TEXTURE MAPPING BY MULTI-IMAGE BLENDING FOR 3D FACE MODELS

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

BY

HAKAN BAYAR

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

DECEMBER 2007

Approval of the thesis:

TEXTURE MAPPING BY MULTI-IMAGE BLENDING

FOR 3D FACE MODELS

submitted by **HAKAN BAYAR** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and 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 the Department, **Electrical and Electronics Eng.** _______________

Prof. Dr. Uğur Halıcı
Supervisor, **Electrical and Electronics Eng. Dept., METU** _______________

Assist. Prof. Dr. İlkay Ulusoy
Co-Supervisor, **Electrical and Electronics Eng. Dept., METU** _______________

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu _______________
Electrical and Electronics Eng. Dept., METU

Prof. Dr. Uğur Halıcı _______________
Electrical and Electronics Eng. Dept., METU

Assoc. Prof. Dr. Veysi İşler _______________
Computer Eng. Dept., METU

Assist. Prof. Dr. İlkay Ulusoy _______________
Electrical and Electronics Eng. Dept., METU

Soner Büyükatalay _______________
TÜBİTAK – BİLTEN

Date: **05.12.2007**

ABSTRACT

TEXTURE MAPPING BY MULTI-IMAGE BLENDING

FOR 3D FACE MODELS

Bayar, Hakan

M.S., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Uğur Halıcı

Co-Supervisor: Assist. Prof. Dr. İlkay Ulusoy

December 2007, 67 pages

Computer interfaces has changed to 3D graphics environments due to its high number of applications ranging from scientific importance to entertainment. To enhance the realism of the 3D models, an established rendering technique, texture mapping, is used. In computer vision, a way to generate this texture is to combine extracted parts of multiple images of real objects and it is the topic studied in this thesis. While the 3D face model is obtained by using 3D scanner, the texture to cover the model is constructed from multiple images. After marking control points on images and also on 3D face model, a texture image to cover the 3D face model is generated. Moreover, effects of the some features of OpenGL, a graphical library, on 3D texture covered face model are studied.

Keywords: 3D face model, Texture mapping

ÖZ

3B YÜZ MODELLERİ İÇİN ÇOKLU RESİMLERİ BİRLEŞTİREREK

DOKUM EŞLEME

Bayar, Hakan

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Ortak Tez Yöneticisi: Yard. Doç. Dr. İlkay Ulusoy

Aralık 2007, 67 sayfa

Bilgisayar uygulamaları, gerek bilimsel gerekse eğlence sektörü gibi çok sayıda kullanım alanına sahip 3 boyutlu (3B) grafiklere geçmektedir. 3B modellerin gerçekliğini arttırabilmek için dokum oluşturma denilen bir teknik kullanılmaktadır. Bilgisayarla görme alanında, dokum oluşturmanın bir yolu, çoklu resimlerin uygun parçalarını birleştirmektir ve bu tezde çalışılan konudur. 3B yüz modeli, 3B tarayıcı sayesinde elde edilirken, bu modeli kaplayacak dokum çoklu resimlerden elde edilir. 3B yüz modeli ve resimler üzerinde kontrol noktaları işaretlendikten sonra 3B yüz modelini saracak dokum resmi oluşturulur. Ayrıca bir grafik kütüphanesi olan OpenGL'in bazı özelliklerinin dokum kaplı 3B yüz modeli üzerindeki etkileri incelenmiştir.

Anahtar Kelimeler: 3B yüz modeli, Dokum oluşturma

To My Dear Family

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Uğur Halıcı and Assist. Prof. Dr. İlkay Ulusoy for their valuable supervision, support and tolerance throughout the development and improvement of this thesis.

I would like to thank Erdem Akagündüz and Soner Büyükatalay for their support, guidance and valuable advices.

This thesis is partially supported by TÜBİTAK Research Project 105E128 entitled “3D face reconstruction where soft and hard tissue of the face is modeled by different techniques, and texture mapping of the face model”.

This thesis is completed as a part of the research going on in METU Computer Vision and Intelligent Systems Research Lab. The 3D face shape models are obtained by 3D cylindrical scanner at University of York, UK

The special thanks to my family their encouragement throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

3D reconstruction, that is extracting the 3D model (both shape and texture to cover it) of the object from images, is a hot topic of investigation and received significant attention from research communities. Its application areas vary from virtual reality, robotics applications and medical science to gaming and animation which means the field involves from industry to education, security to entertainment. After 3D models began to be used in internet and began to be displayed in low cost computers, the interest gradually increased. Surely, it demands different requirements according to usage; while animation and games require high visual quality, medical applications require accurate models. Even though it is very easy to create a simple model, complex models may need great efforts.

3D modeling is a process for developing of a mathematical representation of any 3D object in computers. Methods of representing 3D objects include mesh modeling, spline, surfaces and equation-based representations used in ray tracers. A vertex is the basic object used in mesh modeling, a point in three dimensional space. Connecting two vertices by a straight line generates an edge. Three vertices, connected to the each other by three edges, define a triangle, simplest polygon. A group of polygons which are connected together by shared vertices is referred to as a mesh**.** 3D model may be displayed as a 2D image on the computer screen via a process called 3D rendering or used for computer simulations or used for 3D printers to generate solid models. 3D model can be created automatically or manually. To create realistic representation of the object, a process called texture mapping, has been developed for applying wrapping image onto 3D model. At Fig 1.1, 3D face model, texture image to cover the face and texture covered 3D face model are shown.

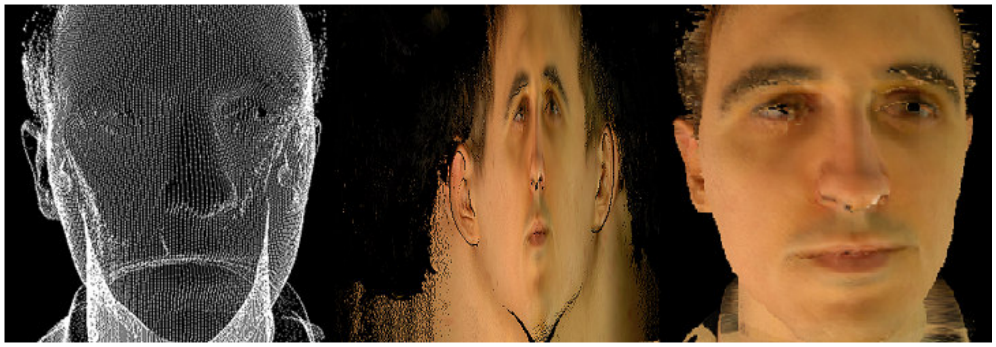


Figure 1.1 : a) 3D Face Model, b) Texture Image to Cover the Face, c) Texture Covered 3D Face Model

1.1 Literature Survey

Face reconstruction is one of the topics that has high interest due to importance of its application areas. The main approaches are recovering the shape from different techniques like stereo cameras by computing disparity map [26] [8], by illumination and shading [3], by structured light [20] [39], by morphing model [32] [36] [14] and by finding shape from silhouettes [33] [17].

In [33], a method is proposed to find the corresponding points between 3D face model and images by using silhouettes. It is stated that if the input images are given, 3D face reconstruction can be performed by estimating the shape of the face. The silhouette is only depended on shape and poses but it is invariant of light. They stated that accuracy of the model is depended on the number of cameras and priori knowledge of shape will give better performance. After obtaining 3D face model from laser-based cylindrical scanner, they construct a texture image; from a set of multi-pose input images of a human face under unknown lighting. They estimate the pose parameters and shape parameters by minimizing the difference between the silhouette of the face model and the input images. At the last step, they recover the model from illumination parameters.

The introduction of texture mapping to computer graphics started at mid-1970s and significantly enhanced the realism of models. First work in texture mapping belongs to Calmutt [4] in 1975 who demonstrate the mapping of a brick on some surfaces with un-weighted average computation. Unfortunately, textures are suffering from aliasing since they contain high frequencies. Moreover, there were many texture points to place in a small screen area. Blinn and Newell [15] refined the technique with the use of filters. They also mapped textures to known 3D shape teapot model in 1976. Feibush, Levoy and Cook [5] suggested Gaussian filters and they achieved to render synthetic texture to a house model in 1980. Gangnet, Perny, and Coueignoux [22] proposed filters in screen space rather than texture space. In 1984, Economy and Bunker [29] has worked on texture to cover ground terrain in visual flight simulator. Heckbert [23] made contributions to various transformations to map on planar and non-planar polygons.

Below are some problems with texture mapping [2].:

- Image brightness distortion: This is an artifact that caused by the use of different images acquired in different positions, with different cameras or in different time due to mainly lighting condition. Therefore in the texture covered 3D model has discontinuities and artifacts at the edges of adjacent triangles textured with different images. This artifact is suppressed in this thesis by not using triangular mapping, but instead by calculating a color mapping for each point of the 3D face model,
- Scene distortion: This is caused by wrong camera calibration or inaccurate registration or having large triangular meshes. This will lead to discontinuity in the surface of the model,
- Occlusions: Objects in front of the model might cause lack of quality in the texture image. Occlusion-free images can be generated as described in [26] [27]

Although generating the 3D model is the most critical part, texture mapping is also that important since it adds much to the realistic appearances. From the different view of the scene, pixel color can be computed by combining image information. The texture covered 3D model must be pre-computed and be independent of the viewing directions of the camera. The color information from images may be different due to the light variations or camera color balance. This will produce undesirable artifacts in texture image. To overcome this, color adjustments must be performed [7]. In general, to form a texture, a texture image is created where the points are matched with vertex positions of the 3D model. Images with different viewing directions are blended together to generate a texture image which will be mapped to 3D model during rendering.

There are additional methods that can enhance the texture quality. One of them is to use synthetic texture for the occluded parts [14] or to use artificial pixels that are randomly generated from skin-color statistics.

There are also lots of works continuing on medical science. After Computed Tomography (CT) or Magnetic Resonance (MR) is becoming available in many hospitals, there appears the demand for model guided surgery. This enables surgeons to be able to use texture covered 3D model during operations and improves accuracy, reduces operation time and improves results for the patients. A model is obtained from CT/MR data without color and texture image is extracted from images of a commercial camera within seconds. In [38], they propose a method to find control points automatically and then they performed similar steps like camera calibration and texture mapping in this thesis. Since their model doesn't contain any data that may reduce quality of the texture like hair or beard, the results are satisfactory. In [21], they tried to register images with MR data based on photo-consistency.

1.2 Scope of the Thesis

The purpose of this study is texture mapping of CT/MR data for the use of clinical diagnosis. After having CT/MR data, surgeon will take pictures of the patients and using the software, s/he should be able to see texture covered 3D face model in details in the computer. The critical thing is to have high quality of texture model and accuracy of model that is the accuracy of the texture registration. Since 3D models which are extracted using CT/MR data, are not available at the moment, 3D laser scanner data that are obtained from University of York, are used during the study. Furthermore, the techniques that are used by S. Büyükatalay in his MS Thesis [32] are also tried in this study to be able to compare results. The similarities are the number of feature points and camera calibration technique. The differences are model resolution and blending technique. In contrary of using general face model in [32], laser scanner data and multiple images of the object' face are used in this thesis.

In our case a 3D face model is created with laser scanner and images taken with an ordinary digital camera, images are blended and mapped to the surface of the

3D face model in order to have photo-realistic model. After manually selecting control points on both 3D face model and images, the interior and exterior parameters of the camera are calculated. And then, for every point on the 3D face model; a color value is blended from images according to their orientation relative to 3D face model. Front view has highest contribution to texture image since it contains almost all details of face.

The work in this thesis uses a technique to extract a composite texture map from a set of images. Multiple images taken from different view-points are blended together based on weights computed using different criteria. [32][14]

The face modeling process followed in this thesis is summarized below:

1. 3D face model is obtained by a 3D laser scanner. Its data is converted to a known file format,
2. Meanwhile, photographs of the face is captured from multiple directions which will form texture,
3. Control points are marked on both images and 3D face model manually. These points are used in camera calibration,
4. Control points normalization and camera parameter estimation are applied,
5. Texture is generated using the images considering their viewing directions.

1.3 Organization

The rest of this thesis is organized as follows: The second chapter describes control points normalization, camera calibration and calculation of 3D coordinates of control points. In addition, the texture extraction from multiple images is described in the second chapter. The third chapter describes the texture mapping software developed in this thesis, in which Microsoft Visual C++ and OpenGL library is used. The results of the thesis are given in the fourth chapter. Finally in the last chapter, conclusion and future work are explained.

CHAPTER 2

BACKGROUND

In this chapter, the necessary background is given. In the first section, camera calibration that deals with interior and exterior camera parameters is explained. In the second section, texture extraction by blending multi-images is studied.

2.1 Camera Calibration

The camera calibration problem is an important issue for computer vision. The purpose of the camera calibration is to establish a relation between 3D and 2D points. In literature, almost all studies are performed by a pinhole camera model which is explained in detail in section 2.1.2. According to this model, we need to determine two kinds of parameter:

- Intrinsic parameters, including optical and electronics properties,
- The extrinsic parameters, including rotation and translation between the camera frame and a 3D object frame.

These parameters can be determined separately by zooming or vanishing points algorithms or can be found simultaneously from point or line correspondences [12]. According to [40], in a single view system, there are 11 parameters to be determined. Since each 2D point can be used to solve 2 equations, at least 6 points are needed to solve all of them. But more points may also be used to increase precision. In [40], it is shown that after using 20 control points, relative error is decreasing very slowly. So 18 points are used in this thesis.

Camera calibration can be divided to two parts:

- Photogrammetric calibration: Camera calibration is performed by observing a calibration object with a known 3-D model,
- Self-calibration**:** it does not use any calibration object. Calibration is performed by moving the camera in a static scene.

The problem of camera calibration can be stated as: "Given a set of *n* control points in 2D images such that their corresponding 3D coordinates are known, determine camera direction for all images and find a transformation matrix with an error of transforming same control point to what from different images is below a threshold value" [19]. We can divide this problem into four sub-problems:

Assume 3D coordinates of *n* control points are known and their corresponding 2D coordinates on *m* images are also known (if a control points on 2D image is not visible, and then its value is selected as "*null*").

1. Normalize 2D and 3D points separately for preventing the effects of scaling and translation of points,
2. Calculate camera parameters for *m* images,

3. Calculate new 3D points from projection matrices of the *m* images. Calculate the distance between initially marked 3D points and new calculated 3D points. While the error is not below threshold level, go back to step 2 and find camera parameters with new calculated 3D points. (At step 2, *m* transformation matrices for m images are found. With these matrices, all 2D points from each image are transformed and combined in 3D coordinates).
4. Denormalize control points in 2D and 3D.

In [9] [24], nonlinear self-camera calibration by Levenberg-Marquardt algorithm is studied. In this thesis, linear camera calibration is applied

2.1.1 Normalization of 2D and 3D points

While doing camera calibration, all points should contribute same amount to the result. To achieve this, all of the 2D and 3D control points should be normalized. The normalization will lead to scale and translation invariance to the images and face shape model [32].

Normalization matrices $\mathbf{N_{2D}}$, $\mathbf{N_{3D}}$, are defined as follows:

For a image having *n* 2D control points, $\mathbf{d}'_i = (u_i, v_i) \quad i = 1...n$, and a model with *n* 3D control points $\mathbf{d}_i = (X_i, Y_i, Z_i) \quad i = 1...n$, the translation parameters o_x, o_y, o_z and the scaling factors s_2, s_3 for 2D and 3D are defined as follows:

$$o_x = -\frac{\sum_{i=1}^{n} x_i}{n} \quad , \quad o_y = -\frac{\sum_{i=1}^{n} y_i}{n} \quad , \quad o_z = -\frac{\sum_{i=1}^{n} z_i}{n} \tag{2.1}$$

$$\begin{aligned} s_2 &= \frac{\sqrt{2}\, n}{\sum_{i=1}^{n} \sqrt{(x_i + o_x)^2 + (y_i + o_y)^2}} \\ s_3 &= \frac{\sqrt{3}\, n}{\sum_{i=1}^{n} \sqrt{(x_i + o_x)^2 + (y_i + o_y)^2 + (z_i + o_z)^2}} \end{aligned} \tag{2.2}$$

Then the normalization matrix for 2D and 3D are:

$$\mathbf{N_{2D}} = \begin{bmatrix} s_2 & 0 & s_2 o_x \\ 0 & s_2 & s_2 o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$\mathbf{N_{3D}} = \begin{bmatrix} s_3 & 0 & 0 & s_3 o_x \\ 0 & s_3 & 0 & s_3 o_y \\ 0 & 0 & s_3 & s_3 o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The normalization matrices will be used at the end of camera calibration.

2.1.2 Camera Parameters Calculation

The pinhole camera model describes the mathematical relationship between the coordinates of a 3D point and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is described as a point and no lenses are used to focus light. The model does not include geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures. The pinhole camera model can only be used as a first order approximation of the mapping from a 3D scene to a 2D image. Its validity depends on the quality of the camera and, in general, decreases from the center of the image to the edges as lens distortion effects increases.

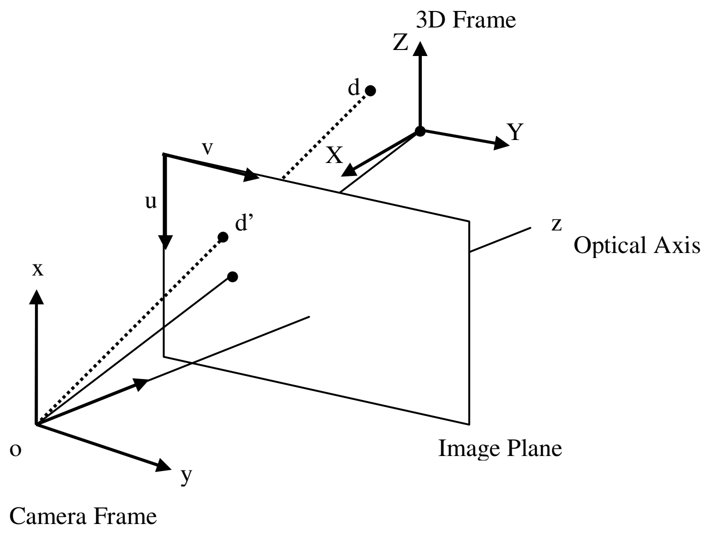


Figure 2.1 : Pinhole Camera

In Figure 2.1, pinhole camera model with center of projection o and optical axis parallel to z is shown. The image plane is in the focus so the distance from image plane to camera frame is focal length, f. A 3D point $\mathbf{d}_i = (X_i, Y_i, Z_i)$ is mapped to image plane $\mathbf{d}'_i = (u_i, v_i)$ as satisfying Eq. 2.5.

$$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y} \tag{2.5}$$

The Eq. 2.5 can be reformulated as:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2.6}$$

If the origin of the 2D image coordinates does not coincide with optical axis, we need a translation ($u_{0,}$ v_0) in Eq. 2.6. Moreover, if the camera pixels are not unit square but rectangle shape, a scale factor will be added for $m_{u,}$ m_v, pixels per mm. In u and v directions respectively. With the skew parameter, s of camera, intrinsic camera parameter matrix can be expressed as:

$$K = \begin{pmatrix} m_u f & s & u_0 \\ 0 & m_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

After normalizing control points, we can calculate camera parameters. For an ideal pinhole camera delivering a true perspective image, perspective projection from 3D to 2D can be represented in 3x4 camera projection matrix, **P:**

$$\lambda \begin{bmatrix} u \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \qquad (2.8)$$

In open form:

$$\lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} p_{12} p_{13} p_{14} \\ p_{21} p_{22} p_{23} p_{24} \\ p_{31} p_{32} p_{33} 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \qquad (2.9)$$

The projection matrix has 11 degrees of freedom and can be decomposed into the orientation and position of the camera relative to the world coordinate system (a 3x3 rotation matrix **R** and a 3x1 translation vector **T**) and a 3x3 camera calibration matrix, **K**, as given in Eq. 2.10.

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R} \;\middle|\; \mathbf{T}\right] \qquad (2.10)$$

Image coordinates of $\mathbf{d}_i' = (u_i, v_i)$ is given by Eq. 2.6:

$$u_i = \frac{X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + 1}$$
$$v_i = \frac{X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + 1} \qquad (2.11)$$

where X_i; Y_i; Z_i are the coordinates of **d** in the 3D frame; and the p_{ij} are the coefficients of the matrix **P**.

Eq 2.11 can be solved as follows:

$$\begin{aligned} X_i(p_{11}-u_i p_{31})+Y_i(p_{12}-u_i p_{32})+Z_i(p_{13}-u_i p_{33})+p_{14}&=u_i \\ X_i(p_{21}-v_i p_{31})+Y_i(p_{22}-v_i p_{32})+Z_i(p_{23}-v_i p_{33})+p_{24}&=v_i \end{aligned} \qquad (2.12)$$

The matrix **P** can be linearly estimated if at least 6 image-point-to-scene-point correspondences are provided. In the general case, for n points correspondences, one has to solve for an overconstrained system of $2n$ linear equations with eleven unknowns:

$$\underset{2mx11}{\mathrm{G}} \ \underset{11x1}{\mathrm{p}} = \mathrm{u} \qquad (2.13)$$

with $p=[p_{11};p_{12};p_{13};p_{14};p_{21}\cdots p_{33}]^T$

$u=[u_1,v_1,\ldots,u_i,v_i,\ldots,u_m,v_m]^T \quad i=1...m$ and

$$G=\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -u_NX_N & -u_NY_N & -u_NZ_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -v_NX_N & -v_NY_N & -v_NZ_N \end{bmatrix}$$

P is computed by using pseudo inverse method:

$$[\mathrm{p}] = \left[\mathbf{G}\mathbf{G}^T\right]^{-1}\mathbf{G}^T u \tag{2.14}$$

After the coefficients of the matrix **P** have been determined by using standard linear algebra techniques, one can decompose it into intrinsic parameters (matrix **K**) and extrinsic parameters (rotation matrix **R** and translation **T**):

Camera Center can be calculated as $\mathbf{Pc} = \mathbf{0}$ where $\mathbf{c} = (x/t, y/t, z/t)$

$$x = \begin{vmatrix} p_{12} p_{13} p_{14} \\ p_{22} p_{23} p_{24} \\ p_{32} p_{33} p_{34} \end{vmatrix} \quad y = -\begin{vmatrix} p_{11} p_{13} p_{14} \\ p_{21} p_{23} p_{24} \\ p_{31} p_{33} p_{34} \end{vmatrix} \quad z = \begin{vmatrix} p_{11} p_{12} p_{14} \\ p_{21} p_{22} p_{24} \\ p_{31} p_{32} p_{34} \end{vmatrix} \quad t = -\begin{vmatrix} p_{11} p_{12} p_{13} \\ p_{21} p_{22} p_{23} \\ p_{31} p_{32} p_{33} \end{vmatrix} \tag{2.15}$$

2.1.3 3D Points Calculation

After finding projection matrices for all images separately, new 3D points from the projection matrices of the *m* images are calculated and the distance between new calculated points and initial marked points are found. Until the distance becomes less than a preset threshold value, camera parameters in section 2.1.2 are recalculated with new calculated 3D points. To have a valid solution each control point must appear in at least two images.

The new 3D points from the projection matrices can be calculated as:

$$\begin{bmatrix} (p_{11}^1 - u_1 p_{31}^1) & (p_{12}^1 - u_1 p_{32}^1) & (p_{13}^1 - u_1 p_{33}^1) \\ (p_{21}^1 - v_1 p_{31}^1) & (p_{22}^1 - v_1 p_{32}^1) & (p_{23}^1 - v_1 p_{33}^1) \\ \vdots & \vdots & \vdots \\ (p_{11}^N - u_N p_{31}^N) & (p_{12}^N - u_N p_{32}^N) & (p_{13}^N - u_N p_{33}^N) \\ (p_{21}^N - v_N p_{31}^N) & (p_{22}^N - v_N p_{32}^N) & (p_{23}^N - v_N p_{33}^N) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u_1 - p_{14}^1 \\ v_1 - p_{24}^1 \\ \vdots \\ u_N - p_{14}^N \\ v_N - p_{24}^N \end{bmatrix} \qquad (2.16)$$

The new 3D points can be found by pseudo inverse method as in Eq. 2.14

2.1.4 Denormalization of 2D and 3D points

Using inverse of the matrices given in Eq 2.3 and 2.4, the control points are converted to the original scale.

2.2 Texture Mapping

Texture mapping has become very popular in the computer vision area in the last decade because it is simple and enhance the realism of the models. Texture mapping can be divided into two steps:

i) Texture extraction, that is to generate a texture image from images,

ii) Transformation of the texture to screen space, rendering.

The texture extraction is to compute a texture color from images by recovering viewing parameters for each point on the surface of the 3D face model. Any point in the 3D face model may appear in more than one image; therefore a method should be applied to blend these points. There are two ways to combine these values:

i) View-independent blending. It results in a texture map that is used to render model from any viewpoint,

ii) View-dependent blending. The blending weights are changed according to the current viewing point. Rendering takes longer with view-dependent blending, but the resulting image has higher quality.

The texture value **T**(p) at each point *p* on the model can be found as a weighted combination of corresponding color of 2D image:

$$\mathbf{T}(p) = \frac{\sum_i k_i(p) C_i(p)}{\sum_i k_i(p)} \qquad (2.17)$$

C$_i$(p) is the color data from *i-th* image and **k**$_i$(p) is the weight value for that image. The texture value is calculated for all pixels of the model.

As stated at [11], there are different properties of texture extraction,

- Self-occlusion: **k**$_i$(p) should be zero if *p* is invisible at that image,
- Smoothness: if the weight function varies smoothly, it will prevent artifacts in the texture that caused by blending different images,

- Positional certainty: $\mathbf{k}_i(p)$ should depend on the "positional certainty" of *p* with respect to the *i-th* image. The positional certainty is defined as the dot product between the surface normal at *p* and the *i-th* image direction of projection,
- View similarity: it is related to view-dependent texture mapping, beside dependency on the *i-th* image direction of projection**,** $\mathbf{k}_i(p)$ should also depend on the angle between the direction of projection of *p* onto the *j-th* image.

Authors may use any or all of the properties above according to their interest. For example, while Kurihara and Arai [35] use positional certainty, they do not care about self-occlusion. Akimoto [34] and Ip and Yin [13] blend the images smoothly, but they didn't mention about self-occlusion and positional certainty. Debevec [26], who worked on view-dependent texture mapping buildings from photographs, studied occlusion but not positional certainty. In this study, view-independent blending is used and positional certainty is taken into account.

The second step of texture mapping is transforming texture image to screen coordinates. There are several ways to do that like orthogonal projection, perspective projection.

In addition to texture mapping, some techniques have been developed like illumination - based shading effects, shadows, reflection, refraction, motion blur, and lens defocus.

2.2.1 Face Modeling Using Multiple Images

In [32] which is a master thesis conducted in METU Computer Vision and Intelligent Systems Research Laboratory, face model generation by morphing an initial 3D face model with uncalibrated 2D images is studied. Manually marked feature points on 2D images are used to deform initial 3D face shape model. The same steps in section 2.1 are followed with 100 iterations. As a result a transformation matrix is found. The next step is deforming a generic 3D face shape model considering the coordinates of the 3D feature points found in the iteration phase. After deformation, if results are not satisfactory, then more new control points can be added to the initial images for refining the model. The aim of the refining step is to specify more control points which are specific to the current face. The last step is forming of a texture image to wrap the final 3D face shape model. An initial texture map that defines the wrapping of texture image to the 3D face shape model is used. Deformed 3D face shape model is projected to each image considering camera position and a new texture image is formed by combining triangles from the images with respect to the projected 3D coordinates.

In this thesis, the selected control points and number of 2D images are the same as those in [32]. In addition, the camera calibration is performed in the same way with [32] except the number of iterations is one instead of hundred. The deformation step is not needed for this study. Since the 3D face shape model and 2D images are belonged to same person. But for extracting the texture image, a different approach is applied. In [32] a texture image is obtained by copying triangular areas from the most suitable images. However in this thesis, for each vertices of the 3D shape model, color values from the images are blended. This approach leads to a more smooth texture image and gives a better appearance to the model

At first, surface normals are calculated for every triangle in the 3D face model. Then, in order to obtain more realistic appearance, vertex normals are calculated from surface normals. The camera directions were already calculated as explained in section 2.1.2. So by computing angle between vertex normals and camera direction, we can determine which image has better view for that particular point. A texture is mapped from all images with their contributions angle.

$\mathbf{k}_i$(p), the weight value in Eq. 2.17 is found as follows:

$$k_i(p) = \mathbf{n}_v(p) \bullet \mathbf{n}_i \qquad (2.18)$$

where $\mathbf{n}_v(p)$ is the vertex normal for point p and $\mathbf{n}_i$ is the direction of *i-th* image. "●" is the dot product.

The differences between [32] and this thesis are:

i) Since shape and images belongs to the same person, only one iteration is enough to compute projection matrix,

ii) The shape deformation is not needed,

iii) None of the shape transformation like affine is used. Since pixel color from 2D is transferred instead of triangular areas from images,

iv) Models with more vertices are used (261760 vertices instead of 800 vertices).

CHAPTER 3

TEXTURE MAPPING SOFTWARE

3.1 Introduction

At the previous chapter, theoretical aspect of texture mapping is described. This chapter gives information about the developed 3D texture mapping software.

This application is developed with MS Visual C++ 6.0 with linking OpenGL 2.0 as 3D graphics library and Matlab 6.5 as math library. The detailed explanation of compiling the developed program is given in the Appendix-II.

3.2 3D Texture Mapping Software Interface

In this thesis, the outputs of the laser scanner at the University of York - UK, are used for generating the 3D face model. The output of the laser scanner is in cylindrical coordinate system (r, Φ, z) as a point cloud 1024 pixels in z (height) direction and 480 pixels in Φ (cylindrical angle) direction with equally sample distance. These points are converted to the well known 3D data format, “obj” with Matlab (The details of the “obj” format are given in the Appendix-I) and then triangles are generated. Since laser scanner output is a regular output pattern, it is not hard to generate triangles from point clouds.

At the moment only three models are available. Although 1024 x 480 = 491.520 sample points are obtained from laser scanner, among these data 261760 vertex and 485711 meshes are valid for the first model. The second and third models are composed of 293255 vertices and 566366 triangles; 227547 vertices and 420781 triangles respectively.

For generation of the texture, the digital camera, NIKON D200 with 1162x778 resolution is used and images taken without any calibration from five different directions.

The program is designed to work on obj format data of any size. After opening 3D data, a scaled version will appear at the screen. User can easily translate, rotate or scale 3D face model. Additionally, functions to select points on the model and save them to a text file are implemented. 3D model can also be seen in predefined x, y and z direction in 2D or view it in perspective or orthogonal 3D space. There are three kinds of displaying model in 3D on the screen as solid, wire and point. (Figure 3.1 to 3) Moreover, user can see 2D images, with/without control points and projection of 3D model. Furthermore, texture covered 3D face model can be viewed from different directions.

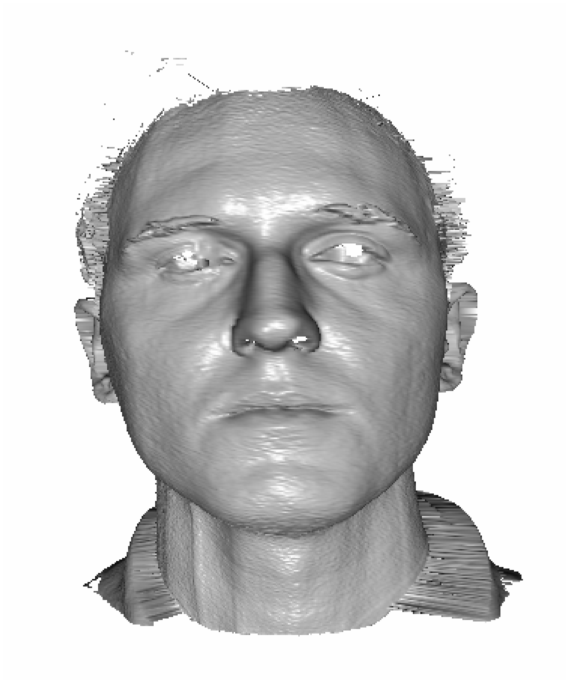


Figure 3.1 : Solid View of 3D Model

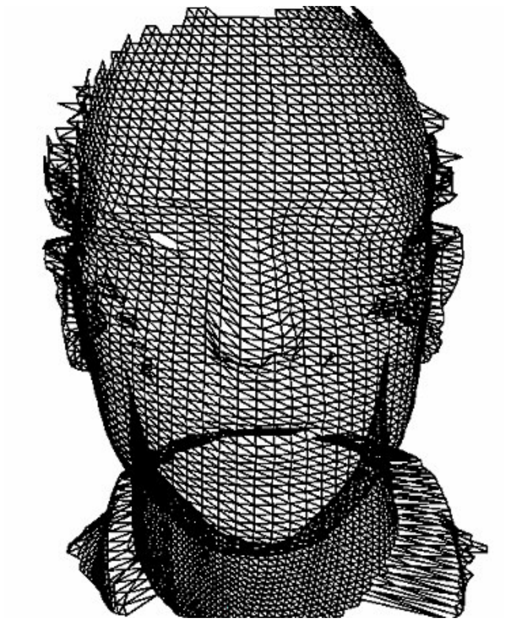


Figure 3.2 : Wire View of 3D Model

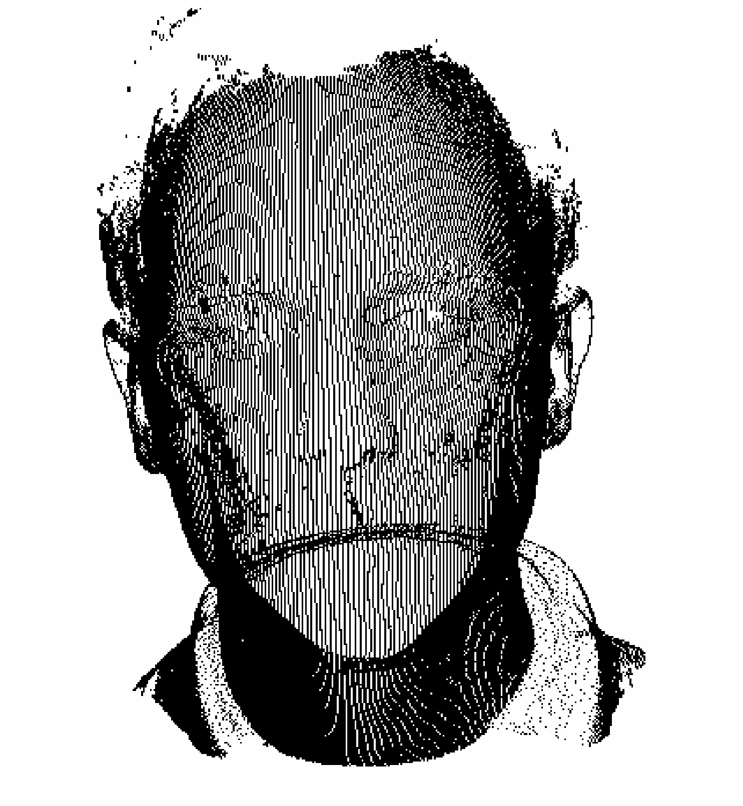


Figure 3.3 : Point View of 3D Model

The first step of the texture mapping software is manually marking the control points on both 3D face model and images. There exist 18 control points on the model. This number is selected to be similar to [32] and as stated [40] that using more than 18 points will not increase accuracy of the calibration. These points are ear up and ear down for left and right ears; eye in and eye out for left and right eyes; mount up, mount down, mount left and mount right; nose tip, nose left, nose right and nose saddle; up-middle head and down-middle of head. This step will take few minutes and the only user interactive part of the study. In Figure 3.4 and 3.5, marking in 2D image and 3D model is shown respectively.

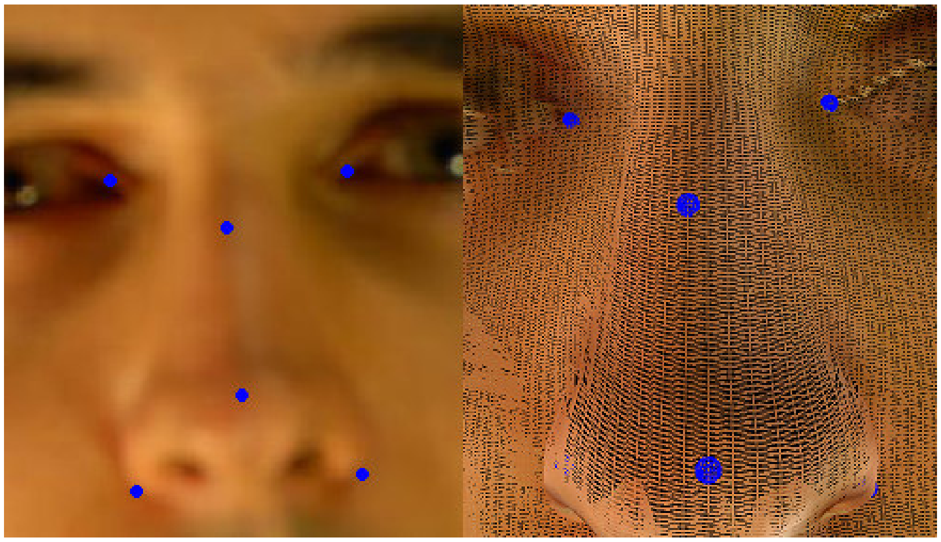


Figure 3.4 : Marking Control Points in 2D and in 3D (Nose and Eyes)

After selection of control points, camera calibration is performed and camera parameters are calculated.

To calibrate camera, for each image, there should be at least six of control points and each control point should be visible at least in two images. For this study, since shape and images belongs to the same person, only one iteration is enough to compute projection matrix. More than one iteration will destroy projection of the model onto images. Moreover, 3D face model modification is not necessary. Then, all 3D points are projected onto 2D image to get the appropriate color information.

Finally, color information is mapped to 3D model. Some parts on the 3D texture model haven’t obtained color information from the 2D images. This is caused by not matching viewing directions with the camera and these parts are covered with the skin color. The obtained results are presented in Chapter 4.

CHAPTER 4

RESULTS AND COMPARISONS

4.1 The Results Obtained by the Texture Mapping Software Developed

In Figures 4.1 to 12, five original images, 3D face model with control points, 2D image with projected model, zoomed 2D image with projected model from five different directions, 3D face model in point form and texture covered 3D face model from five different direction are shown for the first model. In Figures 4.13 to 24 and Figures 4.25 to 36 are the same images for the second and third model respectively.

The results are quite satisfactory since the shape and images belongs to same person. So there is almost perfect match between image and projection of the model onto that image. Despite the forehead or cheek of the head gives good match and texture, there are some artifacts at down part of the nose, at hair and eyebrow. The reason for that there is no available texture with defined surface normal between image and model. In other words, the surface normal is changing so rapid that camera directions is not available at that direction. Due to the same reason, the third model with long hair has very serious artifact at forehead at the front view.

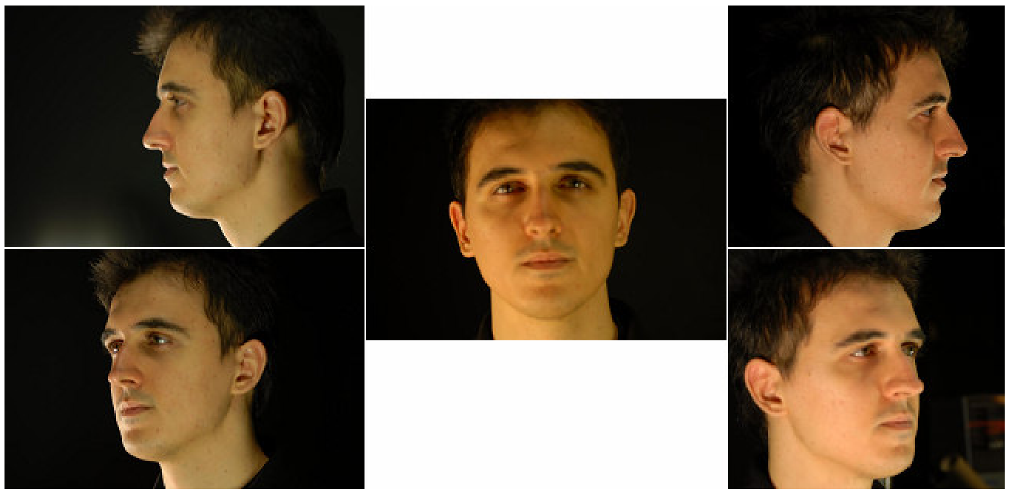


Figure 4.1 : Input Images

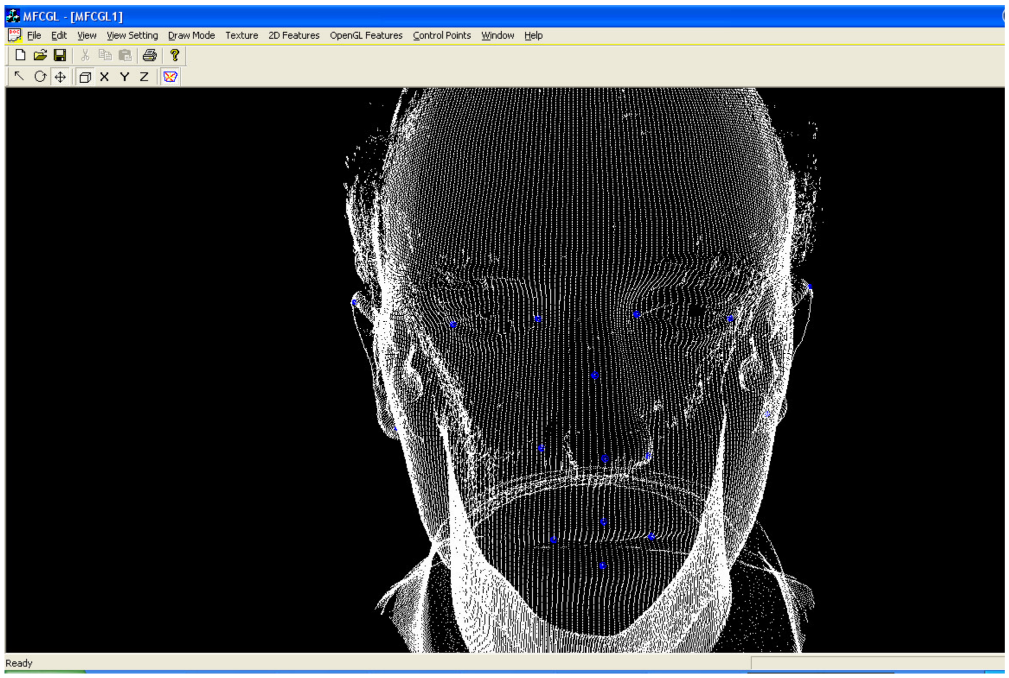

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.2 : 3D Model with Selected Points

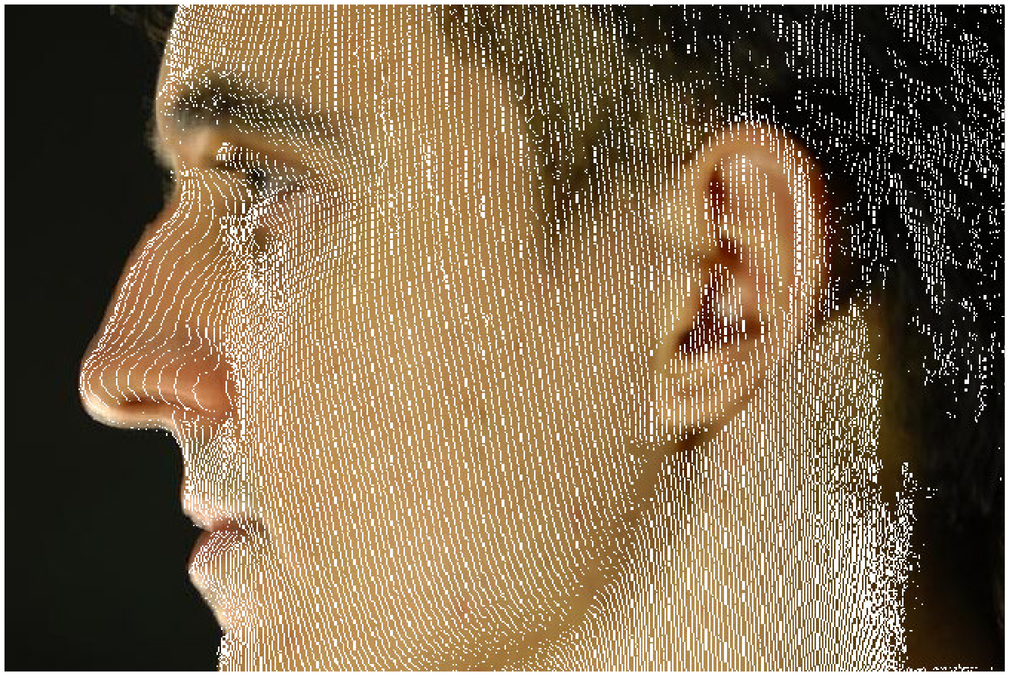


Figure 4.3 : Projection of 3D Points on Left View (Zoomed)

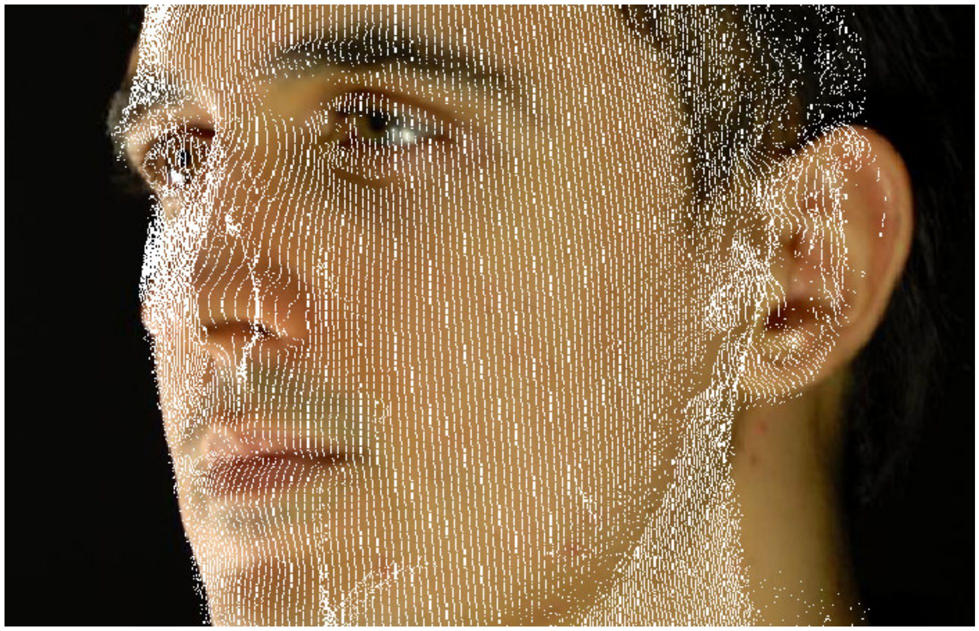


Figure 4.4: Projection of 3D Points on Front Left View (Zoomed)

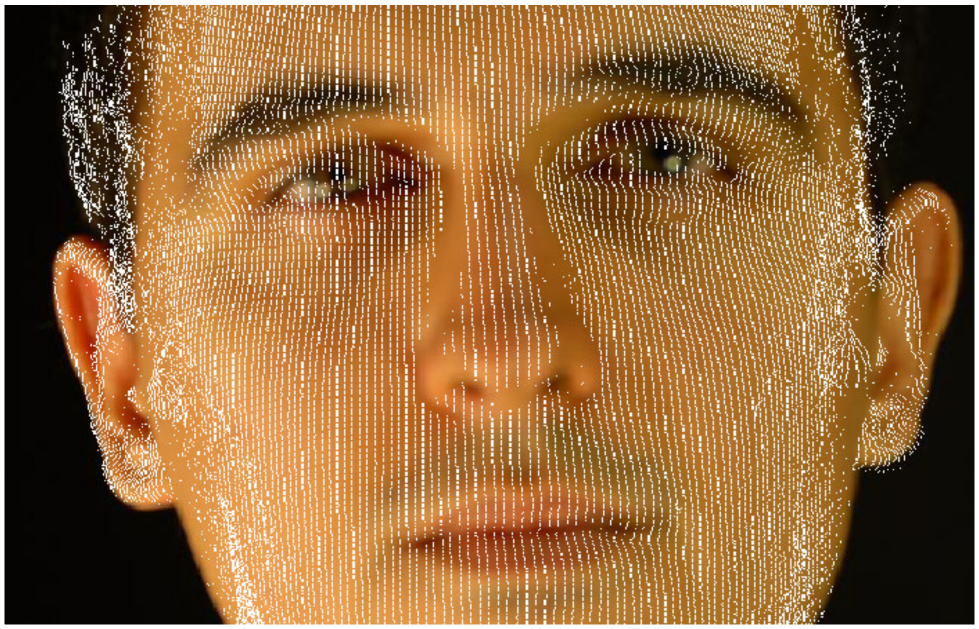


Figure 4.5: Projection of 3D Points on Front View (Zoomed)

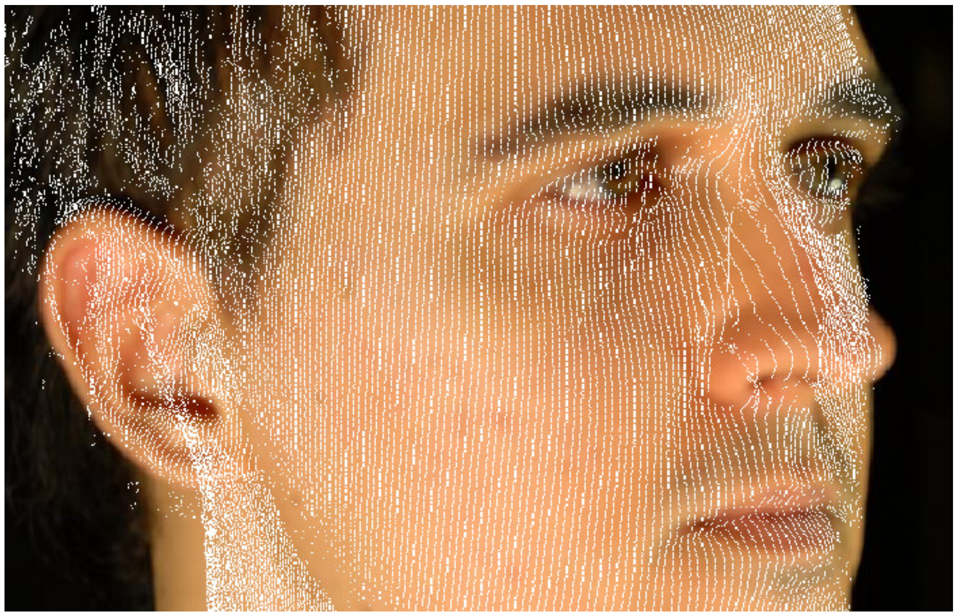


Figure 4.6 : Projection of 3D Points on Front Right View (Zoomed)

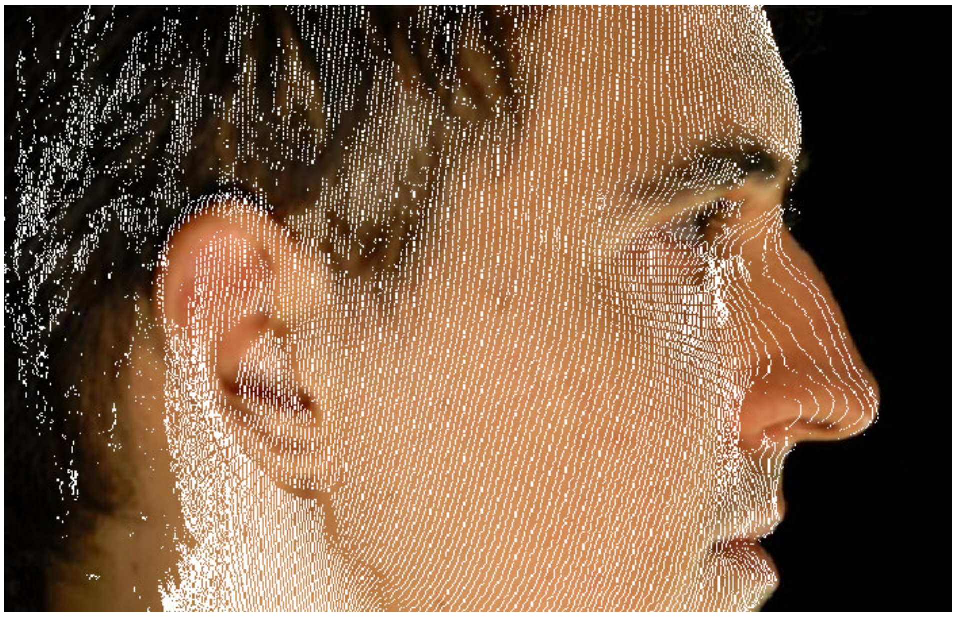


Figure 4.7 : Projection of 3D Points on Right View (Zoomed)

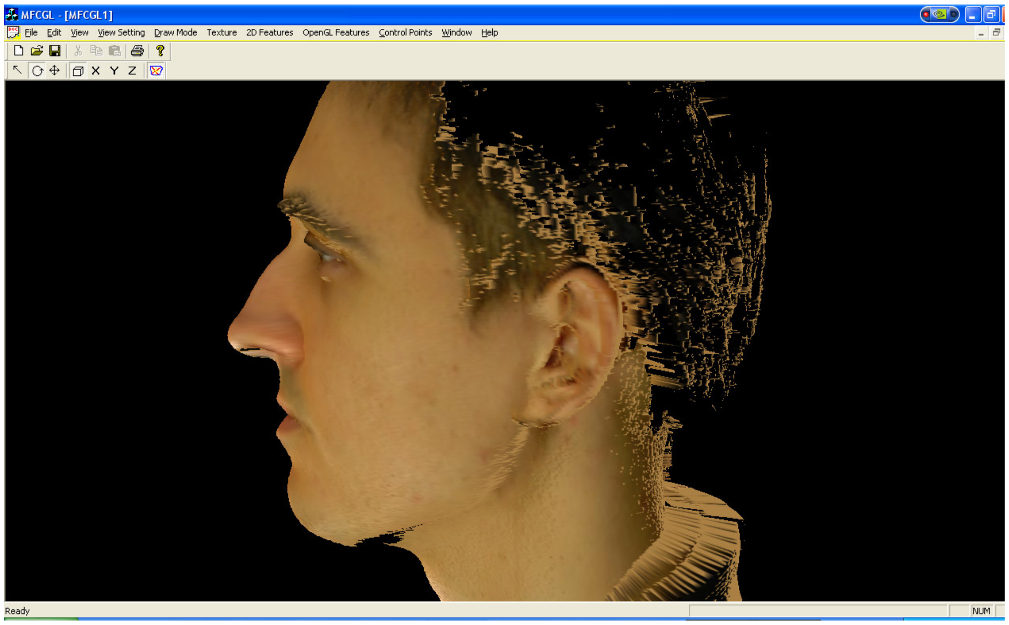

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready
NUM

Figure 4.8: 3D Model with Texture on Left View

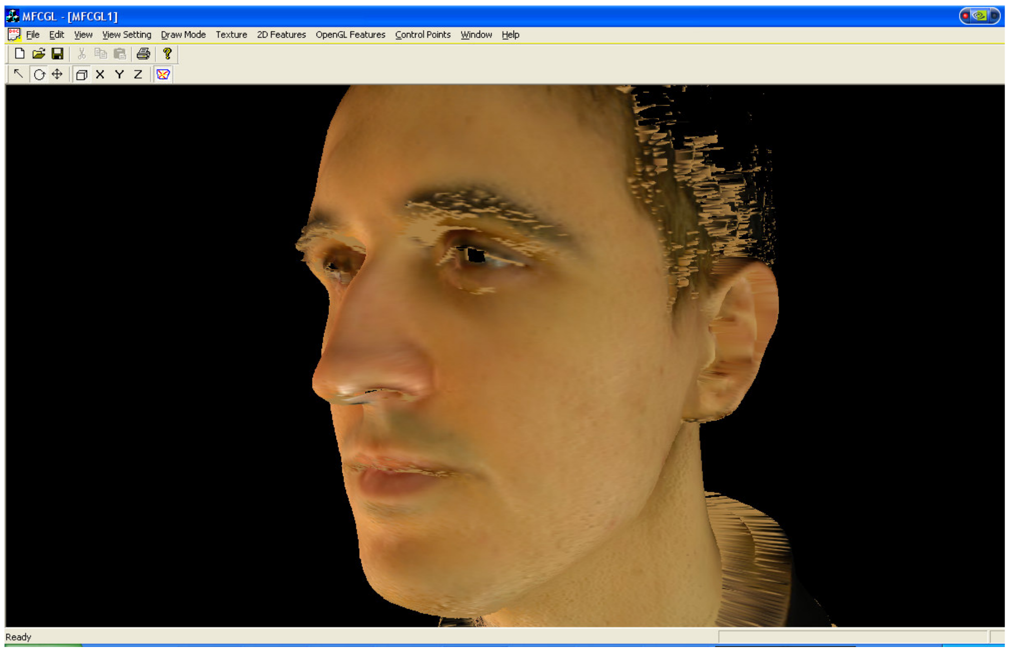

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.9: 3D Model with Texture on Front Left View

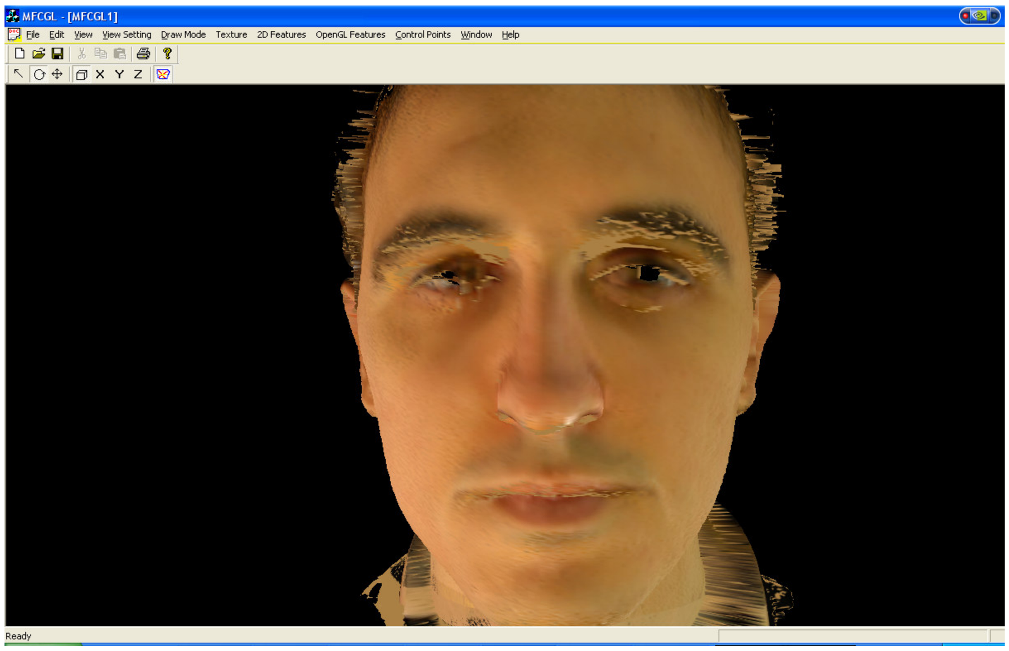

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.10: 3D Model with Texture on Front View

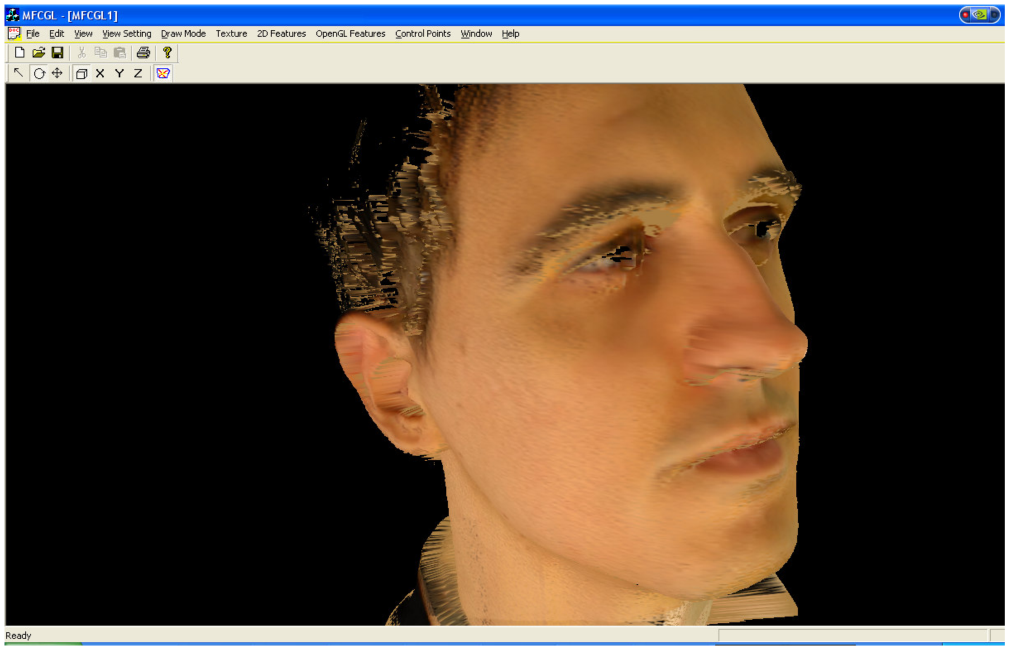

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.11: 3D Model with Texture on Front Right View

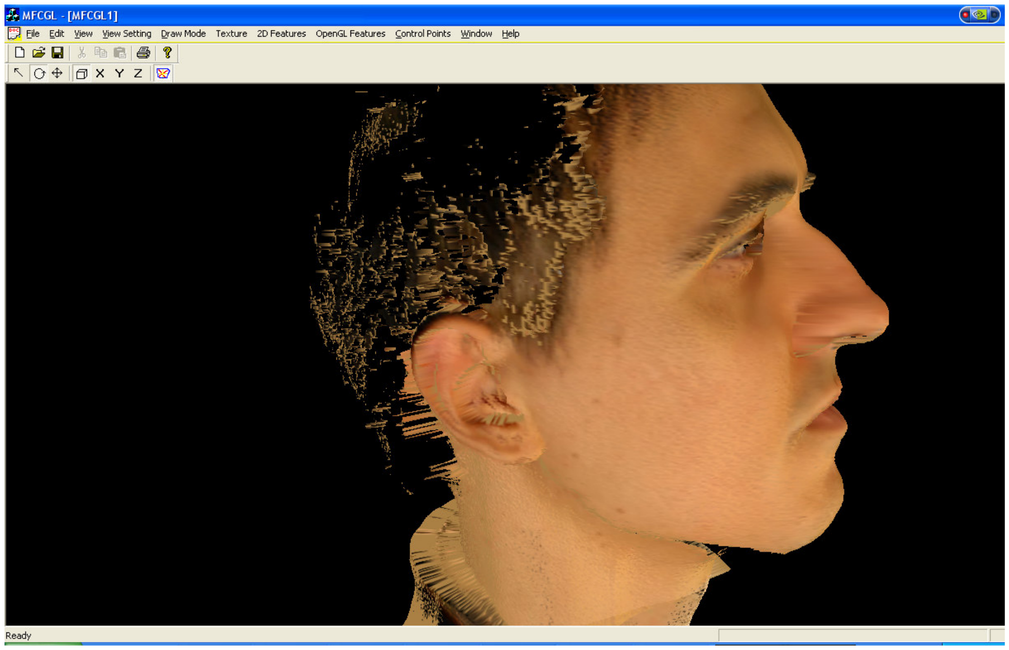

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.12: 3D Model with Texture on Right View

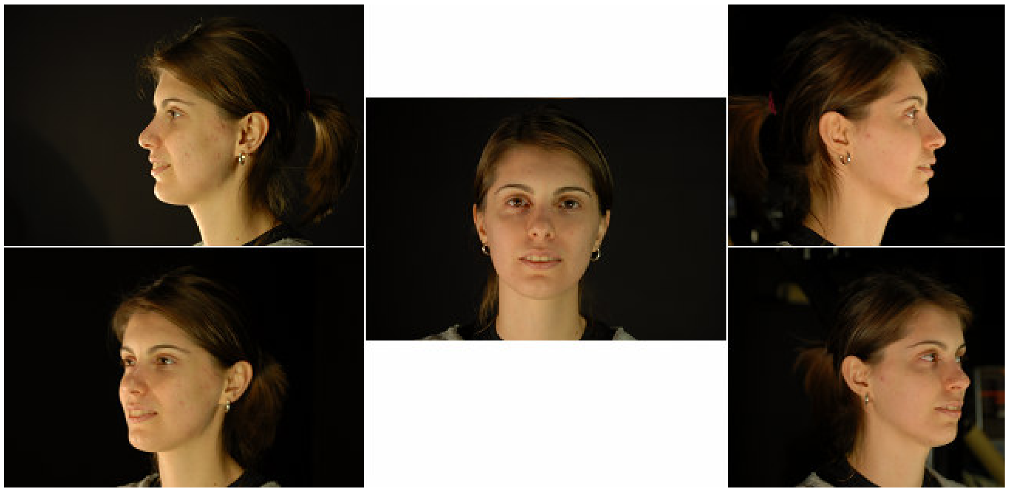


Figure 4.13 : Input Images

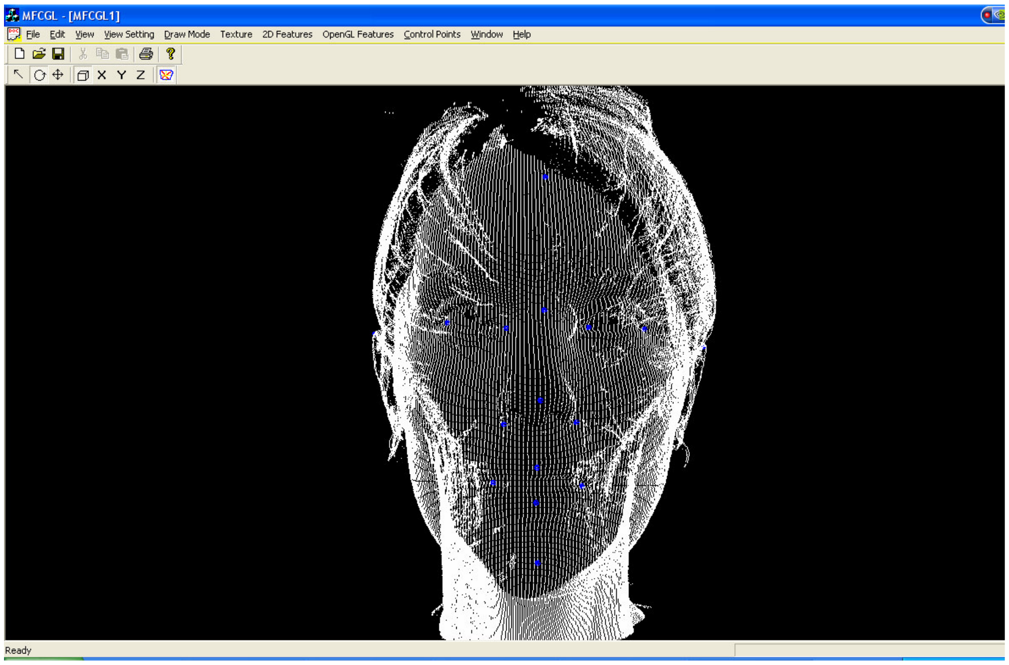

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.14 : 3D Model with Selected Points

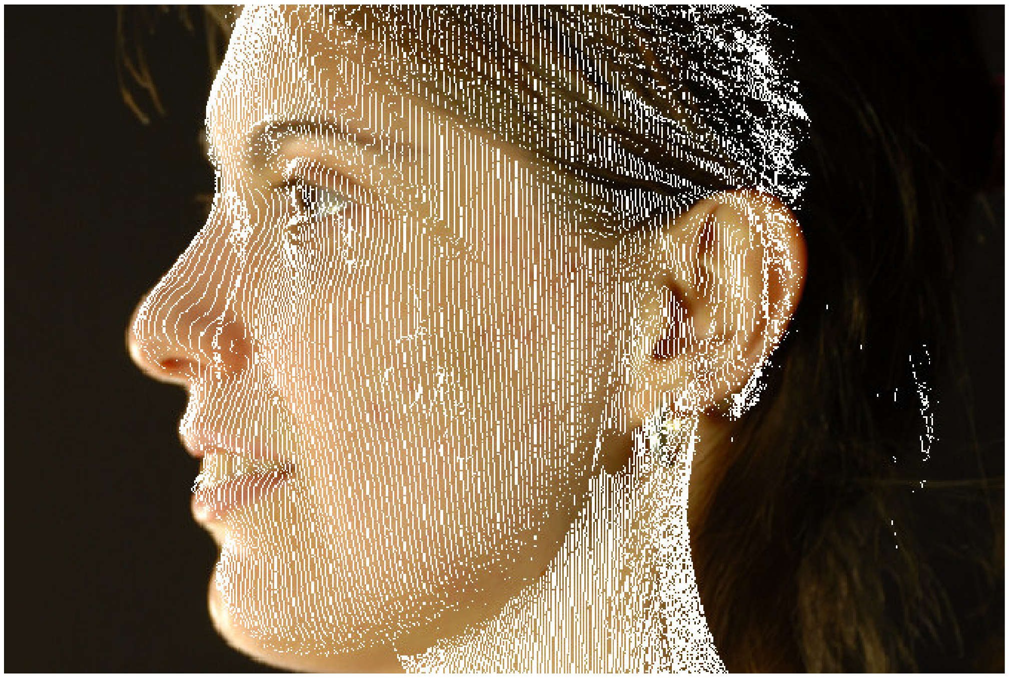


Figure 4.15 : Projection of 3D Points on Left View (Zoomed)

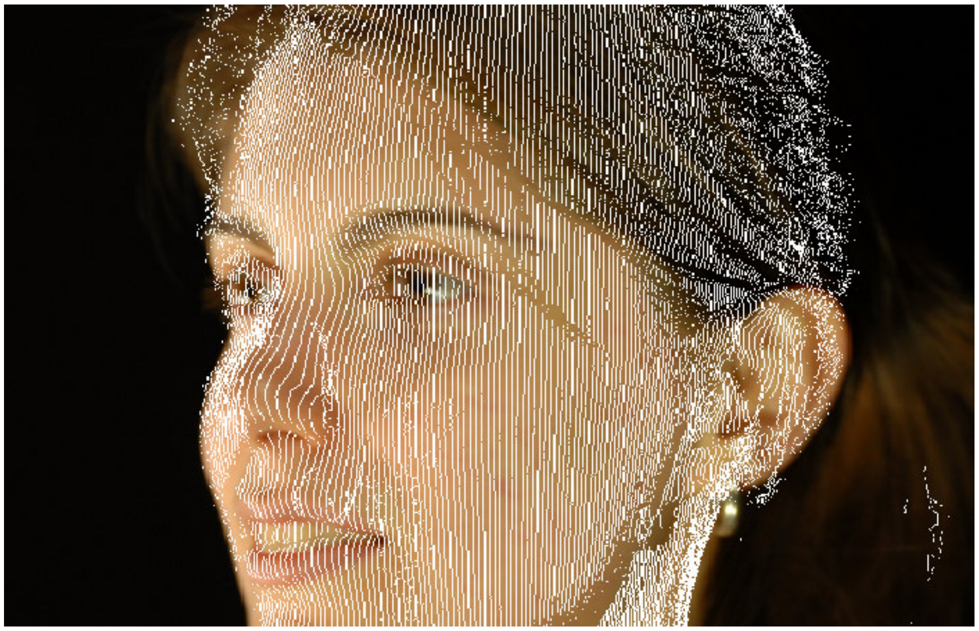


Figure 4.16 : Projection of 3D Points on Front Left View (Zoomed)

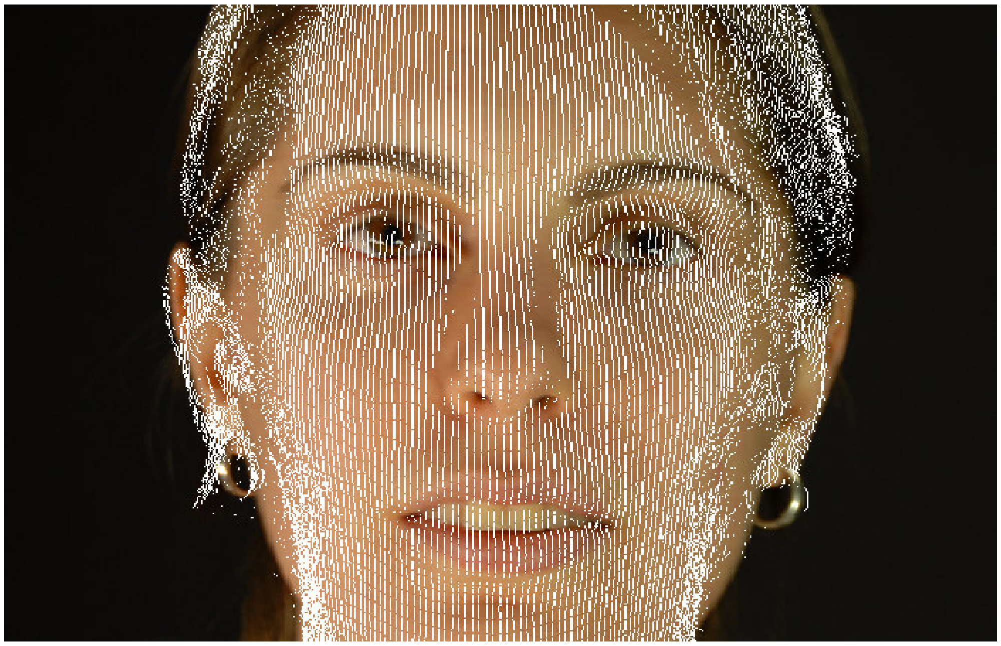


Figure 4.17 : Projection of 3D Points on Front View (Zoomed)

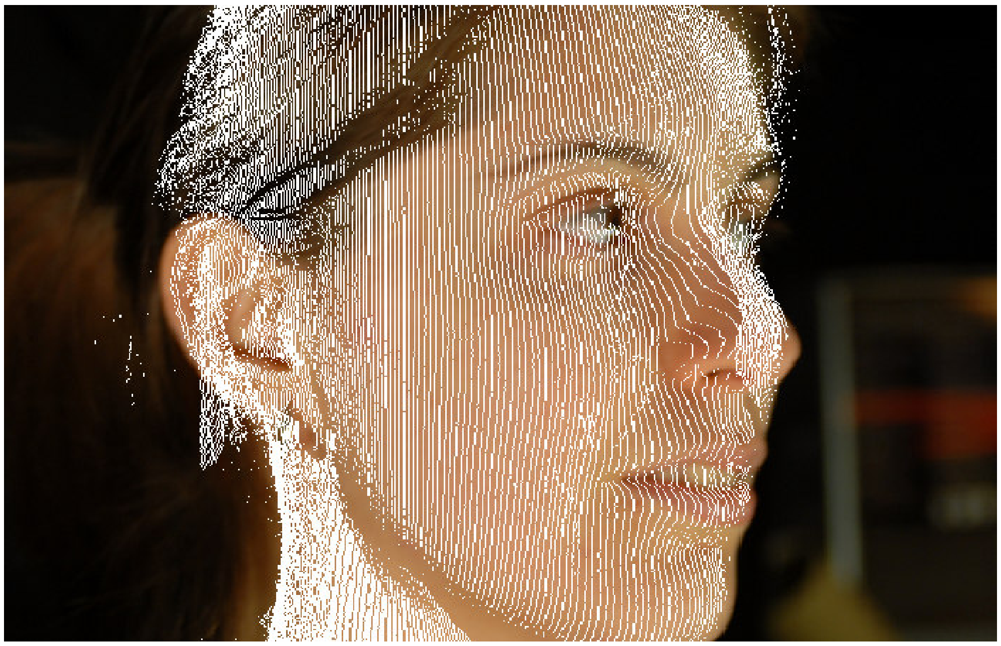


Figure 4.18 : Projection of 3D Points on Front Right View (Zoomed)

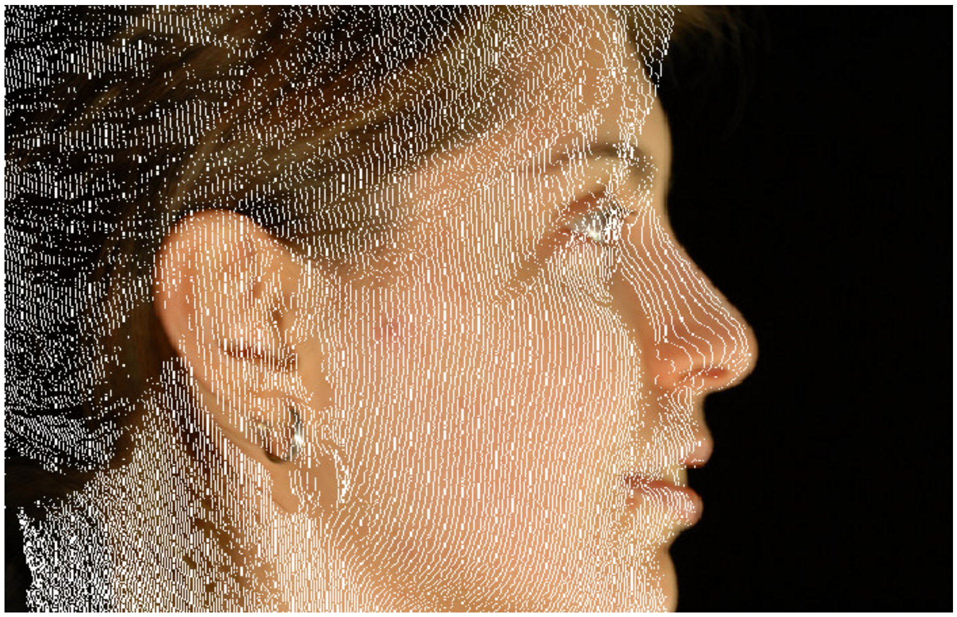


Figure 4.19 : Projection of 3D Points on Right View (Zoomed)

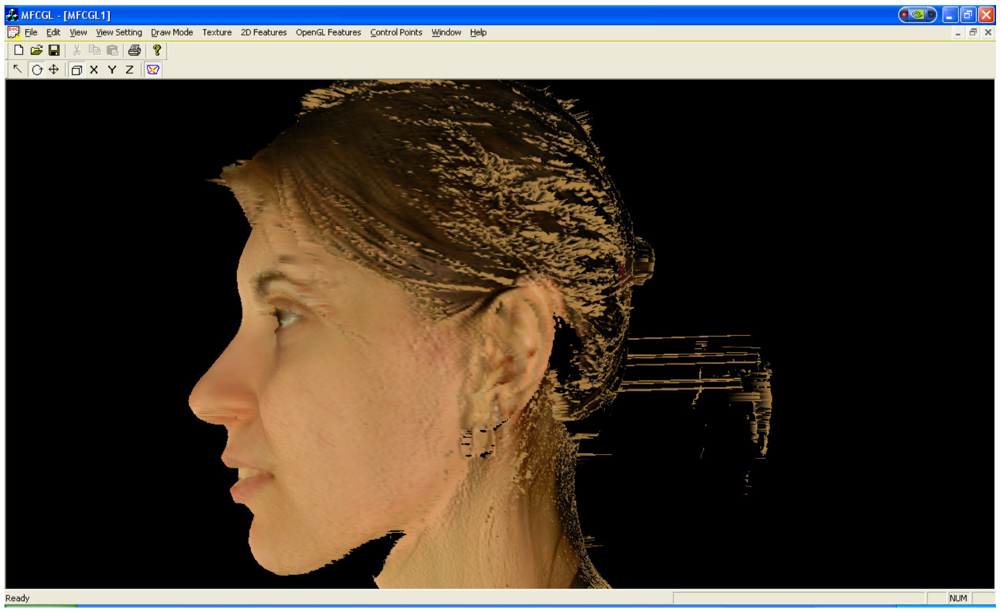

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready
NUM

Figure 4.20 : 3D Model with Texture on Left View

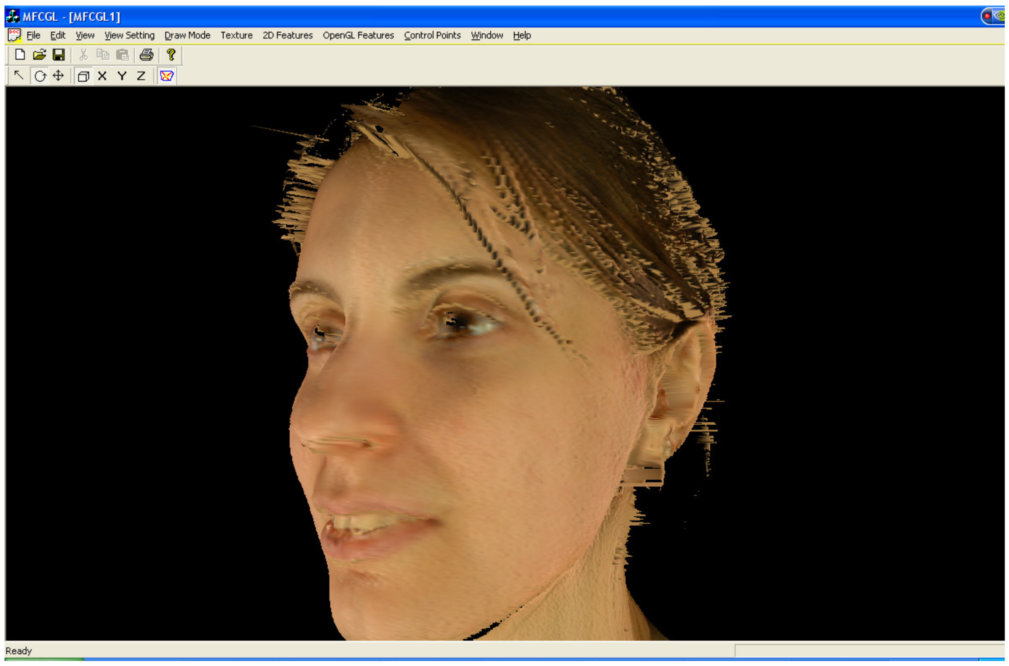

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.21 : 3D Model with Texture on Front Left View

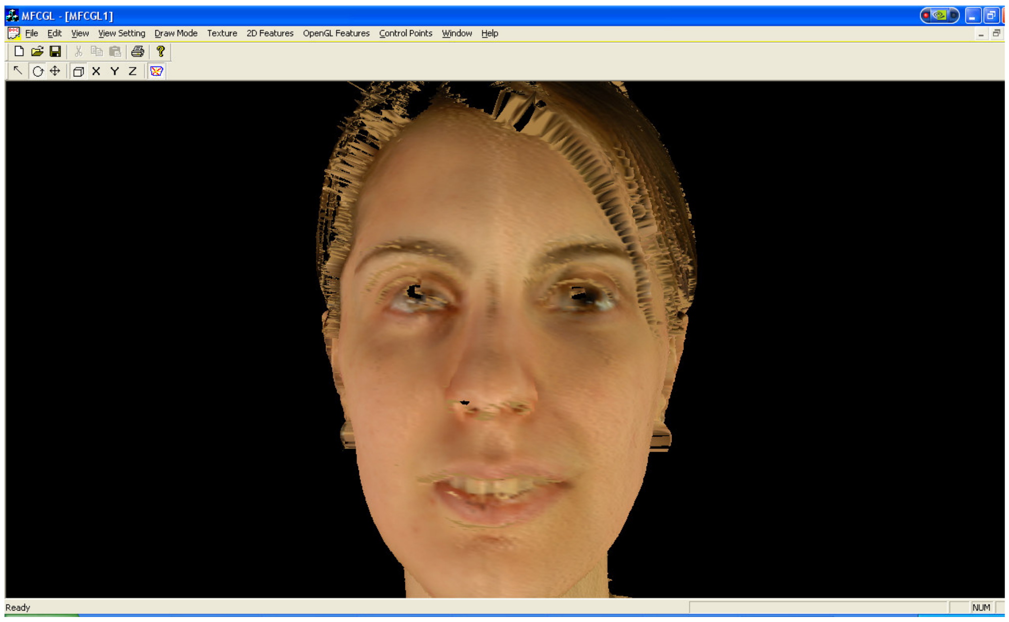

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready
NUM

Figure 4.22 : 3D Model with Texture on Front View

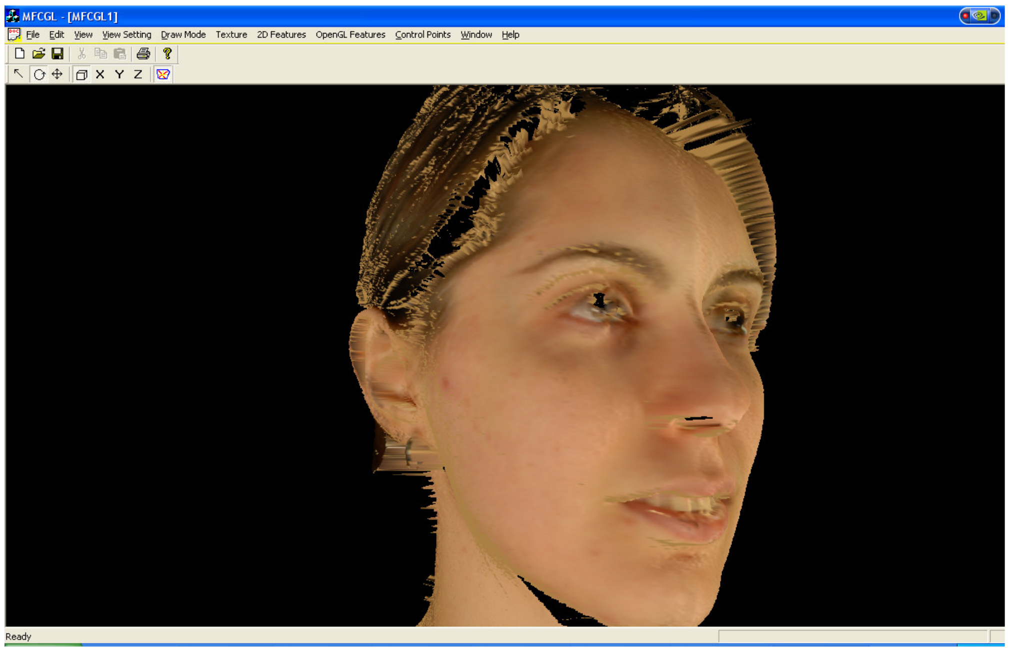

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.23 : 3D Model with Texture on Front Right View

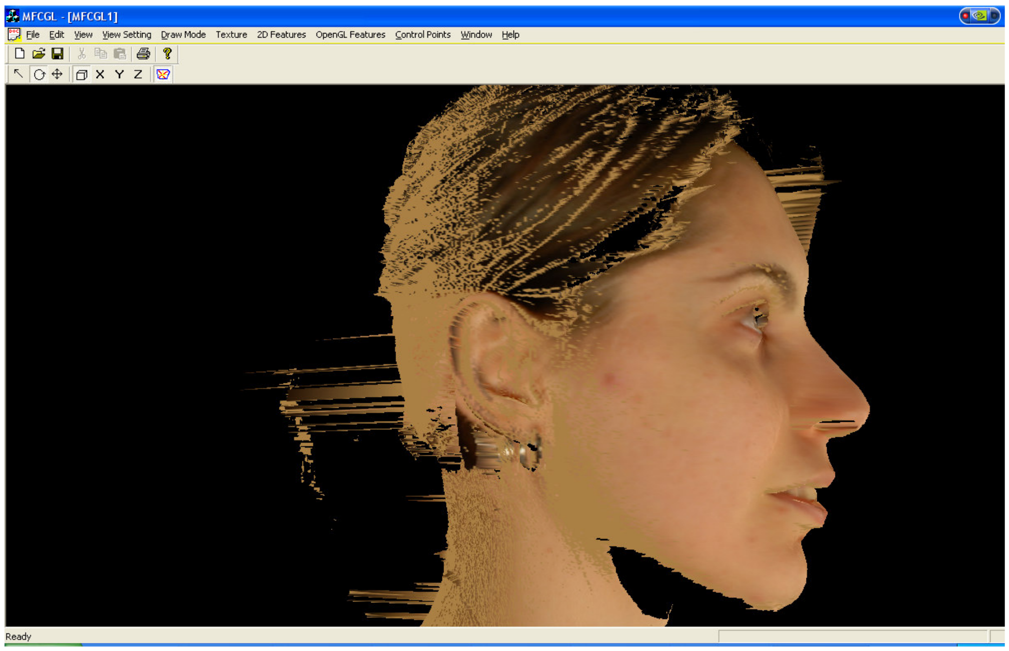

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.24 : 3D Model with Texture on Right View

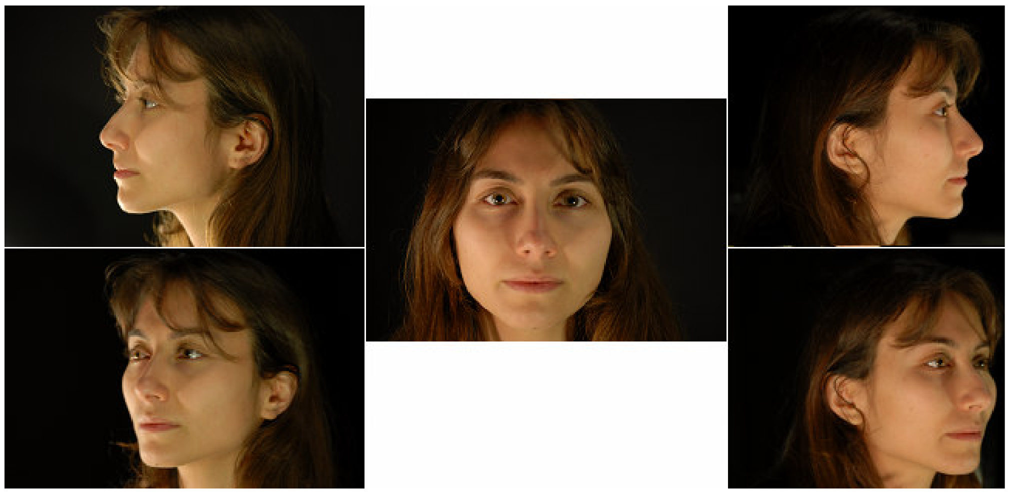


Figure 4.25 : Input Images

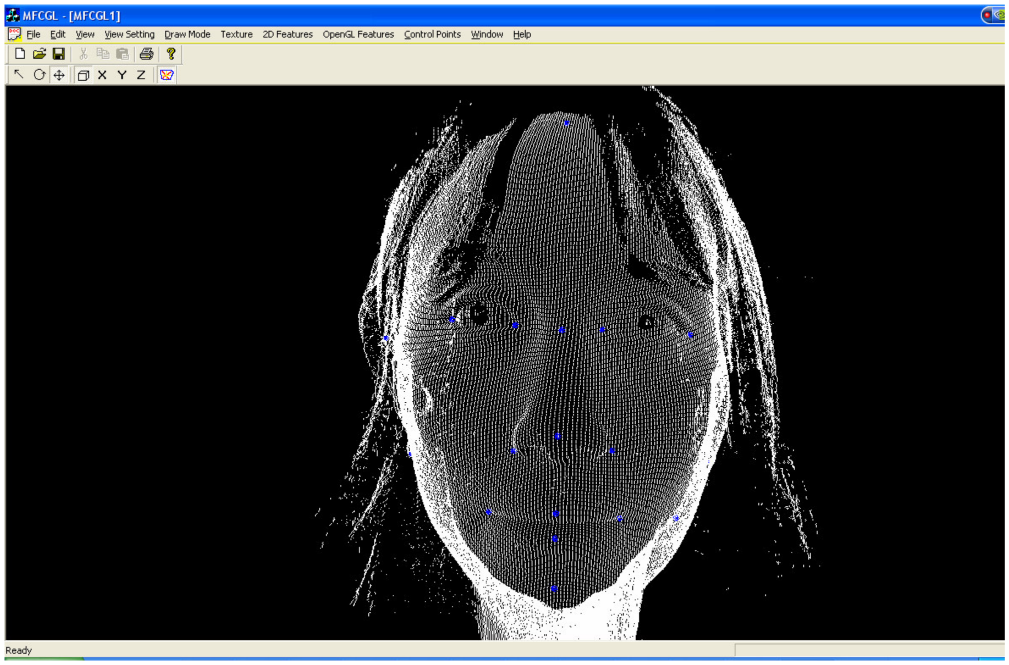

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.26 : 3D Model with Selected Points

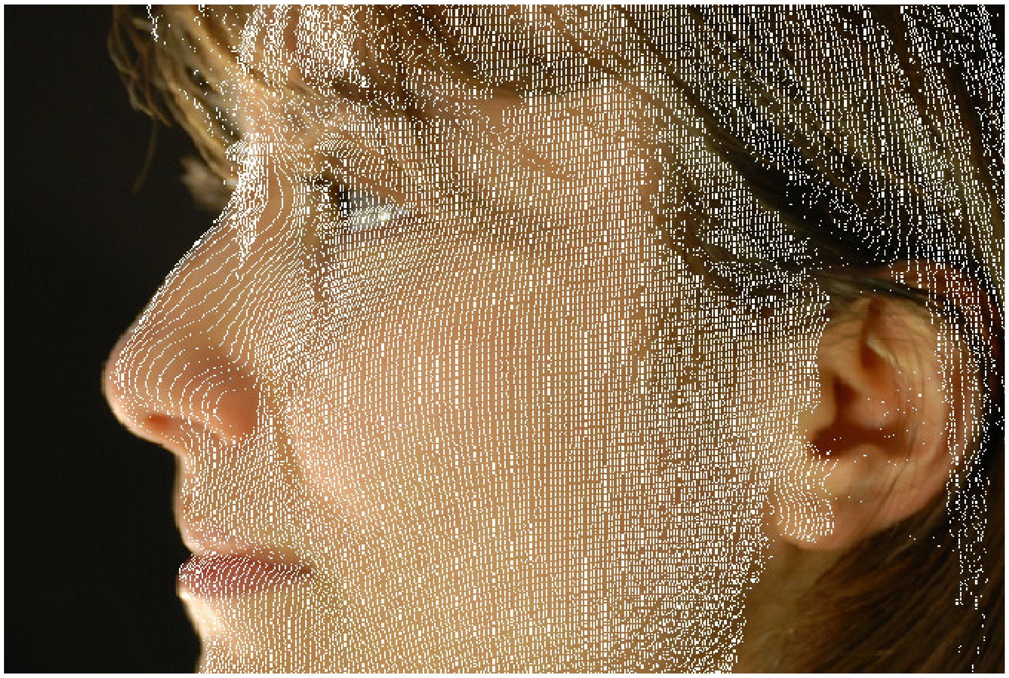


Figure 4.27 : Projection of 3D Points on Left View (Zoomed)

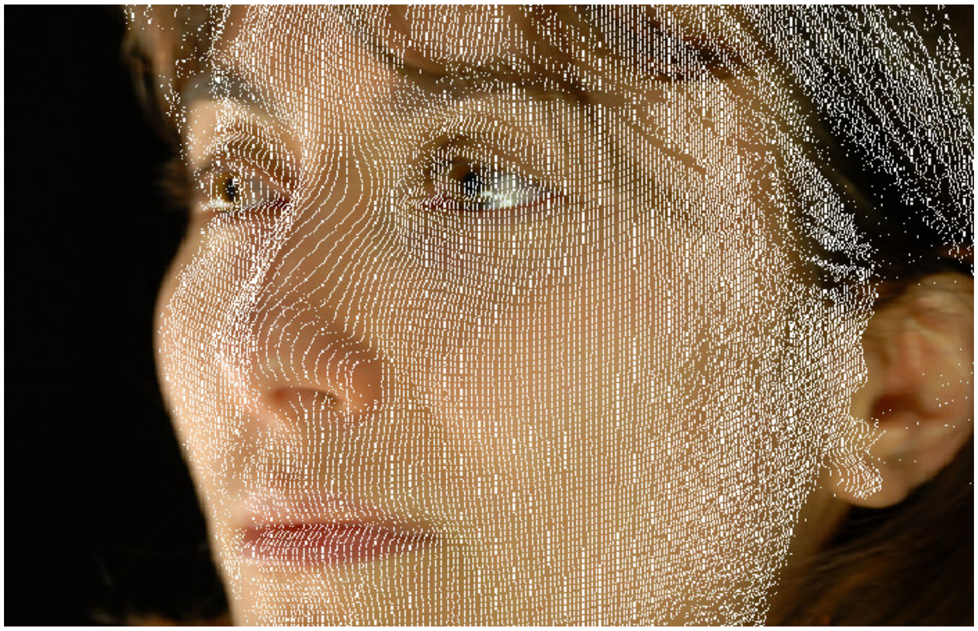


Figure 4.28 : Projection of 3D Points on Front Left View (Zoomed)

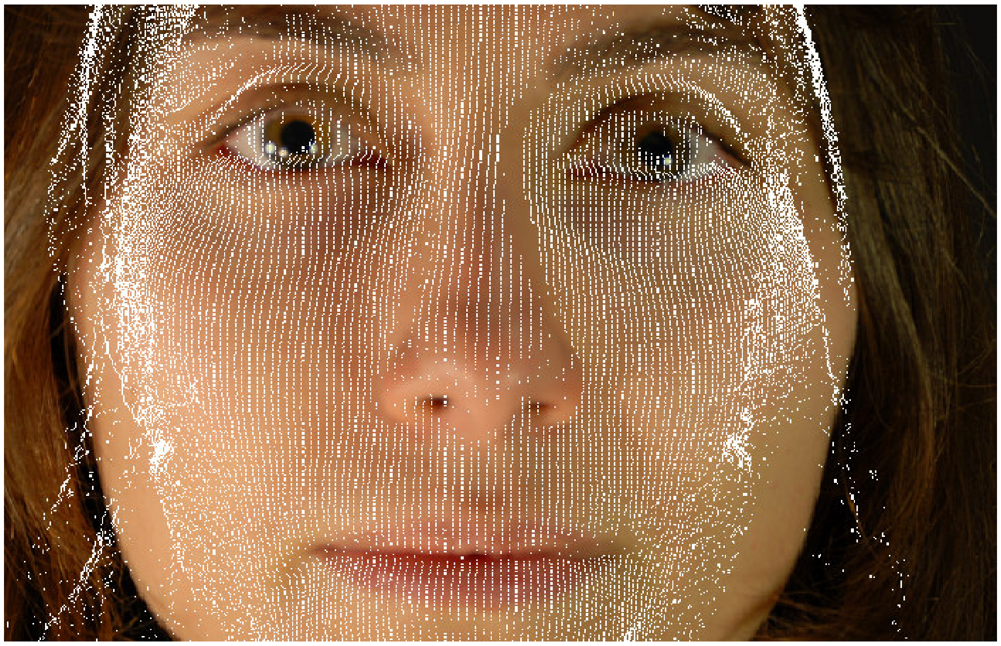


Figure 4.29 : Projection of 3D Points on Front View (Zoomed)

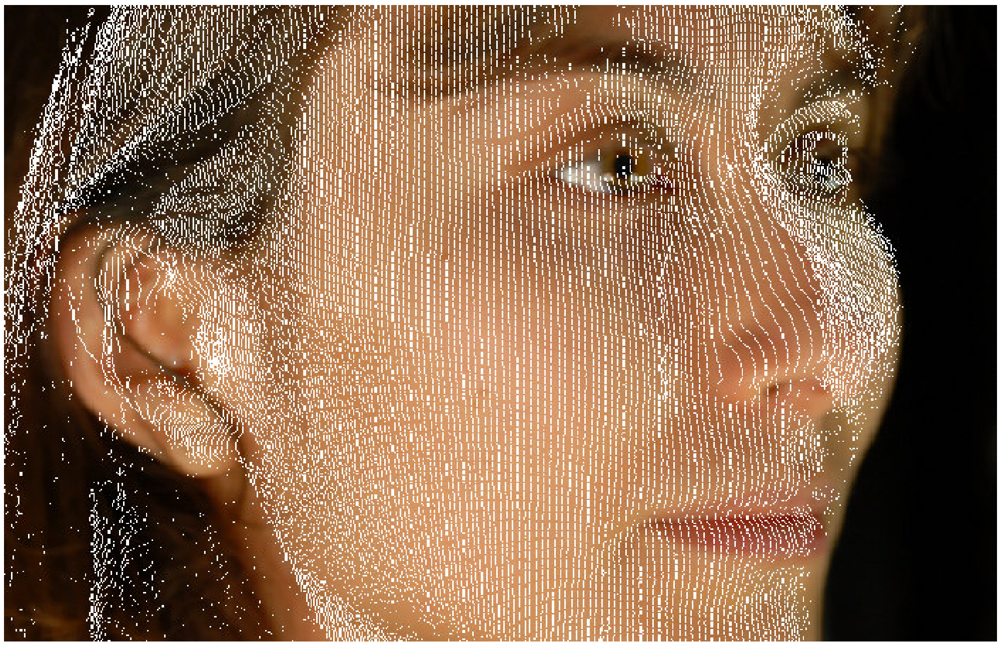


Figure 4.30 : Projection of 3D Points on Front Right View (Zoomed)

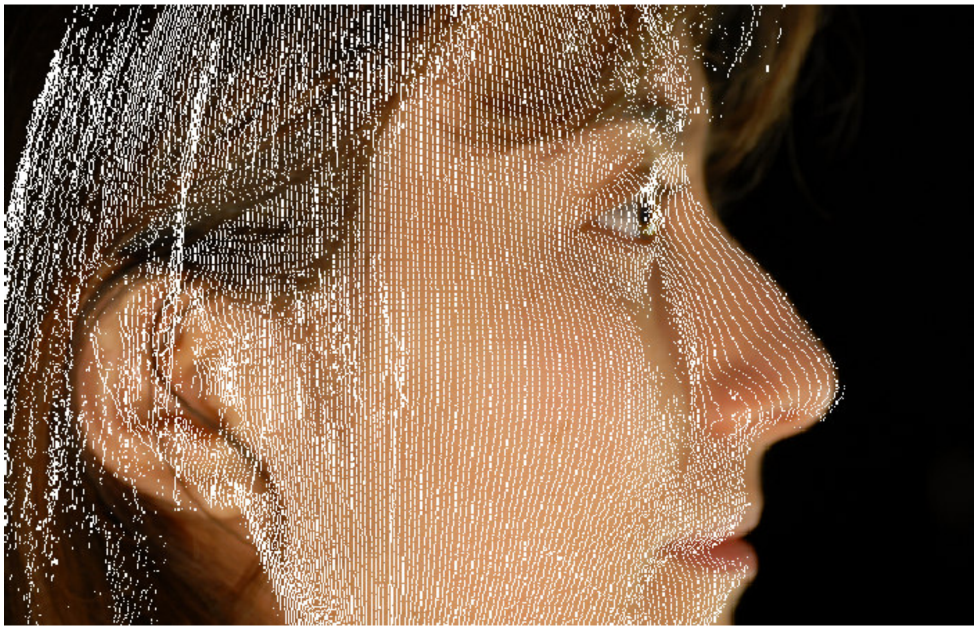


Figure 4.31 : Projection of 3D Points on Right View (Zoomed)

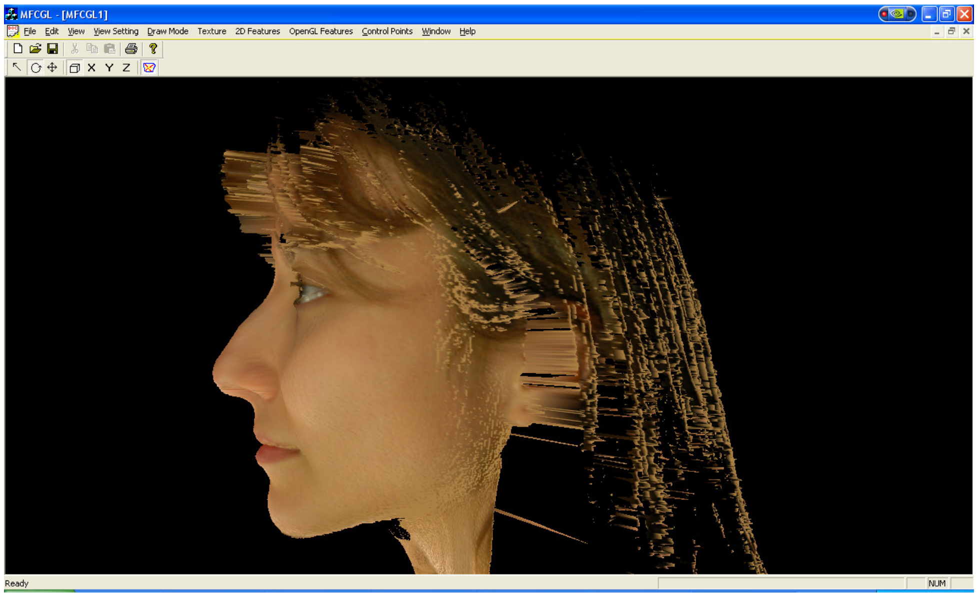

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready
NUM

Figure 4.32 : 3D Model with Texture on Left View

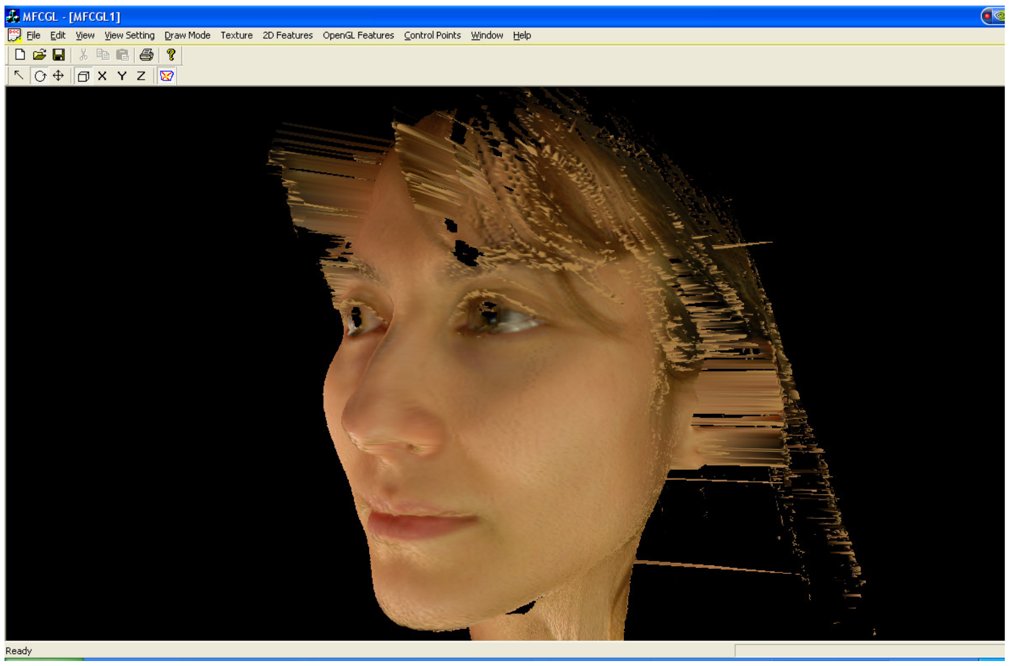

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.33 : 3D Model with Texture on Front Left View

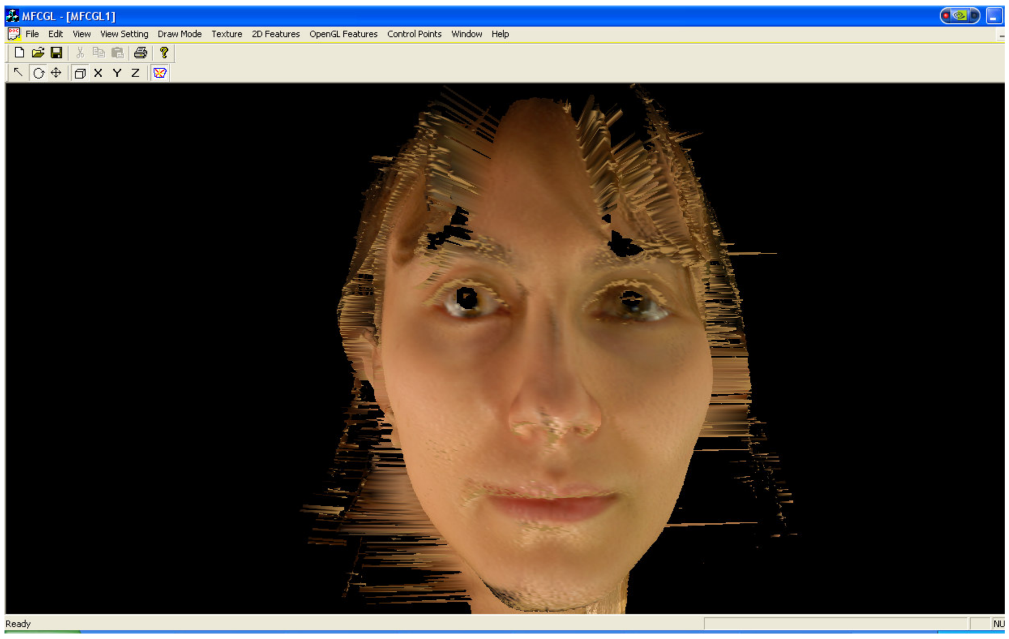

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready

Figure 4.34 : 3D Model with Texture on Front View

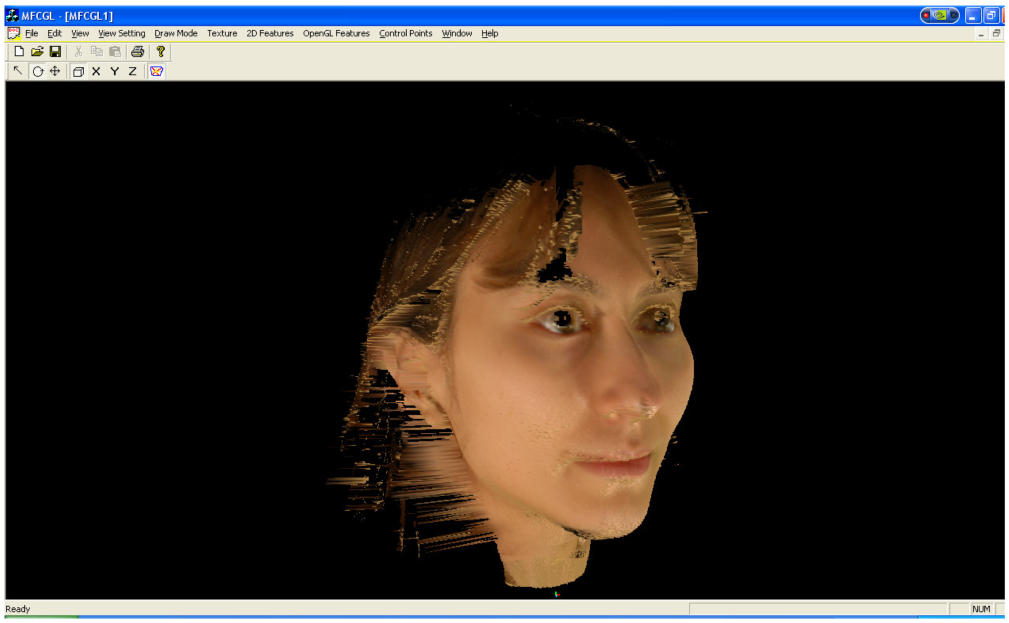

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready
NUM

Figure 4.35 : 3D Model with Texture on Front Right View

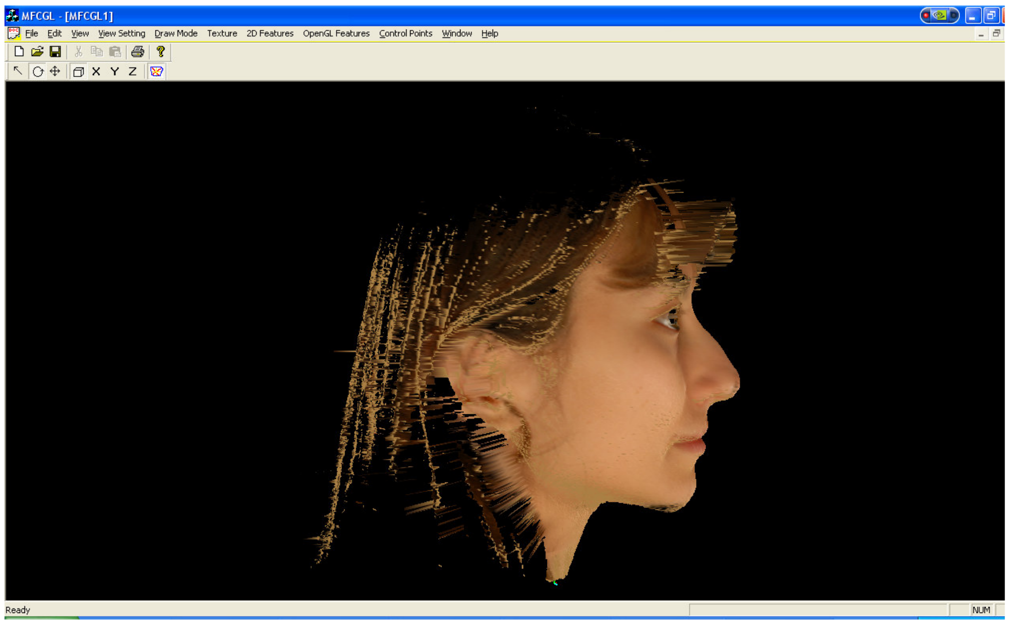
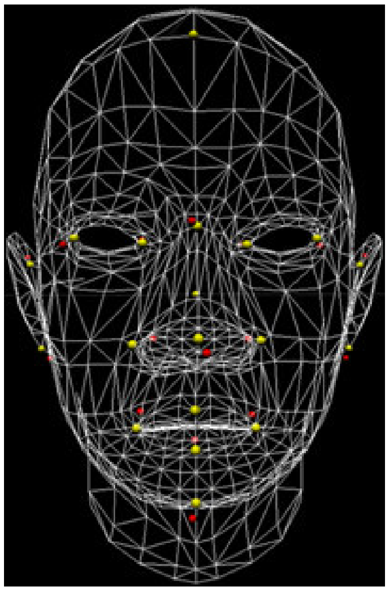

MFCGL - [MFCGL1]
File Edit View View Setting Draw Mode Texture 2D Features OpenGL Features Control Points Window Help
X Y Z
Ready
NUM

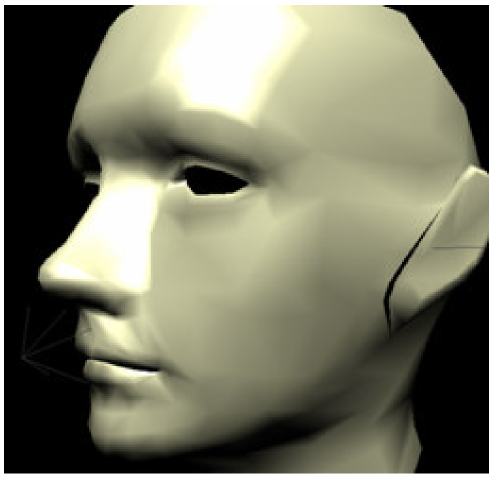
Figure 4.36 : 3D Model with Texture on Right View

4.2 Comparison with the Results of 3D Face Model with Multiple Images Using Generic Face Model

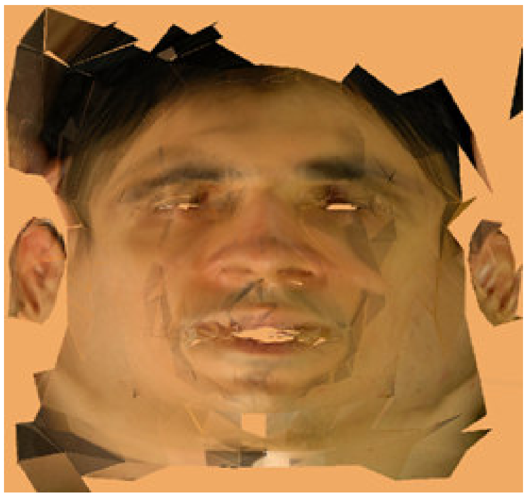
In this section, the software that developed in [32] is used to generate texture with the images in Fig. 4.1. Since a low resolution generic face model is deformed, the shape is not accurate as 3D scanner. Moreover, due to texture image is formed by using triangular areas taken from appropriate images; there is discontinuity at the edges of triangles.



(a)



(b)


(c)

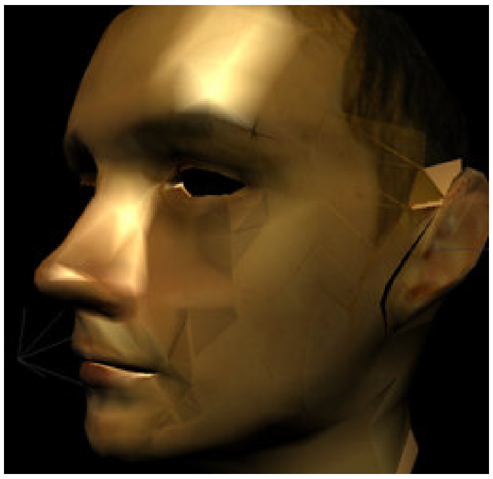
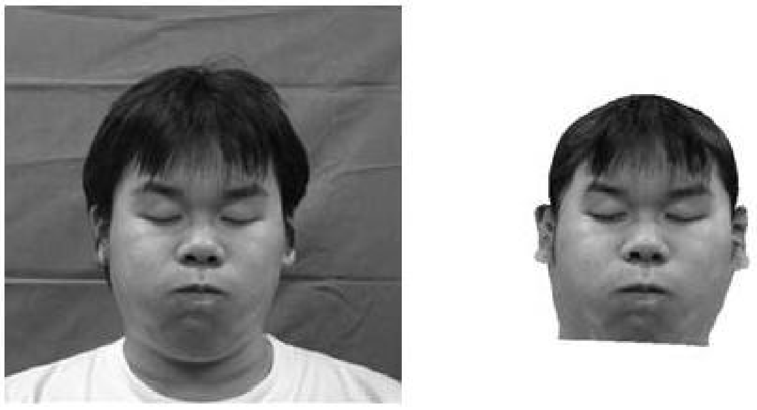

(d)

Figure 4.37: (a) Generic Face Shape Model used in [32],
(b) Refined Face Shape Model Generated in [32] in Solid Mode,
(c) Texture Image, (d)Texture Covered 3D Refined Face Model

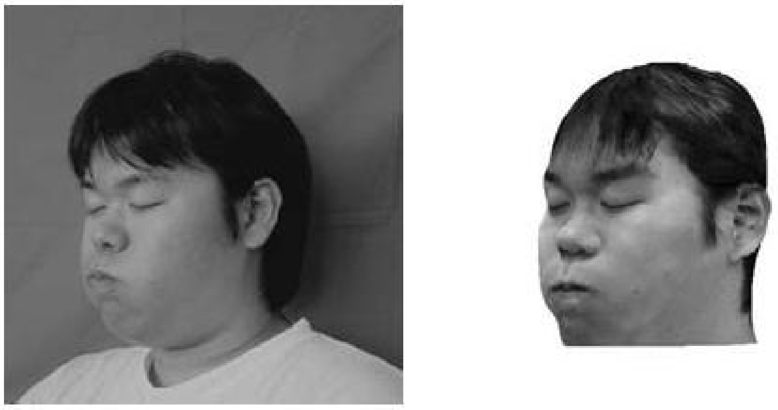
4.3 Comparison with the Results of 3D Laser Scanner Outputs

In [38], a method to find control points automatically is proposed and similar steps like camera calibration and texture mapping in this thesis, are performed. CT data that is used for the 3D face shape model, does not contain hair or eyebrow parts which may be thought as noisy part of the face model. The results look like the result that is obtained in this thesis except the hair and eyebrow parts of the face.



(a) (b)

Figure 4.38 : (a) Input Image, (b) Texture Covered 3D Face Model



(a) (b)

Figure 4.39 : (a) Input Image, (b) Texture Covered 3D Face Model

CHAPTER 5

CONCLUSION

In this study, 3D face models generated by a 3D laser scanner are textured by blending multi images. By manually selecting 18 control points on 3D model and also on five images with different viewing directions, all the camera parameters are calculated. By view independent texture mapping technique, blended color information is transformed onto 3D model.

Although camera calibration technique and number of control points are decided according to [32] which is a MS thesis, completed in METU Computer Vision and Intelligent Systems Research Laboratory in 2004, a different approached is used for texture blending in this thesis. Even though there are some artifacts at hair or nose of the face model, the generated texture is satisfactory. This is due to the fact that a high accurate shape model is used and images and this shape model belongs to the same person. Moreover, blending pixel by pixel results in a smooth texture image.

The inaccuracy in selection of the control points does not disturb as much as the generic face approach in [32] since the images and the 3D face model are almost perfectly matched. In addition, since the resolution in terms of vertices is higher than the generic face model, this will enable to select more accurately the points.

5.1 Future Works

Several additional works to improve the results obtained in the thesis can be listed. First thing is automatical selection of the control points. Manually selection is somehow time consuming process and it should be accurate.

Second, despite high resolution is very critical for accurate pose estimation and more realistic texture, it causes the program to be slow in computation time. Reduction in number of vertices can be performed at flat surfaces of the face.

Moreover, instead of using 3D laser scanner output, CT/MR data can be used or 3D model can be formed from multiple images too. Multi-camera or structure light can also be used for generating 3D shape. It will make the process affordable since laser scanners are expensive and not very common.
To reduce artifacts at the hair and nose regions of the face model, synthetic texture that are preloaded from generalized face texture, can be used.

Finally, since the multi image is used for texturing, a shadow removing algorithm can be implemented

REFERENCES

[1] Adnan Ansar and Kostas Daniilidis, “Linear Pose Estimation from Points or Lines” IEEE Transactions on Pattern Analysis And Machine Intelligence, Vol. 25, No. 5, pp. 578-589, (2003)

[2] Diego Ortin and Fabio Remondino, “Occlusion-Free Image Generation For Realistic Texture Mapping” International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVI, part 5/W17 (CD-Rom), (2005)

[3] Dimitris Samaras, “Incorporating Illumination Constraints in Deformable Models for Shape from Shading and Light Direction Estimation” IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 25, No. 2, pp. 247-264, (2003)

[4] Ed Catmull, "A Subdivision Algorithm for Computer Display of Curved Surfaces," PhD thesis, Univ. of Utah, (1974)

[5] Eliot A. Feibush, Marc Levoy, and Robert L. Cook, "Synthetic Texturing Using Digital Filters" Computer Graphics (Proc.SIGGRAPH 80), Vol. 14, No. 3, pp.294-301, (1980)

[6] Elsayed E. Hemayed, “A Survey of Camera Self-Calibration” In Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS’03), pp.351-357, (2003)

[7] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, “Color transfer between images” IEEE Comput. Graph. Appl., vol. 21, no. 5, pp. 34–41, (2001)

[8] F. Devernay and 0. D. Faugeras, “Computing Differential Properties of 3-D Shapes from Stereoscopic Images without 3-D Models” In Proceedings of Computer Vision and Pattern Recognition, pp. 208-213, (1994)

[9] F. Dornaika and C. Garcia, “Robust camera calibration using 2D to 3D feature correspondences” In Proceedings of the International Symposium SPIE - Optical Science Engineering and Instrumentation, Videometrics V, Volume 3174, pp. 123-133, (1997)

[10] Frederick M. Weinhaus and Venkat Devarajan, “Texture Mapping 3D Models of Real-World Scenes” ACM Computing Surveys, Vol. 29, No. 4, pp. 325-365, (1997)

[11] Frederic Pighin, Jamie Hecker, Dani Lischinski Richard Szeliski and David H. Salesin, “Synthesizing Realistic Facial Expressions from Photographs” International Conference on Computer Graphics and Interactive Techniques, pp. 75-84, (1998)

[12] G. Csurka and O. Faugeras, “Algebraic and Geometric Tools to Compute - Projective and Permutation Invariants” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 1, pp.58-65, (1999)

[13] Horace H. S. Ip and Lijun Yin, “Constructing a 3D Individualized Head Model from Two Orthogonal Views” The Visual Computer, pp. 254–266, (1996)

[14] I. Kyu Park, Hui Zhang and Vladimir Vezhnevets, “Image-Based 3D Face Modeling System” EURASIP Journal on Applied Signal Processing, pp. 2072–2090, (2005)

[15] James F. Blinn and Martin E. Newell, "Texture and Reflection in Computer Generated Images," Comm ACM, Vol. 19, No. 10, pp.542-547, (1976)

[16] Joseph S.C. Yuan, “A General Photogrammetric Method for Determining Object Position and Orientation” IEEE Transactions on Robotics and Automation, Vol. 5, No. 2, pp. 129-142, (1989)

[17] Li-an Tang, Thomas S. Huang, “Analysis-Based Facial Expression Synthesis” IEEE International Conference on Image Processing, Vol. 3, pp. 98-102, (1994)

[18] Long Quan and Zhongdan Lan, “Linear N-Point Camera Pose Determination” IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 21, No. 8, pp. 774-780, (1999)

[19] M. A. Fischer and R. C. Bolles, “Random Sample consessus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography” Comm. of ACM, Vol. 24, No. 6, pp. 381-395, (1981)

[20] Marc Proesmans, Lug J. Van Gool and A. J. Oosterlinck, “Active Acquisition Of 3d Shape For Moving Objects” IEEE International Conference on Image Processing, Vol. 3, pp.647-650, (1996)

[21] Matthew J. Clarkson, Daniel Rueckert, Derek L.G. Hill and David J. Hawkes, “Using Photo-Consistency to Register 2D Images of Human Face to a 3D Surface Model” IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 23, No. 11, pp. 1266-1280, (2001)

[22] Michel Gangnet, Didier Perny, Philippe Coueignoux, “Perspective Mapping of Planar Textures” Eurographics ’82, pp. 57-71, (1982)

[23] Ned Greene and Paul S. Heckbert, "Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter" IEEE Computer Graphics and Applications, Vol. 6, No.6, pp.21-27, (1986)

[24] Nian Wang, Jun Tang, Yi-Zheng Fan and Dong Liang, “An Algorithm of Camera Self-calibration” In Proceedings of Signal Processing, Vol. 2, (2006)

[25] O. D. Fruferas and G. Toscani, “The calibration problem for stereo” In Proceedings of Computer Vision and Pattern Recognition, pp.15-20 (1986).

[26] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik, “Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach” In Proceedings of SIGGRAPH 96, pp. 11–20, (1996).

[27] Paul E. Debevec, Yizhou Yu, and George D. Borshukov, “Efficient view-dependent image-based rendering with projective texture-mapping” Eurographics Rendering Workshop, pp. 105–116, (1998).

[28] R. Cipolla, T. Drummond and D. Robertson, “Camera calibration from vanishing points in images of architectural scenes” The British Machine Vision Conference, pp. 382-391, (1999)

[29] R Economy and M Bunker, “Advanced video object simulation” In Proceedings of the National Aerospace and Electronics Conference, pp. 1065–1071, (1984)

[30] Reimar K. Lenz and Roger Y. Tsai, “Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology” In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 4, pp. 68-75, (1987)

[31] Sami Romdhani, Jeffrey Ho,Thomas Vetter and David J. Kriegman, “Face Recognition Using 3-D Models: Pose and Illumination” In IEEE Proceedings, Vol. 94, No. 11, pp. 1977-1999, (2006)

[32] S. Büyükatalay, “3D Face Model Generation” Master’s Thesis, Middle East Technical University, Ankara, (2004)

[33] Sen Wang, Lei Zhang, and Dimitris Samaras, “Face Reconstruction Across Different Poses And Arbitrary Illumination Conditions” AVBPA 2005, LNCS 3546, pp. 91–101, (2005)

[34] Takaaki Akimoto, Yasuhito Suenaga, and Richard S. Wallace, “Automatic Creation of 3D Facial Models” IEEE Computer Graphics and Applications, pp. 16–22, (1993)

[35] Tsuneya Kurihara and Kiyoshi Arai, “A Transformation Method for Modeling and Animation of the Human Face from Photographs” In Nadia Magnenat Thalmann and Daniel Thalmann, editors, Computer Animation 91, pp. 45–58. Springer-Verlag, (1991)

[36] Volker Blanz Thomas Vetter, “A Morphable Model For The Synthesis Of 3D Faces”, In International Conference on Computer Graphics and Interactive Techniques, pp.187-194, (1999)

[37] Y. Lui, T.S. Huang and O.D. Faugeras, “Determination of Camera Location from 2D to 3D Line and Point Correspondence” In Proceedings of Computer Vision and Pattern Recognition, pp.82-88, (1988)

[38] Yuya Iwakiri, Keisuke Yorioka and Toyohisa Kaneko, “Fast Texture Mapping of Photographs on a 3D Facial Model” Image and Vision Computing, pp. 390-395, (2003)

[39] Z. Chen, S.-Ying Ho, and D.-Chang Tseng, “Polyhedral Face Reconstruction and Modeling from a Single Image with Structured Light” IEEE Transactions on Systems, Man, And Cybernetics, Vol. 23, No. 3, pp.864-872, (1993)

[40] Zhonggen Yang and Fang Cao, “Linear Six-Point Algorithm of Camera Self-calibration and 3D Reconstruction from Single-View” In Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), Vol. 2, pp. 390-395, (2006)

APPENDIX A

"OBJ" FILE FORMAT

The "obj" is a text file format. The first character of each line specifies the type of command. If the first character is a #, the line is a comment and the rest of the line is ignored. The vertex command, **v** *x y z*, specifies a vertex by its three coordinates. The texture command, **vt** *u v*, specifies texture information. The normal command, **vn** *x y z* specifies normal vector of facet. The facet command, **f** *v1 v2 v3*, specifies which vertices generate a triangle and also texture list or normal list may be added. The group command, **g** *name*, specifies subgroup of the model. The **usemtl** *name* command lets you to link a material file to the model. Below is an example of a obj file.

```
#example obj file
usemtl faceshading
g face
v -28.1210  291.3750  2.2132
v -26.7285  291.3750  6.4169
v -9.7140  286.5000  -0.7645
v -20.3873  286.5000  20.3873
…
```

```
vt 0.736585  0.414583

vt 0.736585  0.427083

vt 0.736585  0.439583

vt 0.736585  0.452083

vt 0.736585  0.464583

…

f   1057/  1057   1136/  1136    1058/  1058

f   1058/  1058   1136/  1136    1137/  1137

f   1058/  1058   1137/  1137    1059/  1059

f   1059/  1059   1137/  1137    1138/  1138

f   1059/  1059   1138/  1138    1060/  1060

…
```

APPENDIX B

HOW TO COMPILE THE PROGRAM

The steps to be followed in order to compile and run the Texture Mapping Software, is as follows:

1. Copy *Textured_Face_Model* folder to your local drive.
2. Install Microsoft Visual C++ 6.0
3. Install Matlab 6.5 (or any newer version)
4. Start Matlab
    a. In the prompt window, type *mbuild –setup*
    b. Follow the menu and choose MS Visual C++ 6.0
    c. Type the following command in the Matlab prompt window

       cd ..\Textured_Face_Model

       mccsavepath
    d. exit from Matlab
5. Copy *glut32.lib* to *C:\Program Files\Microsoft Visual Studio\VC98\Lib*
6. Copy *glut32.dll* to *C:\WINDOWS\system32*

(*glut32.lib* and *glut32.dll* are in ..\Textured_Face_Model\Lib_files)

7. Start MS VC++ 6.0
    a. Select **Tools→Customize** from the menu

b. Click on **Add-ins and Macro Files** tab

c. Check **Matlab Add-in** on the **Add-ins and Macro Files** list

d. Close the **Customize** window

(The floating *MATLAB add-in for Visual Studio* toolbar appears)

e. Click **Files→Open Workspace**

f. Load *..\Textured_Face_Model\MFC\MFCGL.dsw*

g. Select *FileView* tab from *Workspace* window and delete all **.m* files

h. Click on *Matlab Add-in Project Setup* from *MATLAB add-in for Visual Studio* toolbar and click **OK**

i. Select all **.m* from *..\Textured_Face_Model\Matlab_files* and click **Open**

j. Click **Files→Save Workspace**

k. Click **Compile (Ctrl+F7)** and **Go (F5)** in the toolbar or in the **Build** menu

The program is ready to run and the above steps will only be needed in the first installation.

The inputs of the Texture Mapping Software are;

Face_Model.obj: An object file of the 3D face model

Face_imagex.bmp: Five images of the face

The outputs of the Texture Mapping Software are;

Face_Model_Control_Points.txt: A text file showing the control points of the 3D face model and the control points of the images of the face (It will be created automatically when the software runs)

Face_Texture_Image.bmp: Texture image file (It will be created automatically when the software runs)

To run Texture Mapping Software:

1. Copy *Face_model.obj, Face_image1.bmp, Face_image2.bmp, Face_image3.bmp, Face_image4.bmp, Face_image5.bmp* to *..\Textured_Face_Model\MFC\data*,
2. Start MS VC++ 6.0,
    a. Click **Files→Open Workspace,**
    b. Load *..\Textured_Face_Model\MFC\MFCGL.dsw,*
    c. Click **Compile (Ctrl+F7)** and **Go (F5)** in the toolbar or in the **Build** menu,

or

2. Click MFCGL.exe in *..\Textured_Face_Model\MFC,*
3. The 3D face model will appear,
    a. Click **View Setting→Select** from the menu bar,
    b. Click **Control Points→Left Eye Out** from the menu items,
    c. Click corresponding point on the 3D face model,

    (Use **View Setting→Rotate** and **View Setting→Pan** to rotate or zoom in/out the 3D face model),

d. Repeat 3.a-c for other 17 control points,

e. Click **Control Points→Save,**

(*Face_Model_Control_Points.txt* will be updated automatically),

4. Click **2D Features→Switch to 2D image** from the menu bar. The *Face_image1.bmp* will appear,

   a. Click **View Setting→Select** from the menu items,

   b. Click **Control Points→Left Eye Out** from the menu items,

   c. Click corresponding point on the 2D image,

   (If the point does not exist on the image, continue with next point)

   d. Repeat 3.a-c for other 17 control points,

   e. Click **Control Points→Save,**

   (*Face_image_Control_Point.txt* will be updated automatically),

   f. Click **2D Features→Switch to 2D image** from the menu bar to go back 3D face model,

5. Click **3D Features→Texture** from the menu bar to cover the 3D face model with the texture image. (*Face_Texture_Image.bmp* will be generated),

6. Use **View Setting→Rotate** and **View Setting→Pan** to rotate or zoom in/out the model. The *TAB* key will change the rotation axis in the rotate mode.