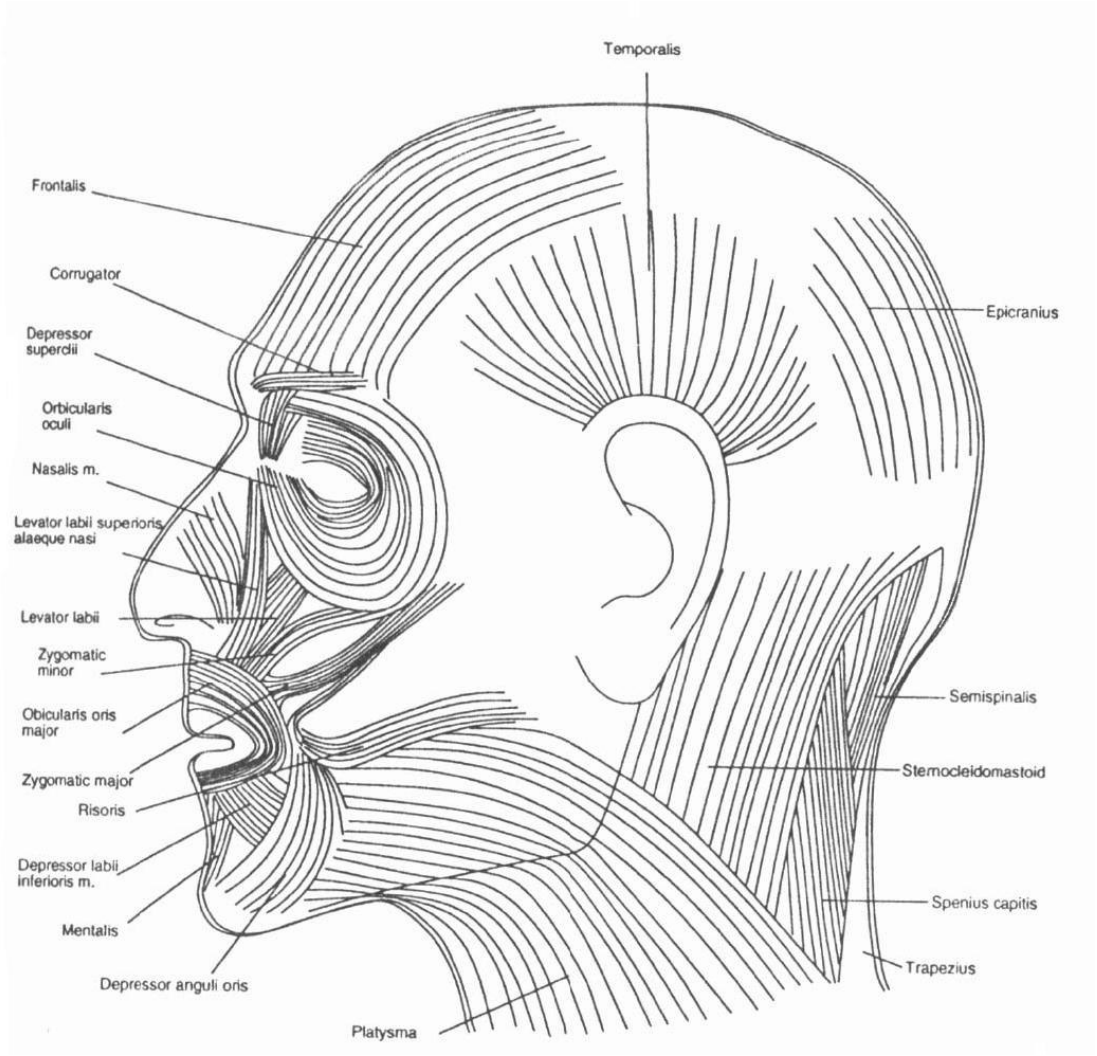

Figure 12-Lateral view of human face muscles [3]

Some of the responsibilities of the muscles of the human face can be listed as the movements of jaw, mouth, lips, cheeks, eyelids and eyebrows. So we can conclude that facial muscles aid digestion by the motion of jaw; aid speech by the motion of mouth and lips; create facial expressions which have a vital role in the communication of individuals.

2.4.1.3 Skin

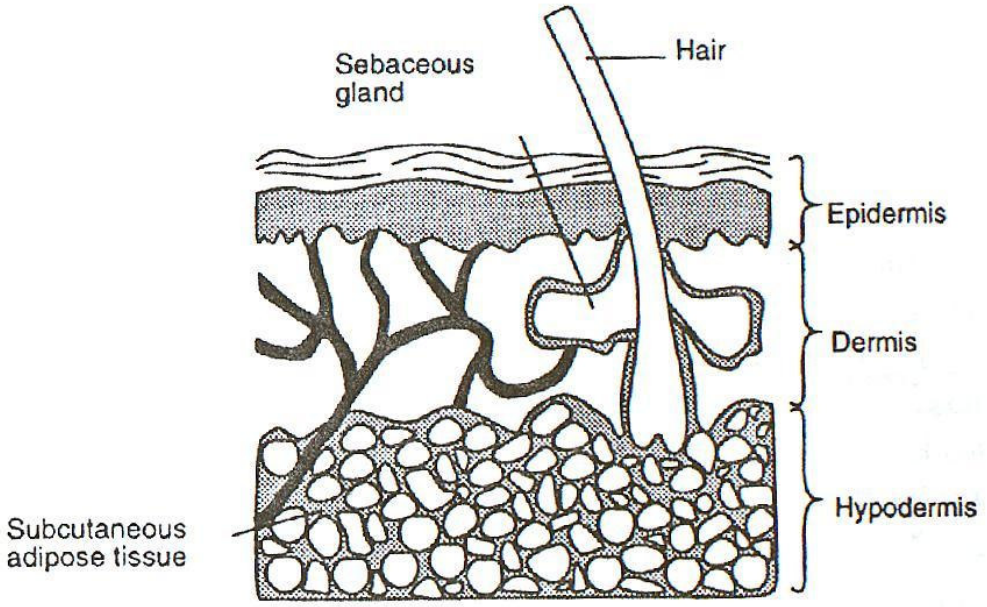

Figure 13- Layers of human skin [3]

Skin is the organic interface between the human body and its environment, so it should be studied carefully to create realistic human face models. Skin has a three layered structure including epidermis, dermis and hypodermis when looked from the body surface through inner regions of the body. These layers can be seen in **Figure 13**. Epidermis is the outermost layer and is composed of keratin. Dermis is the second layer including elastin fibers, proteoglycans, fibronectin, blood vessels, lymphatic vessels and nerves [3]. Hypodermis includes adipose tissue which is mostly formed by fat cells.

2.4.1.4 Eyes

The eyes are safely settled in skeletal orbits and are responsible for the vision in human. The motion of the eyes is controlled by nerves and neurons. Since eyes have an important contribution on the facial appearance and facial expressions of a person, care must be given to the eyes in the modelling process.

2.4.2 Human Face Modelling Techniques

As we introduced in the first part of this chapter, human face has a complex anatomical structure. Different techniques have been used to realize 3D face models considering the requirements of the aimed output. Visual quality, animation capabilities, morphing capabilities, allocated rendering time, storage requirements, texture detail level are some of the factors which effect the decision of modelling technique. In this section we will give more detail about these modelling techniques.

2.4.2.1 Anatomy Based Modelling

This technique uses detailed anatomy knowledge to create human face models. In this approach firstly skull, which forms the innermost and base layer, is modelled. Skull model can be formed by computer tomography scans, magnetic resonance imaging scans or the volume or laser scanning of a real skull.

Then muscle tissue is modelled in relation with skull model. Dealing with only the muscles which are large in volume and contribute to facial expressions generally produces satisfactory results. On the top of the skull and muscle layers, skin is modelled and integrated with the other layers.

Skull model is simpler compared to other models, because its shape is fixed. On the other hand shape of the muscles and skin can change by the contraction and the relaxation of facial muscles. So great attention should be paid to model the characteristics of muscles and skin if an anatomically complete face model is aimed.

Anatomy based modelling is the most complete and realistic way of human face modelling. The drawbacks of this technique may be listed as complexity, high rendering time and necessity of detailed anatomical knowledge. An example for anatomy based face modelling can be seen in **Figure 14**.

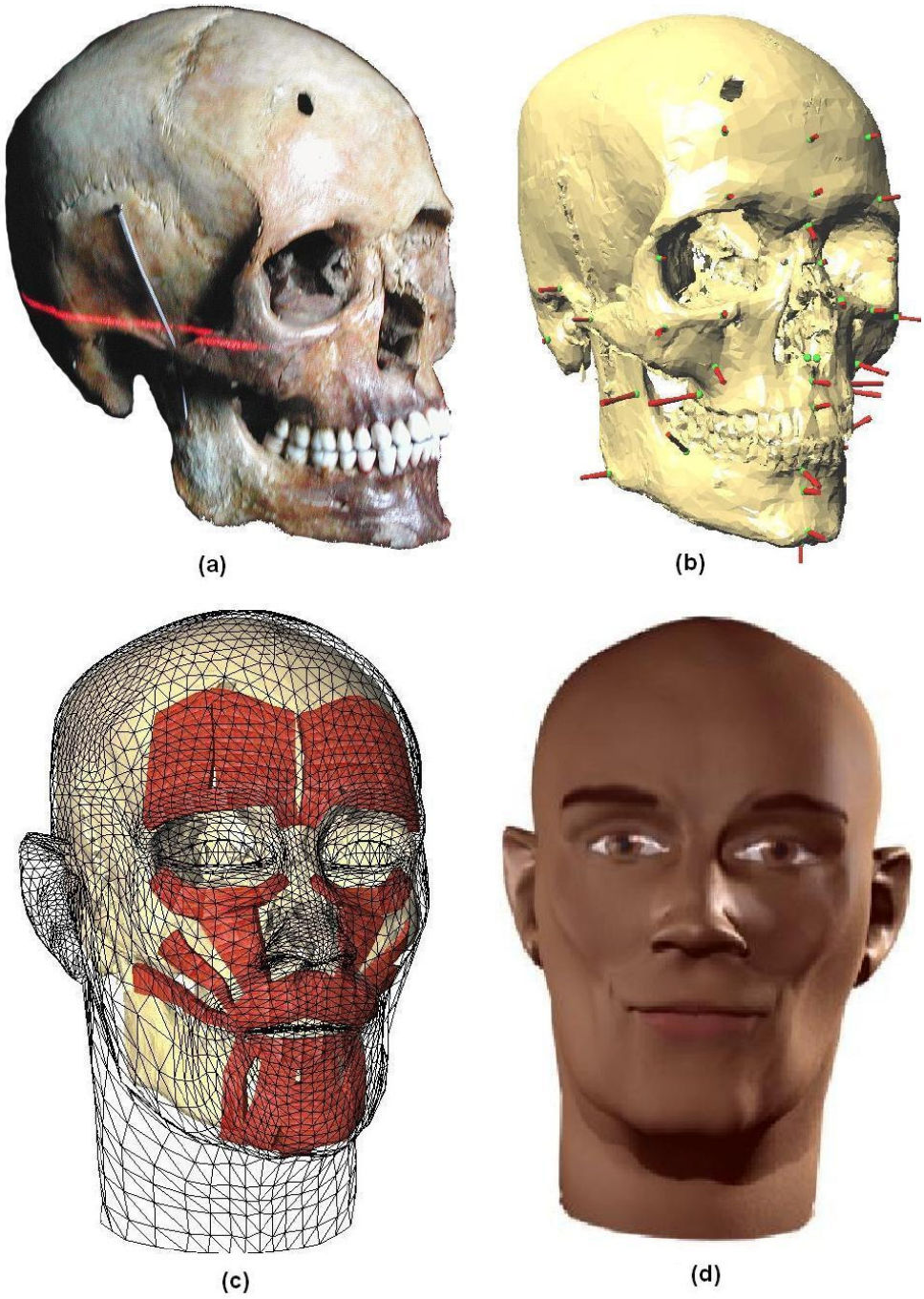

Figure 14 – Anatomy based face model [8]

(a) Original 3D scanned skull image (b) Skull model with connection points of muscles

(c) Skull model covered with muscles and skin (d) Final model with textures coated

2.4.2.2 Point Based Modelling

Point based modelling is the simplest face modelling approach if a visually high quality model is not mandatory. In this technique, only skin, which is the outermost layer of the face is of importance and included in the model. The resulting model is defined by a fixed number of surface points, which form surface polygons. To modify a model, each of the relevant surface points should be edited one by one. Surface points may be grouped to make modifications easier, but it is still very difficult to make believable changes on the face models. If a visually satisfactory model is aimed, a large number of surface points are needed and this makes the creation and management of a point based model very difficult. The visual quality of the model can be improved by texture mapping and by using an advanced polygon rendering method, such as Phong surface rendering rather than flat surface rendering, which is not visually satisfactory.

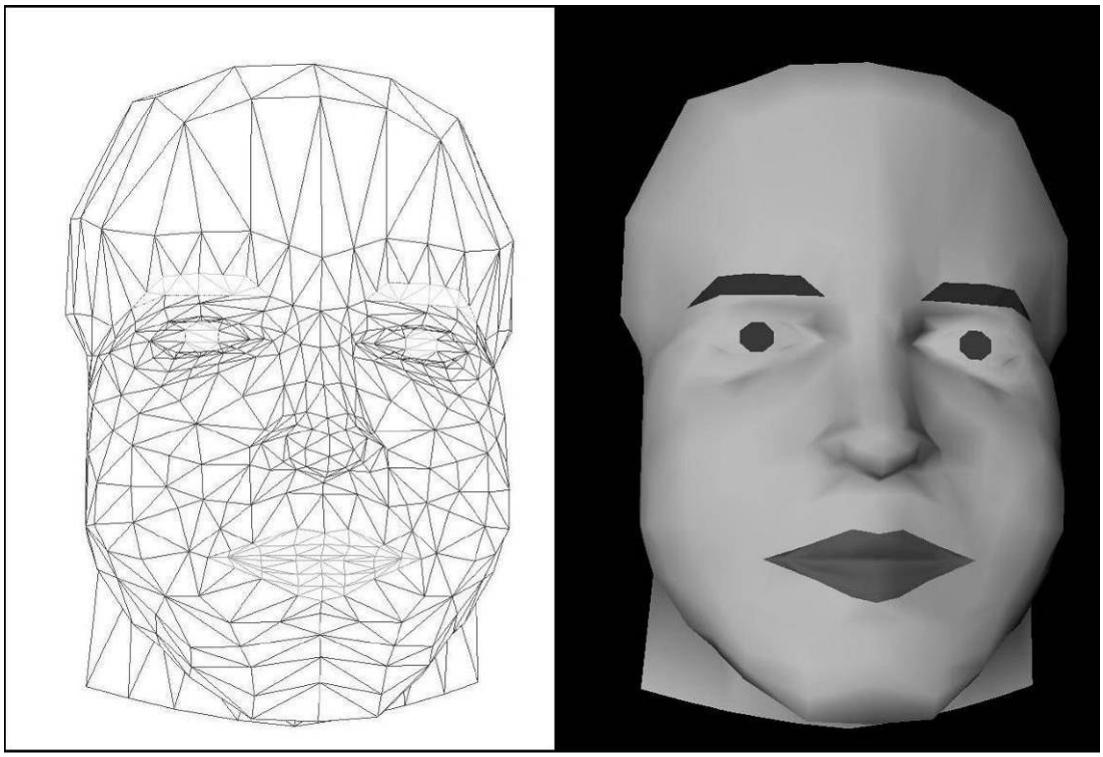

Figure 15- Low resolution point based human face model [10]

A point based synthetic human face model can be seen in **Figure 15**. This model has a small number of surface points and thus a low resolution 3D model is obtained. Since there are few surface points, modifications can be carried easily and model can be rendered in real-time.

Another point based synthetic human face model can be seen in **Figure 16**. This model has a large number of surface points and thus produces a visually satisfactory result. On the other hand management and modification of it becomes pretty difficult due to the density of surface points.

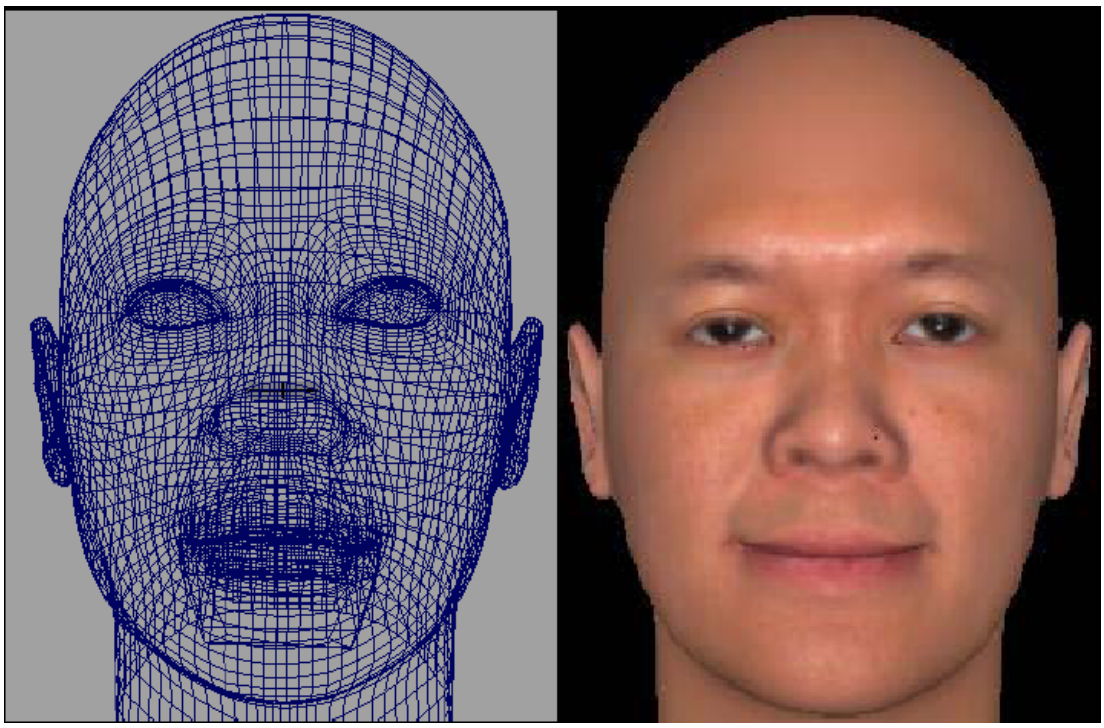

Figure 16- High resolution point based human face model [9]

2.4.2.3 Surface Based Modelling

This modelling approach is based on defining a human face model using surfaces. Similar to point based modelling technique, this technique deals with the outermost layer, skin. Mostly parametric surface descriptions such as Bézier and B-spline surfaces are employed in these surface based models.

Parametric surfaces are defined via a set of control points. The surface points are calculated using the defining control points and each control point effects the surface in a predefined manner. Rendering a surface based model requires more time compared to rendering a point based model, because of the calculations of surface points coordinates. On the other hand, surface based modelling requires far less number of points to define a surface than point based modelling. This makes the management and modification of a surface based model easier compared to a point based model.

Surface based models can be grouped into two as static surface models and dynamic surface models [7]. In a static surface model, topological layout and number of control points is fixed, whereas the number of control points and their layout can be changed in a dynamic surface model. Although static surfaces are simpler, dynamic surface models can be preferred if the ability to add unusual elements such as scars, creases or pimples is desired in the models.

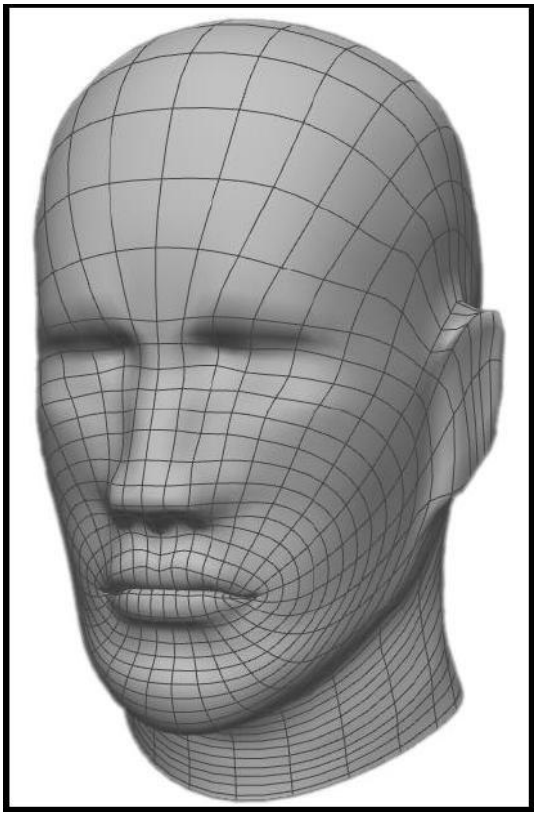

Figure 17- Low resolution surface based human face model [17]

A surface based face model defined by a B-spline surface is given in **Figure 17**. This model has a cylindrical topology originating from the mouth, extending to neck. The set of control points form a rectangular grid. It can be observed that the surface of the model is smooth and does not have enough details. To enhance the visual quality and realism of this model, new control points should be added in eye, nose and ear regions of the face.

Another implementation of surface based modelling using NURBS surfaces can be seen in **Figure 18**. To produce the detailed model on the right, a total of 4712 control points are employed. Blue control points on the figure are responsible for all the details needed. We can observe that a high resolution of blue control points is needed in the parts of the face which have high curvature surfaces. Because of the

topological constraints of the NURBS surfaces, a rectangular mesh of control points is mandatory. To satisfy this condition, several red control points are added in low curvature regions such as forehead and cheeks, however they do not introduce more detail to the model.

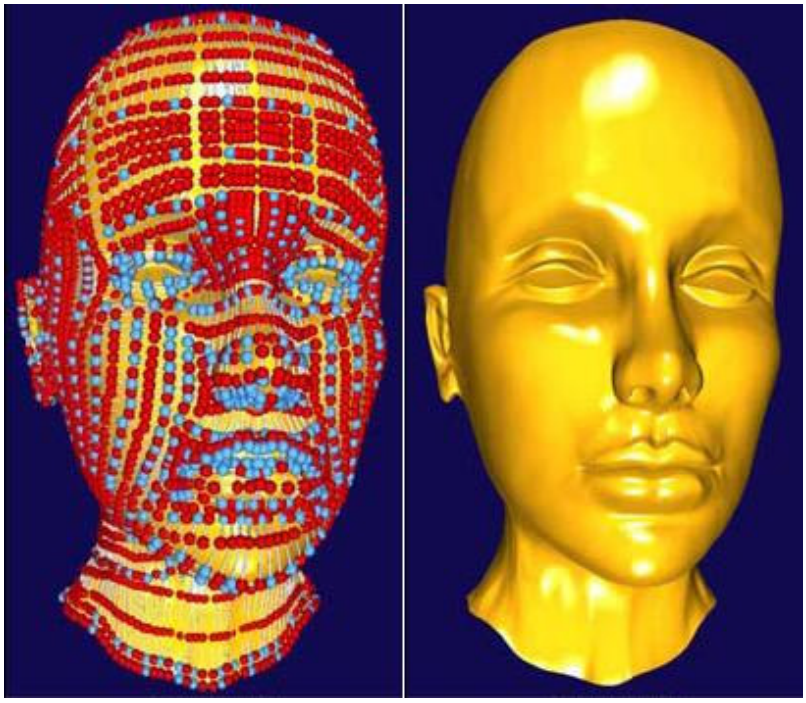

Figure 18-High resolution surface based human face model [5]

CHAPTER 3

DESIGN AND IMPLEMENTATION

In this chapter, design and implementation of 3D Human Face Modelling Software is discussed. Firstly, requirements are introduced and secondly the selection of methods and structures are explained. Thirdly, implementation of the project is explained with the corresponding results.

3.1 Requirements

The purpose of this thesis work is, to design and implement 3D Human Face Modelling Software. To implement this software, capabilities and requirements of the final product has to be defined first. The general properties and requirements of our software are as follows:

In terms of structure, the software is required

- to have a user-friendly graphical user interface
- to be easy to operate
- to be making use of OOP to provide modularity
- to support upgrades

In terms of capabilities, the software is required

- to model a wide range of human faces
- to save a model

- to load a previously saved model
- to edit an existing model
- to have preview tools to be used during modelling process
- to create human face models from scratch with operator in control
- to have modules for modelling the subparts of the face separately
- to have predefined library of subparts
- to allow adding new partial models to the corresponding library

3.2 Selection of 3D Object Representation Method

As previously explained in Chapter 2, space-partitioning representations and boundary representations are two choices for 3D object representations.

Space-partitioning representations require a vast amount of knowledge about the interior structure of the object to be modelled. In our case if a complete space-partitioning representation is employed, modeller has to deal with the anatomical details of the human face. Introductory level information on human face anatomy was presented in Chapter 2 to show the complex structure of human face. Bones, muscles and skin of the face should be modelled individually to have a complete and single volume model. This kind of scheme is more preferable if anatomical models and animation are to be implemented, which is not the purpose of our work.

After a comparative research on both schemes, boundary representation scheme is found to be the better choice for the purposes of this work. Mainly, this scheme is simpler than its space-partitioning counter part, while still having the capabilities to meet the requirements of the project. Secondly, modeller has to deal with only the outer surface of the object to be modelled, which is sufficient in our case. This helps the design process, since less anatomical knowledge is needed.

3.3 Selection of Boundary Representation Method

After deciding on boundary representation method, the technique for representing the boundary surface of the object should be chosen. As the first possible technique, implicit surfaces were considered but not preferred because of:

- Difficulties in definition and implementation of them.
- Difficulties in the definition and modification of a human face model in implicit form.

Polygonal surfaces deserve a careful concern for the purposes of our work. It is obvious that, defining all polygons lying on the boundary surface of the object densely enough would deliver a satisfactory result in terms of image quality. However, there are some drawbacks of this scheme which can be expressed as follows:

- When a high quality image is aimed, the number of surface polygons increases and the management of the model gets more complex.
- Parameterization of a dense 3D polygon model is difficult to implement.

Observing the drawbacks of implicit surfaces and polygonal surfaces, parametric surfaces have been chosen for the boundary representation technique. The underlying reasoning can be explained as follows:

- Parametric surfaces are simpler to implement compared to implicit surfaces.
- Parametric surfaces can be defined by control points on a control space.
- When we compare the number of control points in a parametric surface description to the number of surface points in a polygonal surface description, the first one is found to be much less than the second. This means that a modeller has to deal with fewer points when a modification is carried on the model.
- Having a smaller number of points in a model makes the parameterization of the model easier.
- Using the control points, resulting model surface can be created in a desired image quality by sweeping the parameter surface with the desired increments in both parameter directions.

3.4 Selection of Parametric Surface Description Method

Bézier Surfaces, B-spline Surfaces, NURBS Surfaces, Hierarchical B-spline Surfaces and T-spline Surfaces were evaluated as the candidates of parametric surface description schemes as summarized in Chapter 2.

Simplicity of Bézier surfaces makes the implementation of these surfaces pretty straightforward, whereas some of its properties create disadvantages when we try to employ them. Firstly, the degrees of a Bézier surface in parametric dimensions are fixed by the defining polygon vertices and this causes a lack of flexibility. Secondly, the basis functions in Bézier surface formulation are real and nonzero for all parameter values. Thus, a change in the position of a control point is reflected throughout the entire curve, which makes applying local changes to the shape of the curve difficult. Because of this drawback, Bézier surfaces can be used in the modelling of the objects which do not have high curvature surface regions and have a smooth surface. Since having strong local control on the model surface is a vital requirement in our work, Bézier surfaces were eliminated.

B-spline surfaces have been widely used in modelling applications for many years. B-splines are flexible in that the degree of a B-spline polynomial can be set independently from the number of control points with some limitations and knot vectors can be chosen to meet the desired output. In addition, more local control can be achieved by using control point insertion, multiple polygon vertices and multiple knot values. Even though B-splines have several flexibilities and nice properties, they also suffer some disadvantages mainly due to topological constraints. As explained in Chapter 2, when we choose to add new control points to the control points set, we have to add several more control points to satisfy the rectangular grid topology of B-splines. This means that modeller has to deal with more control points, which makes parameterization of the model difficult.

NURBS surfaces have been also very popular similarly to B-splines. NURBS surfaces are stronger than B-spline surfaces in terms of local control, with the weight factors included in their formulation. This property helps to pull the surface towards

to a control point more strongly or decrease the influence of a control point on the model surface, thus increasing the local control capabilities. However, the topology of NURBS also requires a rectangular grid of control points, which creates the main drawback of this scheme. With this requirement, several additional control points have to be inserted to form a rectangular grid, after the insertion of a single control point for improving local control in a NURBS surface. B-spline surfaces and especially NURBS surfaces can be used in many modelling applications thanks to their local control capabilities. But merging two NURBS surfaces to form a smooth surface is a pretty difficult process. To merge two B-spline surfaces, these surfaces should have a suitable grid of control points, which coincide to form a single rectangular grid of control points after merging. If the rectangular grids of two surfaces do not coincide to support merging, rows or columns of control points that do not have a counter row or column in the other surface, should be extended into the other surface. This process may introduce a high number of additional control points during merging, which is not desired.

Hierarchical B-spline surface topology has been evaluated as a powerful modelling technique. In addition to the positive aspects of B-splines, it has two main additional properties, namely *overlays* and *offset referencing*. These properties can be summarized as follows:

- With the use of overlays, Hierarchical B-spline surfaces provide local refinement by adding control points only to the area to be refined. In contrast to B-spline surfaces, additional control points through a row or column need not be added to the control grid.
- With the offset referencing feature, a refined region can be defined by taking the hierarchically higher control points as reference. This provides the refined region to move with the main surface when a modification to the main surface is done.

Hierarchical B-Spline surface scheme relaxes some of the topological constraints of B-spline surfaces. However, the control points in a bounded area surrounding the

overlays (depends on the order of the main and overlay surface) cannot be moved freely, not to disturb the continuity of the surface. As a result, Hierarchical B-spline surface topology is not very promising for the purposes of our work. However, this method can be preferred in modelling sharp and thin objects connected to a relatively flat region, like horn of an animal. To model more detailed objects, new overlays must be defined in previously defined overlays, thus extending the hierarchy between surfaces. But as number of hierarchical levels increases, the complexity of the model also increases and management of the surface becomes difficult.

T-spline surface technique is the most recent parametric surface description scheme evaluated and discussed in this thesis work. After evaluating the advantages and disadvantages of Bézier, B-spline, NURBS and Hierarchical B-spline surfaces, T-spline scheme is found to be the best parametric surface description scheme to meet the requirements of this work. The aspects of T-splines which make it the best choice for the purposes of this work can be listed as follows:

- Since it is a generalization of NURBS, it inherits all advantages of NURBS.
- It allows the use of T-junctions, which promise flexibility in terms of topology.
- It does not require the control points to be aligned in a rectangular grid structure in contrast to NURBS. This prevents the inclusion of extra control points into the model. As a result, an equivalent T-spline model of an object can be created with fewer control points compared to a NURBS model.
- Insertion and deletion of control points is possible under some limitations, to achieve local control.
- T-spline models can be converted to NURBS models if needed for compatibility reasons.

3.5 T-spline rendering

To meet the requirements of this work, C++ has been chosen as the programming language to benefit from the OOP concepts, which contributes to the implementation of a modular application. OpenGL is employed as the graphics library, due to its capabilities and ease of use.

T-spline scheme is to be used in this project, thus our face modelling tool must be capable of rendering T-spline surfaces. OpenGL has predefined functions for rendering several types of 3D objects such as polyhedrons, quadratic surfaces and basic spline surfaces (i.e. Bézier and NURBS) with trimming capabilities, but not T-spline surfaces. There are T-spline plugins for commercial CAD products Autodesk Maya and Rhino. Since a standalone application is aimed, implementation of the software was started with a T-spline renderer.

As we already discussed in Chapter 2, knot information of a T-spline surface is deduced from the T-mesh. So we need to represent T-meshes in computer environment. To meet this requirement *Tmesh* class is created. Since control points in a T-mesh have some special properties, *Tpoint* class is created to be used in *Tmesh* class. These two classes are discussed below.

3.5.1 Tpoint class

Tpoint class of our modelling software represents a control point in a T-mesh. Each control point in a T-mesh has to be defined with certain properties to have a complete definition of a T-spline surface. These properties are 3D coordinates, weight factor, knot coordinates (*s-t coordinates*) and neighbourhood information with other control points. To meet these properties following member variables are defined in our *Tpoint* class:

- ***x*** - *x* coordinate of the control point.
- ***y*** - *y* coordinate of the control point.
- ***z*** - *z* coordinate of the control point.
- ***w*** - *weight factor* of the control point.

- ***s*** - *s coordinate* of the control point.
- ***t*** - *t* coordinate of the control point.
- ***id*** - *id* of the control point, which is used for distinguishing each control points and for indexing.
- ***nsu_id*** - *id* of the neighbouring control point, in *+s* direction with the same *t* coordinate. If control point does not have a neighbouring control point in *+s* direction, it is either set to *-1* or *-2* (*-1* if it is an *interior vertex*, *-2* if it is a *boundary vertex)* .
- ***nsd_id*** - *id* of the neighbouring control point, in *-s* direction with the same *t* coordinate. If control point does not have a neighbouring control point in *-s* direction, it is either set to *-1* or *-2* (*-1* if it is an *interior vertex*, *-2* if it is a *boundary vertex)* .
- ***ntu_id*** - *id* of the neighbouring control point, in *+t* direction with the same *t* coordinate. If control point does not have a neighbouring control point in *+t* direction, it is either set to *-1* or *-2* (*-1* if it is an *interior vertex*, *-2* if it is a *boundary vertex)* .
- ***ntd_id*** - *id* of the neighbouring control point, in *-t* direction with the same *t* coordinate. If control point does not have a neighbouring control point in *-t* direction, it is either set to *-1* or *-2* (*-1* if it is an *interior vertex*, *-2* if it is a *boundary vertex)* .

Reading and editing the attributes of control points is needed when surface points are calculated and when changes are necessary respectively. So member functions to read and edit member variables were created to meet this requirement in *Tpoint* class.

3.5.2 Tmesh class

Tmesh class of our software represents a complete T-mesh. In a *Tmesh* object, the list of control points, the number of control points and ranges of parametric dimensions are defined. The member variables of *Tpoint* class can be listed as follows:

- ***TpointList*** – the list of the control points in a T-mesh, where each control point is represented with a *Tpoint* object. The neighbourhood relation of control points with each other can be deduced from this list.
- ***size*** – the number of control points in a T-mesh.
- ***smin*** – minimum *s* coordinate in a T-mesh.
- ***smax*** – maximum *s* coordinate in a T-mesh.
- ***tmin*** – minimum *t* coordinate in a T-mesh.
- ***tmax*** – maximum *t* coordinate in a T-mesh.

Functions for reading and editing these member variables were also created.

The main function *Tmesh* class performs is, defining and drawing a T-spline surface. As we already explained in the T-spline surface formulation in Chapter 2, basis functions in both parametric directions should be calculated to map knot coordinates to 3D coordinates. In addition, for the calculation of basis functions, knot vectors in both parametric directions should be known for each control point.

The calculation of knot vectors and basis functions, simplification of T-spline surface calculations and calculation of a surface point in our software are discussed below.

3.5.2.1 Calculation of s-knot vector

s_knot function of *Tmesh* class is responsible for deducing the *s-knot vector* for the desired control point, which is crucial in the calculation of the basis functions. The s-knot vector was defined in Chapter 2 as:

$$\bar{s}_i = [\ s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$$

As discussed before, for a control point P_i with knot coordinates $(s_{i2},\ t_{i2})$, this function copies s_{i2} directly from the knot coordinates of P_i, $R(\alpha)$ is defined as $R(\alpha) = (s_{i2} + \alpha, t_{i2})$

Then, s_{i3} is to be found as the *s* coordinate of the first *s-edge* intersected by the $R(\alpha)$ in positive *s* direction. To find this constant *s* value, function first looks if P_i has a

neighbouring control point in the positive *s-direction*. If such a control point exists, then s_{i3} is the *s* coordinate of this neighbouring control point. If such a control point does not exist, this means that P_i is a *T-junction*. The function is divided into two procedures at this step. 1st procedure moves in the positive *t-direction* until it finds a control point, which has a neighbouring control point in the positive *s-direction.* In parallel, 2nd procedure moves in the negative *t-direction* until it finds a control point, which has a neighbouring control point in the positive *s-direction.* Then *s* coordinates of the control points found by the two parallel procedures of the function are compared. If the deduced *s* coordinates are identical, this proves that an *s-edge* is found and s_{i3} is the *s* coordinate of *s-edge.* If the deduced *s* coordinates are not identical, both procedures continue to look for an *s-edge*.

s_{i4} is found in a similar manner. The function firstly checks for a neighbouring control point in the positive *s-direction.* If such a control point exists, then s_{i4} is the *s* coordinate of this control point. If such a control point does not exist, s_{i4} is again found by two parallel procedures which look for an *s-edge* in the positive *s-direction.*

s_{i1} and s_{i0} are found in a similar manner but working the algorithm above in the negative *s-direction*.

After all elements of knot vectors are identified for a control point, they are returned in the vector form $\bar{s}_i = [\, s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$.

3.5.2.2 Calculation of t-knot vector

t_knot function of *Tmesh* class is responsible for deducing the *t-knot vector* for the desired control point, which is crucial in the calculation of the basis functions. *t_knot vector* is calculated with a similar approach to *s_knot vector.* The only difference is that, for a control point with knot coordinates (s_{i2}, t_{i2}), $R(\alpha)$ is defined as $R(\alpha) = (s_{i2}, t_{i2} + \alpha)$. In the calculation of the elements of *t_knot vector,* function looks for *t-edges* to deduce t_{i0}, t_{i2}, t_{i3} and t_{i4} values.

After all elements of knot vectors are identified for a control point, they are returned in the vector form $\bar{t}_i = [\, t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$.

3.5.2.3 Calculation of basis functions

In our work, we calculate basis functions in *s* and *t* parametric directions for a control point, for given *s* and *t* values, using *Ni_s* and *Ni_t* functions respectively. For *Ni_s,* this function firstly calls the *s_knot* function to deduce knot vectors for the desired control point. Then, by using the *s-knot vector*, function calculates the value of the basis function $N[\bar{s}_i](s)$ at *s*.

As we explained in Chapter 2, C^2 was chosen for the continuity of our T-spline surface. So, we should have a 3rd degree, i.e. 4th order T-spline surface.

The B-spline basis functions were defined by Cox-deBoor recursion formulas in Chapter 2 as:

$$N_{i,1}(t) = \begin{cases} 1 \ \textit{if} \ x_i \le t < x_{i+1} \\ 0 \ \textit{otherwise} \end{cases} \qquad \text{and}$$

$$N_{i,k}(t) = \frac{(t - x_i)\, N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)\, N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

When we evaluate these formulas for 4th order for all *s* intervals, we reach the corresponding formulas:

- for $s_0 \le s < s_1$

$$N_{1,4}(s) = \frac{(s - s_0)^3}{(s_1 - s_0)(s_2 - s_0)(s_3 - s_0)}$$

- for $s_1 \le s < s_2$

$$N_{1,4}(s) = \frac{(s - s_0)^2 (s_2 - s)}{(s_3 - s_0)(s_2 - s_0)(s_2 - s_1)} + \frac{(s - s_0)(s_3 - s_0)(s - s_1)}{(s_3 - s_0)(s_2 - s_1)(s_3 - s_1)} + \frac{(s_4 - s)(s - s_1)^2}{(s_4 - s_1)(s_3 - s_1)(s_2 - s_1)}$$

- for $s_2 \le s < s_3$

$$N_{1,4}(s)=\frac{(s-s_0)(s_3-s)^2}{(s_3-s_0)(s_3-s_2)(s_3-s_1)}+\frac{(s_4-s)(s-s_1)(s_3-s)}{(s_4-s_1)(s_3-s_1)(s_3-s_2)}$$
$$+\frac{(s_4-s)^2(s-s_2)}{(s_4-s_1)(s_4-s_2)(s_3-s_2)}$$

- for $s_3 \le s < s_4$

$$N_{1,4}(s)=\frac{(s_4-s)^3}{(s_4-s_1)(s_4-s_3)(s_4-s_2)}$$

- for $s < s_0$ *and* $s \ge s_5$

$$N_{1,4}(s)=0$$

This set of formulas is also applied in the implementation of the function *Ni_t*, but in *t* direction. Using the formulas above, function *Ni_s* and *Ni_t* calculates the value of the basis function in *s and t* directions, for given *s* and *t* values.

3.5.2.4 Simplification of T-spline surface calculation formula

T-spline surface definition is expressed in Chapter 2. To implement a T-spline surface, this definition was directly implemented at first attempt in our work. But the time required to render a T-mesh was measured to be very long, especially when the number of control points were large. This was a problem for our software, since quickly previewing the surface defined by the model during the modelling process is needed. A small modification in the T-spline surface definition was made to solve the performance problems which occurred in the calculation of a T-spline surface. The formulation for T-spline surfaces was given as:

$$P^3(s,t)=(x(s,t),y(s,t),z(s,t))=\frac{\sum_{i=1}^{n} P_i^3\, w_i\, B_i(s,t)}{\sum_{i=1}^{n} w_i\, B_i(s,t)}$$

where P_i^3 s' are control points in Cartesian coordinate system and $B_i(s,t)$ are the blending functions defined by:

$$B_i(s,t) = N[\bar{s}_i](s)\, N[\bar{t}_i](t)$$

It is obvious that $B_i(s,t)=0$ for some control points and $B_i(s,t)\neq 0$ for the other control points for a given knot coordinate pair (s,t). By detecting the control points which have contribution to the calculation of resulting surface points, we can decrease the rendering time by excluding the calculations for those control points with no contribution.

FindInnerAndBorderTpoints function of *Tmesh* class creates a list of the control points in the T-mesh, which can contribute to the calculation of the Cartesian coordinates of a point whose *knot coordinates* are given.

3.5.2.5 Calculation of a T-spline surface point

SurfPoint function of *Tmesh* class calculates the Cartesian coordinates of a point, whose knot coordinates are given. It first creates the list of effective control points using *FindInnerAndBorderTpoints* function, then calculates coordinates of the surface point with the aid of previously defined *Ni_s*, *Ni_t*, *s_knot* and *t_knot* functions.

3.5.2.6 T-mesh file format

The ability to save and load 3D models was mandatory throughout our work. Two functions were created to read a T-mesh from file and to write a T-mesh to file, *ReadFromFile* and *WriteToFile* functions of *Tmesh* class respectively. These functions share the same file format for reading and writing purposes. This file format is explained with an example in Appendix A.

3.6 Face Component Modelling

As explained in the requirements of our work, a modular modelling approach is aimed in the modelling process. This means that subparts of the human face (nose, eyes, mouth) are to be modelled separately and then these subparts are to be unified in one complete model, considering the global characteristics of the face in the unification process.

The proposed method for the modelling of subparts of the face is creating an editable generic model for each subpart of the face. Then these sub-models can be unified using a generic face model, which can represent the attributes of the face other than the subparts mentioned above (i.e. width of the face, length of chin, height of the forehead, etc.)

Using the generic models, modelling every human face is aimed theoretically. To achieve this, a parameterized realistic model is needed. In this study, parameterized generic models are produced for face components (nose, eye, mouth) as well as the rest of the face as a whole. These generic models are formed as T-spline surfaces, which are obtained from the B-spline models of a real face. Firstly, B-spline models of a real human face is obtained by fitting B-spline surfaces to 3D scanner output, which provides 3D shape data of a real person. Then these fitted B-spline models are edited to reach T-spline models since they are easy to parameterize. B-spline surface fitting and editing of B-spline models to obtain T-spline models are summarized in this part of the text.

3.6.1 B-Spline Surface Fitting

If we have a rectangular set of data points $D_{i,i}(u,v)$, using the B-spline formulation, we can write for each data point [2]:

$$D_{1,1}(u_1, v_1) = N_{1,k}(u_1)\left[M_{1,l}(v1)B_{1,1} + M_{2,l}(v1)B_{1,2} + ... + M_{m+1,l}(v1)B_{1,m+1}\right] +$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots +$$

$$N_{n+1,k}(u_1)\left[M_{1,l}(v1)B_{n+,1} + M_{2,l}(v1)B_{n+1,2} + ... + M_{m+1,l}(v1)B_{n+1,m+1}\right]$$

where we have an *r by s* rectangular set of data such that

$$2 \le k \le n+1 \le r \text{ and } 2 \le l \le m+1 \le s$$

Here, u and v values for each data point are found using a chord length approximation. According to this chord length approximation, for r data points, parameter value u for the p^{th} data point in can be expressed as follows:

$$\frac{u_p}{u_{\max}} = \frac{\sum_{g=2}^{p} \left| D_{g,s} - D_{g-1,s} \right|}{\sum_{g=2}^{r} \left| D_{g,s} - D_{g-1,s} \right|}, \ (u_1=0).$$

$D_{g,s}$ represents the data points, u_{max} is the maximum value of the knot vector in u direction and u_1 is assumed to be zero by definition. u_{max} is selected arbitrarily since all u_p are scaled by the formula. u_{max} can be selected as 1 by convention.

For s data points, parameter value v for the q^{th} data point can be expressed as follows:

$$\frac{v_q}{v_{\max}} = \frac{\sum_{g=2}^{q} \left| D_{r,g} - D_{r,g-1} \right|}{\sum_{g=2}^{s} \left| D_{r,g} - D_{r,g-1} \right|}, \ (v_1=0).$$

Again, $D_{g,s}$ represents the data points, v_{max} is the maximum value of the knot vector in v direction and v_1 is assumed to be zero by definition.

Writing each point in the form above, we can form the matrix equation:

$$[D] = [C][B]$$

where $C_{i,j} = N_{i,k} M_{j,l}$

If [C] is square, control points can be found by matrix inversion

$$[B] = [C]^{-1} [D]$$

If [C] is not square, problem can be solved in a mean sense by obtaining pseudoinverse of C matrix, such that:

$$[B] = \left[[C]^T [C] \right]^{-1} [C]^T [D]$$

Using a software which employs the procedure above, implemented in [21], we have obtained several B-spline human face models. These models had large number of control points to fit to the high detailed point clouds obtained by 3D scanning. Two examples of these fitted models are given in **Figure 19**.

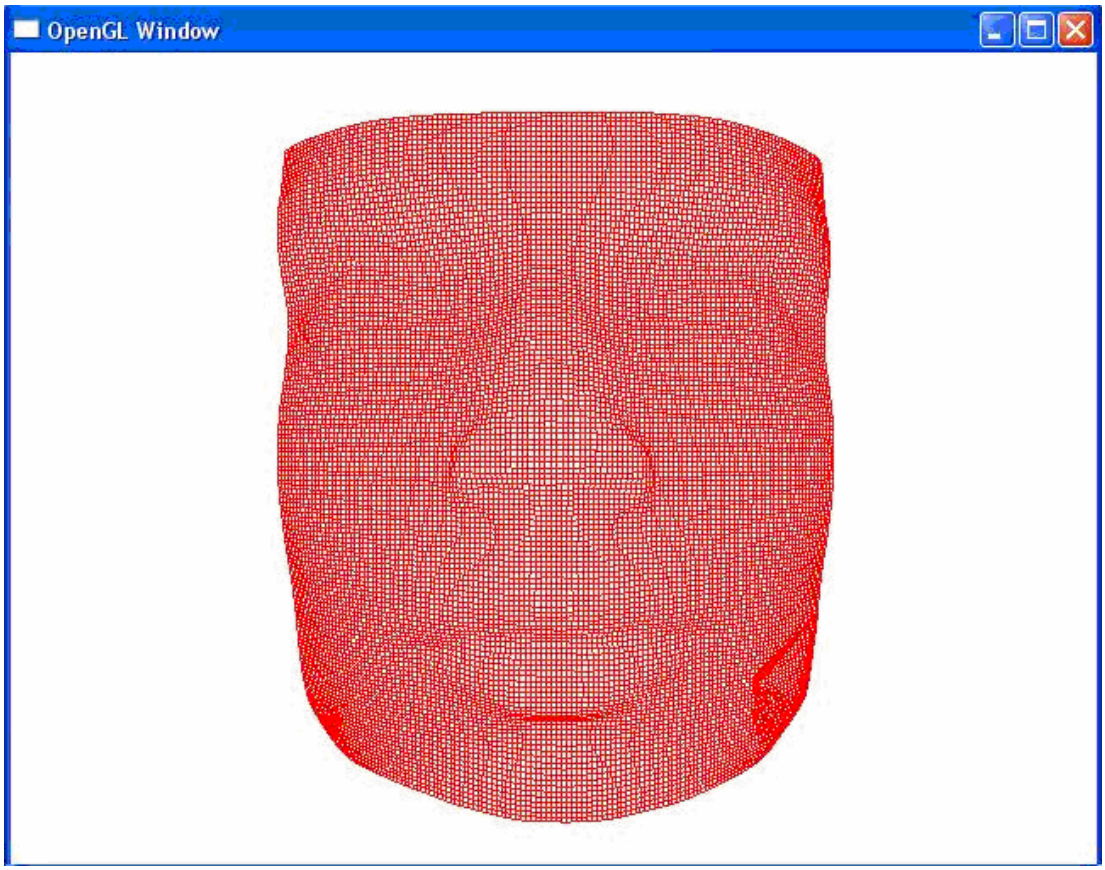

Figure 19 – B-spline fitted human face models

3.6.2 Editing B-spline models

After obtaining B-spline fitted models for the face, regional models for the nose, mouth and eye are extracted from these fitted models for further processing and parameterization. The extracted models have a large number of control points to define the details of the data point cloud to be fitted. But since our aim is to have the ability of creating several models by modifying the generic models, we should exclude as many points as possible to have an easy parameterization step. Then properties of each model are associated with the remaining points in the model. Generic models of nose, mouth and eye are created with this approach as explained in the following parts of the text.

3.6.2.1 Creation of Generic Nose Model

Settling in the centre of the human face, nose has an important contribution to the complete shape of a human face. With two nostrils at the bottom, it has a challenging structure to model realistically.

In our work, firstly a raw nose model was created by extracting the nose region from a fitted B-spline model. Then this model was simplified, to decrease the number of control points, which is vital to have a successful parameterization. The control points which have small effects on the shape of the nose surface were deleted, using the advantages of T-spline topology. The positions of the control points were adjusted manually, to match the high detailed nose regions such as nostrils. The successive simplification of the model is visualized in **Figure 21** in 15 steps. As we move to the final generic model, the number of control points is decreased and model is simplified for parameterization.

Screenshots of the resulting generic nose model rendered as wire-frame and as surface point cloud are presented in **Figure 20.**

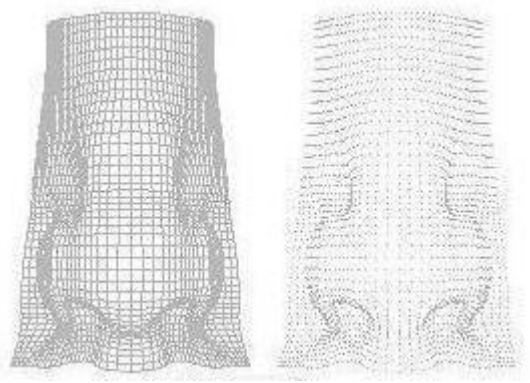

Figure 20- Generic nose model rendered as wireframe and surface point cloud

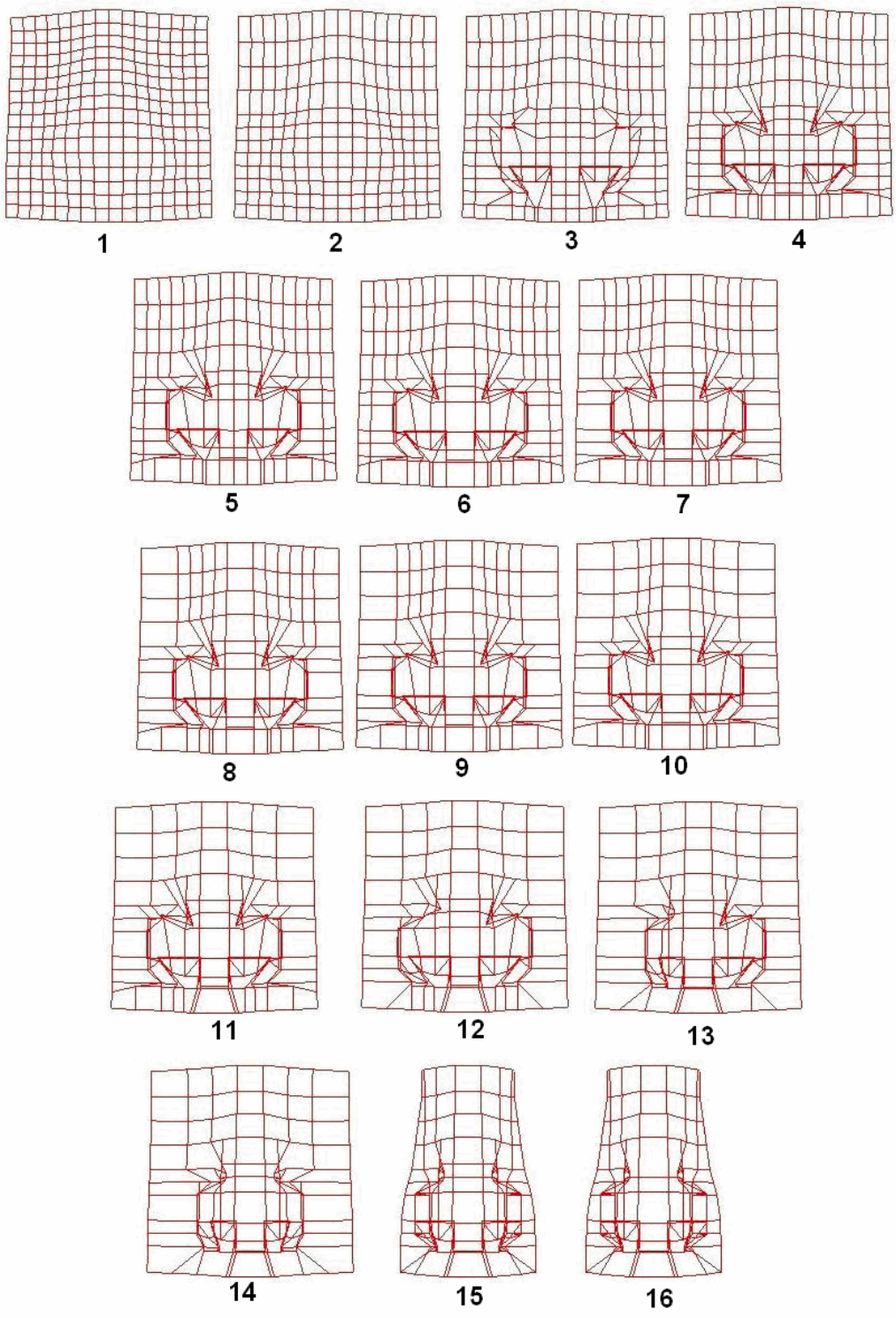

Figure 21- Generic T-spline Nose Model Creation

Screenshots of the generic nose model rendered as quadrilaterals with illumination are presented in **Figure 22.**

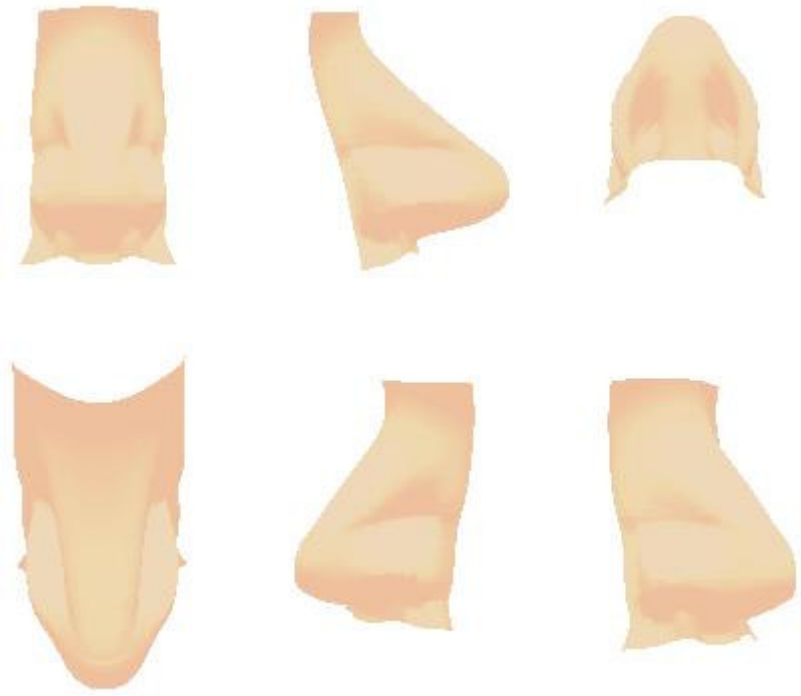

Figure 22- Generic nose model rendered as quadrilaterals with illumination

After the creation of a visually satisfactory generic nose model, it is parameterized to have the ability to model noses with different properties. During parameterization, a key characteristic of the generic model is associated with the control points which are responsible for that specific characteristic. For example to parameterize the width of the generic nose model, the relative positions of the control points surrounding the nostril with reference to the vertical symmetry axis of the nose should be edited. Characteristics of the nose that are currently parameterized in our generic nose model are width, upper height, lower height, nostril width and upwards-downwards orientation. To verify the parameterization of our generic nose model, several nose models are created by varying the predefined parameters. These models are presented in **Figure 23.**

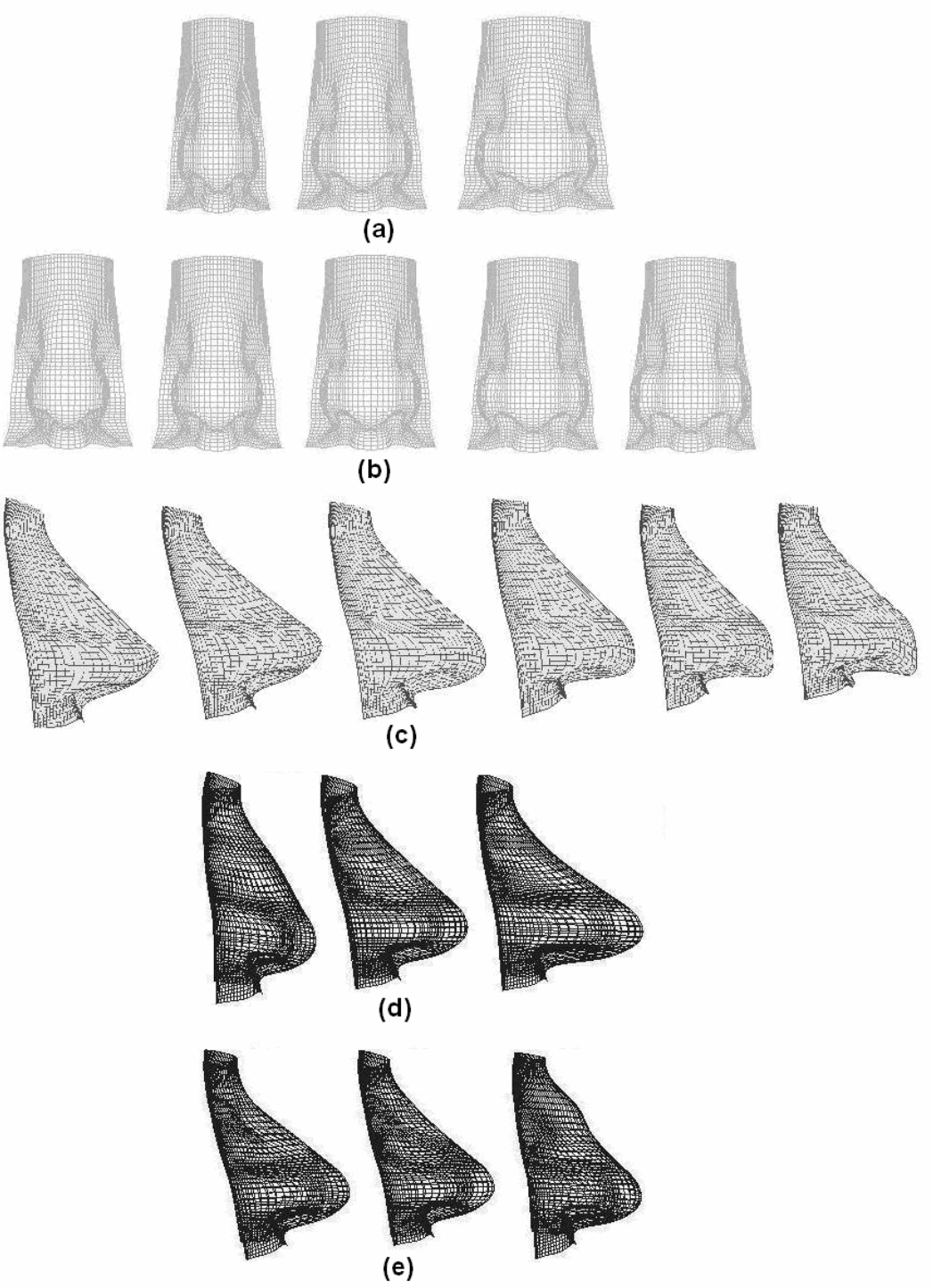

Figure 23- Nose models created from generic nose model
(a) varying width, (b) varying nostril width, (c) varying upwards-downwards orientation
(d) varying lower height, (e) varying upper height

3.6.2.2 Creation of Generic Mouth Model

Settled in the lower part of the human face between nose and chin, mouth has a simpler structure compared to nose. If the inner part of the mouth is included in the modelling process, it becomes pretty difficult to obtain a complete model. Modelling teeth and tongue is mandatory when facial animation is the purpose of a work. Since we are not concerned with facial animation and we work with surface based modelling, inner part of the mouth is not included in our generic mouth model.

Similar to the creation of the generic nose model, a raw mouth model was created by extracting the mouth region from a fitted B-spline model. Then this model was simplified and parameterized. The successive simplification of the model is visualized in **Figure 25.**

Screenshots of the generic mouth model rendered as wire-frame and as surface point cloud are presented in **Figure 24.**

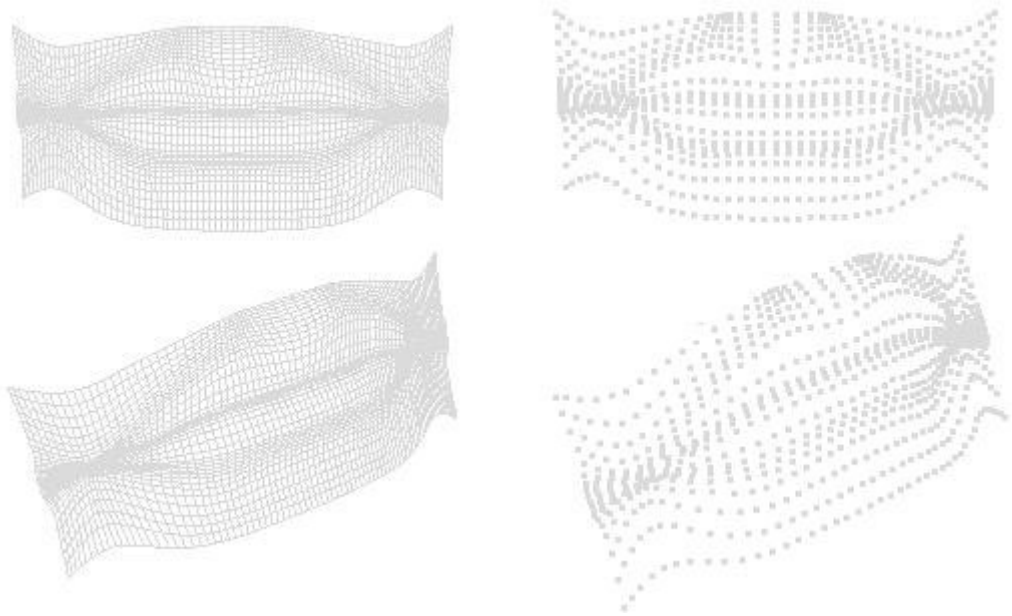

Figure 24- Generic mouth rendered as wireframe and surface point cloud

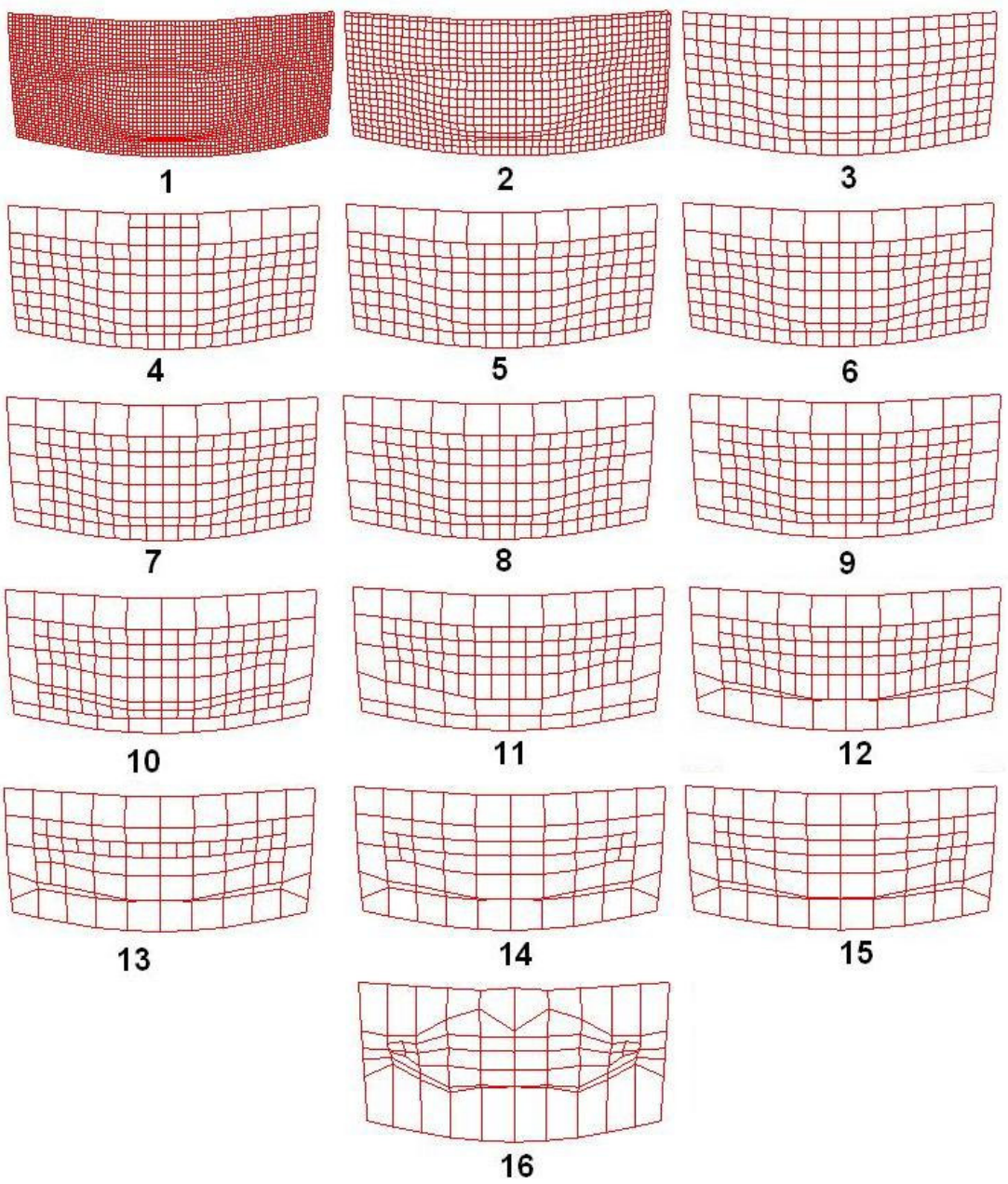

Figure 25- Generic T-spline Mouth Model Creation

Screenshots of the generic mouth model rendered as quadrilaterals with illumination are presented in **Figure 26.**

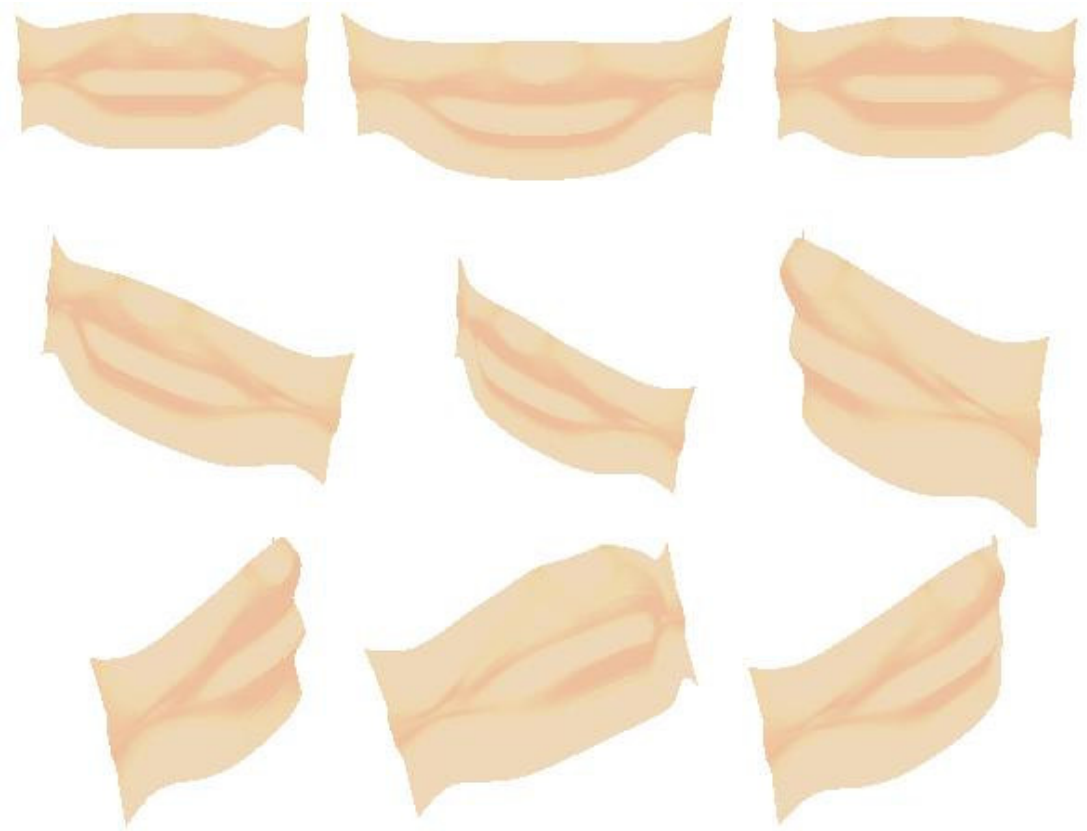

Figure 26- Generic mouth model rendered as quadrilaterals with illumination

After the creation of a visually satisfactory generic mouth model, the model was parameterized to have the ability to modify the model as desired. Characteristics of the mouth that are currently parameterized are width, upper lip thickness, lower lip thickness, upper lip inflation, lower lip inflation.

To verify the parameterization of our generic mouth model, several mouth models were created by varying the predefined parameters. These models are presented in **Figure 27.**

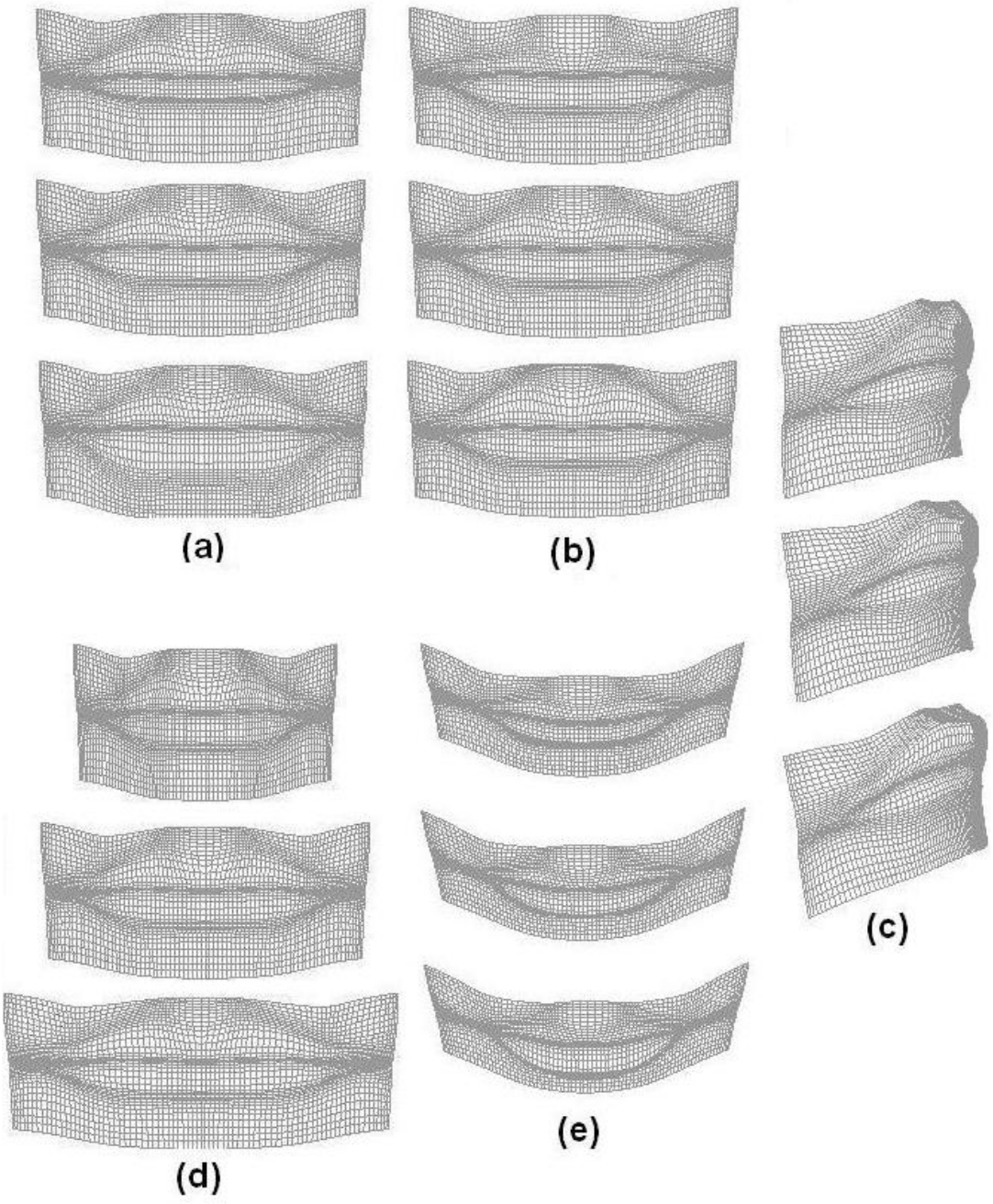

Figure 27-Mouth models created from generic mouth model

(a) varying lower lip thickness, (b) varying upper lip thickness, (c) varying upper lip inflation (d) varying width, (e) varying lower lip inflation

3.6.2.3 Creation of Generic Eye Model

Human eye is the most complex element of the human face from a modelling perspective. Presence of the eyeball, eyelid and eyelashes in a human eye makes realistic modelling of eye pretty difficult. If facial animation is aimed to allow the motion of eyes and eye lids, eyeballs should be separately modelled and then should be integrated with the surroundings. But in our work we do not aim facial animation and thus we modelled eyeball together with eyelids in a single model.

Similar to the creation of the models for nose and mouth, a raw eye model was created by extracting the eye region from a fitted B-spline model. Then this model was simplified and parameterized. The successive simplification of the model is visualized in **Figure 29** and **Figure 30.**

Screenshots of the generic eye model rendered as wire-frame and as surface point cloud are presented in **Figure 28.**

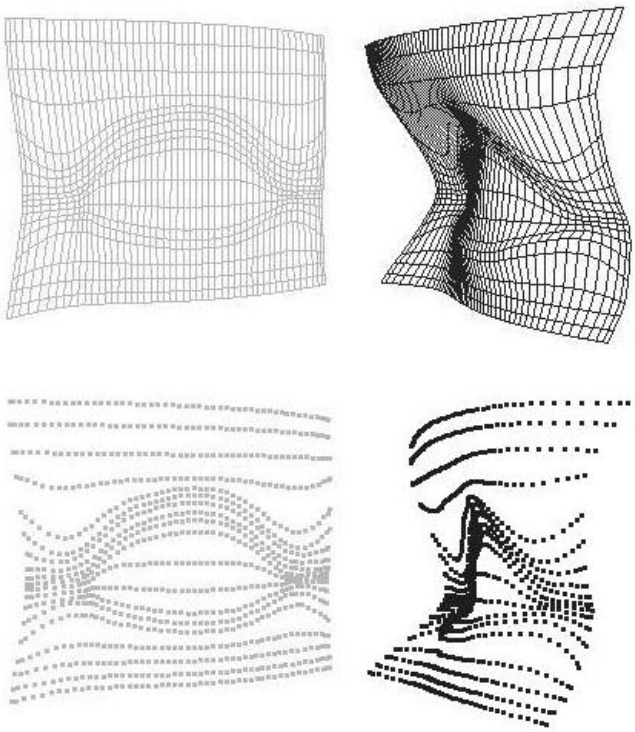

Figure 28 -Generic eye rendered as wireframe and surface point cloud

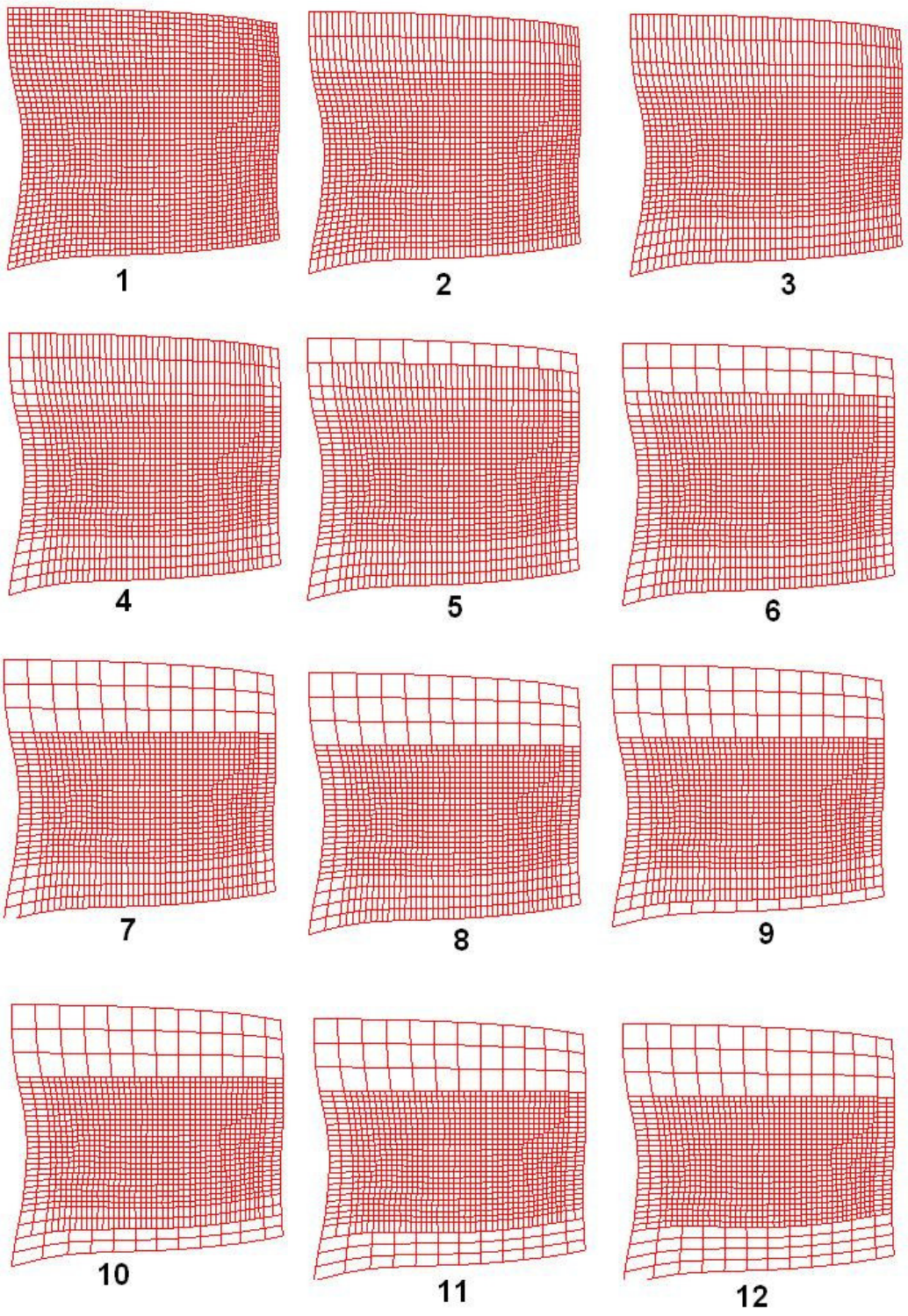

Figure 29- Generic T-spline Eye Model Creation Part 1

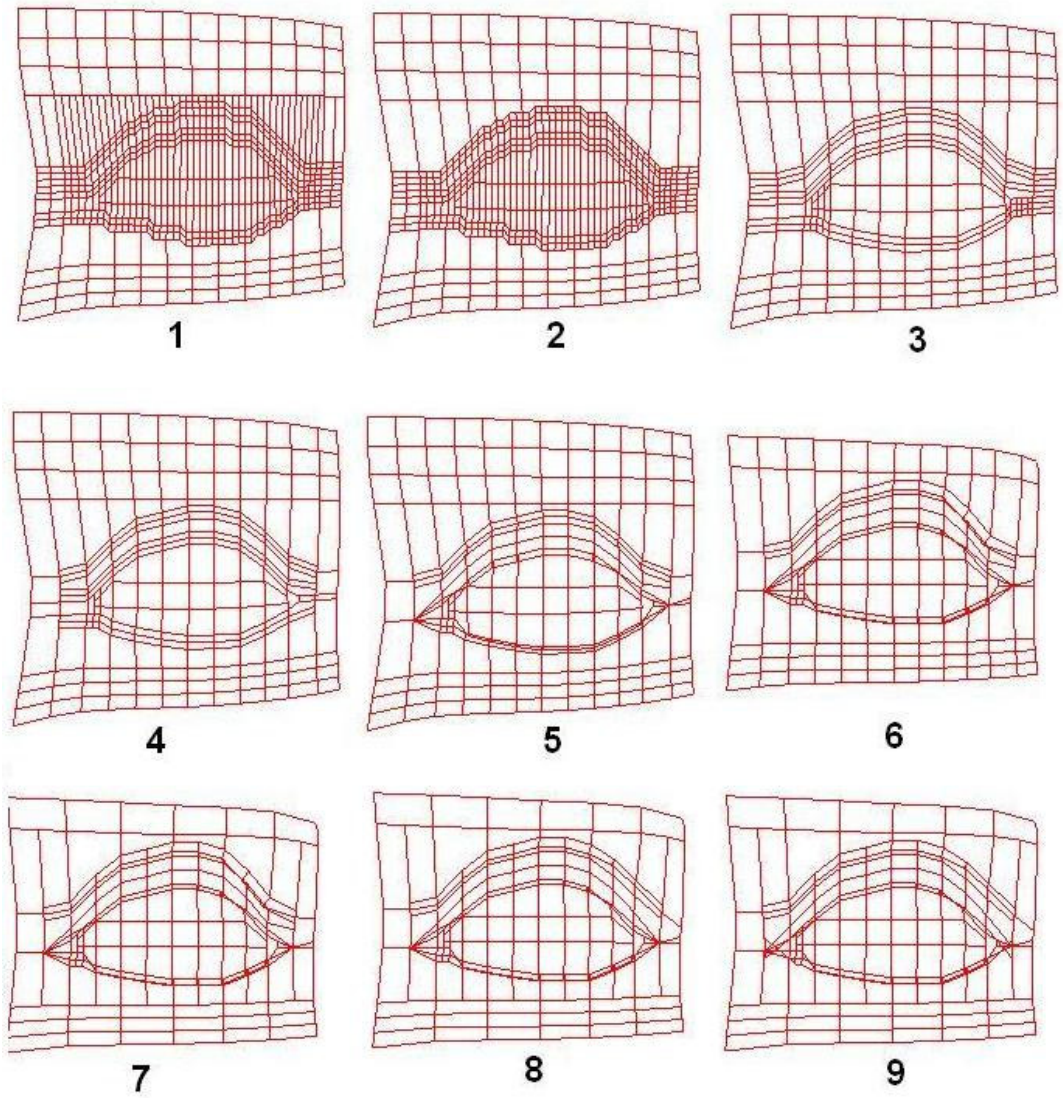

Figure 30- Generic T-spline Eye Model Creation Part 2

Screenshots of the generic eye model rendered as quadrilaterals with illumination are presented in **Figure 31.**

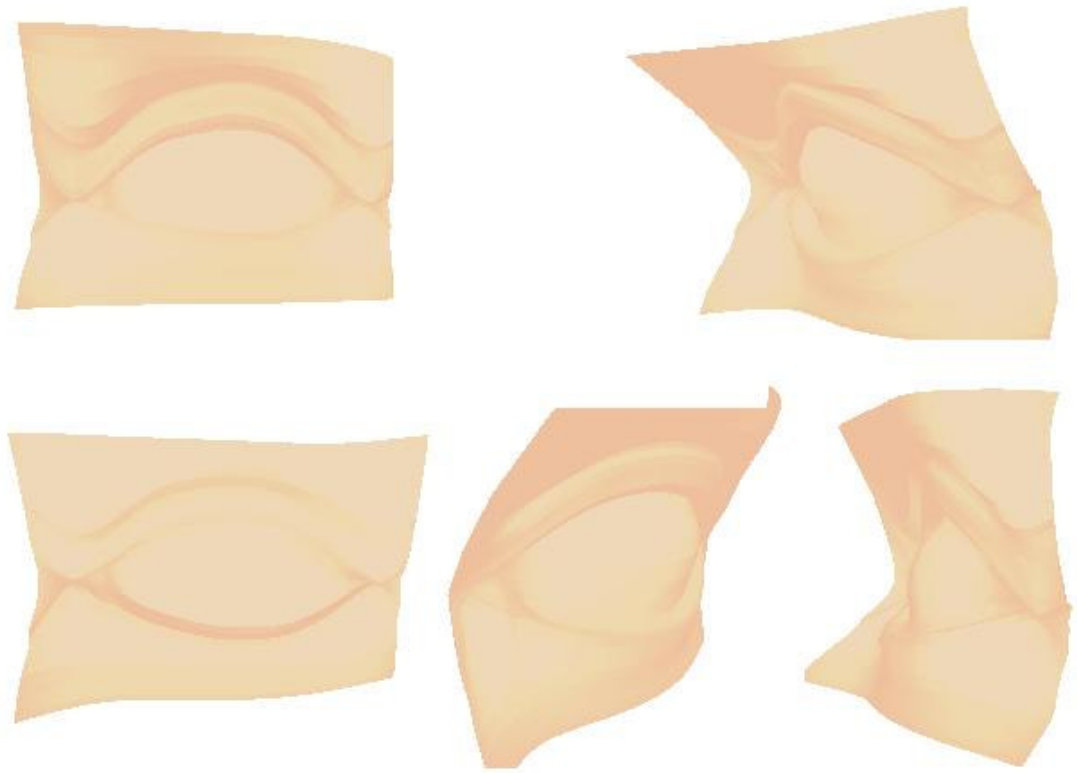

Figure 31- Generic eye model rendered as quadrilaterals with illumination

After the creation of a visually satisfactory generic eye model, the model was parameterized. Characteristics of the eye that are currently parameterized are upwards-downwards orientation of inner eye region, upwards-downwards orientation of outer eye region, upper eye curve, lower eye curve and emphasis of eyelid.

To verify the parameterization of our generic eye model, several eye models were created by varying the predefined parameters. These models are presented in **Figure 32.**

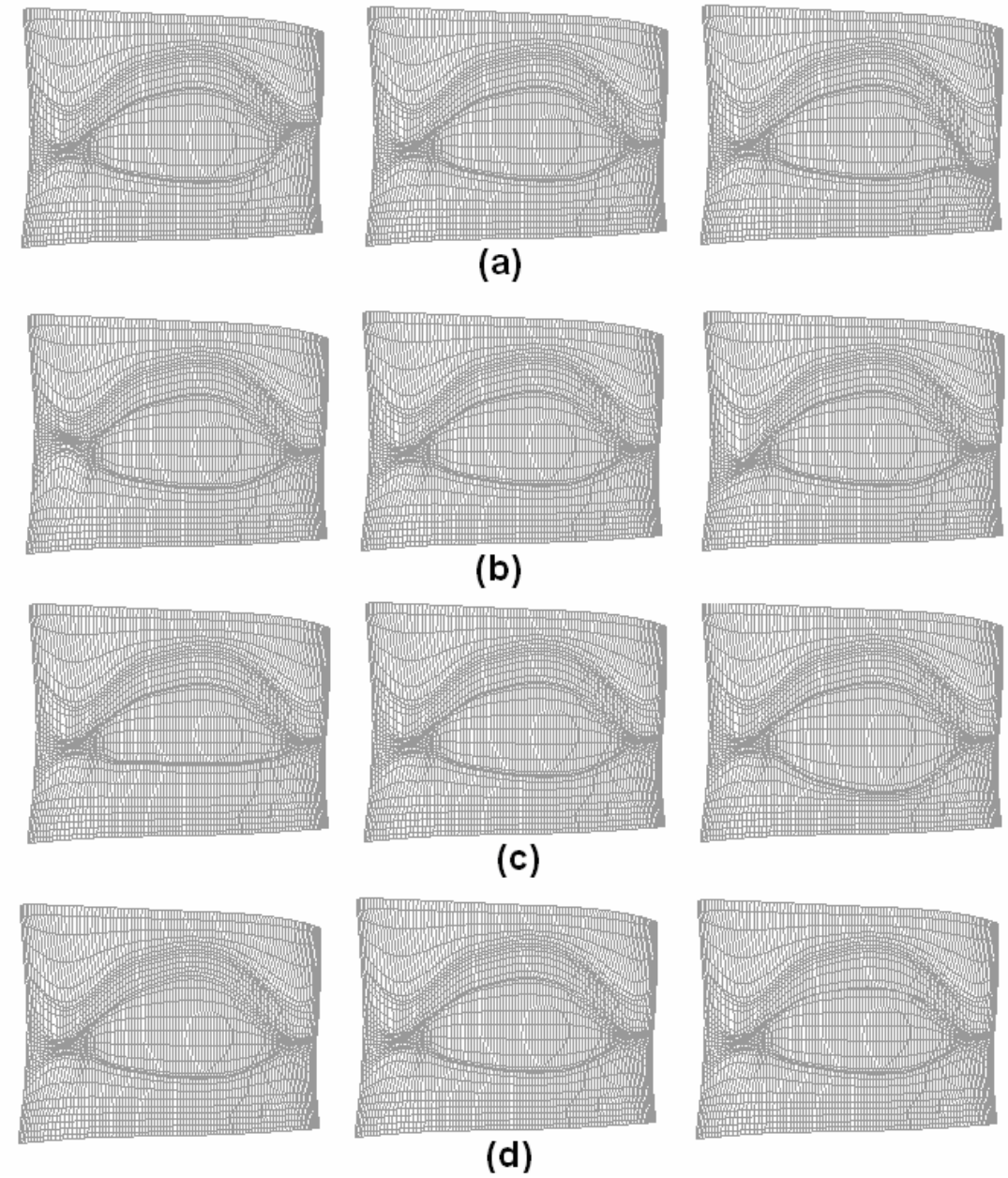

Figure 32 - Eye models created from generic eye model

(a) varying orientation of outer eye region, (b) varying orientation of inner eye region,

(c) varying lower eye curve (d) varying upper eye curve

3.6.2.4 Creation of Generic Face Model

Similar to the creation of the face component models, a generic face model was created to accommodate the face component models, again by simplifying a fitted B-spline surface. The successive simplification of the model is visualized in **Figure 33.**

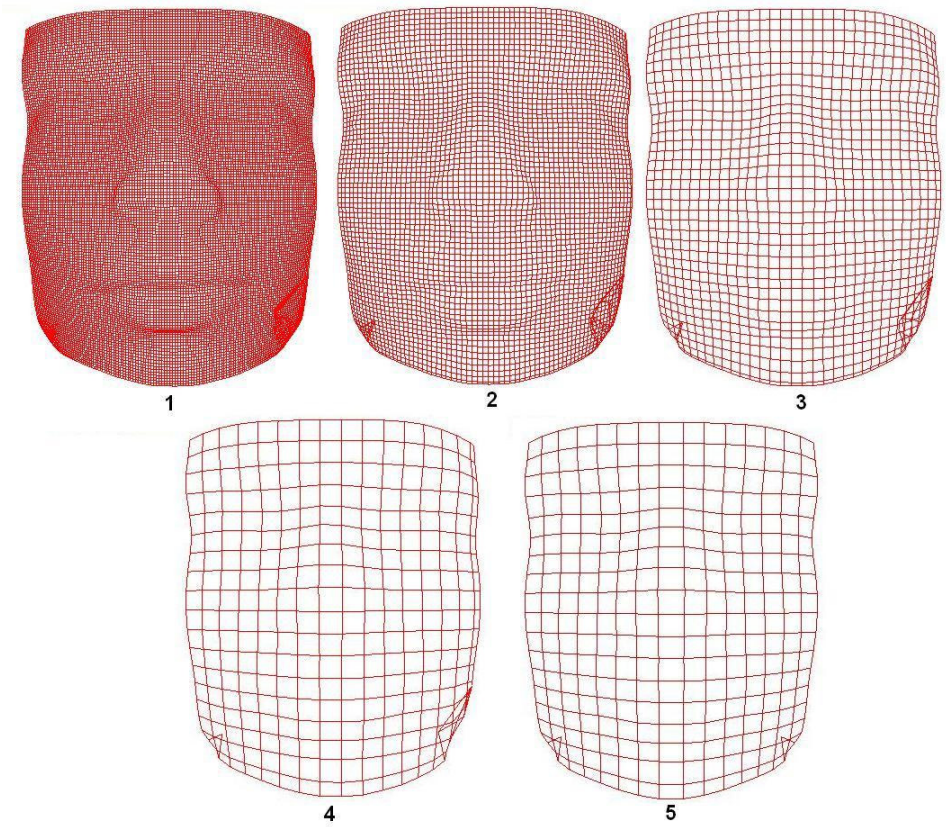

Figure 33 - Generic T-spline Face Model Creation

After the generic face model was obtained, the model was parameterized to have the ability to modify the model as desired. Characteristics of the face that are parameterized are length of the forehead, emphasis of zygomatic bone, mouth position, eye orientation, width of chin, length of chin, shape of chin. The results of the parameterization of the face model on a unified face model are visualized in **Figure 39, Figure 40** and **Figure 41.**

3.6.2.5 Merging Face Component Models with Face Model

After having generic models for nose, eye, mouth and face, these component models should be inserted into the face model. For this purpose, the region of the face model, where the component model will be inserted, should be cleared before insertion. This procedure is visualized in **Figure 34** for the insertion of nose model. For example, before the insertion of nose model, the control points lying in the nose region of the generic face model are deleted from the model. Then the boundary points of the nose and nose region of the generic face model are adjusted to supply a smooth transition between two models. In this adjustment, 3D coordinates of the boundary control points which correspond to the same knot coordinates in the models are equalized. 3D coordinates of other boundary control points are adjusted such that, there is not a sharp transition between nose and face models. The same procedure is followed for mouth and eye models and they are visualized in **Figure 35** and **Figure 36** respectively.

Figure 34 - Merging nose model with the face model

Figure 35- Merging mouth model with the face model

Figure 36 - Merging eye models with the face model

Resulting face model merged with nose, mouth and eye models, which are rendered and illuminated, are presented in **Figure 37** and **Figure 38.** The resulting model has second order geometric continuity thanks to the alignment of control points of the models according to each other.

Figure 37 -Merged face model rendered as quadrilaterals and illuminated 1

Figure 38 - Merged face model rendered as quadrilaterals and illuminated 2

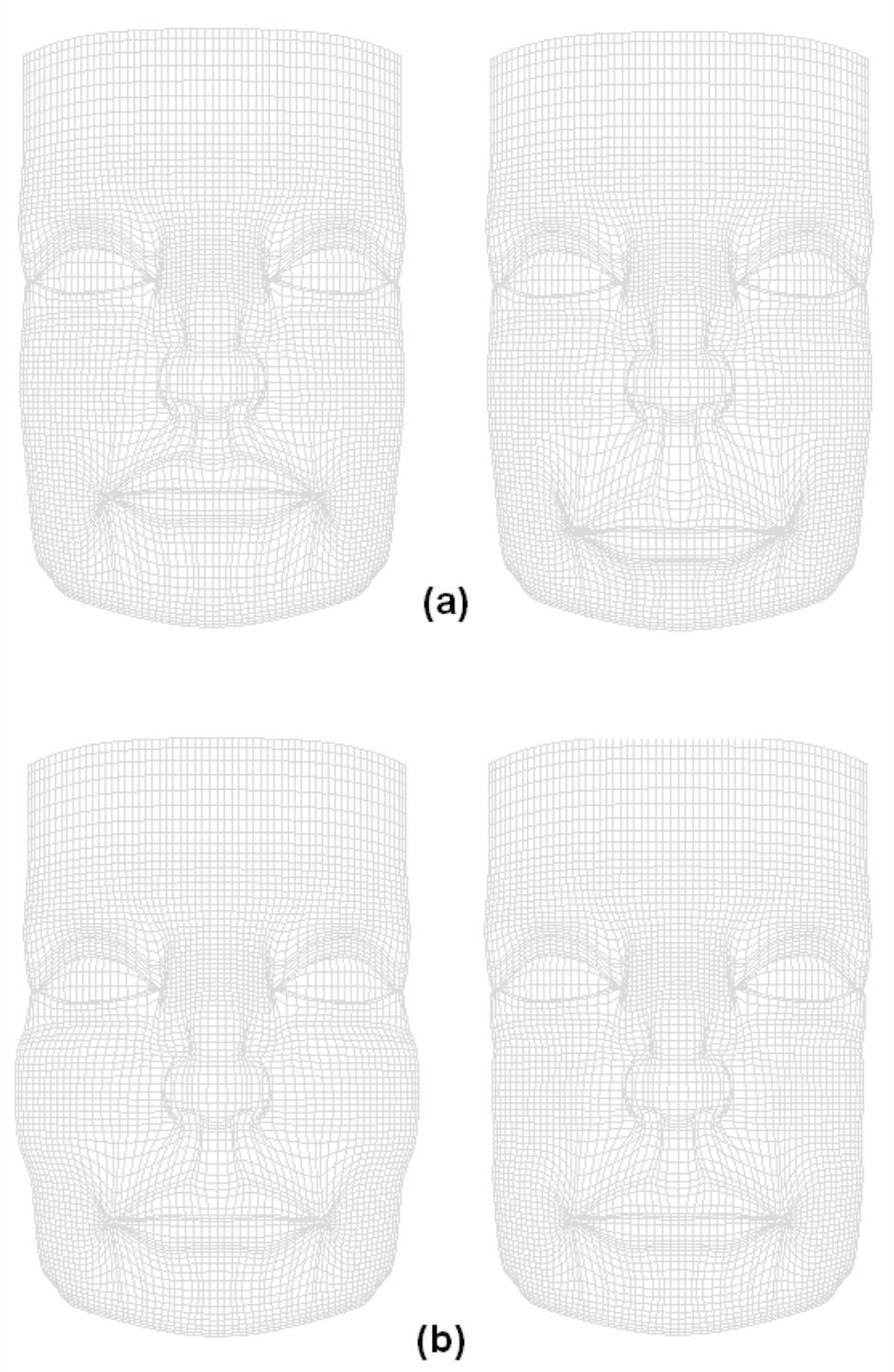

Figure 39 –Face models 1

(a) varying mouth position (b) Face models with varying emphasis of zygomatic bone

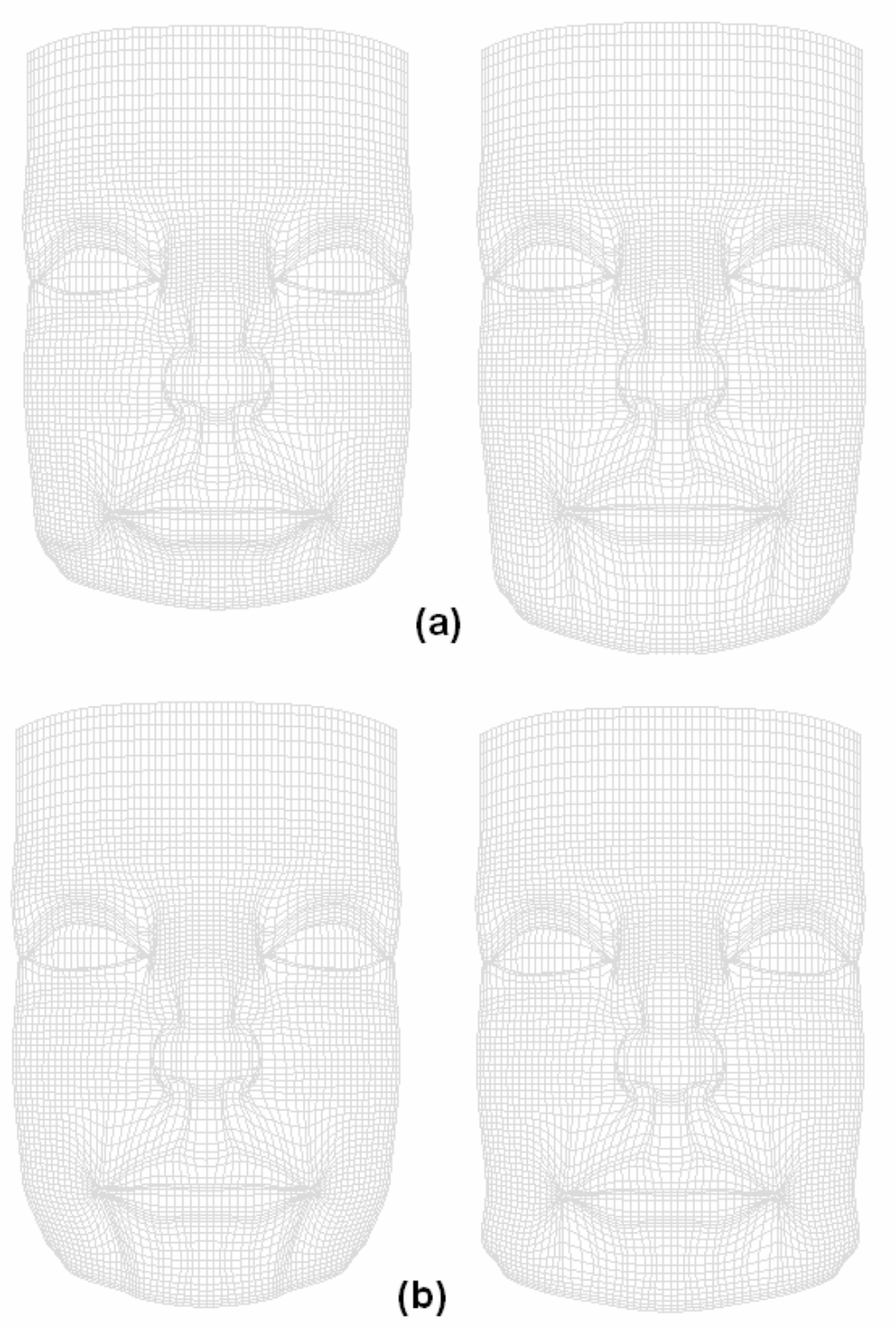

Figure 40 - Face models 3

(a) varying chin length (b) varying chin width

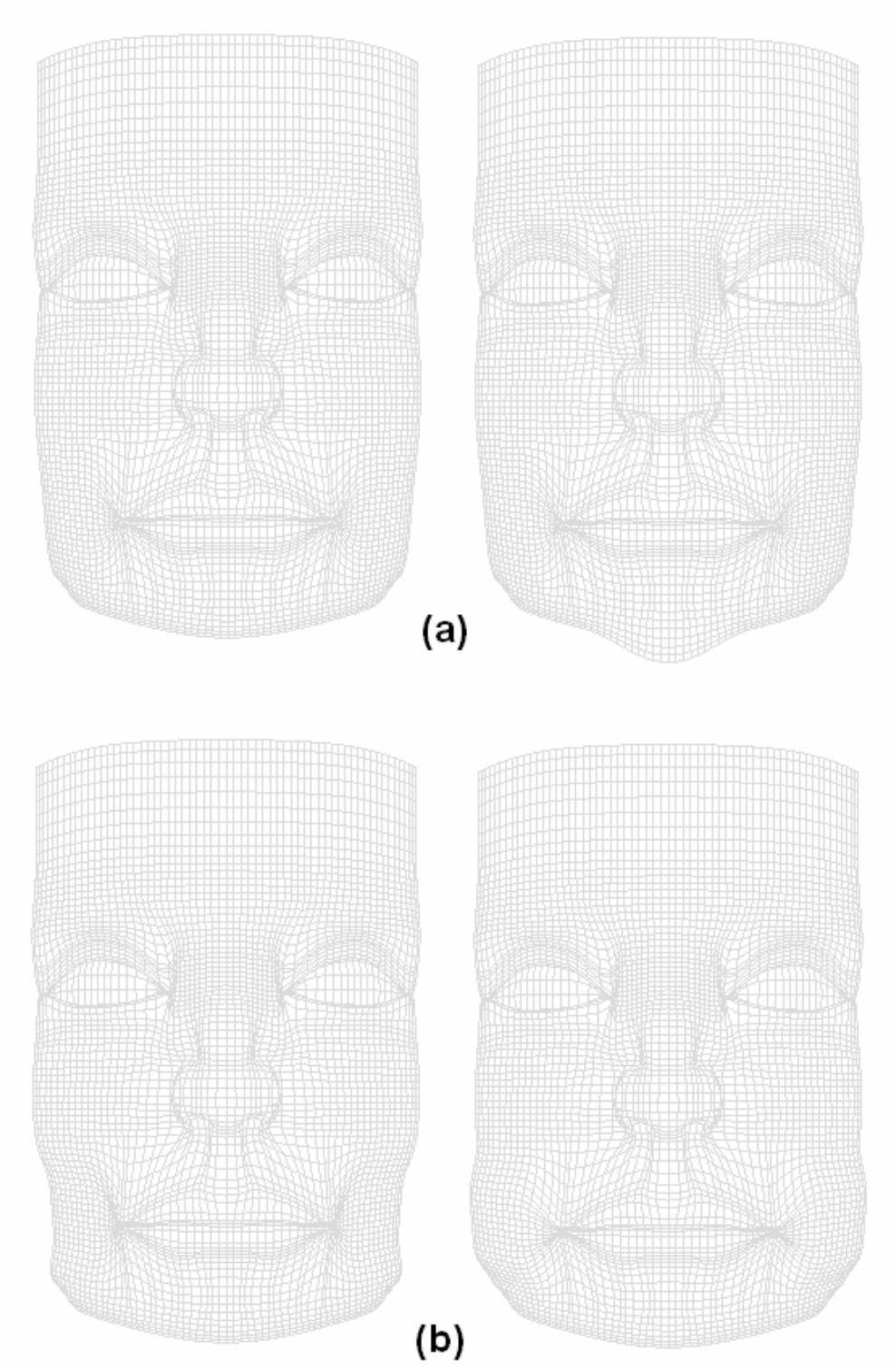

Figure 41 - Face models 3

(a) varying chin sharpness (b) Face models with varying cheek inflation

3.6.2.6 Graphical User Interface

For automation of the 3D synthetic human face modelling, a user friendly GUI was implemented. Separate GUIs were created for each generic model, where parameters of each model can be adjusted by the user and saved for future use. The modelling process starts with a complete face model where all parameters of the generic component models are set to default values and user adjusts the parameters of the generic models to define the desired face model. Screenshots of the GUI for each generic model are presented in Appendix B.

CHAPTER 4

CONCLUSION

4.1 Conclusion

In this thesis work, 3D Synthetic Human Face Modelling Tool Based On T-spline Surfaces was implemented using C++, MFC and OpenGL. The main steps and conclusions of the work can be listed as follows:

- Representation methods for solid objects in computer environment, namely Volume Representations and Surface Representations were researched. Surface Representation method was chosen for implementation, since it was found to be the simpler method and still meets the requirements of the project. The outer surface of the human face was of interest in our work, so implementation of this method prevented us from dealing with the volume information of the human face, which would require detailed anatomical knowledge. As far as observed in our work, Surface Representation is a sufficient method for 3D object modelling, if the internal structure of the object is not to be included in the model.

- Surface representation techniques, namely Implicit Surfaces, Polygonal Surfaces and Parametric Surfaces were researched. Parametric Surfaces were chosen for implementation, since they represent surfaces with comparably fewer points and modifications can be applied relatively easily on them. In

our implementation, parametric models to represent the human face surface were created and characteristics of the models were associated with relevant control points in the model. It was observed that high detailed 3D images can be produced with the implementation of parametric surfaces and it was concluded that parametric surfaces can be easily modified to define the desired model, by the modification of the positions of the relevant control points. As the disadvantage of this technique, time required to calculate the coordinates of model surface points was found to be long if a high quality surface is being rendered.

- Parametric surfaces, namely Bézier Surfaces, B-spline Surfaces, NURBS Surfaces, Hierarchical B-Spline Surfaces and T-spline Surfaces were researched. T-spline Surfaces were chosen for implementation, since they are found to be the superior approach within parametric surfaces, in terms of capabilities and topological flexibility.

  With the use of T-spline surfaces, number of control points in the models was reduced, thanks to T-junctions introduced by this scheme. It was observed that having less number of control points in a surface model is a desired characteristic, to reduce the memory cost and the processing time required to render the model surface, as well as to make parameterization of the surface easier.

  Having less topological constraints compared to B-spline and NURBS surfaces, merging different surfaces with T-spline definition is possible and this property supports the modular structure of our design. In our work, generic models of nose, mouth, eye and face were formed separately and then merged in a single T-spline surface model. With the merging capabilities of T-spline surfaces, a smooth output surface was obtained after merging.

- *Tpoint* class to represent and store control points in a T-mesh was designed and implemented.

- *Tmesh* class to represent, store and display a T-mesh, which defines a T-spline surface was designed and implemented.

- A simplified version of the T-Spline surface description formula was used. With this formula, computational and memory cost of the calculation of a T-spline surface was decreased and performance of the software was enhanced.

- A B-spline Fitting algorithm was used to obtain B-spline models of human face data, which was formed by scanning real human faces. Generic T-spline models for nose, eye, mouth and face were created by manual simplification of fitted B-spline models. With the aid of B-spline fitting in the creation of models, a satisfactory level of realism was reached in the generic models.

- Generic models of nose, eye, mouth and face were parameterized to have the capability of creating different nose, eye, mouth and face models quickly. A new nose, eye, mouth or face model can be quickly defined by a new set of parameters for the corresponding generic model.

- A user friendly GUI was implemented to cover the modelling process. A new face model can be created with the aid of the user interfaces implemented for nose, eye, mouth and face models. Parameters defined for each generic model can be set to new values and a new synthetic human face model is created by merging the models with the new parameters.

4.2 Proposed Future Work

Following further studies are suggested to improve the capability of the current modelling tool:

- New parameters may be introduced for the generic models of nose, eye, mouth and face to enhance the set of models that can be created.

- Parameters of the generic models can be associated with anthropometric data of the human face to enhance the realism of the models formed by the application.
- 3D models created by the application can be converted to an equivalent NURBS model for compatibility with 3D modelling environments which do not support T-spline surfaces but supports NURBS surfaces.
- 3D model surfaces created by the application can be saved in *.obj or another common file format.
- A face recognition toolbox can be added to the software to automatically create 3D models of a person from two facial images.
- Texture mapping can be applied on the resulting model to enhance the realism of the models. With texture mapping, face models with different eye, skin and lip colours can be achieved; visual enhancement such as eyebrows, beard and moustache can be added on the models. Also, personal attributes such as age, gender and race can be visualized in the output models.

REFERENCES

[1] Hearn, D., Baker, P., *Computer Graphics with OpenGL*, 3rd ed., Pearson Prentice Hall, 2004.

[2] Rogers, D., Adams, A., *Mathematical Elements for Computer Graphics*, 2nd ed., Mc Graw Hill, 1990.

[3] Parke, F., Waters, K., *Computer Facial Animation,* A K Peters, Ltd, Wellesley, 1996.

[4] Sederberg, T.W., Zheng, J., Bakenov, A., Nasri A.H., *T-splines and T-nurccs*, ACM Transactions on Graphics 22.3, pp 477-484, July 2003.

[5] Sederberg, T.W., Cardon, D.L., Finnigan, G. T., North, N. S., Zheng, J., and Lyche, T., *T-spline simplification and local refinement* , ACM Transactions on Graphics 23.3, pp 276-283, 2004.

[6] Forsey, D.R., Bartels, R.H., *Hierarchical B-splineRefinement,* Computer Graphics 22.4, pp 205-212, August 1988.

[7] Gabor, S., *Interactive Human Face Modelling,* Master's Thesis, Budapest University of Technology and Economics, 2001.

[8] Kahler, K., Haber, J., Seidel, H.P., *Reanimating the Dead: Reconstruction of Expressive Faces from Skull Data*, ACM Transactions on Graphics 22.3, pp 554-561, July 2003.

[9] Yang, B., Jia, P., *A Facial Expression Model for Human-Like Agent,* Tsinghua University, Beijing, 2006.

[10] Nagel, B., Wingbermühle, J., Weik, S., Liedtke, C.E., *Automated Modelling of Real Human Faces for 3D Animation,* Proceedings of the 14th International Conference on Pattern Recognition vol.1, pp 693-696, August 16-20, 1998.

[11] Valente, S., Dugelay, J.L., *3D Face Modeling and Encoding for Virtual Teleconferencing,* Proc. Workshop Very Low Bit-Rate Video (VLBV 98), Beckman Inst., Urbana-Champaign, Ill., October 1998.

[12] Kerstin, D., ed. 1999, *Human Cognition and Social Agent Technolog*y, John Benjamins Publishing Company, Amsterdam, pp 85-110.

[13] Wikimedia Commons, August 2005, http://commons.wikimedia.org/wiki/Image:Csg_tree.png, Last access:November 2007.

[14] Martin, S., 2003, U.C. Berkeley, http://stevezero.com/eecs/cs294proj1/, Last access:November 2007.

[15] Koch, R.M., Gross, M.H., Carls F.R., Büren D.F, Fankhauser, G., Parish, Y.I.H. *Simulating Facial Surgery Using Finite Element Models,* Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp 421-428, August 1996.

[17] DeCarlo, D., Metaxas, D., Stone, M. *An anthropometric face model using variational techniques*, Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp 67-74, July 1998.

[18] Vuoskoski, J., *Exchange of Product Data between CAD systems and a Physics Simulation*, Doctoral Thesis, Tampere University of Technology, April 1996.

[19] Orlans, N.M., Piszcz, A.T., Chavez, R.J. *Parametrically controlled synthetic imagery experiment for face recognition testing*, Proceedings of the 2003 ACM SIGMM Workshop on Biometrics Methods and Applications, pp 58-64, 2003.

[20]Gray, H., *Anatomy of the Human Body.* Philadelphia: Lea & Febiger, 1918, Bartleby.com, 2000.

[21] Akagündüz, E., Ulusoy, İ., *Extraction of 3D Transform and Scale Invariant Patches from Range Scans,* 2nd Beyond Patches Workshop "Patches Everywhere" Workshop in conjunction with CVPR 2007.

APPENDIX A

A.1 T-mesh file format

The beginning portion of a T-mesh file text is given below:

154 0.000000 13.000000 0.000000 11.000000

v0 -32.000000 -19.500000 -51.068260 1.000000 0.000000 0.000000 1 -2 14 -2

v1 -24.000000 -19.500000 -46.380264 1.000000 1.000000 0.000000 2 0 15 -2

v2 -16.000000 -19.500000 -44.183632 1.000000 2.000000 0.000000 3 1 16 -2

v3 -12.000000 -19.500000 -41.676472 1.000000 3.000000 0.000000 4 2 17 -2

v4 -9.000000 -19.500000 -39.105301 1.000000 4.000000 0.000000 5 3 18 -2

v5 -8.000000 -19.500000 -37.187820 1.000000 5.000000 0.000000 6 4 19 -2

v6 -4.000000 -19.500000 -36.473736 1.000000 6.000000 0.000000 7 5 20 -2

v7 4.000000 -19.500000 -36.473736 1.000000 7.000000 0.000000 8 6 21 -2

v8 8.000000 -19.500000 -37.187820 1.000000 8.000000 0.000000 9 7 22 -2

v9 9.000000 -19.500000 -39.105301 1.000000 9.000000 0.000000 10 8 -1 -2

v10 12.000000 -19.500000 -41.676472 1.000000 10.000000 0.000000 11 9 24 -2

v11 16.000000 -19.500000 -44.183632 1.000000 11.000000 0.000000 12 10 -1 -2

v12 24.000000 -19.500000 -46.380264 1.000000 12.000000 0.000000 13 11 26 -2

v13 32.000000 -19.500000 -51.068260 1.000000 13.000000 0.000000 -2 12 27 -2

v14 -32.000000 -14.500000 -52.025433 1.000000 0.000000 1.000000 15 -2 28 0

v15 -24.000000 -12.500000 -48.006184 1.000000 1.000000 1.000000 16 14 29 1

……..

The first line of a *T-mesh file* stores the number of control points, minimum and maximum *s* coordinates, minimum and maximum *t* coordinates in the T-mesh. Member variables of *Tmesh* class (*size, smin, smax, tmin* and *tmax)* represent these properties. For the T-mesh in our example file above:

size=154

smin=0.000000

smax=13.000000

tmin=0.000000

tmax=11.000000

All the lines after first line of the file are preceded with a 'v' character. This shows that this line represents a *Tpoint* object in a T-mesh. The number following the 'v' character denotes the *id* of a control point. Second line of our sample file starts with "v0", which shows that this line stores the information of the control point with *id* 0. The other numbers on a line represents member variables *x, y, z, w, s, t, nsu_id, nsd_id, ntu_id* and *ntd_id* respectively. For v0 in the example file above:

x= -32.000000

y= -19.500000

z= -51.068260

w= 1.000000

s= 0.000000

t= 0.000000

nsu_id= 1

nsd_id= -2

ntu_id= 14

ntd_id= -2

APPENDIX B

B.1 GUI of the software

GUI of our software are presented in **Figure 42, Figure 43, Figure 44** and**Figure 45.**

A 3D screen is placed on the left-hand side of the program window. Models created by the software are visualized in this screen. Operator can modify the camera position in 3D Cartesian coordinates and also rotate the displayed object by a desired angle in three axis. The camera position and angle of rotation can be reset to the original values.

The adjustments of the parameters of the generic models are carried by the menus created for each generic model, on the right-hand side of the program window. A menu is created for each of the face, nose, mouth and eye models. New models can be formed by setting the parameters of the generic models to new values by the relevant slider controls. These models can be saved with a new name and stored in the software for further use. Previously defined models can be loaded during the modelling process and a bitmap image of the model to be loaded is displayed in a small preview window. After all components of the face are defined with their parameters, these models are merged by the software to form a complete face model and output is displayed on the program screen.

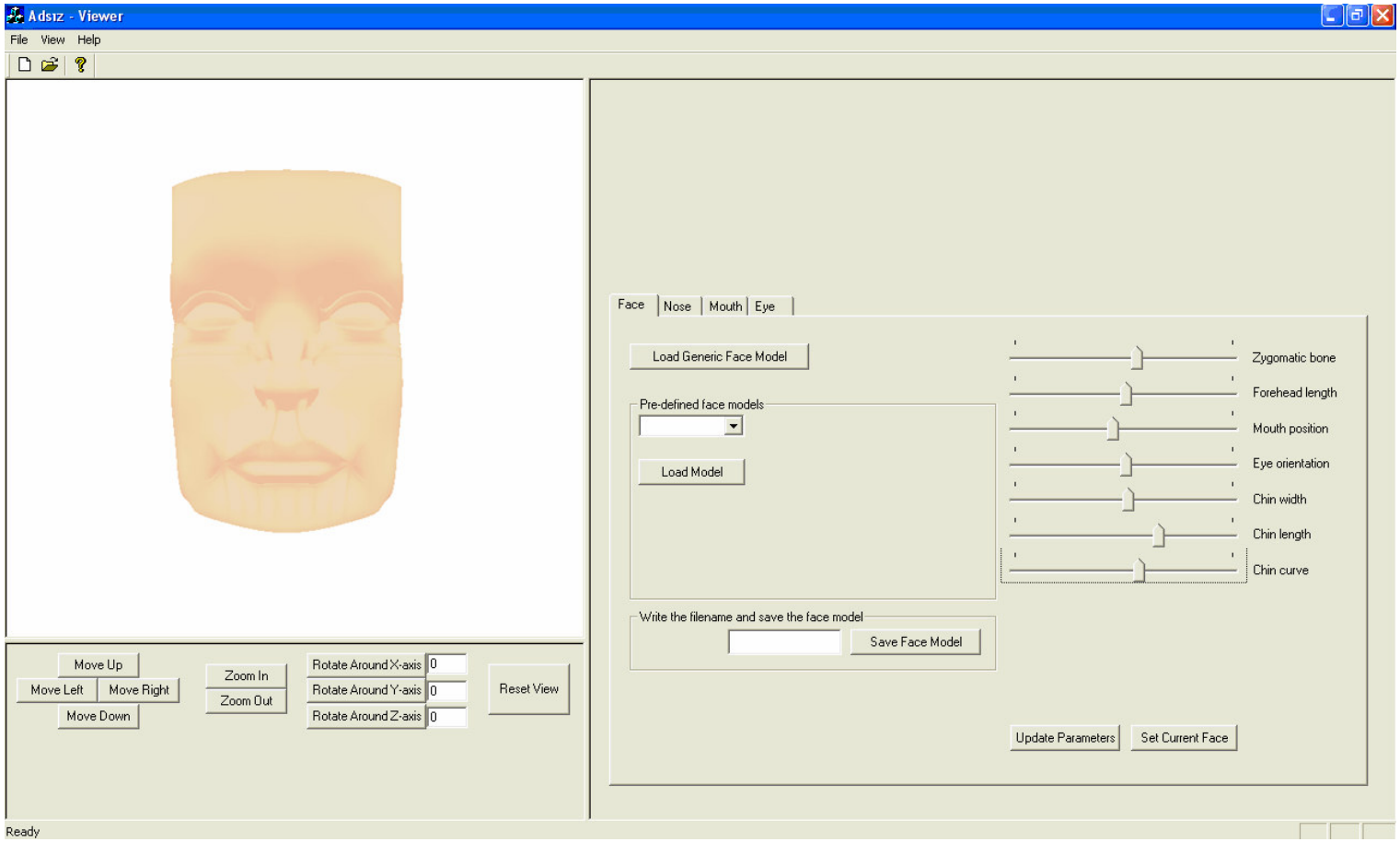

Figure 42 - GUI for face model

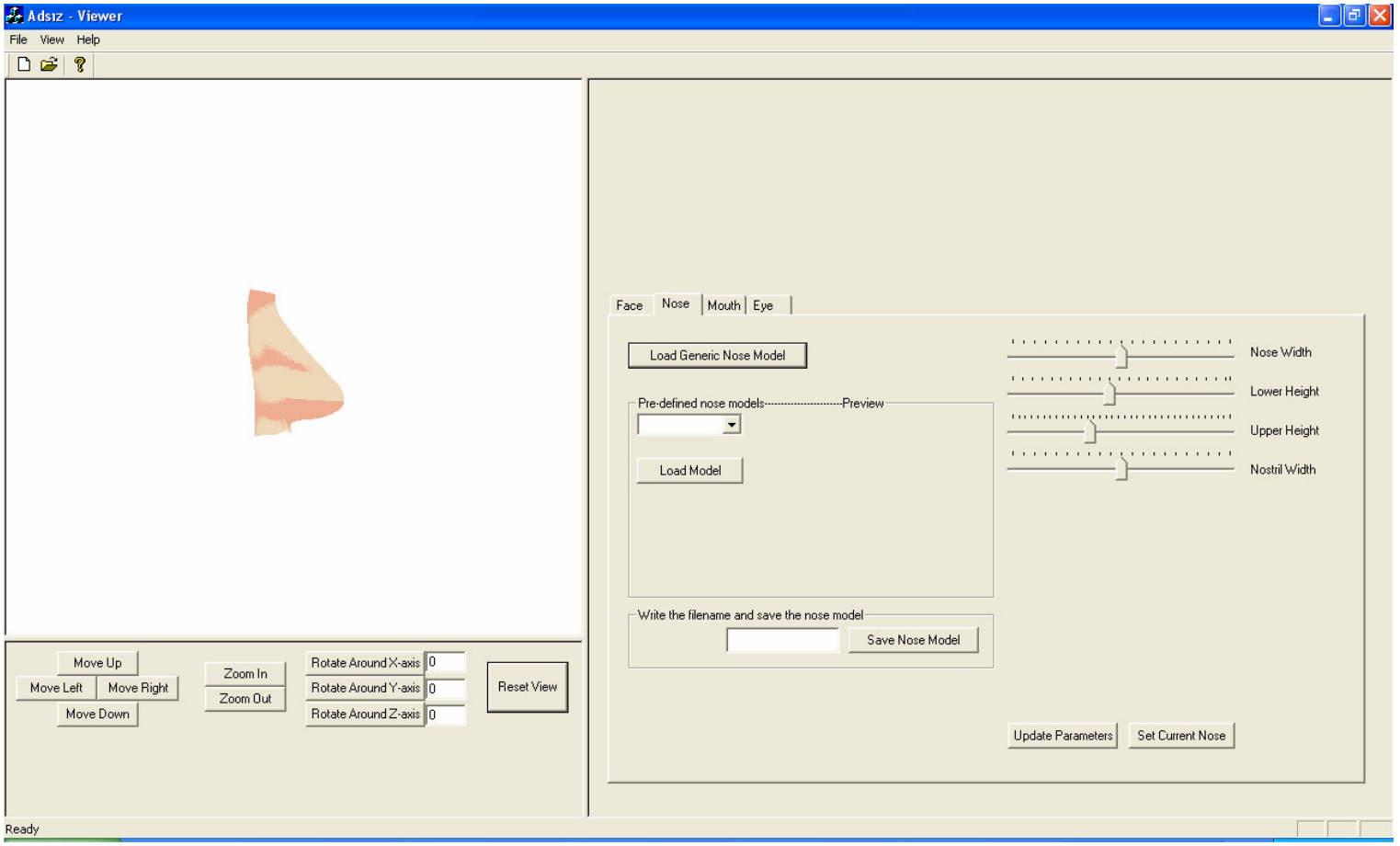

Figure 43 - GUI for nose model

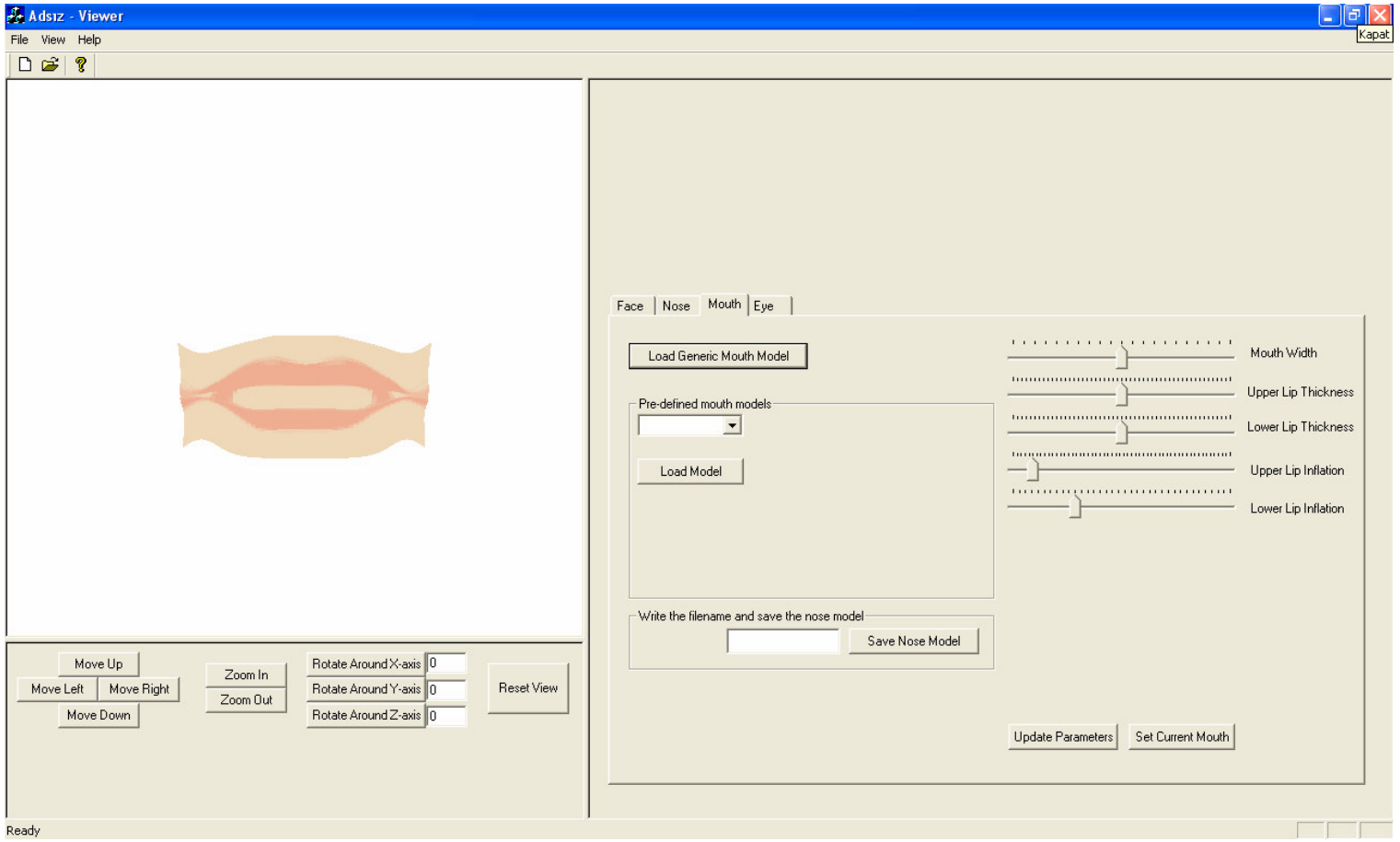

Figure 44 - GUI for mouth model

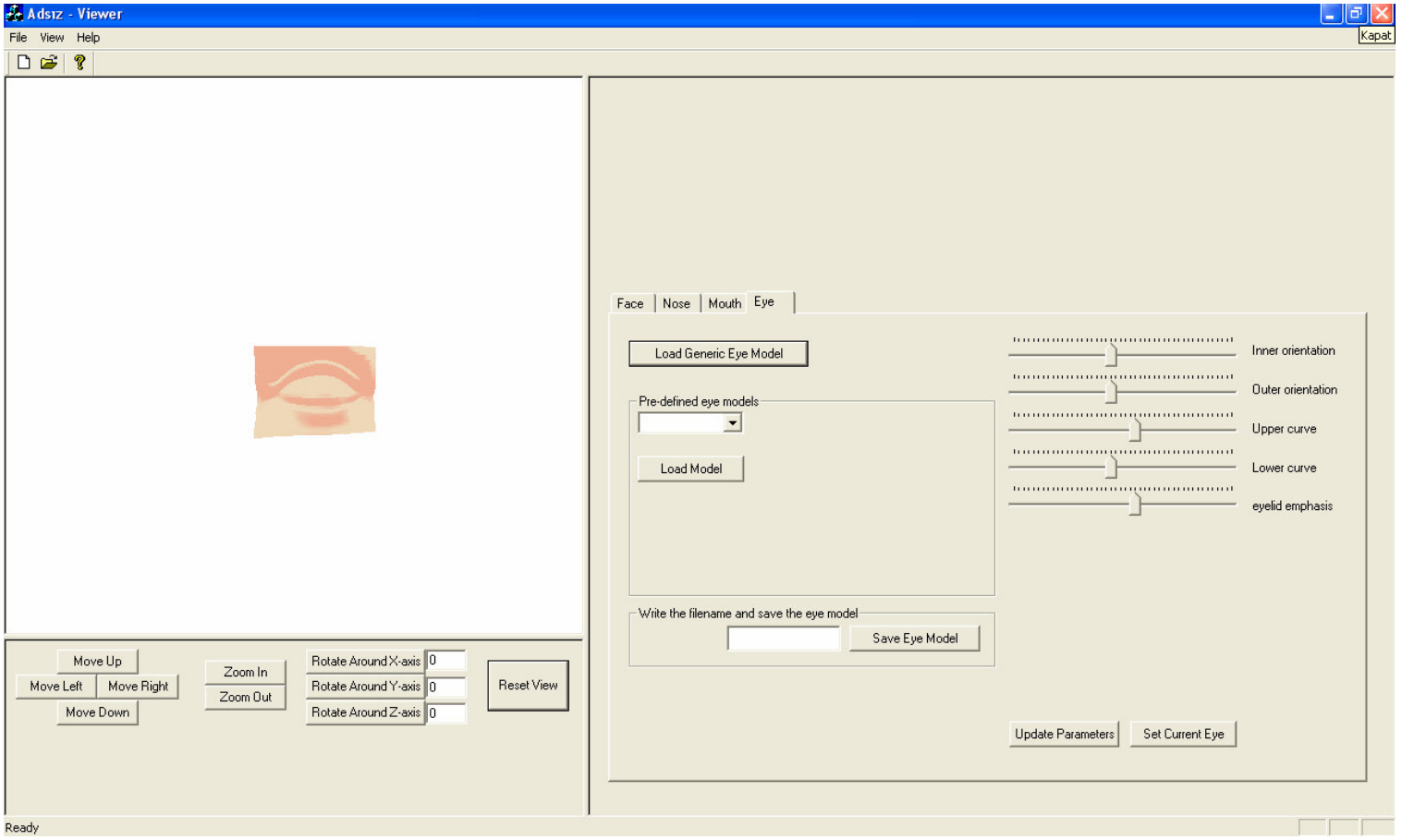

Figure 43 - GUI for eye model